TUGAS AKHIR - KS184822

DETEKSI KAPAL DI LAUT INDONESIA MENGGUNAKAN METODE *YOU ONLY LOOK ONCE CONVOLUTIONAL NEURAL NETWORK* (YOLO-CNN)

ADAM FAHMI FANDISYAH
NRP 062116 4000 0065

Dosen Pembimbing
Prof. Drs. NUR Iriawan, M.IKomp., Ph.D.
Dra. Wiwiek Setya Winahju, M.S.

PROGRAM STUDI SARJANA
DEPARTEMEN STATISTIKA
FAKULTAS SAINS DAN ANALITIKA DATA
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA 2020

TUGAS AKHIR - KS184822

DETEKSI KAPAL DI LAUT INDONESIA MENGGUNAKAN METODE *YOU ONLY LOOK ONCE CONVOLUTIONAL NEURAL NETWORK* (YOLO-CNN)

ADAM FAHMI FANDISYAH
NRP 062116 4000 0065

Dosen Pembimbing
Prof. Drs. NUR Iriawan, M.IKomp., Ph.D.
Dra. Wiwiek Setya Winahju, M.S.

PROGRAM STUDI SARJANA
DEPARTEMEN STATISTIKA
FAKULTAS SAINS DAN ANALITIKA DATA
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA 2020

FINAL PROJECT - KS184822

SHIP DETECTION IN INDONESIAN SEA USING YOU ONLY LOOK ONCE CONVOLUTIONAL NEURAL NETWORK (YOLO-CNN)

ADAM FAHMI FANDISYAH
SN 062116 4000 0065

Supervisors
Prof. Drs. NUR Iriawan, M.IKomp., Ph.D.
Dra. Wiwiek Setya Winahju, M.S.

UNDERGRADUATE PROGRAMME
DEPARTMENT OF STATISTICS
FACULTY OF SCIENCE AND DATA ANALYTICS
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA 2020

LEMBAR PENGESAHAN
DETEKSI KAPAL DI LAUT INDONESIA MENGGUNAKAN METODE *YOU ONLY LOOK ONCE CONVOLUTIONAL NEURAL NETWORK* (YOLO-CNN)

TUGAS AKHIR

Diajukan untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Statistika
pada
Program Studi Sarjana Departemen Statistika
Fakultas Sains dan Analitika Data
Institut Teknologi Sepuluh Nopember

Oleh :
Adam Fahmi Fandisyah
NRP. 062116 4000 0065

Disetujui oleh Pembimbing Tugas Akhir :

Prof. Drs. NUR Iriawan, M.IKomp., Ph.D.
NIP. 19621015 198803 1 002 ( )

Dra. Wiwiek Setya Winahju, M.S.
NIP. 19560424 198303 2 001 ( )

Mengetahui,
Kepala Departemen Statistika

KEMENTERIAN PENDIDIKAN DAN KEBUDAYAAN INSTITUT TEKNOLOGI SEPULUH NOPEMBER FAKULTAS SAINS DAN ANALITIKA DATA
DEPARTEMEN STATISTIKA

Dr. Dra. Kartika Fithriasari, M.Si.
NIP. 19691212 199303 2 002

SURABAYA, NOVEMBER 2020

DETEKSI KAPAL DI LAUT INDONESIA MENGGUNAKAN METODE *YOU ONLY LOOK ONCE CONVOLUTIONAL NEURAL NETWORK* (YOLO-CNN)

Nama Mahasiswa : Adam Fahmi Fandisyah
NRP : 062116 4000 0065
Departemen : Statistika-FSAD-ITS
Dosen Pembimbing : Prof. Drs. NUR Iriawan, M.IKomp., Ph.D.
Dra. Wiwiek Setya Winahju, M.S.

Abstrak

Negara Kesatuan Republik Indonesia (NKRI) adalah negara kepulauan terbesar di dunia. Posisi negara Indonesia yang strategis juga berbatasan langsung dengan 10 negara di laut dan 3 di darat. Bentangan garis pantai Indonesia dengan panjang 81.000 km^2, menjadikan laut Indonesia dan wilayah pesisir Indonesia memiliki kandungan kekayaan dan sumber daya alam laut yang sangat berlimpah. Kawasan laut Indonesia masih sering terjadi peristiwa seperti illegal fishing, illegal mining, illegal logging, drugs trafficking dan people smuggling. Kondisi ini menunjukkan bahwa selama ini kurang maksimalnya pengawasan wilayah laut Indonesia. Pesatnya perkembangan teknologi di bidang kecerdasan buatan mendorong ditemukannya deep learning. YOLO-CNN adalah salah satu teknik deep learning yang dikembangkan dengan algoritma untuk mendeteksi sebuah objek secara realtime. Dalam penelitian ini, deteksi tipe kapal dilakukan dengan menggunakan YOLO-CNN dan dievaluasi dengan menghitung nilai Mean Average Precission (mAP) yang dilakukan dengan membandingkan hasilnya dengan ground truth. Hasil deteksi tipe kapal menggunakan YOLO-CNN dengan model yang menggunakan k-means anchor box dapat mengenali tipe kapal pada citra satelit, diperoleh nilai mAP hingga 95,06% pada data training serta 50,41% pada data testing.

Kata kunci: ***Convolutionanl Neural Network, Deep Learning, K-means Anchor Box, Mean Average Precision, YOLO***

(Halaman ini sengaja dikosongkan)

SHIP DETECTION IN INDONESIAN SEA USING YOU ONLY LOOK ONCE CONVOLUTIONAL NEURAL NETWORK (YOLO-CNN)

Name	**:**	**Adam Fahmi Fandisyah**
Student Number	**:**	**062116 4000 0065**
Department	**:**	**Statistics**
Supervisors	**:**	**Prof. Drs. NUR Iriawan, M.IKomp., Ph.D.** **Dra. Wiwiek Setya Winahju, M.S.**

Abstract

The Unitary State Republic of Indonesia (NKRI) is the largest archipelagic country in the world. The strategic position of the Indonesian state is also directly adjacent to 10 countries at sea and 3 on land. The stretch of the Indonesian coastline with a length of 81,000 km^2, makes the Indonesian sea and Indonesian coastal areas contain abundant marine natural resources. The Indonesian marine area still frequently occurs such as illegal fishing, illegal mining, illegal logging, drugs trafficking and people smuggling. This condition shows that so far the supervision of the Indonesian sea area has not been maximal. The rapid development of technology in the field of artificial intelligence is driving the discovery of deep learning. YOLO-CNN is a deep learning technique developed with algorithms to detect an object in realtime. In this study, vessel type detection was carried out using YOLO-CNN and evaluated by calculating the Mean Average Precission (mAP) value which was carried out by comparing the results with ground truth. The results of ship detection using YOLO-CNN with a model using a k-means anchor box can be accessed from ships on satellite imagery, obtained mAP values of up to 95.06% in training data and 50.41% in data testing.

Keywords: ***Convolutionanl Neural Network, Deep Learning, K-means Anchor Box, Mean Average Precision, YOLO***

(Halaman ini sengaja dikosongkan)

KATA PENGANTAR

Puji syukur penulis panjatkan kehadirat Allah SWT yang telah melimpahkan segala rahmat dan hidayah-Nya, sehingga penulis dapat menyelesaikan Laporan Tugas Akhir yang berjudul : "**Deteksi Kapal di Laut Indonesia Menggunakan Metode *You Only Look Once Convolutional Neural Network* (YOLO-CNN)".** Selama proses penyusunan Laporan Tugas Akhir ini, penulis dapat menyelesaikan dengan baik dan lancar tidak lepas dari adanya bantuan berbagai pihak. Oleh karena itu, dengan penuh hormat, ketulusan, dan rendah hati, penulis ingin mengucapkan terima kasih kepada :

1. Ibu Dr. Dra. Kartika Fithriasari, M.Si. selaku Kepala Departemen Statistika FSAD, Ibu Dr. Santi Wulan, S.Si., M.Si. selaku Sekretaris Departemen Bidang Akademik dan Ibu Dr. Vita Ratnasari, S.Si., M.Si. selaku Sekretaris Departemen Bidang Keuangan yang telah menyediakan fasilitas untuk menyelesaikan Tugas Akhir ini.
2. Bapak Prof. Drs. NUR Iriawan, M.IKomp., Ph.D. dan Ibu Dra. Wiwiek Setya Winahju, M.S. selaku dosen pembimbing yang telah meluangkan waktu, mengarahkan, membimbing dengan sabar dan memberikan dukungan yang sangat besar bagi penulis dalam menyelesaikan Tugas Akhir.
3. Ibu Dr. Dra. Kartika Fithriasari, M.Si. dan Ibu Adatul Mukarromah. S.Si., M.Si. selaku dosen penguji yang telah memberikan koreksi dan saran-saran untuk kesempurnaan Tugas Akhir.
4. Bapak Dr. Ir. Setiawan, M.S. selaku dosen wali yang telah memberikan nasehat, motivasi dan bimbingan selama ini.
5. Seluruh dosen Statistika ITS yang telah memberikan ilmu dan pengetahuan, serta segenap karyawan Departemen Statistika ITS.
6. Kedua orang tua Bapak Nurhadi serta Ibu Farida Masviyah yang terhebat dan tertangguh yang selalu mendoakan,

memberikan nasehat, kasih sayang, dan dukungan kepada penulis sehingga termotivasi untuk menyelesaikan Tugas Akhir ini.

7. Semua pihak yang telah memberikan dukungan dan membantu dalam keberhasilan penulis baik secara langsung maupun tidak langsung yang tidak dapat penulis sebutkan satu persatu.

Semoga laporan yang penulis susun dapat bermanfaat dan mampu digunakan sebagaimana mestinya. Penulis menyadari apabila pembuatan Laporan Tugas Akhir ini masih jauh dari kesempurnaan, besar harapan dari penulis untuk menerima kritik dan saran yang berguna untuk perbaikan di masa mendatang. Serta tidak lupa penulis memohon maaf apabila terdapat banyak kekurangan dalam laporan yang telah penulis susun. Atas perhatian dan dukungannya penulis sampaikan ucapan terima kasih.

Surabaya, Juli 2020

Penulis

DAFTAR ISI

DAFTAR TABEL

(Halaman ini sengaja dikosongkan)

DAFTAR GAMBAR

DAFTAR LAMPIRAN

(Halaman ini sengaja dikosongkan)

BAB I
PENDAHULUAN

1.1 Latar Belakang

Negara Kesatuan Republik Indonesia (NKRI) adalah negara kepulauan terbesar di dunia (*the biggest archipelagic state in the world*) yang terdiri dari 17.499 pulau (16.056 pulau sesuai verifikasi PBB pada tanggal 18 Agustus 2017), dimana kepulauan Indonesia menjadi pemisah antara Samudera Pasifik dengan Samudera Hindia. Perairannya menjadi jalur perhubungan laut dua benua antara Asia dan Australia. Dua pertiga wilayah Indonesia merupakan perairan yang luasnya mencapai 5,8 juta km^2, terdiri dari Zona Ekonomi Eksklusif (ZEE) 2,7 juta km^2 dan wilayah laut teritorial 3,1 juta km^2. Luas wilayah perairan Indonesia tersebut telah diakui sebagai Wawasan Nusantara oleh *United Nation Convention of the Law of the Sea* (UNCLOS) pada tahun 1982 (Jaelani & Basuki, 2014). Berdasarkan Pasal 3 UU RI No. 6 tahun 1996 tentang Perairan Indonesia, menyebutkan bahwa Wilayah Perairan Indonesia meliputi : (1) Laut Teritorial Indonesia adalah jalur laut selebar 12 (dua belas) mil laut yang diukur dari garis pangkal kepulauan Indonesia; (2) Perairan Kepulauan Indonesia adalah semua perairan yang terletak pada sisi dalam garis pangkal lurus kepulauan tanpa memperhatikan kedalaman atau jaraknya dari pantai; dan (3) Perairan Pedalaman Indonesia adalah semua perairan yang terletak pada sisi darat dari garis air rendah dari pantai-pantai Indonesia, termasuk di kedalamannya semua, bagian dari perairan yang terletak pada sisi darat dari suatu garis penutup (Kementrian Kelautan dan Perikanan, n.d.).

Posisi negara Indonesia yang strategis juga berbatasan langsung dengan 10 negara tetangga di laut dan 3 negara di darat. Bentangan garis pantai Indonesia dengan panjang 81.000 km^2, menjadikan laut Indonesia dan wilayah pesisir Indonesia memiliki kandungan kekayaan dan sumber daya alam hayati laut yang sangat berlimpah, seperti ikan, terumbu karang, mutiara, mangrove, dan sebagainya. Sejak awal dikumandangkannya Deklarasi Djoeanda

(1957) yang memberikan keteguhan atas konsepsi Indonesia sebagai negara kelautan yang besar, berdaulat dan sejahtera, maka dari itu untuk menjaga kelestarian laut negara ini, terutama guna menjaga sumberdaya ikan perlu didasarkan pada suatu Rencana Pengelolaan Perikanan (RPP). Hal ini sejalan dengan amanat Pasal 33 Undang-Undang Dasar Tahun 1945, yang diatur lebih lanjut dalam UU No. 31 tahun 2004 dan perubahannya UU No. 45 tahun 2009 tentang Perikanan (Pratiwi, 2016). Berdasarkan riset yang dilakukan Lembaga Ilmu Pengetahuan Indonesia (LIPI), potensi kekayaan laut Indonesia terhitung hingga Maret 2019 mencapai Rp 1.772 triliun (LIPI, 2019). Namun yang perlu diketahui, kondisi hingga hari ini tidak memberikan sesuatu yang cukup berarti bagi mereka yang menggantungkan hidupnya pada pengelolaan sumber daya pesisir dan laut Indonesia, seperti nelayan dan petambak tradisional. Mereka ini menjadi komunitas masyarakat yang rapuh secara ekonomi, pendidikan, kesehatan, dan hal-hal yang mendasar lainnya. Ini semua disebabkan salah satunya dari maraknya kasus pencurian ikan atau dengan kata lain adanya tindak kejahatan yang terjadi di perairan Indonesia yaitu *illegal fishing* (Pratiwi, 2016).

Potensi perikanan Indonesia sendiri sangat melimpah ruah, namun hanya sedikit menghasilkan apabila dibandingkan dengan negara lain dengan panjang pantai yang lebih pendek. Jika dibandingkan Cina dengan panjang pantai hanya 14.500 km mampu menjadi penghasil ikan terbesar pada tahun 1999 sebesar 30 juta ton, maka potensi Indonesia masih bisa lebih besar lagi. Dahuri, mantan Menteri Kelautan dan Perikanan dalam Ringkasan Eksekutif Kemenkeu (2015) mengatakan bahwa potensi perikanan Indonesia pada tahun 2011 sekitar 65 juta ton per tahun, dan baru dimanfaatkan 13,4 juta ton atau 20,7%. Hal tersebut mencerminkan bahwa sumber daya alam dari sektor perikanan di Indonesia masih sangat berpotensi untuk digali lagi. Menurut data dari FAO (2014), hasil tangkap ikan laut Indonesia menduduki posisi kedua di dunia dengan hasil tangkapan sebesar 6 juta ton. Posisi pertama sebagai negara penghasil ikan tangkap terbesar yaitu Cina sebesar 14,8 juta ton (Databoks, 2016). Melihat potensi sumber daya alam hasil laut

Indonesia ini menjadikan kapal-kapal asing pun berusaha untuk mencari pendapatan dari laut Indonesia secara ilegal. Menurut Susi Pudjiastuti, mantan Menteri Kelautan dan Perikanan dalam Wardah (2015) mengatakan bahwa kapal-kapal asing tersebut di antaranya berasal dari Thailand, Cina, Filipina, Taiwan, dan Korea Selatan (Syahrani, Al Musadieq, & Darmawan, 2017). Penangkapan ikan secara ilegal telah menyebabkan total kerugian hingga 23 miliar dolar per tahun di seluruh dunia (FAO, 2015), dengan kerugian yang dialami oleh Indonesia mencapai Rp 100 triliun (Jaelani & Basuki, 2014). Kondisi ini menunjukkan bahwa selama ini kurang maksimalnya pengawasan pada wilayah perairan laut Indonesia.

Presiden Joko Widodo telah menetapkan visi untuk bangsa Indonesia berbasis kemaritiman yaitu menjadikan negara Indonesia sebagai Poros Maritim Dunia (PMD). Menurut Presiden RI ke-7 tersebut, sebagaimana dikemukakan dalam pidatonya pada saat pelantikan di hadapan Majelis Permusyawaratan Rakyat (MPR) pada 20 Oktober 2014, “Kita telah terlalu lama memunggungi laut, memunggungi samudera, dan memunggungi selat dan teluk, dan kini saatnya kita mengembalikan semuanya sehingga *‘Jalesveva Jayamahe’*, di laut justru kita jaya, sebagaimana semboyan kita di masa lalu, bisa kembali” (BAPPENAS, 2016). Realisasi dan aktualisasi dalam mewujudkan cita-cita Indonesia sebagai Poros Maritim Dunia ini perlu memiliki kerangka yang mampu untuk mengakomodasi karakter pembangunan bangsa yang heterogen dan dinamis sekaligus mampu menjadi generator bagi kemajuan dalam berbagai sektor dan aktivitas maritim Indonesia yang salah satunya dengan menggunakan pendekatan pengembangan teknologi bidang kemaritiman. Mengingat suatu negara dapat dikatakan sebagai Poros Maritim Dunia apabila salah satunya adalah dapat menjaga sumber daya laut dan kedaulatan negaranya. Pengelolaan wilayah perairan Indonesia, terutama di daerah perairan yaitu perbatasan yang dikelola dengan baik, sehingga dapat digunakan untuk kesejahteraan masyarakat Indonesia.

Dalam mengawasi kegiatan kapal-kapal yang ada di wilayah perairan Indonesia, saat ini digunakan sistem yang disebut dengan

Vessel Monitoring System (VMS) atau Sistem Pemantauan Kapal Perikanan (SPKP). VMS adalah suatu skema pengawasan kegiatan perikanan berbasis satelit dan GPS, yang menggunakan peralatan yang terpasang di kapal perikanan memberikan informasi mengenai kegiatan dan posisi kapal. Setiap alat VMS yang terpasang pada kapal akan memancarkan sinyal ke satelit sehingga dapat dipantau oleh pusat pemantauan kapal di daerah tersebut melalui *website*. Meski memiliki alat kontrol canggih berupa VMS, kawasan laut Indonesia masih rawan terhadap peristiwa seperti *illegal fishing, illegal mining, illegal logging, drugs trafficking* dan *people smuggling*. Hadinata menyatakan bahwa terdapat alat berupa *transmitter online* dan *offline* dalam sistem VMS saat ini. Dalam penelitiannya, penggunaan *transmitter* memiliki beberapa kelemahan dalam hal pendeteksian suatu kapal. Penerimaan sinyal oleh pusat pemantauan kapal sering terganggu, dan tampilan gambar pergerakkan kapal di *website* sulit dimengerti (Hadinata, 2010). Yunus Husen (2015) sebagai Wakil Ketua Tim Satgas Anti Mafia *Illegal Fishing* mengatakan bahwa maraknya kapal-kapal *illegal* dikarenakan kapal laut asing saat masuk ke perairan Indonesia akan selalu mematikan VMS sehingga kapal-kapal tersebut tidak terdeteksi oleh aparat keamanan di dalam sistem (detik.com, 2015). Menurut Suharta (2014) sebagai Kasubid Pemantauan, Pemanfaatan Sumber Daya Perikanan mengatakan, “Kalau untuk semua kapal yang terdeteksi, kita mencoba menggunakan alat baru yaitu dengan sistem potret via radar satelit. Setelah itu proses hasil sensor potret bisa kita dapatkan maksimal dengan waktu 2 jam. Ini sedang kita coba” (detik.com, 2014). Berdasarkan pernyataan tersebut maka diperlukan suatu sistem baru yang dapat menutupi kelemahan ini yang dapat dilakukan dengan deteksi kapal menggunakan citra digital atau citra satelit.

Di sisi lain, pesatnya perkembangan teknologi di bidang kecerdasan buatan mendorong ditemukannya teknologi dalam pembelajaran mesin (*machine learning*). Teknologi ini dapat memungkinkan mesin untuk berpikir selayaknya manusia dengan algoritma-algoritma yang terus menerus dikembangkan. *Deep*

learning merupakan salah satu teknik dari pembelajaran mesin. Penggunaan teknik *Deep Learning* dengan metode *Convolutional Neural Network* pertama kali berhasil diaplikasikan oleh Yann LeCun pada tahun 1998 (LeCun, Haffner, Bottou, & Bengio, 1998). Pada penelitian ini, LeCun mengemukakan bahwa metode *Convolutional Neural Network* (CNN) untuk mengenal tulisan tangan yang digunakan untuk keperluan pembacaan pada suatu dokumen. Dibandingkan algoritma pembelajaran mesin yang konvensional, *deep learning* lebih unggul karena melakukan ekstraksi fitur yang lebih dalam, sehingga performa pembelajaran lebih maksimal. Metode ini merupakan salah satu metode *deep learning* yang terus mengalami perkembangan. Penelitian sebelumnya mengenai penerapan CNN pada citra telah dilakukan oleh Alex Krihevsky pada tahun 2012 yang terbukti berhasil mengungguli metode *machine learning* lainnya yaitu *Support Vector Machine* (SVM) (Putra, Wijaya, & Soelaiman, 2016).

Pada penelitian sebelumnya, Shao dkk (2017) mencoba mendeteksi kapal pada citra satelit optik dengan menggabungkan metode *ReedXiaoli* (RX) dengan *Principal Component Analysis Network* (PCAnet). Keberadaan kapal pada citra satelit dengan latar belakang lautan lepas dapat dianggap sebagai suatu anomali pada citra dan diselesaikan dengan metode RX, dan PCAnet akan melakukan ekstraksi fitur pada citra. Kemudian, Zhang dkk (2016) menggunakan arsitektur S-CNN untuk pendeteksian kapal menggunakan citra *Synthetic Aperture Radar* (SAR). Liu dkk (2017) menggunakan arsitektur *Sea-Land Segmentation-based* (SLS) CNN dalam mendeteksi kapal pada penelitiannya. Citra yang digunakan juga berupa citra SAR namun yang telah dinormalisasi. Sedangkan Li dkk (2018) menggunakan arsitektur CNN dengan tambahan menggunakan metode *Histogram of Oriented Gradient* (HOG) untuk ekstraksi fiturnya. Dalam proses pengujiannya, penelitian tersebut menggunakan metode *sliding window*, namun tidak mencantumkan spesifikasi metode yang digunakan, seperti ukuran *window* dan *stride* (langkah) pada *window*.

Redmon & Farhadi (2018) mengemukakan algoritma *You Only Look Once Convolutional Neural Network* (YOLO-CNN). YOLO-CNN adalah sebuah algoritma yang dikembangkan untuk mendeteksi sebuah objek secara *realtime* dengan akurasi yang cukup baik. Sistem pendeteksian yang dilakukan adalah dengan menggunakan *repurpose classifier* atau *localizer* untuk melakukan deteksi. Sebuah model diterapkan pada sebuah citra di beberapa lokasi dan skala. Daerah dengan citra yang diberi *score* paling tinggi akan dianggap sebagai sebuah pendeteksian (Redmon & Farhadi, 2018). Berbeda dengan penelitian sebelumnya, dalam penelitian ini penulis menggunakan metode YOLO-CNN sebagai metode untuk ekstraksi fitur pada citra dengan menggunakan data *remote sensing* pada citra satelit berukuran 768×768 *pixel* sebagai data penelitian yang diunduh dari *Website* OneAtlas Sand*box* dengan nama dataset "*Ships Detection Machine Learning Dataset*". Menggunakan sumber data yang sama pada sebelumnya telah dilakukan penelitian menggunakan data berjumlah 10.000 citra yang telah diambil secara random, pada penelitian ini dilakukan deteksi menggunakan 2 kategori yaitu kapal dan non kapal pada sebuah citra dengan menggunakan beberapa metode *machine learning* (*Naive Bayes, Linear Discriminant Analysis, K-Nearest Neighbors, Random Forest, Support Vector Machine*) dan *deep learning* (*Convolutional Neural Networks*), untuk metode *machine learning* diperoleh hasil terbaik dengan metode *Random Forest* yang memiliki hasil akurasi sebesar 93%, sedangkan untuk metode *deep learning* dengan menggunakan *Convolutional Neural Networks* diperoleh akurasi sebesar 94% (Chen, Zheng, & Zhou, 2018).

Penulis mengajukan penggunaan *deep learning* dengan arsitektur YOLO-CNN dalam mendeteksi posisi suatu kapal pada citra satelit sehingga dapat dilakukan pendeteksian kapal beserta kategorinya secara *realtime*. Diharapkan cara ini mampu menutupi kelemahan dari sistem VMS yang telah ada dan dapat membantu petugas Angkatan Laut untuk meningkatkan keamanan di

perbatasan perairan Indonesia tanpa harus selalu melakukan pengawasan wilayah perbatasan tersebut.

1.2 Rumusan Masalah

Dalam mengawasi kegiatan kapal-kapal yang ada di Indonesia, saat ini digunakan sistem yang disebut dengan *Vessel Monitoring System* (VMS) atau Sistem Pemantauan Kapal Perikanan (SPKP) yaitu suatu skema pengawasan kegiatan perikanan berbasis satelit dan GPS, yang menggunakan peralatan yang terpasang di kapal perikanan memberikan informasi mengenai kegiatan dan posisi kapal. Di Indonesia terdapat banyak kapal-kapal *illegal* dikarenakan kapal laut asing saat masuk ke perairan Indonesia selalu mematikan VMS sehingga kapal-kapal tersebut tidak terdeteksi di dalam sistem. Oleh karena itu diperlukan sistem yang dapat menutupi kelemahan VMS, sehingga dalam penelitian ini akan dilakukan deteksi kapal menggunakan citra digital atau citra satelit dengan menggunakan *You Only Look Once Convolutional Neural Network* (YOLO-CNN). YOLO-CNN merupakan metode *deep learning* yang bagus dengan algoritma yang dikembangkan untuk mendeteksi sebuah objek secara *realtime*, serta menentukan akurasi dari hasil deteksi kapal.

1.3 Tujuan

Berdasarkan rumusan masalah, tujuan yang ingin dicapai dalam penelitian ini adalah sebagai berikut.

1. Memperoleh model asitektur terbaik YOLO-CNN dalam melakukan deteksi kapal dan tipe kapal.
2. Mendapatkan performa pendeteksian keberadaan kapal pada citra satelit dengan teknik *deep learning* terutama dengan arsitektur YOLO-CNN, dengan memperoleh nilai *loss* dan mAP dari data *training* dan mAP dari data *testing* pada deteksi kapal.

1.4 Manfaat

Manfaat sisi akademis dari tugas akhir ini adalah dapat diketahui kinerja teknik *deep learning* dengan arsitektur YOLO-CNN yang sudah dipilih sehingga dapat menjadi pembanding pada

penelitian-penelitian selanjutnya. Secara umum, penelitian ini diharapkan dapat memberikan kontribusi terhadap pengawasan kedaulatan laut Indonesia dengan dapat dilakukannya deteksi kapal yang terdapat di citra satelit yang berada di perairan Indonesia. Hasil kajian ini akan dapat dimanfaatkan oleh Angkatan Laut Indonesia dalam pengamanan perairan dan pengawasan kapal yang ada di wilayah laut Indonesia. Serta diharapkan dapat mendukung industri maritim Indonesia untuk meningkatkan ilmu pengetahuan pada bidang kelautan.

1.5 Batasan Masalah

Dataset gambar kapal diunduh dari *Website OneAtlas Sandbox* untuk data *training* dan data *testing*. Pada penelitian ini jumlah *anchor box* dibatasi yaitu sebanyak 9 *anchor box* sesuai dengan penelitian pada YOLOv3.

BAB II
TINJAUAN PUSTAKA

Pada bagian ini diuraikan tinjauan pustaka terkait dengan analisis deteksi kapal di Laut Indonesia menggunakan metode YOLO-CNN, yaitu sebagai berikut.

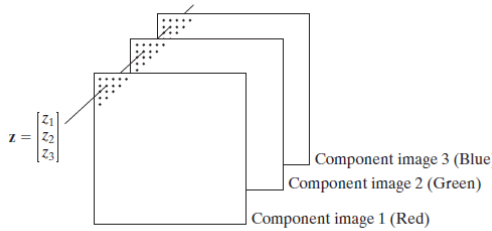
2.1 *Citra Digital*

Secara harfiah, citra adalah gambar pada dua dimensi. Ditinjau dari sudut pandang matematis, citra merupakan fungsi dari intensitas cahaya yang termaktub dalam dua dimensi. Citra digital adalah data digital pada komputer yang merepresentasikan sebuah citra. Sebuah citra yang diwakili oleh $f(a, b)$ berbentuk matriks yang terdiri dari A baris dan B kolom, dimana perpotongan antara kolom dan baris inilah yang disebut *picture element*, *image element*, *pels*, atau *pixel*. Citra digital juga dapat digambarkan sebagai fungsi $f(a, b)$ dengan a dan b merupakan koordinat pada sebuah bidang datar yang merepresentasikan kumpulan *pixel* dalam dua dimensi (Gonzalez & Woods, 2008).

Dalam penelitian tentang citra digital, terdapat tipe-tipe dasar citra digital yaitu citra RGB dan citra *grayscale*. Citra warna RGB biasanya disebut Citra *True Color* atau citra asli yang ditangkap oleh kamera. Setiap *pixel* pada citra warna RGB memiliki tiga komponen warna, yaitu merah (*Red* = R), hijau (*Green* = G), dan biru (*Blue* = B). Setiap komponen warna memiliki nilai minimum 0 dan nilai maksimum 255. Warna pada *pixel* ditentukan dari kombinasi ketiga komponen warna tersebut. Citra *grayscale* merupakan citra yang hanya memiliki satu nilai pada setiap *pixel*-nya, nilai tersebut disebut sebagai derajat keabuan yang memiliki nilai minimum 0 (warna hitam) dan nilai maksimum 255 (warna putih). Pada praktiknya, citra RGB direpresentasikan dalam susunan tiga fungsi yang ditulis sebagai berikut.

$$f(a,b) = \left[r(a,b) \quad g(a,b) \quad b(a,b) \right] \tag{2.1}$$

Warna suatu gambar adalah sebuah fungsi. Namun, pada kasus ini, nilai *pixel* pada posisi (*a*, *b*) bukanlah nilai tunggal, melainkan merupakan vektor yang memiliki tiga tingkat intensitas warna yang berbeda sesuai warnanya seperti Gambar 2.1 (Sewak, Karim, & Pujari, 2018).

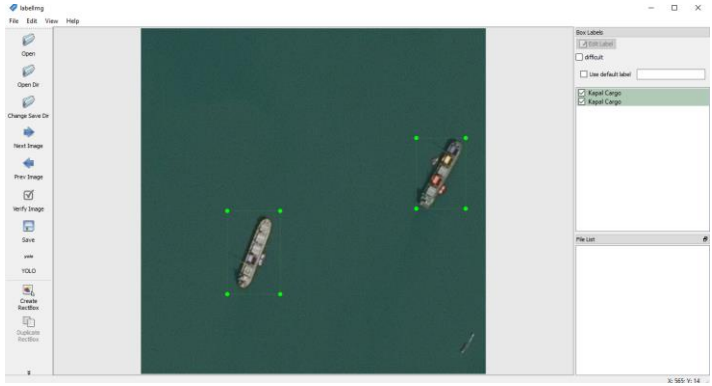


Gambar 2. 1 Gambar Bercitra RGB sebagai Vektor

2.2 *Image Labelling*

Image Labelling atau pelabelan gambar atau anotasi gambar adalah proses mengidentifikasi atau mengenali unit yang berbeda dalam suatu gambar (Gonzalez & Woods, 2008). Label maksudnya adalah *tag* dari data yang ditambahkan dalam *machine learning* atau *deep learning model*. Contohnya gambar kucing diberikan *tag*/ *label* "kucing" di tiap masing masing gambar kucing dan gambar anjing diberikan *tag*/*label* "anjing" di tiap masing gambar anjing. Proses ini membantu kita membuat gambar dapat dibaca untuk *computer vision*. Beberapa jenis pelabelan gambar untuk *computer vision* adalah *bounding boxes*, *polygonal segmentation*, *line annotation*, *landmark annotation*, *3D cuboids*, *semantic segmentation*, dll. *Bounding boxes* adalah jenis anotasi pelabelan data yang paling umum digunakan dalam *computer vision*. *Bounding box* adalah kotak persegi panjang yang digunakan untuk menentukan lokasi objek target. Mereka dapat ditentukan oleh koordinat sumbu x dan y di sudut kiri atas dan koordinat x dan y di sudut kanan bawah persegi panjang. *Bounding box* umumnya digunakan dalam pendeteksian objek. *Bounding box* biasanya

diwakili oleh dua koordinat (x_1, y_1) dan (x_2, y_2) atau dengan satu koordinat (x_1, y_1) dan lebar (w) dan tinggi (h) dari kotak pembatas.



Gambar 2. 2 *Image Labelling*

Gambar 2.2 merupakan contoh anotasi atau pelabelan citra yang dilakukan dengan menggunakan bantuan *software labelImg*, selanjutnya akan diperoleh matriks $\mathbf{Y_j}$ yang memiliki elemen diantaranya kategori kelas (c) , nilai koordinat $(\mathrm{bx}, \mathrm{by})$, serta lebar (bw) dan tinggi (bh) *ground truth bounding box* dalam citra.

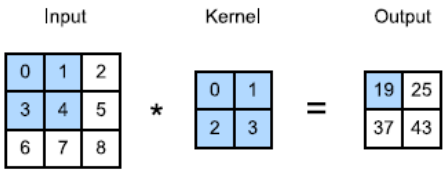
2.3 *Convolutional Neural Network* (CNN)

CNN didesain untuk mengenali pola visual dari suatu gambar secara langsung dengan proses minimal. CNN pada dasarnya adalah *neural network* dengan banyak *layer* (*multilayer neural network*). Tiap neuron menerima beberapa *input* dan melakukan perhitungan *dot product*. CNN memiliki *loss function* yang terletak di *fully connected layer* terakhir dan memiliki fungsi aktivasi. Nama "*convolutional neural network*" mengindikasikan bahwa *neural network* tersebut menggunakan operasi matematika yang disebut *convolution*. CNN sebagai *neural network* yang menggunakan *convolution* sebagai pengganti perkalian matriks umum di setidaknya satu *layer* (Goodfellow, Bengio, & Courville, 2017). CNN mengatur neuron dalam bentuk tiga dimensi, yaitu

lebar, panjang, dan *width*. Pada CNN terdapat dua fase operasi yaitu *feature extraction* serta *classification*. Secara umum, terdapat tiga *layer* utama dalam CNN, yaitu *convolutional layer*, *pooling layer*, dan *fully connected layer* (Sewak, Karim, & Pujari, 2018). Dalam CNN juga terdapat operasi layer *upsampling dan downsampling*. Kontribusi CNN pada *feature extraction layer* adalah pada *convolution layer* dan *pooling layer*. Selanjutnya hasil dari beberapa operasi convolutional dan pooling dihasilkan *feature map* sebagai *input* pada proses *classification* yang dilakukan pada *fully connected layer*. Berikut dijelaskan definisi dan proses operasi layer dalam CNN lebih detail.

1. *Convolutional Layer*

Pada *layer* ini terdapat fungsi *convolution* yang berguna untuk mengekstrak suatu nilai dari *input* gambar. Manfaat lain dari *convolution layer* berfungsi untuk mengurangi kompleksitas perhitungan. *Convolution* didefinisikan sebagai operasi matematika yang menjelaskan cara untuk menggabungkan dua set informasi. *Convolutional layer* membutuhkan *input*, kemudian mengaplikasikan *convolutional kernel*/*filter*, dan memberikan hasil berupa *feature map* sebagai *output*.



Gambar 2. 3 Ilustrasi *Convolutional Layer*

Perlu diketahui bahwa dimensi *output* lebih kecil daripada dimensi *input* seperti Gambar 2.3. Hal ini disebabkan oleh *kernel* yang memiliki lebar lebih dari 1 *pixel*, dan *convolution* hanya bisa dilakukan ketika dimensi *input convolutional* sama dengan dimensi *kernel*. Secara matematis, apabila dimensi *input* $A \times B$ dan dimensi *kernel* $a \times b$, maka dimensi *output* adalah $(A - a + 1) \times (B - b + 1)$.

Untuk menghitung semua *output*, *kernel* digeser untuk setiap *pixel* (Zhang, Lipton, Li, & Smola, 2020).

Operasi *convolution* apabila *input* dan *kernel* berukuran dua dimensi ditulis pada persamaan sebagai berikut.

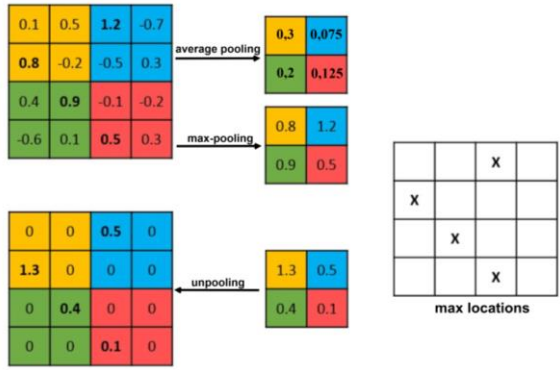
$$FM_{a,b} = bias + \sum_{c}^{C} \sum_{d}^{D} Z_{c,d} \times X_{a+c-1,b+d-1} \tag{2.2}$$

dimana,

$X_{a+c-1,b+d-1}$: variabel *input*
$FM_{a,b}$: *feature map* pada *pixel* ke-*a,b*
Bias : *bias* pada *feature map*
$Z_{c,d}$: bobot pada *convolution kernel* ke-*c,d*
a : 1, 2, …, *A*. *A* merupakan panjang *pixel* pada *feature map*
b : 1, 2, …, *B*. *B* merupakan lebar *pixel* pada *feature map*
c : 1, 2, …, *C*. *C* merupakan panjang *pixel* pada *convolution kernel*
d : 1, 2, …, *D*. *D* merupakan lebar *pixel* pada *convolution kernel*

2. *Pooling Layer*

Pooling layer pada umumnya disisipkan diantara *convolutional layer*. *Pooling layer* setelah *convolutional layer* digunakan untuk mengurangi dimensi *input*.



Gambar 2. 4 Ilustrasi *Pooling Layer*

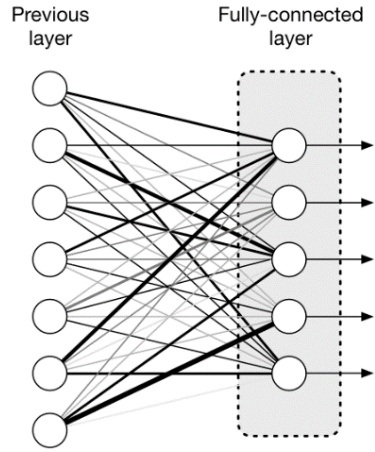
Pooling layer digunakan untuk merangkum informasi oleh suatu *convlution layer*. Tidak seperti *convolutional layer*, p*ooling layer* tidak memiliki parameter sehingga operasi *pooling* bersifat deterministik (Zhang, Lipton, Li, & Smola, 2020). Contoh operasi *pooling* diberikan pada Gambar 2.4, kebalikan dari *pooling layer* adalah *unpooling layer*. Dalam operasi *pooling layer* digunkan fungsi *max*() atau mean() untuk mengurangi *input* data. Operasi ini dinamakan *max pooling* dan *average pooling*.

Pada umumnya, *pooling layer* berukuran 2 × 2 dengan *stride* 2 (Patterson & Gibson, 2017). *Max pooling layer* digunakan sebagai operasi *downsampling* yang menurunkan sampel gambar *input*, dimana sering digunakan pada proses *encoder* yang dilakukan untuk mempelajari representasi dari data *input*. Selanjutnya, *decoder* digunakan untuk mendekompresi atau merekontruksi kembali data *input* dengan dilakukan proses *upsampling*.

Upsampling merupakan teknik dalam CNN untuk membuat dimensi spasialnya sama dengan gambar *input* setelah dilakukan *downsampling*. Salah satu teknik upsampling dapat dilakukan dengan menggunkan kebalikan dari teknik *max-pooling* yaitu *max-unpooling*. Pertama, posisi indeks nilai maksimum disimpan untuk setiap *max-pooling layer* selama langkah encoder. Posisi indeks yang disimpan kemudian digunakan selama langkah Decoder di mana piksel *input* dipetakan ke posisi indeks yang disimpan dan mengisi nol pada posisi lainnya.

3. *Fully-connected layer*

Fully connected layer (juga dikenal sebagai FNN atau *dense layer*) ditambahkan pada bagian akhir arsitektur CNN. FNN bekerja mirip seperti *Multi Layer Perceptron* (MLP) yang terdiri dari beberapa FNN (Sewak, Karim, & Pujari, 2018).



Gambar 2. 5 lustrasi *Fully-Connected Layer*

Pada Gambar 2.5 *output* hasil dari *feature extraction* pada operasi *convolution layer* dan *pooling layer* digunakan sebagai *input* pada *fully connected layer* untuk dilakukan klasifikasi. Sebelum dilakukan operasi *fully connected layer* dilakukan *flatenning* atau mengubah *output* pada *feature extraction* yang memiliki bentuk multi dimensi menjadi satu dimensi vektor, lalu dilanjutkan dengan perkalian *dot product* sebagai langkah untuk dilakukan klasifikasi dengan bantuan *softmax fuction* untuk memperoleh *confidence score* dari setiap kelas yang digunakan dalam penelitian.

2.4 *Transfer Learning*

Suatu arsitektur CNN yang ada dan sudah dilatih dapat digunakan pada data yang berbeda. Maka daripada membuat arsitektur CNN baru dan melatihnya dari awal, model CNN yang sudah dilatih tadi diadaptasi untuk data yang baru dengan teknik yang disebut *transfer learning*. *Transfer learning* adalah proses menyalin pengetahuan dari jaringan yang sudah ada ke jaringan yang baru untuk menyelesaikan masalah serupa (Sewak, Karim, & Pujari, 2018). Teknik ini diimplementasikan untuk mempercepat

proses *training* dan meningkatkan performa model (Patterson & Gibson, 2017).

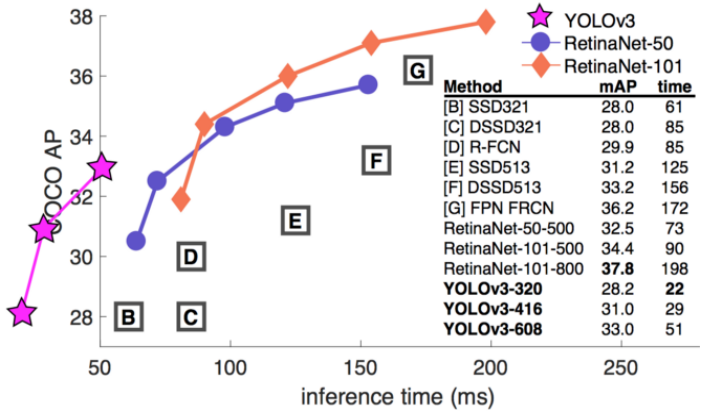
Suatu model yang telah dilatih disebut *pre-trained* model. *Pre-trained* model dibentuk oleh seseorang/organisasi yang memiliki akses terhadap dataset skala besar dan sumber daya yang mumpuni dalam mengolah model. Contoh *pre-trained* model adalah Darknet, Xception, VGGNet, ResNet, Inception, MobileNet, DenseNet, NASNet dan GoogleNet. *Pre-trained* model tersebut dilatih pada dataset skala besar seperti *ImageNet*, yaitu dataset gambar yang disusun sesuai hierarki *WordNet*, mempunyai gambar sebanyak 14.197.122 buah dan 1,2 juta gambar yang terlabeli (ImageNet, 2016). Dua penggunaan *pre-trained* model yaitu, *Feature extractor* dan *Fine-tuning existing model*. *Feature extractor* yaitu penghapusan *Fully-connected layer* terakhir dan menggunakan sisanya sebagai *fixed feature extractor* untuk dataset yang lebih kecil, sedangkan *Fine-tuning existing model* yaitu melatih ulang *pre-trained* model dengan data baru dan melakukan *tuning* parameter (*weights*) dengan *backpropagation*.

2.5 *You Only Look Once (*YOLO*)*

Sebuah pendekatan untuk sistem pendeteksian objek, yang ditargetkan untuk pemrosesan secara *realtime*. YOLO membingkai pendeteksian objek sebagai masalah regresi tunggal, dimana dari *pixel* gambar langsung ke kotak pembatas (*bounding box*) spasial yang terpisah dan probabilitas kelas yang terkait. YOLO melakukan pendeteksian dan pengenalan objek dengan sebuah jaringan syaraf tunggal (*single neural network*), yang memprediksi kotak-kotak pembatas dan probabilitas kelas secara langsung dalam satu evaluasi (Redmon, Divvala, Girshick, & Farhadi, 2016).

Untuk mendapatkan prediksi final, faktor penentunya adalah *class confidence score* yang didapat, berdasarkan probabilitas kondisional kelas dan *box confidence score*. *Class confidence score* mengukur nilai kepercayaan pada klasifikasi dan lokalisasi objek. *Class confidence score* memberi nilai kepercayaan kelas spesifik untuk setiap kotak, yang mengkodekan kemungkinan kelas yang

muncul di kotak dan seberapa sesuainya kotak yang diprediksi dengan objek.



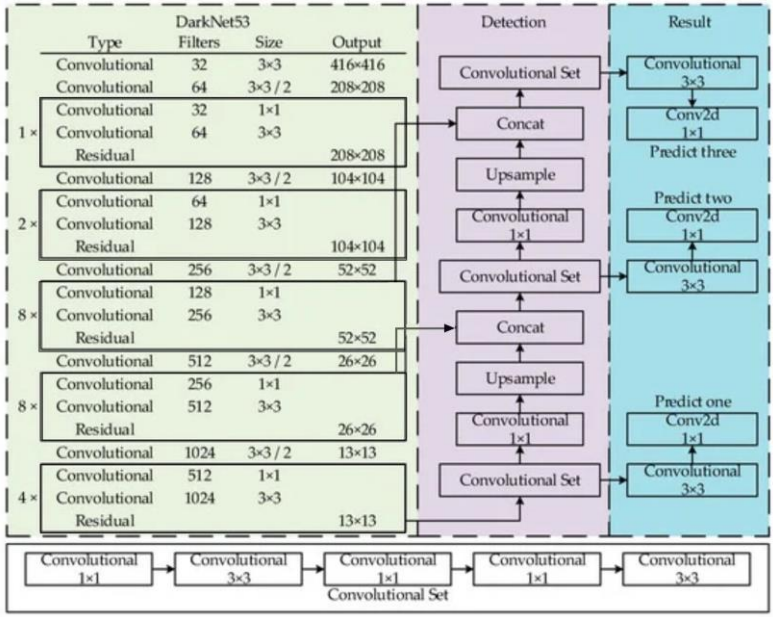
Gambar 2. 6 Performa YOLO-CNN

Berdasarkan Gambar 2.6 YOLO-CNN memiliki performa yang cukup baik dibandingkan dengan metode deteksi objek lainnya dengan waktu pembelajaran yang tercepat, performa terbaik pada deteksi objek dimiliki oleh RetinaNet namun memiliki proses pembelajaran yang lebih lama dibandingkan dengan YOLOv3.

2.5.1 Arsitektur YOLO

YOLOv3 menggunakan arsitektur dari Darknet 53 yang artinya memiliki 53 *convlutional layers*. YOLOv3 memiliki 53 lapisan konvolusional yang disebut Darknet 53 yang terdiri dari struktur *convolutional* dan *residual* (Redmon & Farhadi, 2018). Pada YOLOv3 *convolutional layer* selalu diikuti dengan *batch normalization* dan *leaky ReLu*. *Residual block* atau *shortcut connection* pada YOLOv3 dilakukan dengan menjumlahkan *input* sebelum *convolutional layer residual block* dengan hasil dari *convolutional layer filter* 1x1 diikuti *batch normalization* dan *leaky ReLu*, dilanjutkan dengan *convolutional layer filter* 3x3 dengan *batch normalization*, dan pada bagian akhir dilakukan *leaky ReLu*. *Residual layer* pada YOLOv3 didefinisikan sebagai operasi

penjumlahan antara *input* awal dengan *output* hasil sesudah dilakukan convolutional layer pada *residual block*.



Gambar 2. 7 Arsitektur YOLOv3

Berdasarkan arsitektur YOLOv3 pada Gambar 2.7, *input* citra awal pada arsitektur YOLOv3 diatur berukuran 416x416, pada YOLOv3 gambar *input* diatur agar memiliki *input* dengan kelipatan 32. *Convolutional layer* yang pertama memiliki *filter* berukuran 3x3 sebanyak 32 dengan *stride* 1 *zero padding* 1, sehingga diperoleh *output* berukuran 416x416. Selanjutnya dilakukan operasi *convolution layer* kedua dengan *filter* 3x3 sebanyak 64 *stride* 2 *zero padding* 1 sehingga diperoleh *output* berukuran 208x208. Lalu dilakukan *residual block* 1 kali pengulangan yang terdiri dari 2 *convolutional layer* yang diakhiri residual, *convolutional layer* yang pertama memiliki *filter* 1x1 sebanyak 32 dengan *stride* 1 dan dilanjutkan dengan *convolutional layer* dengan *filter* berukuran 3x3

sebanyak 64 dengan *stride* 1 *zero padding* 1 sehingga diperoleh *output* pada *residual* dengan ukuran yang sama dengan ukuran sebelumnya yaitu 208x208, pada langkah ini sudah digunakan total sebanyak 4 *convolutional layer*.

Setelah itu dilanjutkan dengan *convolutional layer* kelima dengan *filter* 3x3 sebanyak 128 dengan *stride* 2 *zero padding* 1 sehingga didapatkan *output* dengan ukuran 104x104. Kemudian dilanjutkan lagi dengan *residual block* 2 kali perulangan yang terdiri dari 2 *convolutional layer* dan diakhiri *residual*, *convolutional layer* pertama menggunakan *filter* berukuran 1x1 sebanyak 64 dengan *stride* 1 dan dilanjutkan lagi dengan *convolutional layer* dengan *filter* berukuran 3x3 sebanyak 128 dengan *stride* 1 *zero padding* 1 sehingga diperoleh *output* pada *residual* yang memiliki ukuran tetap dengan ukuran sebelumnya yaitu 104x104, pada langkah ini telah dilakukan total sebanyak 9 *convolutional layer*. Selanjutnya pada *convolutional layer* kesepuluh digunakan *filter* berukuran 3x3 sebanyak 256 dengan *stride* 2 *zero padding* 1 sehingga diperoleh *output* berukuran 52x52. Lalu dilanjutkan lagi dengan *residual block* dengan 8 kali perulangan yang terdiri dari 2 *convolutional layer* dan diakhiri *residual*, *convolutional layer* yang pertama memiliki *filter* berukuran 1x1 sebanyak 128 dengan *stride* 1 dan dilanjutkan *convolutional layer* dengan *filter* 3x3 sebanyak 256 dengan *stride* 1 *zero padding 1*, maka diperoleh *output* pada *residual* dengan ukuran yang sama dengan sebelumnya yaitu 52x52, pada langkah ini sudah digunakan sebanyak 26 *convolutional layer*.

Selanjutnya pada *convolutional layer* ke-27 digunakan *filter* 3x3 sebanyak 512 *stride* 2 *zero padding* 1 dan diperoleh *output* berukuran 26x26. Kemudian dilakukan *residual block* dengan 8 kali perulangan yang terdiri dari 2 *convolutional layer* dan diakhiri *residual*, convolutional layer yang pertama memiliki *filter* berukuran 1x1 sebanyak 256 dengan *stride* 1 dan dilanjutkan *convolutional layer* dengan *filter* berukuran 3x3 sebanyak 512

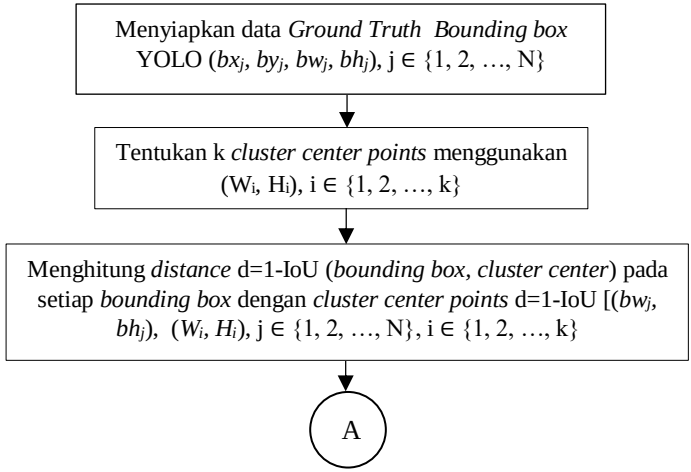
stride 1 *zero padding* 1, sehingga diperoleh *output* dengan ukuran yang sama dengan sebelumnya pada *residual* yaitu 26x26, pada langkah ini sudah digunakan sebanyak 43 *convolutional layer*. Selanjutnya pada *convolutional layer* ke-44 digunakn *filter* berukuran 3x3 sebanyak 1024 dengan *stride* 2 *zero padding* 1 sehingga diperoleh *output* 13x13. Setelah itu dilanjutkan *residual block* dengan 4 kali perulangan yang terdiri dari 2 *convolutional layer* dan *residual*, *convolutional layer* yang pertama menggunakan *filter* 1x1 sebanyak 512 dengan *stride* 1 lalu dilanjutkan *convolutional layer* dengan *filter* 3x3 sebanyak 1024 dengan *stride* 1 *zero padding* 1, maka diperoleh *output* pada *residual* dengan ukuran yang sama dengan sebelumnya yaitu 13x13, sampai langkah ini sudah digunakan sebanyak 52 *convolutional layer* sebagai *feature extraction* pada Darknet 53, kemudian untuk melakukan deteksi objek digunakan *fully connected layer* maka total manjadi 53 *convolutional layer* sehingga YOLOv3 dikenal dengan arsitektur Darknet 53.

Dalam melakukan deteksi objek pada arsitektur YOLOv3 dilanjutkan dengan beberapa *convolutional layer* lagi sehingga YOLOv3 dapat dilakukan deteksi objek menggunakan 3 skala yang berbeda untuk mendeteksi objek dengan ukuran kecil, sedang, dan besar. Untuk mendeteksi objek dengan ukuran besar *output* terakhir *feature map* pada *convolutional layer* ke 52 berukuran 13x13 dilanjutkan dengan *convolutional set* yang terdiri dari 5 *convolutional layer* dengan ukuran *filter* 1x1, 3x3, 1x1, 1x1, dan 3x3 yang dilakukan secara berurutan. Selanjutnya dilakukan lagi *convolutional layer* dengan *filter* 3x3, yang diakhiri dengan *convolutional layer* 2d berukuran filter 1x1, pada bagian akhir diperoleh *output* berupa deteksi objek dengan algoritma YOLO dengan *grid* 13x13 yang digunakan untuk mendeteksi objek dengan ukuran yang cenderung besar, informasi pada citra *input* berukuran 416x416 dibagi menjadi 13x13 *grid*, maka dimensi resolusi spasial pada *final feature map* 1/32 kali lebih kecil daripada *input image*.

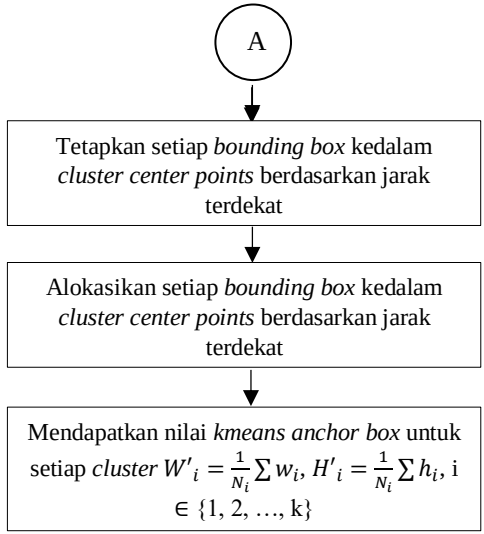
Selanjutnya untuk melakukan deteksi pada skala selanjutnya digunakan untuk objek berukuran sedang, dilakukan operasi *concat* atau penggabungan antara *output* berukuran 26x26 pada *residual block* pada langkah 43 *convolutional layer*, dengan hasil *upsample* 2 kali dari lanjutan *convolutional set* yang dilanjutkan dengan *convolutional layer* dengan *filter* 1x1. Setelah itu dilanjutkan lagi dengan *convolutional set* yang terdiri dari 5 *convolutional layer*, lalu dilanjutkan dengan *convolutional layer* dengan *filter* berukuran 3x3, hingga pada bagian akhir digunakan *convolutional layer* 2d dengan *filter* 1x1, dan diperoleh *output* berupa deteksi objek dengan algoritma YOLO dengan menggunakan *grid* 26x26, pada skala ini bertujuan untuk mendeteksi objek dengan ukuran sedang karena informasi pada citra *input* berukuran 416x416 dibagi menjadi 26x26 *grid*, sehingga dimensi resolusi spasial pada *final feature map* 1/16 kali lebih kecil daripada *input image*. Pada skala deteksi yang terakhir digunakan untuk mendeteksi objek berukuran cenderung kecil, dimana pada langkah ini dilakukan operasi *concat* atau penggabungan antara *upsample* 2 kali dari *output* pada *convolutional set* sebelumnya yang dilakukan lagi *convolutional layer* 1x1, dengan *output residual block* pada total digunakan 26 *convolutional layer* yang berukuran 52x52. Setelah itu dilanjutkan lagi dengan *convolutional set* yang terdiri dari 5 *convolutional layer*, lalu dilanjutkan dengan convolutional layer dengan *filter* berukuran 3x3 dan *convolutional layer* 2d dengan *filter* 1x1, sehingga pada bagian akhir diperoleh *output* berupa deteksi objek dengan algoritma YOLO dengan menggunakan *grid* 52x52, pada skala ini bertujuan untuk mendeteksi objek dengan ukuran yang cenderung kecil, karena informasi pada citra *input* berukuran 416x416 dibagi menjadi 52x52 *grid*, sehingga dimensi resolusi spasial pada *final feature map* 1/8 kali lebih kecil daripada *input image*.

2.5.2 *Anchor Box*

Anchor box adalah satu set kotak pembatas yang telah ditentukan dengan tinggi dan lebar tertentu. Kotak-kotak ini didefinisikan untuk menangkap skala dan rasio aspek kelas objek tertentu yang ingin dilakukan deteksi dan biasanya dipilih berdasarkan ukuran objek dalam kumpulan data pelatihan. Dalam deteksi objek perlu dilakukan *k-means clustering* agar dapat disesuaikan *anchor box* yang digunakan pada model dengan *ground truth bounding box* pada *dataset* sehingga memiliki hasil prediksi yang baik. *K-means* merupakan salah satu metode *clustering* non-hirarki untuk mengelompokan data dalam beberapa kelompok. Data dengan karakteristik yang sama dikelompokan dalam satu *cluster* dan data yang memiliki karakteristik yang berbeda dikelompokan dengan kelompok yang lain sehingga data yang berada dalam satu kelompok memiliki tingkat variasi yang kecil (Agusta, 2007). Pada Gambar 2.8 ditampilkan algoritma *k-means anchor box* mulai dari menyiapkan data *Ground Truth Bounding box* hingga mendapatkan nilai *k-means anchor box*.



Gambar 2. 8 Diagram Alir Algoritma *K-means Anchor Box*



Gambar 2. 8 Diagram Alir Algoritma *K-means Anchor Box* (Lanjutan)

Keterangan :

N : jumlah semua *ground truth bounding box*

N_i : jumlah *ground truth bounding box* pada cluster ke-i

k : jumlah cluster

bx_j, by_j : kordinat x,y *bounding box*

bw_j, bh_j : lebar dan tinggi *bounding box*

W_i, H_i : lebar dan tinggi *anchor box*

d : jarak ke *cluster center points*

2.5.3 *Intersection over Union* (IoU)

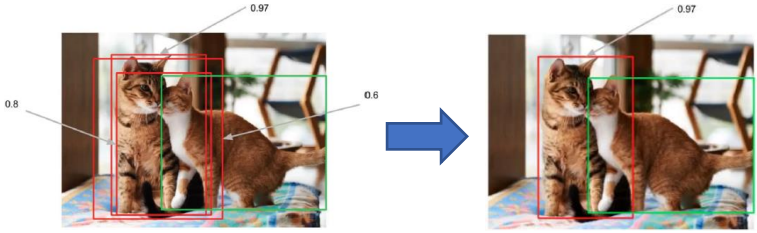
Intersection over union menghitung luas area yang berpotongan lalu membaginya dengan luas area gabungan antar 2 *bounding box*. Nilai IoU digunakan untuk menentukan kesesuaian *bounding box* yang diprediksi dengan luas objek yang sesungguhnya pada citra (*ground truth*). IoU dapat dihitung dengan persamaan sebagai berikut.

$$IoU = \frac{Area\ Irisan}{Area\ Gabungan} =$$  (2.3)

2.5.4 *Non Max Supression*

Non Max Suppression berhubungan dengan algoritma objek deteksi untuk memfilter *bounding box*. *Non Max Suppression* membuat seleksi berdasarkan IoU dengan mengurangi *bounding box* yang muncul secara berlebihan. Seleksi dilakukan hingga tersisa satu saja *bounding box* dan memiliki nilai *confidence* tertinggi. *Input Non Max Suppression* yaitu daftar kotak Proposal B, skor kepercayaan kelas yang sesuai S dan *overlap* ambang/ *threshold* N, sehingga diperoleh *output* daftar proposal yang dilakukan filter pada D. Berikut Algoritma dalam melakukan *Non Max Supression*.

1. Pilih proposal dengan skor kepercayaan tertinggi, hapus dari B dan tambahkan ke daftar proposal akhir D. (Awalnya D kosong).
2. Sekarang bandingkan proposal ini dengan semua proposal lalu hitung IoU (*Intersection over Union*) dari proposal ini dengan setiap proposal lainnya. Jika IoU lebih besar dari ambang/*threshold* N, hapus proposal itu dari B.
3. Sekali lagi ambil proposal dengan kepercayaan tertinggi dari sisa proposal di B dan hapus dari B dan tambahkan ke D.
4. Sekali lagi hitung IOU proposal ini dengan semua proposal di B dan hilangkan kotak-kotak yang memiliki IoU lebih tinggi dari ambang batas/*threshold* ditambahkan di D.
5. Proses ini diulangi hingga tidak ada lagi proposal yang terlewat di B.
6. Mendapatkan satu *confidence score* yang paling maksimal pada D sebagai *output bounding box* yang terbaik untuk menandai akan adanya objek kelas.



Gambar 2. 9 *Non-Max Supression*

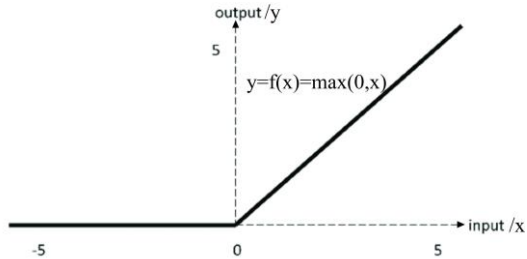
Pada gambar 2.9, terdapat tiga *bounding box* berwarna merah dan masing-masing memiliki nilai IoU 0.8, 0.97, dan 0.6. NMS menyeleksi nilai tertinggi dan menghapus *bounding box* yang memiliki nilai IoU yang lebih kecil, sehingga hanya terdapat satu *bounding box* dengan nilai *confidence* yang tertinggi.

2.5.5 Fungsi Aktivasi

Fungsi aktivasi merupakan fungsi yang digunakan pada jaringan saraf untuk mengaktifkan atau tidak mengaktifkan neuron. Hasil perhitungan antara *input*, *weight* dan bias akan dihitung lagi dengan persamaan dari fungsi aktivasi untuk mendapatkan *output* dari setiap layer. Pada penelitian ini, penulis menggunakan fungsi aktivasi ReLU pada convolutional layer dan fungsi aktivasi *softmax* pada *output layer* untuk mendapatkan hasil yang merupakan data kategoris. Fungsi ReLU akan merubah nilai *pixel* pada *input* namun tidak merubah dimensi *input* sehingga dimensi *output* berukuran sama dengan *input* (Patterson & Gibson, 2017). ReLU (*Rectified Linear Unit*) merupakan fungsi aktivasi sebagai solusi dari menghilangnya gradien/*vanishing gradient* dengan cara menerapkan fungsi aktivasi tersebut sebagai persamaan (2.4).

$$f(x) = \max(0, x) \tag{2.4}$$

ReLU sangat mempercepat proses konvergensi yang dilakukan dengan *stochastic gradient descent*. ReLU mudah untuk dihitung karena tidak ada matematika yang rumit (*cheap to compute*). Oleh karena itu, model ini memerlukan waktu lebih sedikit untuk dilatih atau dijalankan. Gambar 2.10 merupakan grafik fungsi aktivasi ReLU.



Gambar 2. 10 Fungsi Aktivasi ReLU

Gambar 2.10 menunjukkan bawah fungsi aktivasi ini akan memaksa nilai negatif untuk menjadi 0, sesuai dengan (Krizhevsky, Sutskever, & Hinton, 2012). Fungsi *softmax* menghitung probabilitas terhadap sejumlah kejadian. Dalam kasus *machine learning* / *deep learning* fungsi ini akan menghitung probabilitas setiap label yang ditebaknya. Kelebihan dari fungsi aktifasi ini adalah nilai keluarannya yang berupa rentang probabilitas 0 sampai 1. Jika setiap hasil fungsi *softmax* dijumlahkan maka akan bernilai 1. *Softmax layer* digunakan apabila pada permasalahan *multiclass classification*, *output* layer biasanya memiliki lebih dari satu neuron. Berikut adalah rumus dari fungsi aktivasi *softmax*. Misalkan $a = [a_1, a_2, \cdots, a_m]^T$ merupakan sebuah vektor dengan m buah elemen, *softmax* didefinisikan sebagai berikut.

$$\sigma(a_j) = \frac{\exp(a_j)}{\sum_{k=1}^{m} \exp(a_j)} \tag{2.5}$$

dimana $\sum_{j=1}^{m} \sigma(a_j) = 1$ untuk *softmax*. Berikut merupakan contoh perhitungan fungsi aktivasi *softmax*.

$$a \begin{bmatrix} 2,0 \rightarrow \\ 1,0 \rightarrow \\ 0,1 \rightarrow \end{bmatrix} \sigma(a_j) = \frac{\exp(a_j)}{\sum_{k=1}^{m} \exp(a_j)} \begin{matrix} \rightarrow p = 0,7 \\ \rightarrow p = 0,2 \\ \rightarrow p = 0,1 \end{matrix} \tag{2.6}$$

Berdasarkan contoh pada persamaan (2.6), diperoleh nilai *a* yang merupakan nilai *output* dari *fully connected layer* untuk tiga kelas deteksi objek yang berbeda, dengan menggunakan fungsi *softmax* diperoleh probabilitas untuk ketiga kelas tersebut, dimana masing-masing nilai probabilitas dari kelas kesatu hingga kelas ketiga yaitu 0,7, 0,2, dan 0,1.

2.5.6 *Batch Normalization*

Batch Normalization adalah teknik untuk melatih *deep neural networks* dengan melakukan *standardizes input layer* untuk setiap *mini-batch*. *Batch Normalization* diperkenalkan oleh Ioffe dan Szgedy dalam tulisannya pada tahun 2015. Lapisan *batch normalization*, seperti namanya digunakan unutuk melakukan normalisasi aktivasi sebuah *input* sebelum meneruskannya ke lapisan berikutnya dalam jaringan. *Batch normalization* digunakan untuk mempercepat pelatihan jaringan dengan mengurangi *internal covariate shift*. Dimana *internal covariate shift* mengacu pada adanya perubahan distribusi *input* pada *learning system*. BN berguna untuk meningkatkan kecepatan, performasi dan kestabilan pada jaringan (Ioffe & Szegedy, 2015). Berikut persamaan untuk *batch normalization*.

$$\mu_B^{(k)^2} = \frac{1}{m}\sum_{i=1}^{m} x_i^{(k)} \tag{2.7}$$

$$\sigma_B^{(k)^2} = \frac{1}{m}\sum_{i=1}^{m} (x_i^{(k)} - \mu_B^{(k)})^2 \tag{2.8}$$

$$\widehat{x}_i^{(k)} = \frac{x_i^{(k)} - \mu_B^{(k)}}{\sqrt{\sigma_B^{(k)^2} + \varepsilon}} \tag{2.9}$$

$$y_i^{(k)} = \gamma^{(k)}\widehat{x}_i^{(k)} + \beta^{(k)} = BN_{\gamma^{(k)},\beta^{(k)}}(x_i^{(k)}) \tag{2.10}$$

Keterangan:

$x_i^{(k)}$: *input* ke-i

μ_B : mean *mini batch*

σ_B^2 : varians *mini batch*

ε : *numerical stability* (konstanta bernilai 1e-5)

$\hat{x}_i^{(k)}$: *normalize* x_i

$y_i^{(k)}$: *output batch normalization*

dimana $k \in [1, d]$ dan $i \in [1, m]$.

Berikut merupakan diferensial untuk meminimumkan nilai *loss*.

$$\frac{\partial l}{\partial y_i^{(k)}}$$

$$\frac{\partial l}{\partial \hat{x}_i^{(k)}} = \frac{\partial l}{\partial y_i^{(k)}} \gamma^{(k)}$$

$$\frac{\partial l}{\partial \gamma^{(k)}} = \sum_{i=1}^{m} \frac{\partial l}{\partial y_i^{(k)}} \hat{x}_i^{(k)}, \quad \frac{\partial l}{\partial \beta^{(k)}} = \sum_{i=1}^{m} \frac{\partial l}{\partial y_i^{(k)}} \tag{2.11}$$

$$\frac{\partial l}{\partial \sigma_B^{(k)^2}} = \sum_{i=1}^{m} \frac{\partial l}{\partial y_i^{(k)}} \left(x_i^{(k)} - \mu_B^{(k)} \right) \left(-\frac{\gamma^{(k)}}{2} \left(\sigma_B^{(k)^2} + \varepsilon \right)^{-3/2} \right)$$

$$\frac{\partial l}{\partial \mu_B^{(k)}} = \sum_{i=1}^{m} \frac{\partial l}{\partial y_i^{(k)}} \frac{-\gamma}{\sqrt{\sigma_B^{(k)^2} + \varepsilon}} + \frac{\partial l}{\partial \sigma_B^{(k)^2}} \frac{1}{m} \sum_{i=1}^{m} (-2) \left(x_i^{(k)} - \mu_B^{(k)} \right)$$

2.5.7 *Hyperparameter*

Pada proses pembangunan model CNN, akan diterima sejumlah *hyperparameter* yang mendefinisikan jaringan CNN yang digunakan, lalu akan dibangun sebuah model CNN awal sesuai *input*. Karena dalam *machine learning* terdapat banyak sekali parameter dalam pembuatan model, maka dikenal sebuah istilah yaitu *hyperparameter*. *Hyperparameter* didefinisikan sebagai parameter dari sebuah distribusi di luar distribusi pada model. Dalam konteks model klasifikasi, distribusi model adalah distribusi dari permasalahan klasifikasi yang sedang dipelajari oleh model. *Hyperparameter* perlu diatur sedemikian rupa dan tepat untuk menghasilkan model dengan performa terbaik (Bergstra, Bardenet, Bengio, & Kegl, 2011).

a. Epoch, Batch Size, dan Iterations

Epoch adalah pengulangan yang terjadi pada satu siklus proses pelatihan di dalam *neural network* dalam memperbaiki *error*. Pengulangan ini akan terus berlangsung hingga toleransi *error* pada saat *training* atau nilai *epoch* yang ditetapkan telah tercapai. *Batch Size* adalah jumlah sampel data yang disebarkan ke *Neural Network*. Contohnya yaitu jika kita mempunyai 100 dataset dan *batch size* kita adalah 5 maka algoritma ini akan menggunakan 5 sampel data pertama dari 100 data yang kita miliki (ke-1, ke-2, ke-3, ke-4, dan ke-5) lalu disebarkankan atau di*training* oleh *Neural Network* sampai selesai kemudian mengambil kembali 5 sampel data kedua dari 100 data (ke-6, ke-7, ke-8, ke-9, dan ke-10), dan begitu seterusnya sampai 5 sampel data ke 20 (100⁄5=20). *Iterations* adalah jumlah *batch* yang diperlukan untuk menyelesaikan satu *epoch*. Jumlah *batch* sama dengan jumlah iterasi untuk satu *epoch*.

b. Learning Rate

Learning rate (α) merupakan parameter pembelajaran didalam *neural network* yang digunakan untuk mempercepat proses *training*. Nilai parameter yang digunakan berkisar antara 0 sampai 1. Semakin *learning rate* mendekati nilai 1 kecepatan pembelajaran yang digunakan akan menyebabkan perubahan yang besar dan pengaruh terhadap pembelajaran menjadi kurang baik. Sedangkan dengan tingkat kecepatan pembelajaran yang rendah akan menghasilkan pembelajaran yang akurat namun proses akan berjalan lebih lambat.

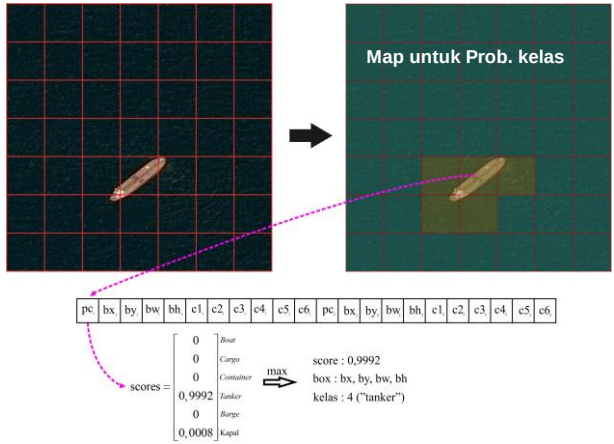
c. Data Augmentasi

Augmentasi citra merupakan teknik pengolahan citra yang digunakan untuk menambahkan data *training* sehingga dapat mengurangi *overfitting* (Wang & Perez, 2017). Beberapa *hyper-parameter* untuk melakukan data augmentasi yang terdapat dalam cfg YOLOv3 yaitu diantaranya *angle* (memutar gambar secara acak selama pelatihan), *saturation* (secara acak mengubah saturasi gambar selama pelatihan), *exposure* (secara acak mengubah *esposure*/kecerahan selama pelatihan).

2.5.8 Deteksi Objek YOLO

Klasifikasi secara umum adalah proses untuk mengidentifikasi label dari data yang diuji, pada klasifikasi YOLO dilakukan dengan localization, yaitu terdapat tambahan pemberian lokasi objek dalam bentuk *bounding box* (*bx, by, bh, bw*). Tahapan deteksi objek YOLO terdapat pada langkah-langkah berikut.

a. Membaca data citra dengan ukuran sebarang.
b. Mengubah ukuran citra menjadi 448×448, lalu membuat *grid* pada citra dengan ukuran *S*×*S grids*.
c. Melakukan pemetaan deteksi objek pada setiap *grid cell* dengan *fully connected layer* dan fungsi *softmax*.

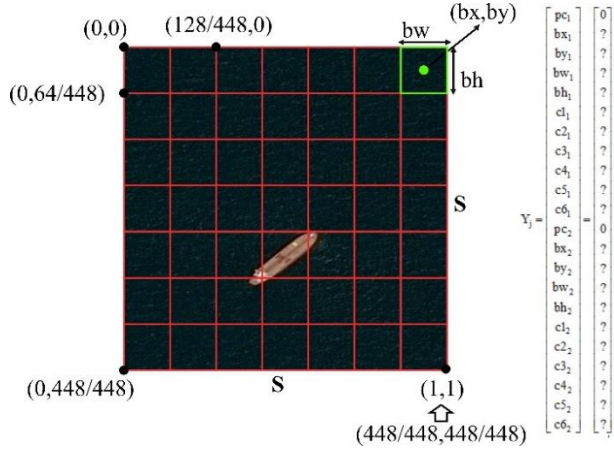


Gambar 2. 11 Contoh Gambar dalam Algoritma YOLO-CNN

Berdasarkan Gambar 2.11, sebagai contoh dari output YOLO diperoleh *grid S*=7, maka tiap *grid cell* ukurannya 64×64 diperoleh dari 448×448 dibagi 7. Total terdapat sebanyak 49 *grid cell*. Dalam penelitian digunakan 6 kelas (*nC*=6), yaitu "*Boat*" (c1), "*Cargo*" (c2), "*Container"* (c3), "*Tanker"* (c4), "*Barge*" (c5), dan "Kapal" (c6). Dalam setiap *grid cell* dilakukan *fully connected* dan fungai *softmax* untuk diperoleh *scores* deteksi objek, sehingga diperoleh *map* untuk probabilitas kelas untuk masing-masing *grid cell*.

Sebagai contoh berdasarkan Gambar 2.11 pada *grid cell* ke 32 diperoleh nilai pc untuk kelas "*Tanker*" (c4) 0,9992 dan untuk kelas "Kapal" (c5) 0,0008, diperoleh probabilitas maksimal yaitu kelas "*Tanker*" (c4) maka *map* untuk *grid cell* ke 32 lebih dominan untuk kelas "*Tanker*" (c4).

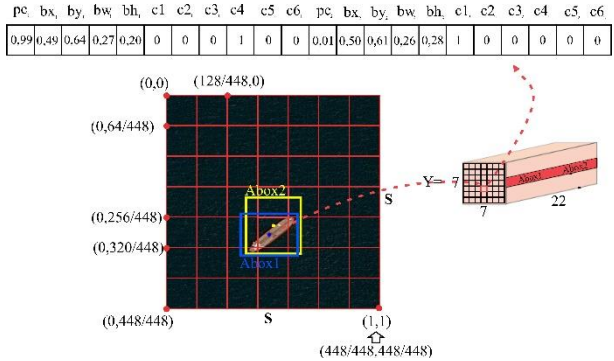
d. Melakukan deteksi objek pada masing-masing *grid cell*, masing-masing *grid cell* yang terdapat objek akan bertanggung jawab untuk melakukan deteksi. Pada ilustrasi ini digunakan *anchor box* atau banyaknya *bounding box* yang digunakan untuk deteksi pada masing-masing *grid cell* dimisalkan digunakan 2 *anchor box*.



Gambar 2. 12 Penjelasan Algoritma YOLO-CNN

Gambar 2.12 menunjukkan koordinat x,y sudah dinormalisasi dengan membagi maksimal *pixel* yaitu 448, sehingga nilai berselang antara 0 hingga 1. Ketika dalam *grid cell* tidak ditemukan objek kelas hanya terdapat *background* maka nilai *confidence score* bernilai 0 dan elemen lainnya akan tidak memiliki nilai. Pada masing-masing *grid cell* terdapat *bounding box* yang berisi 11 nilai, yaitu nilai *confidence*, lokasi koordinat *x*, koordinat *y*, ukuran *bounding box*, dan 6 kelas yang digunakan (*pc, bx, by, bw, bh, c1, c2, c3 c4, c5, c6*)

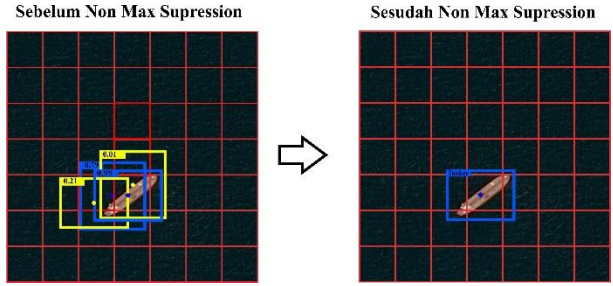
sebanyak 2 kali dalam setiap *grid cell* karena digunakan sebanyak 2 *anchor box*. Selanjutnya jika terdapat objek dalam *grid* cell maka akan dilakukan *bounding box* untuk memberi tanda akan adanya objek yang telah ditentukan sebelumnya.



Gambar 2. 13 Penjelasan Algoritma YOLO-CNN

Pada Gambar 2.13 setiap *grid cell* dilakukan deteksi objek sehingga diperoleh dimensi 7x7x22 untuk satu citra. Sebagai contoh pada *grid cell* ke 32 ditemukan objek dengan kelas "*Tanker*" dan "Kapal" dengan masing-masing probabilitas, maka selanjutnya diberikan *bounding box* sehingga diperoleh spesifikasi koordinat x,y serta ukuran *bounding box*. Nilai pada *grid cell* ke 32 yang menandai objek yang ditemukan terdiri dari pc_1, bx_1, by_1, bw_1, bh_1, $c1_1$, $c2_1$, $c3_1$, $c4_1$, $c5_1$, $c6_1$, pc_2, bx_2, by_2, bw_2, bh_2, $c1_2$, $c2_2$, $c3_2$, $c4_2$, $c5_2$, $c6_2$.

e. Melakukan *non max supression* untuk mendapatkan *bounding box* dengan *confidence score* yang maksimal. Selanjutnya setelah pada semua *grid cell* dilakukan proses deteksi objek, maka diperoleh beberapa *bounding box* yang menandai akan adanya objek. Diantara *bounding box* tersebut akan dipilih yang menandai akan adanya objek dengan probabilitas atau *confidence score* yang tertinggi, sedangkan *bounding box* dengan objek yang sama yang memiliki *confidence score* yang lebih rendah akan dihilangkan.



Gambar 2. 14 Penjelasan Algoritma YOLO-CNN

Gambar 2.14 merupakan contoh ilustrasi *non max supression* objek dengan *confidence score* tertinggi diperoleh pada titik koordinat yang jatuh pada *grid cell* ke 32 untuk kelas "*Tanker*" dengan *confidence score* tertinggi 0,99 dan "Kapal" 0,01, sedangkan pada *grid cell* ke 31 ditemukan kelas dengan probabilitas tertinggi yaitu kelas "*Tanker*" 0,79 dan "Kapal" 0,21. Sehingga diperoleh hasil *output* deteksi pada citra dengan kelas "*Tanker*" yang memiliki *confidence score* 99%.

2.5.9 *Loss Function*

Loss adalah nilai dari error antara nilai hasil prediksi dan nilai sebenarnya. Pada *machine learning* digunakan *loss function* digunakan untuk menghitung nilai dari error (Zhang, Lipton, Li, & Smola, 2020). *Loss functions* pada jaringan YOLO didapatkan dari hasil penjumlahan beberapa perhitungan *partial loss functions*. Dengan menghitung kesalahan koordinat, kesalahan klasifikasi, dan kesalahan IoU lalu menjumlahkannya, *total loss function* dapat dinyatakan sebagai berikut,

$$loss = \sum_{i=0}^{s^2} coordErr + clsErr + IoUErr \tag{2.12}$$

Nilai *loss* untuk prediksi koordinat pusat dinyatakan sebagai berikut,

$$\lambda_{coord} \sum_{i=0}^{s^2} \sum_{j=0}^{B} \ell_{ij}^{obj} \left[\left(x_i - x_i \Lambda \right)^2 + \left(y_i - y_i \Lambda \right)^2 \right] \tag{2.13}$$

Nilai *loss* untuk prediksi lebar dan tinggi *bounding box* dinyatakan sebagai berikut,

$$\lambda_{coord} \sum_{i=0}^{s^2} \sum_{j=0}^{B} \ell_{ij}^{obj} \left[\left(\sqrt{w_i} - \sqrt{w_i^{\Lambda}} \right)^2 + \left(\sqrt{h_i} - \sqrt{h_i^{\Lambda}} \right)^2 \right] \quad (2.14)$$

Di mana λ_{coord} menunjukkan faktor pembobot eror pada koordinat dalam *overall loss function* / fungsi keseluruhan *loss*. *Loss* untuk perkiraan kategori dinyatakan sebagai berikut,

$$\sum_{i=0}^{s^2} \ell_{ij}^{obj} \sum_{j=0}^{B} \left[\left(p_i(c) - p_i^{\Lambda}(c) \right)^2 \right] \quad (2.15)$$

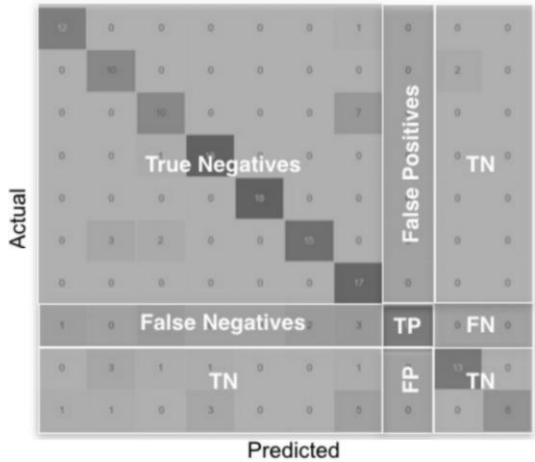
Loss untuk nilai kepercayaan / *confidence* pada prediksi dinyatakan sebagai berikut:

$$\sum_{i=0}^{s^2} \sum_{j=0}^{B} \ell_{ij}^{obj} \left[\left(c_i - c_i^{\Lambda} \right)^2 \right] + \lambda_{noobj} \sum_{i=0}^{s^2} \sum_{j=0}^{B} \ell_{ij}^{obj} \left[\left(c_i - c_i^{\Lambda} \right)^2 \right] \quad (2.16)$$

dimana C adalah skor kepercayaan / *confidence score*; $\hat{C}$ adalah persimpangan antara *bounding box* yang diprediksi dan data real, ketika sebuah objek ada di dalam sel; maka ℓ_{ij}^{obj} sama dengan 1; jika tidak, itu maka bernilai 0; λ_{noobj} mewakili bobot kepercayaan saat tidak ada objek di dalam *bounding box*.

2.6 Evaluasi Performa

Nilai evaluasi pertama yang digunakan pada penelitian ini adalah *confusion matrix*. *Confusion matrix* dapat untuk mengetahui hasil kebaikan klasifikasi pada model. Kebaikan klasifikasi dapat dihitung menggunakan nilai *precision* dan *recall* pada masing-masing kelas pada setiap model. Pada permasalahan klasifikasi, menurut kombinasi kelas *ground truth* dan kelas prediktif, hasil dapat dibagi menjadi empat jenis: *true positive* (TP), *false positive* (FP), *true negative* (TN), dan *false negatif* (FN). Penentuan empat jenis *confusion matrix* untuk multi kelas ditunjukkan pada Gambar 2.15, di mana baris mewakili *ground truth* sedangkan kolom mewakili label yang diprediksi.



Gambar 2. 15 *Confusion Matrix*

Berdasarkan Gambar 2.15, *True Positive* (TP) mengacu pada jumlah prediksi di mana pengklasifikasi kelas positif pada *ground truth* sebagai prediksi positif dengan benar. *False Positive* (FP) mengacu pada jumlah prediksi di mana klasifikasi salah memprediksi kelas negatif pada *ground truth* sebagai prediksi positif, *True Negative* (TN) mengacu pada jumlah prediksi di mana pengklasifikasi kelas negatif pada *ground truth* dengan benar sebagai prediksi negatif, dan *False Negatif* (FN) mengacu pada jumlah prediksi di mana pengklasifikasi salah memprediksi kelas positif pada *ground truth* sebagai prediksi negatif. Dari hasil *confusion matrix* dapat dihitung *precision* dan *recall* dengan persamaan sebagai berikut.

$$\mathrm{Pr}\,ecision = \frac{TP}{TP + FP} = \frac{TP}{\text{Semua Prediksi}} \tag{2.17}$$

$$Re\,call = \frac{TP}{TP + FN} = \frac{TP}{\text{Semua}\,Ground\ Truth} \tag{2.18}$$

Precision merepresentasikan kemampuan model untuk mengidentifikasi objek terkait sebagai nilai persentase prediksi

yang benar. *Recall* adalah kemampuan model untuk menemukan semua objek relevan sebagai nilai persentase positif benar yang dapat terdeteksi di semua *ground truth*. Setelah didapatkan *precision* dan *recall* dapat diperoleh suatu kurva yang disebut sebagai *P–R curve*, dengan *recall* sebagai sumbu horizontal dan *precision* sebagai sumbu vertikal.

Nilai evaluasi kedua yang digunakan pada penelitian ini yaitu mAP. Nilai mAP merupakan nilai evaluasi yang paling populer pada tugas deteksi objek yang dapat digunakan untuk membandingkan kebaikan sebuah model dengan model lainnya. Nilai *mean average precision* (mAP) adalah nilai rata-rata dari *average precision*. *Average precision* didapatkan dari setiap nilai *precision* item relevan yang dihasilkan dan menggunakan nilai 0 untuk item relevan yang tidak dihasilkan oleh sistem. Nilai *precision* untuk *average precision* dihitung dengan memperhatikan urutan item yang diberikan oleh sistem, sehingga nilai *precision* diberikan untuk setiap item yang dihasilkan oleh sistem. Persamaan berikut merupakan persamaan untuk menghitung nilai *mean average precision* (Manning, Raghavan, & Schutze, 2009).

$$mAP = \frac{1}{c}\sum_{i=1}^{c}\frac{1}{m_i}\sum_{k=1}^{m_i} p\left(R_{ik}\right) \tag{2.19}$$

Cara lain untuk mendapatkan nilai mAP menggunakan *Precision Recall Curve*, yaitu sebagai berikut.

$$mAP = \frac{1}{c}\sum_{i=1}^{c} AP_i \tag{2.20}$$

Mean average precision (mAP) adalah rata-rata nilai AP di semua kelas. Perhitungan nilai *average precision* (AP) dapat dilakukan dengan 2 metode interpolasi yaitu 11-*point interpolation* dan *all-point interpolation*. Berikut merupakan persamaan dari metode 11-*point interpolation*.

$$AP = \frac{1}{11}\sum_{r\in\{0;0,1;\ldots;1\}} p_{\text{int}erp}(r) \tag{2.21}$$

$$p_{interp} = \max_{\tilde{r}:\tilde{r} \geq r_n} p(\tilde{r}) \tag{2.22}$$

Interpolasi 11 titik meringkas bentuk *precision recall curve* dengan rata-rata presisi pada sebelas tingkat *recall* yang berjarak sama yaitu 0; 0,1; 0,2, ..., 1. Nilai presisi interpolasi didapatkan dengan mengambil presisi maksimum yang nilai *recall*nya lebih besar dari nilai *recall* saat ini. Untuk metode yang kedua yaitu *all-point interpolation* disajikan pada persamaan berikut.

$$AP = \sum_{n=0} (r_{n+1} - r_n) p_{interp}(r_{n+1}) \tag{2.23}$$

$$p_{interp}(r_{n+1}) = \max_{\tilde{r}:\tilde{r} \geq r_n+1} p(\tilde{r}) \tag{2.24}$$

Pada metode ini, AP diperoleh dengan menginterpolasi presisi di setiap titik, mengambil presisi maksimum yang nilai *recall*nya lebih besar atau sama dengan r_{n+1}.

Keterangan :

c : Jumlah kelas

R : item relevan yang dihasilkan oleh sistem

m : jumlah item relevan yang dihasilkan pada suatu kelas

p : presisi

p_{interp} : interpolasi presisi

r : *recall*

$p(\tilde{r})$: perhitungan presisi pada *recall* $\tilde{r}$

Perhitungan *average precision* (AP) dalam deteksi objek dapat diperoleh dengan mendapatkan nilai IoU terlebih dahulu. IoU didapatkan dari persamaan 2.3 pada subbab 2.5.3. IoU akan digunakan untuk menentukan apakah kotak pembatas yang diprediksi adalah *True Prositive* (TP), *False Positive* (FP) atau *False Negative* (FN). *True Negative* (TN) tidak dievaluasi karena setiap gambar diasumsikan memiliki objek di dalamnya. Prediksi didefinisikan sebagai TP jika IoU $\geq$ 0,5. Ada 2 skenario dalam mendefinisikan FP, pertama apabila klasifikasi kelas benar namun IoU < 0,5 dan kedua apabila terdapat duplikat *bounding box* prediksi. Selanjutnya ada 2 skenario dalam mendefinisikan FN,

yaitu, pertama apabila IoU ≥ 0,5 namun salah klasifikasi dan kedua apabila tidak dideteksi objek yang seharusnya terdapat *ground truth* objek. Hasil deteksi diurutkan sesuai *confidence* yang tertinggi pada perhitungan *Avearage Precision* (AP).

2.7 Kapal

Kapal adalah kendaraan air dengan bentuk jenis apapun, yang digerakkan dengan tenaga mekanik, tenaga angin atau ditunda, termasuk kendaraan yang berdaya apung dinamis, kendaraan di permukaan air, serta alat apung dan bangunan terapung yang tidak berpindah-pindah (Kementrian Luar Negeri, 2008). Berikut beberapa tipe kapal yang digunakan dalam penelitian ini :

a. Kapal motor (*Boat*)

Gambar 2. 16 Kapal Motor *(Boat)*

Gambar 2.16 menunjukkan sebuah kapal cepat yang didesain dengan kebutuhan tertentu untuk transportasi air. Kapal motor (*boat*) yang memiliki mesin tempel dipasang di bagian belakang, memuat mesin pembakaran dalam, kotak gigi dan baling-baling dalam sebuah unit portabel. Mesin dalam/tempel memuat cangkokan pembangkit listrik dan tempel, dan mesin pembakaran dalaman dipasang di dalam perahu, sedangkan kotak gigi dan baling-baling di luar. Kapal motor memiliki beraneka macam ukuran dan konfigurasi, di ruang kemudi tersebut biasanya juga dilengkapi dengan alat komunikasi termasuk juga GPS dan pendeteksi kedalaman air tergantung faktor kebutuhan dan besar

kecilnya sebuah kapal, dari 4 meter jenis konsol terbuka hingga *megayacht* mewah yang sanggup menyeberangi samudera.

b. Kapal Barang/Kargo (*Cargo)*

Gambar 2. 17 Kapal Kargo

Segala jenis kapal yang membawa barang-barang dan kargo dari suatu pelabuhan ke palabuhan lainnya disebut kapal kargo seperti yang ditunjukkan pada Gambar 2.17. Ribuan kapal jenis ini menyusuri laut dan samudera dunia setiap tahunnya memuat barang-barang perdagangan internasional dan nasional. Kapal kargo pada umumnya di desain khusus untuk tugasnya.

c. Kapal Peti Kemas (*Countainer*)

Gambar 2. 18 Kapal Peti Kemas

Gambar 2.18 menunjukkan kapal peti kemas dimana kapal ini khusus digunakan untuk mengangkut peti kemas. Selanjutnya PP 51 tahun 2002 tentang perkapalan, yang dimaksud dengan peti kemas adalah bagian dari alat yang berbentuk kotak serta terbuat dari bahan yang memenuhi syarat bersifat permanen dan dapat di pakai berulang-ulang, yang memiliki pasangan sudut serta dirancang khusus untuk memudahkan angkutan barang dengan satu

atau lebih noda transportasi, tanpa harus dilakukan pemuatan kembali. Termasuk jenis ini adalah kapal semi peti kemas, yaitu perpaduan antara kapal kargo dan peti kemas.

d. Kapal Pengangkut Minyak (*Tanker*)

Gambar 2. 19 Kapal *Tanker*

Kapal yang dirancang untuk mengangkut minyak atau produk turunannya dinamakan kapal *tanker* yang ditunjukkan pada Gambar 2.19. Jenis utama kapal *tanker* termasuk mengangkut minyak, LNG, LPG. Diantara berbagi jenis kapal tenker menurut kapasitas : ULCC (*Ultra large Crude Carrier*) berkapasitas 500.000 Ton dan VLCC (*Very Large Crude Carrier*) berkapasitas 300.000 Ton.

e. Kapal Tongkang/Ponton (*Barge*)

Gambar 2. 20 Kapal Tongkang/Ponton (*Barge*)

Kapal Tongkang/Ponton merupakan suatu kapal dengan lambung datar atau suatu kotak besar yang mengapung, digunakan untuk mengangkut barang dan ditarik dengan pasang-surut seperti pada ikan yang ditunjukkan oleh Gambar 2.20.

BAB III
METODOLOGI PENELITIAN

3.1 Sumber Data

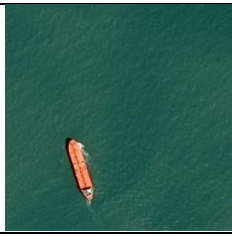
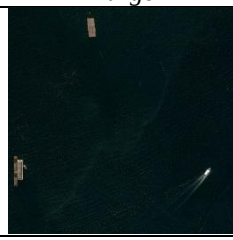
Data yang digunakan dalam penelitian ini merupakan data sekunder gambar citra satelit (*remote sensing*) dari *Website OneAtlas Sandbox* dengan nama dataset "*Ships Detection Machine Learning Dataset* yang diekstrak dari SPOT *satellite imagery* dengan resolusi mencapai 1,5 meter. Data memiliki ukuran dimensi 768×768 *pixel* dengan jumlah lebih dari dua ratus ribu file, terbagi menjadi 192.556 file gambar pada data *training* dan 15.606 file gambar pada data *testing*. Dalam dataset ini terdapat berbagai macam tipe kapal mulai kapal tanker, kapal komersial, maupun kapal nelayan dengan berbagai macam bentuk dan ukuran. Namun dalam penelitian ini akan dilakukan deteksi jenis kapal dengan 5 tipe kapal yaitu *boat*, *cargo*, *container*, *tanker*, dan *barge*, lalu untuk jenis kapal yang tidak masuk kedalam 5 tipe kapal tersebut akan dideteksi sebagai kapal seperti contoh data pada Tabel 3.1.

Pada penelitian ini yang digunakan sebagai data penelitian adalah data *testing* yang terdapat pada dataset "*Ships Detection Machine Learning Dataset*" yaitu sebanyak 15.606 file gambar yang diambil pada tanggal 20 Januari 2020 pukul 15.20, selanjutnya dari sejumlah file tersebut akan dilakukan anotasi atau pelabelan tipe kapal, sehingga hanya menyisakan 1.465 gambar kapal, sedangkan gambar yang tidak terdapat objek kapal tidak digunakan untuk penelitian.

Tabel 3. 1 Contoh Data Tipe Kapal Penelitian

Boat	*Cargo*	*Container*
		

Tabel 3. 1 Contoh Data Tipe Kapal Penelitian (Lanjutan)

Tanker	*Barge*	Kapal
		

Berdasarkan contoh data tipe kapal pada Tabel 3.1, pada kelas pertama yaitu "*boat*" secara visual memiliki ciri-ciri dimana terdapat kecenderungan adanya warna putih panjang yang melintang pada citra. Pada kelas kedua yaitu "*cargo*" secara visual memiliki ciri-ciri dimana terdapat kecenderungan adanya kotak persegi atau persegi panjang yang berjajar pada kapal dimana itu merupakan alat untuk membuka dan menutup penyimpanan barang. Pada kelas ketiga yaitu "*container*" secara visual memiliki ciri-ciri dimana terdapat kecenderungan adanya kotak yang memiliki warna berbeda-beda dimana itu merupakan peti kemas yang diangkut diatas kapal. Pada kelas keempat yaitu "*tanker*" secara visual pada citra satelit memiliki ciri-ciri dimana terdapat kecenderungan adanya garis yang memanjang dari depan kapal hingga belakang kapal dimana itu menunjukkan adanya pipa minyak pada kapal. Pada kelas kelima yaitu "*barge*" secara visual memiliki ciri-ciri dimana terdapat kecenderungan kapal berbentuk kotak persegi panjang, dimana kapal ini digunakan sebagai kapal pengangkut barang-barang seperti batu bara, kayu, dll. Pada kelas keenam yaitu kapal yang secara visual memiliki ciri-ciri diluar kelima kelas yang ada sebelumnya, serta kapal yang terlihat kurang jelas untuk masuk kelima tipe kapal sebelumnya akan diklasifikasi-kan sebagai kelas keenam yaitu "kapal".

3.2 Variabel Penelitian

Variabel penelitian yang digunakan pada penelitian ini terdapat pada Tabel 3.2.

Tabel 3. 2 Variabel Penelitian

Variabel	Keterangan
$X_{1,i,j}$	Nilai *Red* setiap *pixel*
$X_{2,i,j}$	Nilai *Green* setiap *pixel*
$X_{3,i,j}$	Nilai *Blue* setiap *pixel*
$\mathbf{Y}_j$	Matriks *Ground Truth Bounding Box*

dengan,
$0 < i \leq$ K ; K bernilai 589.824 yang merupakan total *pixel* (768×768 = 589.824) dari setiap gambar.
$0 < j \leq$ L; L bernilai sejumlah citra satelit yang digunakan.
$\mathbf{Y_j}$ merupakan matriks *ground truth bounding box* yang merupakan hasil dari anotasi atau pelabelan citra yang dilakukan dengan menggunakan bantuan *software* labelImg, selanjutnya akan diperoleh matriks dengan elemen diantaranya kategori kelas (c), nilai koordinat (bx, by), serta lebar (bw) dan tinggi (bh) pada citra dengan persamaan sebagai berikut.

$$\mathbf{Y_j} = \begin{cases} \begin{bmatrix} c_1 & bx_1 & by_1 & bw_1 & bh_1 \end{bmatrix} & \text{, jika } a=1 \\ \begin{bmatrix} c_1 & bx_1 & by_1 & bw_1 & bh_1 \\ c_2 & bx_2 & by_2 & bw_2 & bh_2 \end{bmatrix} & \text{, jika } a=2 \\ \vdots & \\ \begin{bmatrix} c_1 & bx_1 & by_1 & bw_1 & bh_1 \\ c_2 & bx_2 & by_2 & bw_2 & bh_2 \\ & & \vdots & & \\ c_n & bx_n & by_n & bw_n & bh_n \\ & & \vdots & & \\ c_b & bx_b & by_b & bw_b & bh_b \end{bmatrix} & \text{, jika } a=b \end{cases} \tag{3.1}$$

Keterangan :
$0 < n \leq b$; b bernilai sejumlah *ground truth bounding box* pada citra satelit.

c_n : Tipe kapal pada *ground truth bounding box* ke-n; dimana 1 = *Boat*, 2 = *Cargo*, 3 = *Container*, 4 = *Tanker*, 5 = *Barge*, 6 = Kapal.

bx_n : Titik pusat koordinat x *ground truth bounding box* ke-n.

by_n : Titik pusat koordinat y *ground truth bounding box* ke-n.

bh_n : Tinggi *ground truth bounding box* ke-n.

bw_n : Lebar *ground truth bounding box* ke-n.

3.3 Struktur Data

Struktur data secara umum yang digunakan dalam penelitian ini terdapat pada Tabel 3.3 yang berupa nilai rgb pada setiap *pixel*.

Tabel 3. 3 Struktur Data Penelitian

No	$X_{1,1}$	⋯	$X_{1,589824}$	⋯	$X_{2,589824}$	⋯	$X_{3,589824}$	Y
1	$X_{1,1,1}$	⋯	$X_{1,589824,1}$	⋯	$X_{2,589824,1}$	⋯	$X_{3,589824,1}$	Y_1
2	$X_{1,1,2}$	⋯	$X_{1,589824,2}$	⋯	$X_{2,589824,2}$	⋯	$X_{3,589824,2}$	Y_2
⋮	⋮		⋮		⋮		⋮	⋮
L	$X_{1,1,l}$	⋯	$X_{1,589824,l}$	⋯	$X_{2,589824,l}$	⋯	$X_{3,589824,l}$	Y_l

Tabel 3.3 menunjukkan bahwa dalam setiap citra terdapat nilai *red* setiap *pixel* ($X_{1,i,j}$), nilai *green* setiap *pixel* ($X_{2,i,j}$), nilai *blue* setiap *pixel* ($X_{3,i,j}$), dan nilai $\mathbf{Y_j}$. Sebagai contoh $X_{1,1,1}$ merupakan nilai red pada *pixel* ke-1 citra ke-1. Contoh yang lain, $X_{3,589824,1}$ merupakan nilai blue pada *pixel* ke-589824 citra ke-1. Selanjutnya nilai $\mathbf{Y_j}$ merupakan matriks *ground truth bounding box*.

3.4 Langkah Analisis

Langkah analisis digunakan untuk menggambarkan langkah-langkah penelitian yang dilakukan secara urut. Langkah analisis yang digunakan adalah sebagai berikut.

1. Merumuskan masalah dan studi *literature*.
2. Mengumpulkan data citra kapal.
   a. Gambar terdiri dari panjang dan lebar yang diwakilkan oleh kotak-kotak *pixel* yang tersusun mengikuti ukuran gambar.
   b. Setiap *pixel* mewakilkan satu warna.
   c. Warna terbentuk dari kombinasi tiga komponen warna utama yaitu R (*Red*), G (*Green*), dan B (*Blue*). Setiap

komponen warna utama memiliki tingkat gradasi yang memiliki nilai antara 0-255, dengan semakin kecil nilai maka warna yang dihasilkan akan semakin gelap dan semakin besar nilai maka warna yang dihasilkan akan semakin terang.

d. Sehingga data yang didapatkan seperti struktur data pada Tabel 3.3 yang menunjukkan bahwa dalam setiap citra terdapat nilai *red* setiap *pixel* ($X_{1,i,j}$), nilai *green* setiap *pixel* ($X_{2,i,j}$), nilai *blue* setiap *pixel* ($X_{3,i,j}$), namun secara realita pada *training* dan *testing* tidak perlu mengubah citra dalam bentuk RGB numerik untuk dapat memproses suatu citra pada YOLO-CNN, jadi citra dapat diproses secara langsung dalam bentuk gambar dengan data yang terlampir pada Lampiran 1.

3. Melakukan *pre-processing* data sebagai berikut.
   a. Menyeleksi gambar untuk data penelitian dengan hanya menggunakan gambar yang terdapat objek kapal dan didapatkan 1.465 gambar yang terdapat objek kapal.
   b. Melakukan anotasi pelabelan tipe kapal pada citra gambar dengan 6 tipe kapal yang digunakan yaitu *boat*, *cargo*, *container*, *tanker*, *barge* dan kapal menggunakan bantuan *software labelImg*.
   c. Sehingga didapatkan data seperti struktur data nilai $\mathbf{Y_j}$ pada Tabel 3.3, data terlampir pada Lampiran 2 sebagai *ground truth bounding box*.
4. Membagi data latih dan uji.
5. Melakukan eksplorasi data dengan menggunakan sintaks pada Lampiran 4, dengan terlebih dahulu mengubah format data *ground truth* voc xml menjadi fomat csv dengan menggunakan sintaks pada Lampiran 7.
6. Melakukan proses pelatihan deteksi tipe kapal dengan metode YOLO-CNN.
   a. Mendapatkan nilai *anchor box* dengan metode *K-means* dengan jumlah *cluster anchor* sebanyak 9, dengan langkah-langkah sebagai berikut.

- Mengubah format *ground truth* yolo txt menjadi format pascal voc xml dengan menggunakan sintaks pada Lampiran 5.
- Mengubah format *ground truth* pascal voc xml menjadi format coco json dengan menggunakan sintaks pada Lampiran 6.
- Mendapatkan nilai *anchor box* baru dengan menggunakan *K-means* dengan menggunakan sintaks pada Lampiran 8.

b. Menentukan *hyperparameter* dalam proses pelatihan sesuai pada Lampiran 12 untuk Model 1 dan Lampiran 13 untuk Model 2.

c. Melakukan proses pelatihan dengan menggunakan sintaks pada Lampiran 9.

d. Mendapatkan nilai *loss* untuk setiap model.

7. Memperoleh nilai evaluasi performa untuk kedua model pada data *training* dan *testing* sebagai berikut.

a. Memperoleh *confusion matrix* untuk mengetahui nilai *precision* dan *recall* untuk kedua model dengan menggunakan sintaks pada Lampiran 14.

b. Menghitung nilai mAP pada data untuk kedua model menggunakan sintaks pada Lampiran 11, dengan terlebih dahulu mengubah format data dari *output* yolo txt menjadi partisi beberapa file txt menggunakan sintaks pada Lampiran 10.

c. Mendapatkan grafik mAP dibandingkan dengan *threshold*.

d. Mendapatkan grafik *precision recall curve*.

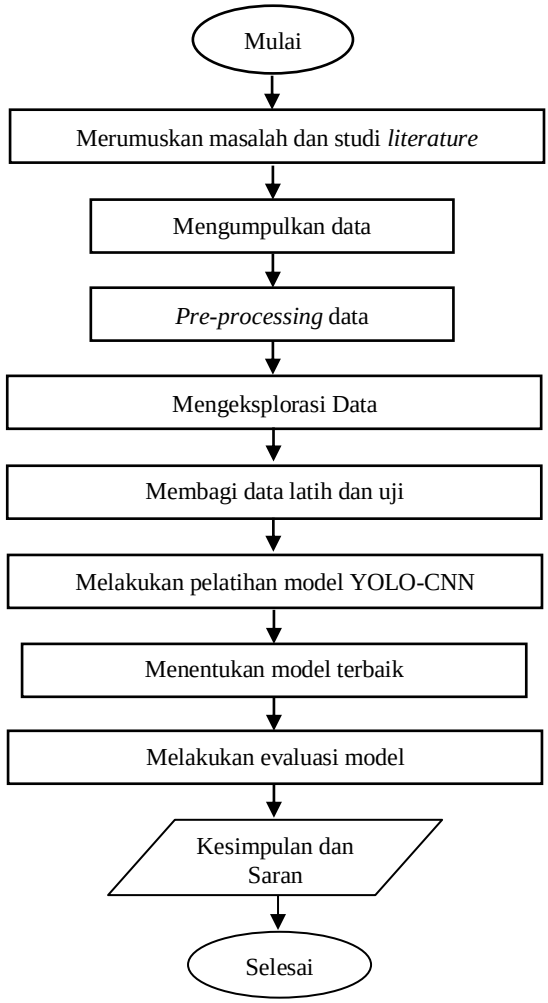
8. Menentukan model terbaik berdasarkan nilai evaluasi performa pada setiap model.

9. Menarik kesimpulan dan saran.

3.5 Diagram Alir

Diagram alir menggambarkan alur perjalanan pembuatan laporan ini, mulai dari proses perumusan masalah dan *studi*

literature, pengumpulan data hingga penarikan kesimpulan dan saran. Diagram alir dari langkah analisis pada penelitian ini diberikan dalam Gambar 3.1.



Gambar 3. 1 Diagram Alir

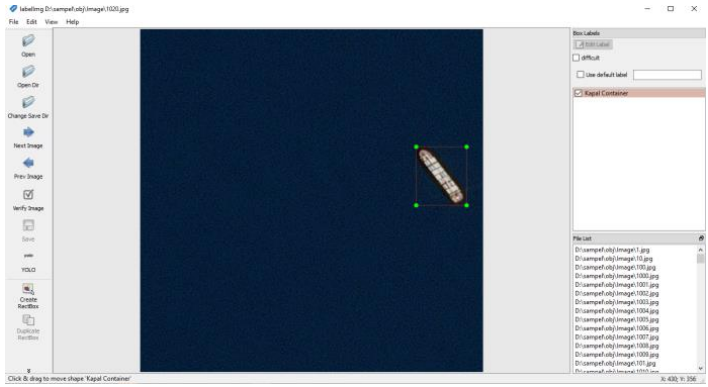
(Halaman ini sengaja dikosongkan)

BAB IV
ANALISIS DAN PEMBAHASAN

Pada penelitian ini dilakukan klasifikasi dan lokalisasi tipe kapal berdasarkan nilai rgb setiap *pixel*. Penelitian ini menggunakan metode *You Only Look Once Convolutional Neural Network* (YOLO-CNN) dengan arsitektur YOLOv3. Kebaikan hasil klasifikasi dan lokalisasi tersebut didapatkan dari hasil nilai mAP dan *Loss*.

4.1 Karakteristik Deskripsi Data Hasil *Preprocessing*

Pada tahap *preprocessing* data dilakukan dengan menyeleksi data citra yang hanya terdapat objek kapal, sehingga diperoleh data sebanyak 1465 citra yang telah terseleksi seperti pada Lampiran 1. Setelah itu agar citra satelit dapat diolah menggunakan metode YOLO-CNN perlu dilakukan anotasi atau pelabelan gambar sebagai data *ground truth bounding box*. Anotasi dilakukan untuk menandai dengan menggunakan *bounding box* pada objek kapal yang terdapat pada citra satelit dengan bantuan *software* labelImg, sehingga diperoleh *output* yang terdapat pada Lampiran 2. Contoh anotasi pada data citra satelit terdapat pada Gambar 4.1 sebagai berikut.



Gambar 4. 1 Anotasi Citra

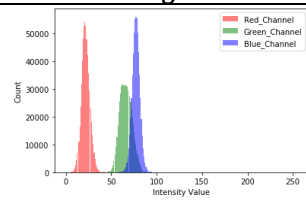
Dari citra tersebut diperoleh data yang terdiri dari jenis atau tipe kapal beserta lokasi dan spesifikasi *bounding box* pada citra sebagai berikut.

2 0,88 0,43 0,15 0,17

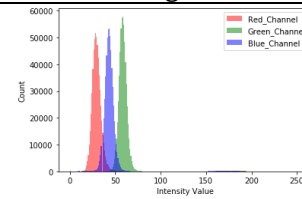
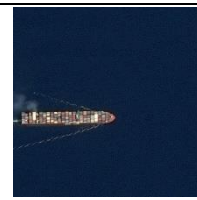
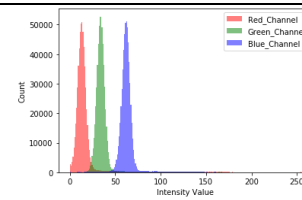
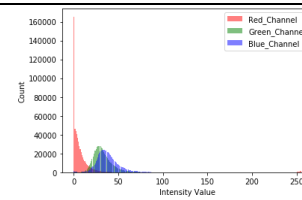
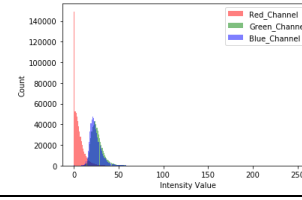
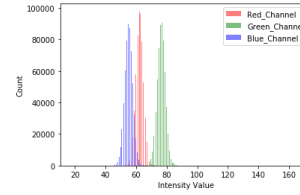
Data tersebut menunjukkan nilai 2 yaitu sebagai kelas kapal yang mana merupakan tipe kapal "*container*". Selanjutnya nilai 0,88 dan 0,43 merupakan nilai koordinat x dan y yang memiliki rentang nilai 0 hingga 1, yang dihitung dari pojok kiri atas citra dimana bernilai (0,0). Yang terakhir merupakan nilai untuk lebar dan tinggi *bounding box* yang terdapat pada citra yaitu bernilai 0,15 dan 0,17 pada rentang nilai 0 hingga 1, dimana nilai tersebut diperoleh dengan pembagian menggunakan nilai *pixel* maksimal yaitu 768. Anotasi atau pelabelan tipe kapal dilakukan dengan melihat karakteristik kapal secara visual. Dari data citra satelit yang digunakan pada penilitian ini yaitu citra yang memiliki objek kapal dengan 6 kelas tipe kapal, dimana pada masing-masing kelas terdapat perbedaan karakteristik.

Dari data citra satelit dengan objek kapal, dibuat histogram citra dengan menggunakan sintaks yang terdapat pada Lampiran 4. Histogram citra terbentuk dari banyaknya *pixel* dengan nilai *red green blue* (rgb), dengan sumbu horizontal dari histogram adalah nilai dari 0 – 255 sedangkan sumbu vertikal adalah banyak *pixel* pada nilai rgb tersebut. Begitu seterusnya untuk keseluruhan citra. Bentuk histogram dari salah satu citra masing-masing kelas yang digunakan terdapat pada Tabel 4.1.

Tabel 4. 1 Histogram Salah Satu Citra Setiap Kelas

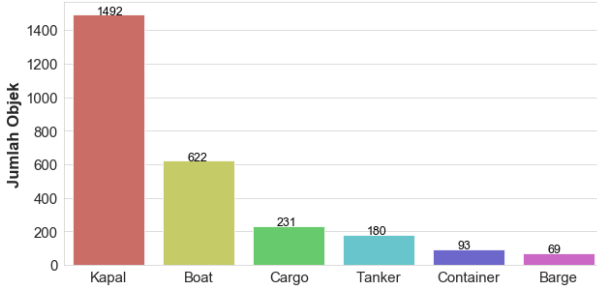
Tipe Kapal	Citra	Histogram
Boat		

Tabel 4. 1 Histogram Salah Satu Citra Setiap Kelas (Lanjutan)

Tipe Kapal	Citra	Histogram
Cargo		
Container		
Tanker		
Barge		
Kapal		

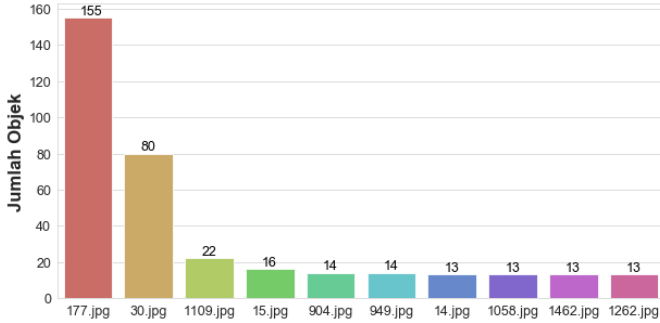
Dari Tabel 4.1 diatas dapat dilihat bahwa hasil seluruh histogram memiliki nilai persebaran yang berbeda-beda, baik histogram warna merah, hijau dan biru sehingga pada histogram RGB memiliki nilai yang berbeda. Hal tersebut menunjukkan

bahwa ada perbedaan karakteristik untuk setiap citra dan pada masing-masing tipe kapal. Dataset yang digunakan dalam penelitian ini yaitu terdiri dari 1465 citra gambar, 6 kategori kapal, dan 2687 objek kapal dalam dataset citra satelit. Jumlah objek untuk setiap kategori tipe kapal pada dataset ditunjukkan pada Gambar 4.2.



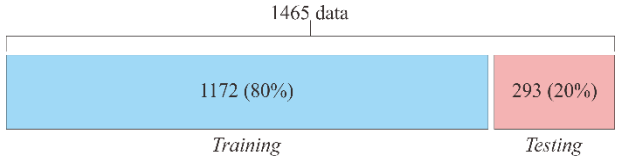
Gambar 4. 2 Bar Chart Jumlah Objek untuk Setiap Tipe Kapal

Berdasarkan Gambar 4.2 kategori dengan jumlah terbanyak adalah kelas keenam yaitu “kapal” yang berjumlah 1492 objek, maka dalam dataset ini paling banyak ditemukan tipe kapal yang berada diluar kelima tipe kapal yang ditentukan. Disusul oleh kelas tipe kapal “*boat*”, “*cargo*”, “*tanker*”, “*container*”, dan kelas dengan objek paling sedikit yaitu kelas tipe kapal “barge” yang hanya terdapat 69 objek. Jumlah objek didalam file citra pada dataset ditunjukkan pada Gambar 4.3.



Gambar 4. 3 Bar Chart 10 Citra dengan Objek Terbanyak didalam Dataset

Berdasarkan Gambar 4.3 citra satelit yang memiliki objek dengan jumlah paling banyak adalah citra 177.jpg dengan jumlah sebanyak 155 objek, diikuti oleh citra 30.jpg dengan jumlah 80 objek, lalu diurutan ketiga yaitu citra 1109.jpg dengan 22 objek. Dalam penelitian YOLOv3 *bounding box*es maksimal dalam satu citra adalah sebanyak 120 citra, maka dalam penelitian ini akan dikembangkan menjadi 155 objek dalam satu citra. Dilakukan pembagian 2 kelompok data citra satelit hasil dari seleksi data dan *pre-processing* yaitu data *training* dan *testing* Gambar 4.4.

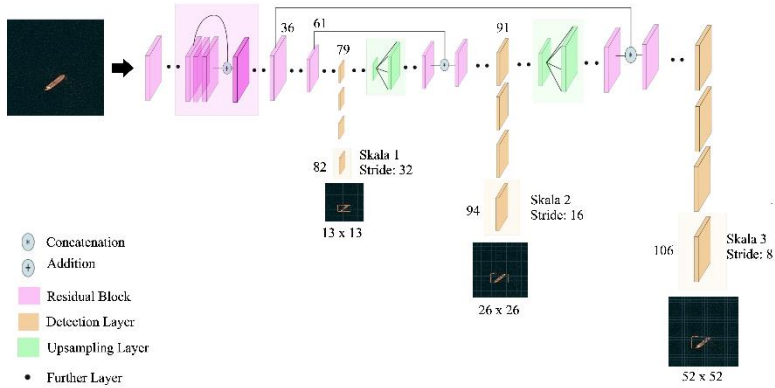


Gambar 4. 4 Pembagian Data *Training* dan *Testing*

Dari seluruh data kapal pada citra satelit dilakukan pengambilan sebanyak 80% data yaitu 1172 citra satelit sebagai data *training* dan sebanyak 20% data yaitu 293 citra satelit sebagai data *testing*. Data *training* digunakan untuk menangkap pola-pola yang dimiliki oleh citra satelit sehingga diperoleh model terbaik dengan hasil perhitungan nilai mAP dan *loss*. Data *testing* dipergunakan untuk melakukan evaluasi dengan menggunakan data yang baru dengan mendapatkan nilai mAP.

4.2 Klasifikasi dan Lokalisasi Tipe Kapal YOLO-CNN

Setelah dilakukan *preprocessing* data dan pembagian data, selanjutnya akan dilakukan pembuatan model untuk melakukan klasifikasi dan lokalisasi tipe kapal dengan mengklasifikasikan citra menurut tipe kapal yang telah ditentukan serta mendapatkan lokasi beserta spesifikasi *bounding box* kapal. Metode klasifikasi dan lokalisasi tipe kapal yang digunakan adalah metode *You Only Look Once Convolutional Neural Network* (YOLO-CNN). Pada proses pelatihan dilakukan *transfer learning* arsitektur YOLOv3 yang disebut juga Darknet53, yang dilakukan pada data *training* sehingga diperoleh model *custom* sesuai *dataset* yang digunakan.



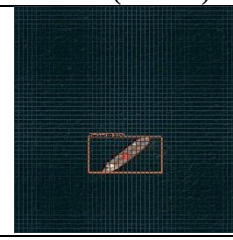
Gambar 4. 5 Ilsutrasi Detail Arsitektur YOLOv3

Sebelumnya arsitektur YOLOv3 atau Darknet-53 telah dijelaskan pada Subbab 2.5.1 di Gambar 2.7, secara singkat YOLOv3 memiliki jumlah 53 lapisan *convolutional layer* sehingga juga disebut sebagai Darknet-53. Pada Gambar 4.5 diatas diilustrasikan detail arsitektur YOLOv3 yang digunakan pada model dalam penelitian ini, YOLOv3 membuat klasifikasi dan lokalisasi objek dalam 3 skala yang berbeda untuk mengakomodasi ukuran objek yang berbeda dengan menggunakan *stride* 32, 16, dan 8 dari ukuran *input* yang telah diatur sebelum masuk kedalam arsitektur. *Input* gambar berukuran 416 x 416 *pixel* yang sudah dilakukan *rescale* sesuai dengan *hyperparameter*, sehingga diperoleh klasifikasi dan lokalisasi tipe kapal skala 13 x 13, 26 x 26, dan 52 x 52.

Pada skala pertama, YOLOv3 melakukan *downsampling* gambar *input* menjadi 13 x 13 dan membuat prediksi pada lapisan ke-82. Skala klasifikasi dan lokalisasi objek kesatu pada penelitian ini menghasilkan *output* tensor 3D dengan ukuran 13 x 13 x 33. Setelah itu, YOLOv3 mengambil *feature maps* dari lapisan 79 dan menerapkan satu *convolutional layer* sebelum dilakukan *upsampling* dengan faktor 2 sehingga memiliki ukuran 26 x 26.

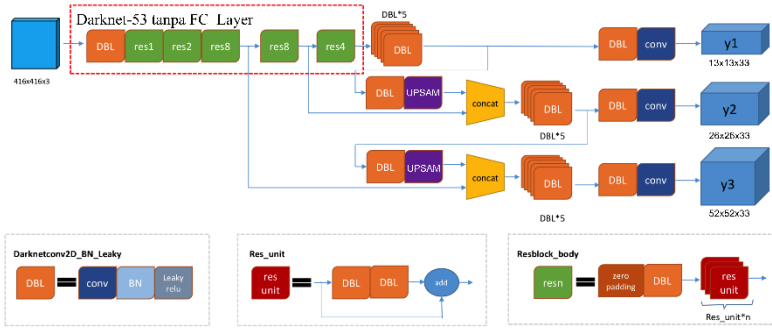
Feature maps yang ditingkatkan ini kemudian digabungkan dengan *feature maps* dari lapisan 61. Rangkuman *feature maps* kemudian mengalami beberapa *convolutional layer* lagi sampai pada skala klasifikasi dan lokalisasi objek kedua yang dilakukan pada lapisan 94. Skala prediksi kedua pada penelitian ini menghasilkan *output* dengan ukuran 26 x 26 x 33. Seperti pada 2 skala sebelumnya, dilakukan sekali lagi untuk memprediksi pada skala ke-3. *Feature maps* pada lapisan 91 ditambahkan satu *convolutional layer* lagi kemudian digabungkan dengan *feature maps* dari lapisan 36. Lapisan prediksi akhir dilakukan pada lapisan 106 menghasilkan *output* dengan ukuran 52 x 52 x 33.

Tabel 4. 2 Tiga Skala YOLOv3 pada Penelitian

Besar (13 x13)	Sedang (26 x 26)	Kecil (52 x 52)
		

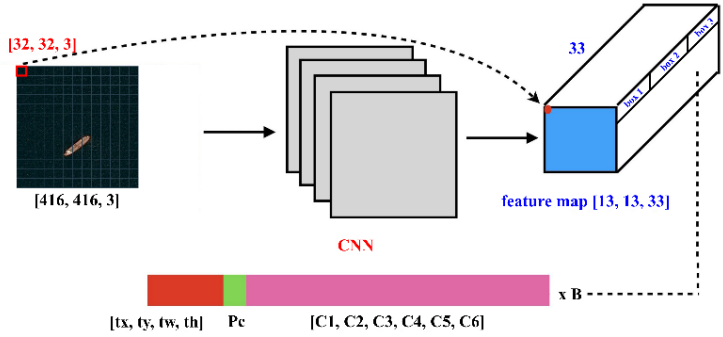
Singkatnya, YOLOv3 memprediksi dengan 3 skala klasifikasi dan lokalisasi tipe kapal yang berbeda, jadi jika kita melakukan *input* gambar berukuran 416 x 416 sesuai *hyperparameter*, akan menghasilkan 3 *output* yaitu 13 x 13 x 33, 26 x 26 x 33, dan 52 x 52 x 33. Berdasarkan Tabel 4.2 klasifikasi dan lokalisasi tipe kapal dengan 3 skala yang dilakukan oleh arsitektur YOLOv3 memiliki tujuan agar dapat dilakukan pada berbagai ukuran objek, baik objek berukuran besar, sedang, dan kecil dengan lebih baik. *Upsampling* yang dilakukan pada arsitektur YOLOv3 bertujuan agar dapat membantu jaringan mempelajari fitur berbutir halus yang berperan penting untuk mengklasifikasikan objek kecil. Pada setiap skala pada YOLOv3 setiap sel *grid* memprediksi 3 *bounding box* menggunakan 3

anchors, sehingga membuat jumlah total *anchors* yang digunakan sebanyak 9 *anchors*.



Gambar 4. 6 Ilsutrasi Detail Arsitektur YOLOv3

Pada Gambar 4.6 dapat dilihat bahwa gambar *input* ukuran 416 x 416 mendapatkan tiga cabang setelah memasuki jaringan Darknet-53. Cabang-cabang ini menjalani serangkaian konvolusi, *upsampling*, penggabungan, dan operasi lainnya. Tiga *feature maps* dengan ukuran yang berbeda akhirnya diperoleh, dengan bentuk [13, 13, 33], [26, 26, 33] dan [52, 52, 33].



Gambar 4. 7 Ilustrasi Singkat Proses Klasifikasi dan Lokalisasi pada Arsitektur YOLOv3

Ilustrasi dari proses klasifikasi tipe kapal dan lokalisasi koordinat objek kapal pada salah satu citra dengan salah satu skala pada dataset yang ditampilkan secara singkat sehingga didapatkan *output feature maps* [13, 13, 33] pada Gambar 4.7. *Input* gambar awal yaitu berukuran 768 x 768 *pixel* dengan tiga *channel* rgb sehingga menghasilkan *input* matriks berdimensi (768, 768, 3). Pada *hyperparameter image size* dilakukan *resize* ukuran gambar menjadi 416 x 416 *pixel* sehingga terjadi perubahan *input* matriks dengan dimensi (416, 416, 3). Secara singkat proses dimulai dengan *input* gambar melewati arsitektur CNN, sehingga pada akhirnya menghasilkan *output* berdimensi (13,13, 3, 11) atau (13, 13, 33), nilai 33 diperoleh dari 3 x 11 karena setiap sel berisi prediksi untuk 3 kotak, sesuai dengan 3 *anchors* yang digunakan pada setiap skala, sedangkan 11 merupakan nilai yang didapatkan dari jumlah 5 nilai untuk (*pc, bx, by, bh, bw*) dimana nilai pertama sebagai *confidence score* untuk kelas yang diprediksi dan nilai selanjutnya merupakan spesifikasi *bounding box* prediksi yang terdiri dari pusat koordinat *bounding box* beserta tinggi dan lebar, lalu 6 nilai sisanya adalah jumlah kelas yang ingin digunakan pada penelitian ini yaitu kelas "*boat*", "*cargo*", "*tanker*", "*container*", "*barge*", dan "kapal". Pada ilustrasi citra ini ditampilkan pada skala 1 dimana digunakan *stride* 32 untuk setiap *pixel* pada citra sehingga diperoleh *input* citra dengan *grid* berukuran 13 x 13, maka setiap sel dalam *grid* 13 x 13 pada gambar *input* citra menghasilkan 33 nilai. Berdasarkan arsitektur yang dimiliki oleh Darknet-53 masih terdapat 2 skala lagi dengan ilustrasi yang sama sesuai yang dijelaskan. Algoritma deteksi objek pada bagian akhir arsitektur YOLO setelah diperoleh output *feature maps* dengan 3 skala deteksi pada masing-masing citra dijelaskan lebih detail pada Subbab 2.5.8.

Pada proses *training* digunakan dua model jaringan konvolusional yang dipertimbangkan, yaitu YOLOv3 dengan 9 *anchors default* serta YOLOv3 dengan 9 *anchors* yang dilakukan

proses *k-means*, keduanya dilakukan pelatihan *transfer learning* pada jaringan arsitektur Darknet-53 yang telah dijelaskan sebelumnya. Perbedaan nilai *hyperparameter* mempengaruhi kecepatan dan kualitas proses pembelajaran yang dapat berpengaruh pada pembentukan model sehingga diperoleh model yang terbaik. Oleh karena itu, pada penelitian ini dilakukan *training* dengan membandingkan beberapa *hyperparameter* yang digunakan. Pada proses *training* digunakan *batch size* sebesar 64 dan jumlah *train step* 12000. *Batch size* dengan nilai 64 menyebabkan terambilnya 64 citra sebagai sampel dan *update weight* dan bias dalam 1 *trainstep*. *Subdivisions* atau *minibatch* dengan nilai 16 artinya dalam 1 *subdivisions* diambil 4 citra sebanyak 16 kali pengambilan dalam 1 *trainstep*. Konfigurasi/*hyperparameter* yang digunakan dalam penelitian secara detail terdapat pada Lampiran 12 untuk Model 1 dan Lampiran 13 untuk Model 2. Nilai *hyperparameter* tersebut ditentukan sebelum proses *training*. Pada Tabel 4.3 ditunjukkan beberapa *hyperparameter* yang digunakan untuk dua model dalam penelitian.

Tabel 4. 3 *Hyperparameter* Setiap Model

Model	***Hyper-parameter***	**Nilai**
Kedua model convolutional network	Image Size	416 x 416
	Batch size	64
	Subdivisions	16
	Training Step	12000
	Learning Rate	0,001
Model1	*Anchors*	(10x13);(16x30);(33,23); (30,61);(62,45);(59,119); (116,90);(156,198);(373,326)
Model2	*Anchors*	(11x20);(24x38);(31x50); (42x65);(53x82);(60,85); (88x108);(100x163);(141x168)

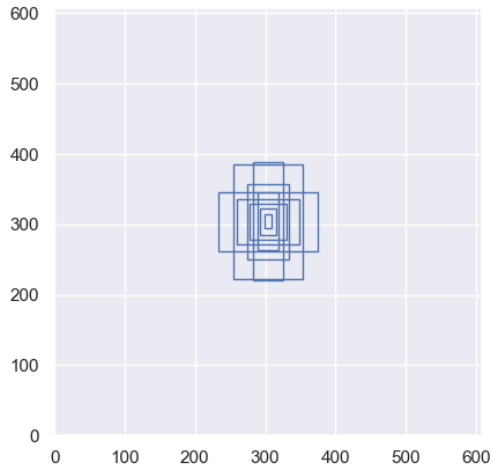
Train step atau iterasi terdiri dari *update weight* dan bias berdasarkan *batch size* dan pengecekan rata-rata *loss* dan akurasi untuk data *training*. Total banyak pelatihan citra yang dilakukan untuk memperoleh satu model adalah sebanyak total 768000 citra yang diperoleh dari pengambilan sampel citra sebanyak nilai *Batch size* yaitu 64 dikalikan dengan *trainstep* sebanyak 12000. Perbedaan kedua model terletak pada *hyperparameter anchors* dengan jumlah *anchors* yang sama-sama berjumlah 9, namun memiliki nilai lebar dan tinggi *anchors* yang berbeda.

Pada Model 1 nilai *anchors* merupakan nilai *anchors default* yang diperoleh dari pelatihan YOLOv3 yang berjumlah 9 dengan ukuran lebar dan tinggi (10 x 13); (16 x 30); (33 x 23); (30 x 61); (62 x 45); (59 x 119); (116 x 90); (156 x 198); dan (373 x 326). Dengan menggunakan sintaks pada Lampiran 8 didapatkan Model 2 dimana nilai *anchors* diperoleh dari perhitungan menggunakan metode *k-means* dengan k=9 seperti halnya *default* pada YOLOv3 dengan hasil sebagai berikut.

Gambar 4. 8 Plot *K-means Hyperparameter Anchor Box*

Pada Gambar 4.8 diperoleh plot pada koordinat x yang merupakan nilai lebar (*bw*) untuk sejumlah *ground truth bounding*

box, sedangkan pada koordinat y merupakan nilai tinggi (*bh*) untuk setiap *ground truth bounding box*. Selanjutnya dilakukan *k-means* dengan k=9 sehingga diperoleh 9 *centroid* sebagai pusat cluster. Dengan menggunakan langkah-langkah yang terdapat pada subbab 2.5.2, sehingga pada model 2 diperoleh 9 nilai *anchor box* sebagai *output* dari *k-means hyperparameter anchor box* dengan nilai (11x20); (24x38); (31x50); (42x65); (53x82); (60,85); (88x108); (100x163); (141x168) seperti yang ditunjukkan pada Gambar 4.9.



Gambar 4. 9 *Output K-means Hyperparameter Anchor Box*

Pada Gambar 4.9 diperoleh hasil dari *k-means anchor box* berjumlah 9 *anchor box* baru yang digunakan sebagai *hyper-parameter anchors* pada Model 2. Tujuan dari klusterisasi ini yaitu dapat disesuaikan *anchor box* yang digunakan dengan *ground truth bounding box* pada *dataset* yang digunakan sehingga memiliki hasil yang baik. Sehingga dalam penelitian untuk mengetahui performa kebaikan pelatihan digunakan model dengan variasi *hyper-parameter anchors* yaitu *default anchor box* pada YOLOv3 pada coco *dataset* pada Model 1 dibandingkan dengan model yang dilakukan *k-means anchor box* pada Model 2.

Pembuatan model menggunakan sintaks pada Lampiran 9 dengan bantuan Google Colab. Setelah iterasi berhenti maka model yang didapatkan akan menjadi model yang digunakan untuk menghitung nilai evaluasi pada *dataset*. Nilai evaluasi pertama pada penelitian ini adalah *confusion matrix*, yang digunakan untuk mengetahui hasil kebaikan klasifikasi pada kedua model, selanjutnya dapat dihitung nilai *precision* dan *recall* pada masing-masing kelas setiap model. Nilai evaluasi kedua yang digunakan yaitu mAP, merupakan nilai evaluasi yang paling populer pada deteksi objek, nilai mAP dapat digunakan untuk membandingkan kebaikan sebuah model dengan model lainnya pada tugas deteksi objek. Tabel 4.4 dan Tabel 4.5 merupakan output *confusion matrix* untuk data *training* pada Model 1 dan Model 2 yang disajikan untuk mengetahui hasil kinerja klasifikasi yang diperoleh dengan menggunakan sintaks pada Lampiran 14, pada tabel dibawah baris mewakili *ground truth* (G) sedangkan kolom mewakili prediksi (P).

Tabel 4. 4 *Confusion Matrix* Model 1 *Training*

G \ P	BO	CA	CO	TA	BA	KA	NO	TO
BO	442	0	0	0	0	1	75	518
CA	0	178	0	0	0	0	0	178
CO	0	0	75	0	0	0	0	75
TA	0	0	0	159	0	0	2	161
BA	0	0	0	0	65	0	1	66
KA	0	3	0	0	0	1136	26	1165
NO	81	14	3	20	7	70		
TO	523	195	78	179	72	1207		

Pada Tabel 4.4 dan 4.5, BO yaitu kelas *"Boat"*; CA yaitu kelas *"Cargo"*; TA yaitu kelas *"Tanker"*; BA yaitu kelas *"Barge"*; KA yaitu kelas *"Kapal"*. Selanjutnya NO atau *None*, pada *ground truth* artinya secara aktual terdapat objek namun tidak terdeteksi,

sedangkan pada prediksi artinya objek terdeteksi namun secara aktual tidak ada objek. Terakhir TO, dimana merupakan Total keseluruhan deteksi.

Berdasarkan hasil *confusion matrix* Model 1 pada Tabel 4.4, kelas BO (*Boat*) memiliki *True Positive* sebanyak 442, *False Negative* sebanyak 76, *False Positive* sebanyak 81; kelas CA (*Cargo*) memiliki *True Positive* sebanyak 75, *False Negative* sebanyak 0, *False Positive* sebanyak 17; kelas CO (*Container*) memiliki *True Positive* sebanyak 75, *False Negative* sebanyak 0, *False Positive* sebanyak 3; kelas TA (*Tanker*) memiliki *True Positive* sebanyak 159, *False Negative* sebanyak 2, *False Positive* sebanyak 20; kelas BA (*Barge*) memiliki *True Positive* sebanyak 65, *False Negative* sebanyak 1, *False Positive* sebanyak 65; kelas KA (Kapal) memiliki *True Positive* sebanyak 1136, *False Negative* sebanyak 29, *False Positive* sebanyak 71.

Tabel 4. 5 *Confusion Matrix* Model 2 *Training*

G \ P	BO	CA	CO	TA	BA	KA	NO	TO
BO	444	0	0	0	0	1	73	518
CA	0	178	0	0	0	0	0	178
CO	0	0	75	0	0	0	0	75
TA	0	1	0	159	0	0	1	161
BA	0	0	0	0	66	0	0	66
KA	0	3	0	2	0	1132	28	1165
NO	83	23	0	9	8	89		
TO	527	205	75	170	74	1222		

Pada Tabel 4.5, diperoleh hasil *confusion matrix* pada Model 2, kelas BO (*Boat*) memiliki *True Positive* sebanyak 444, *False Negative* sebanyak 74, *False Positive* sebanyak 83; kelas CA (*Cargo*) memiliki *True Positive* sebanyak 178, *False Negative* sebanyak 0, *False Positive* sebanyak 27; kelas CO (*Container*)

memiliki *True Positive* sebanyak 75, *False Negative* sebanyak 0, *False Positive* sebanyak 0; kelas TA (*Tanker*) memiliki *True Positive* sebanyak 159, *False Negative* sebanyak 2, *False Positive* sebanyak 11; kelas BA (*Barge*) memiliki *True Positive* sebanyak 66, *False Negative* sebanyak 0, *False Positive* sebanyak 8; kelas KA (Kapal) memiliki *True Positive* sebanyak 1132, *False Negative* sebanyak 33, *False Positive* sebanyak 90. Dari hasil *confusion matrix* untuk Model 1 dan Model 2, dapat dihitung *precision* dan *recall* pada masing-masing kelas sebagai berikut.

Tabel 4. 6 *Precision* dan *Recall Training*

Kelas	**Model 1**		**Model 2**	
	***Precision* (%)**	***Recall* (%)**	***Precision* (%)**	***Recall* (%)**
Boat	84,51	85,33	84,25	85,71
Cargo	91,28	100,00	86,83	100,00
Container	96,15	100,00	100,00	100,00
Tanker	88,83	98,76	93,53	98,76
Barge	90,28	98,48	89,19	100,00
Kapal	94,12	97,51	92,64	97,17
Mean	90,86	96,68	91,07	96,94

Pada Tabel 4.6, diperoleh hasil perhitungan *precision* dan *recall* untuk masing-masing kelas pada Model 1 dan Model 2. *Precission* merupakan rasio prediksi benar positif dibandingkan dengan keseluruhan hasil yang diprediksi positif yang diperoleh dengan persamaan 2.17. Model 1 memiliki *precision* yang lebih tinggi pada tipe kapal "*Boat*", "*Cargo*", "*Barge*", dan "*Kapal*" sedangkan Model 2 memiliki *precision* yang lebih tinggi pada tipe kapal "*Container*" dan "*Tanker*". Namun untuk rata-rata *precision* Model 2 lebih tinggi dibandingkan dengan Model 1, artinya secara rata-rata Model 2 lebih baik dalam melakukan prediksi tipe kapal dengan lebih akurat. *Recall* merupakan rasio prediksi benar positif

dibandingkan dengan keseluruhan data yang benar positif yang diperoleh dengan persamaan 2.18. Model 1 memiliki *recall* yang lebih tinggi pada tipe "*Kapal*", sedangkan Model 2 memiliki *recall* yang lebih tinggi pada tipe kapal "*Boat*" dan "*Barge*". Untuk tipe kapal *"Cargo"*, *"Container"*, dan *"Tanker"* pada kedua model memiliki nilai *recall* yang sama. Namun untuk rata-rata *recall* Model 2 lebih tinggi dibandingkan dengan Model 1, artinya secara rata-rata Model 2 lebih baik dalam mengklasifikasikan jenis kapal yang sesuai dengan *ground truth*.

Selanjutnya pada Tabel 4.7 disajikan hasil kinerja deteksi objek yang didapatkan pada metode YOLO-CNN menggunakan sintaks pada Lampiran 11 dengan bantuan *software* python.

Tabel 4. 7 Nilai Rata-Rata *Loss* dan mAP *Training*

Model	**Avg *Loss***	**mAP (%)**
Model 1	0,2388	94,85
Model 2	0,1978	95,06

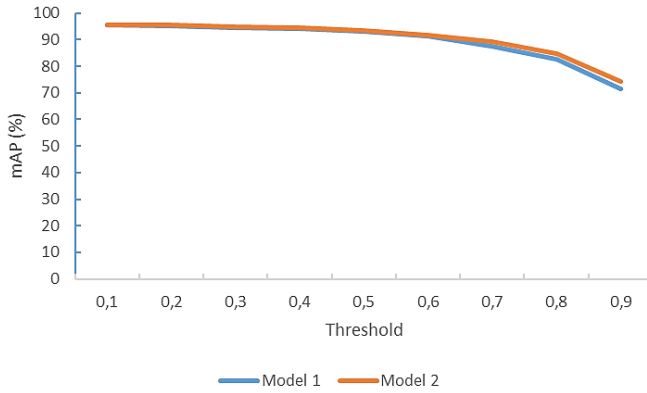
Tabel 4.7 menunjukkan kinerja klasifikasi yang didapatkan dari setiap model. Kinerja klasifikasi yang terbaik merupakan Model 2 dengan model yang dilakukan *training* menggunakan *anchors* sebanyak 9 *anchors* yang melalui proses *k-means*. Hasil ini mendapatkan rata-rata *loss* Model 2 pada data *training* lebih kecil daripada Model 1 yaitu senilai 0,1978, artinya nilai bias prediksi terhadap lokalisasi *bounding box*, *confidence score*, serta klasifikasi secara keseluruhan pada Model 2 yaitu sebesar 0,1978. Nilai mAP pada Model 2 pada data *training* lebih tinggi dibandingkan Model 1 yaitu senilai 95,06%, nilai tersebut memiliki arti bahwa rata-rata ketepatan prediksi kapal pada setiap kelas yaitu sebesar 95,06%. Penggunaan *anchors* dengan menggunakan metode *k-means* mengindikasikan adanya penambahan kinerja klasifikasi. Selanjutnya ditampilkan kinerja kedua model pada setiap kelas. Tabel 4.8 merupakan hasil kinerja untuk setiap kelas

dengan metode YOLO-CNN yang menampilkan nilai *Average Precission* (AP) pada data *training*.

Tabel 4. 8 Nilai *Average Precission Training* Setiap Kelas

Kelas	***Average Precission*/AP (%)**	
	Model 1	**Model 2**
Boat	80,96	79,60
Cargo	99,94	98,06
Container	98,67	100,00
Tanker	98,69	98,11
Barge	95,02	98,39
Kapal	95,82	96,21

Berdasarkan Tabel 4.8 diperoleh hasil *Average Precision* untuk kebaikan deteksi objek pada masing-masing kelas di kedua model. Model 1 dapat melakukan prediksi tipe kapal yang lebih tinggi untuk tipe kapal "*Boat*" dan "*Cargo*", sedangkan Model 2 dapat melakukan prediksi tipe kapal yang lebih tinggi untuk tipe kapal "*Container*", "*Tanker*", "*Barge*", dan "Kapal". Maka berdasarkan Tabel 4.8 didapatkan bahwa model terbaik yaitu Model 2.

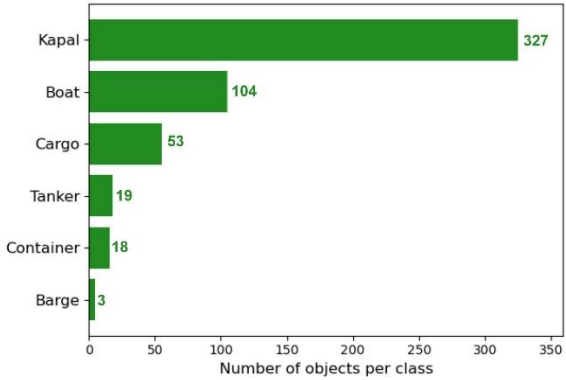


Gambar 4. 10 Grafik mAP setiap *Threshold*

Untuk lebih memperjelas kebaikan model diantara keduanya, dilanjutkan dengan melihat nilai mAP yang dibandingkan untuk setiap *threshold* pada setiap model, untuk menentukan model mana yang memiliki nilai mAP maksimum. Grafik mAP setiap *threshold* tercantum pada Gambar 4.10. *Threshold* dengan nilai 0,1 memiliki nilai mAP yang cenderung lebih tinggi, semakin tinggi *threshold* maka semakin turun nilai mAP pada kedua model. Pada grafik tersebut dapat dilihat bahwa nilai mAP pada model 2 cenderung lebih tinggi dibandingkan dengan model 1, meskipun memiliki selisih nilai yang tidak terlalu besar. Sehingga berdasarkan grafik tersebut dapat disimpulkan bahwa model terbaik dengan metode YOLO-CNN adalah model dengan menggunakan *hyperparameter anchors* sebanyak 9 yang dilakukan *k-mean sclustering*.

4.3 Deteksi Tipe Kapal

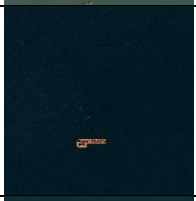
Pada proses *training* diperoleh model 2 merupakan model terbaik. Selanjutnya dilakukan deteksi tipe kapal untuk kedua model dengan *hyperparameter* yang didapat digunakan sebagai acuan untuk melakukan deteksi citra satelit dengan objek kapal pada data *testing*. Berikut ditampilkan *ground truth bounding box* untuk setiap kelas pada data *testing.*



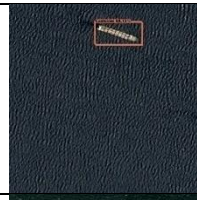
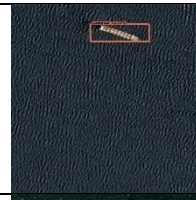
Gambar 4. 11 Jumlah Objek setiap Kelas

Berdasarkan Gambar 4.11 kelas “Kapal” memiliki jumlah objek yang tertinggi pada data *testing* yaitu sebanyak 327 objek, kelas “*Boat*” sebanyak 104 objek, kelas “*Cargo*” sebanyak 53 objek, kelas “*Tanker*” sebanyak 19 objek, kelas “*Container*” sebanyak 18 objek, dan kelas “*Barge*” sebanyak 3 objek. Hasil deteksi kapal terdapat pada Lampiran 3. Untuk menggambarkan efektivitas kedua model yang digunakan, diilustrasikan dengan menunjukkan hasil deteksi tipe kapal. Kedua model dibandingkan hasil deteksi pada setiap kelas.

Tabel 4. 9 Hasil Deteksi dengan *Ground Truth* setiap Kelas

Ground Truth	**Model 1**	**Model 2**
Boat		
Barge		
Kapal		
Cargo		

Tabel 4. 9 Hasil Deteksi dengan *Ground Truth* setiap Kelas (Lanjutan)

Ground Truth	**Model 1**	**Model 2**
Container		
Tanker		

Dari Tabel 4.9 dapat dilihat bahwa hasil deteksi untuk *ground truth* salah satu data gambar masing-masing kelas. Pada masing-masing data gambar memiliki hasil deteksi yang berbeda-beda dari segi spesifikasi ukuran dan koordinat *bounding box*, serta *confidence score* pada kelas yang diprediksi. *Confidence score* yang diperoleh hampir pada semua gambar pada setiap kelas yang ditampilkan cenderung pada model 2 didapatkan *confidence score* yang lebih tinggi dibandingkan dengan pada model 1. Pada kelas dengan *ground truth bounding box* "*Boat*" pada model 1 terdapat kesalahan deteksi, dimana pada *ground truth bounding box* yang seharusnya merupakan "*Boat*" namun dideteksi sebagai "Kapal". Hal ini kemungkinan disebabkan karena adanya karakteristik yang hampir mirip antara kelas "*Boat*" dengan kelas "Kapal". Selanjutnya untuk mengetahui hasil evaluasi klasifikasi pada kedua model, disajikan Tabel 4.10 dan Tabel 4.11 sebagai output *confusion matrix* untuk data *testing* pada Model 1 dan Model 2 yang disajikan untuk mengetahui hasil kinerja klasifikasi yang diperoleh dengan menggunakan sintaks pada Lampiran 14. Pada tabel dibawah baris mewakili *ground truth* (G) sedangkan kolom mewakili prediksi (P). BO yaitu kelas *"Boat"*; CA yaitu kelas *"Cargo"*; TA yaitu kelas

"Tanker"; BA yaitu kelas *"Barge"*; KA yaitu kelas *"*Kapal*"*. Selanjutnya NO atau *None*, pada *ground truth* artinya secara aktual terdapat objek namun tidak terdeteksi, sedangkan pada prediksi artinya objek terdeteksi namun secara aktual tidak ada objek. Terakhir TO, dimana merupakan Total keseluruhan deteksi.

Tabel 4. 10 *Confusion Matrix* Model 1 *Testing*

G \ P	BO	CA	CO	TA	BA	KA	NO	TO
BO	26	0	0	0	0	17	61	104
CA	0	37	1	6	0	5	4	53
CO	0	1	11	0	0	2	4	18
TA	0	0	0	16	0	3	0	19
BA	0	0	0	0	1	0	2	3
KA	16	8	0	13	6	153	131	327
NO	22	15	4	16	11	100		
TO	64	61	16	51	18	280		

Berdasarkan hasil *confusion matrix* Model 1 pada Tabel 4.10, kelas BO (*Boat*) memiliki *True Positive* sebanyak 26, *False Negative* sebanyak 78, *False Positive* sebanyak 38; kelas CA (*Cargo*) memiliki *True Positive* sebanyak 37, *False Negative* sebanyak 16, *False Positive* sebanyak 24; kelas CO (*Container*) memiliki *True Positive* sebanyak 11, *False Negative* sebanyak 7, *False Positive* sebanyak 5; kelas TA (*Tanker*) memiliki *True Positive* sebanyak 16, *False Negative* sebanyak 3, *False Positive* sebanyak 35; kelas BA (*Barge*) memiliki *True Positive* sebanyak 1, *False Negative* sebanyak 2, *False Positive* sebanyak 17; kelas KA (Kapal) memiliki *True Positive* sebanyak 153, *False Negative* sebanyak 174, *False Positive* sebanyak 127. Selanjutnya didapatkan hasil *confusion matrix* untuk Model 2 pada data *testing* sebagai pembanding *confusion matrix* untuk Model 1 pada data *testing* yang disajikan pada Tabel 4.11.

Tabel 4. 11 *Confusion Matrix* Model 2 *Testing*

G \ P	BO	CA	CO	TA	BA	KA	NO	TO
BO	31	0	0	0	0	17	56	104
CA	0	37	1	3	1	8	3	53
CO	0	1	11	0	1	1	4	18
TA	0	1	0	16	0	2	0	19
BA	0	0	0	0	1	0	2	3
KA	11	9	1	12	8	161	125	327
NO	35	13	1	17	10	101		
TO	77	61	14	48	21	290		

Pada Tabel 4.11, diperoleh hasil *confusion matrix* pada Model 2, kelas BO (*Boat*) memiliki *True Positive* sebanyak 31, *False Negative* sebanyak 73, *False Positive* sebanyak 46; kelas CA (*Cargo*) memiliki *True Positive* sebanyak 37, *False Negative* sebanyak 16, *False Positive* sebanyak 24; kelas CO (*Container*) memiliki *True Positive* sebanyak 11, *False Negative* sebanyak 7, *False Positive* sebanyak 3; kelas TA (*Tanker*) memiliki *True Positive* sebanyak 16, *False Negative* sebanyak 3, *False Positive* sebanyak 32; kelas BA (*Barge*) memiliki *True Positive* sebanyak 1, *False Negative* sebanyak 2, *False Positive* sebanyak 20; kelas KA (Kapal) memiliki *True Positive* sebanyak 161, *False Negative* sebanyak 166, *False Positive* sebanyak 129. Dari hasil *confusion matrix* untuk Model 1 dan Model 2, dapat dihitung *precision* dan *recall* pada masing-masing kelas sebagai berikut. Pada Tabel 4.12, diperoleh hasil perhitungan *precision* dan *recall* untuk masing-masing kelas pada Model 1 dan Model 2. *Precission* merupakan rasio prediksi benar positif dibandingkan dengan keseluruhan hasil yang diprediksi positif yang diperoleh dengan persamaan 2.17. *Recall* merupakan rasio prediksi benar positif dibandingkan dengan

keseluruhan data yang benar positif yang diperoleh dengan persamaan 2.18.

Tabel 4. 12 *Precision* dan *Recall Testing*

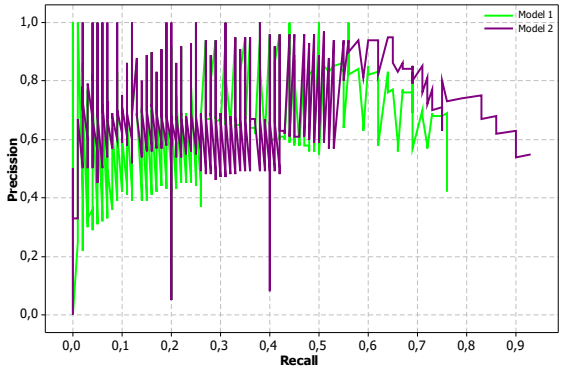
Kelas	**Model 1**		**Model 2**	
	***Precision* (%)**	***Recall* (%)**	***Precision* (%)**	***Recall* (%)**
Boat	40,62	25,00	40.26	29,81
Cargo	60,66	69,81	60.66	69,81
Container	68,75	61,11	78.57	61,11
Tanker	31,37	84,21	33.33	84,21
Barge	5,56	33,33	4,76	33,33
Kapal	54,64	46,79	55.52	49,24
Mean	43,60	53,38	45,52	54,59

Berdasarkan Tabel 4.12, Model 1 memiliki *precision* yang lebih tinggi pada tipe kapal "*Boat*" dan "*Barge*", sedangkan Model 2 memiliki *precision* yang lebih tinggi pada tipe kapal "*Container*", "*Tanker*", dan "Kapal". Pada tipe kapal *"Cargo"* dan *"Barge"* pada kedua model memiliki nilai *precision* yang sama. Rata-rata *precision* pada Model 2 lebih tinggi dibandingkan dengan Model 1, artinya secara rata-rata Model 2 lebih baik dalam melakukan prediksi tipe kapal dengan lebih akurat. Selanjutnya pada *recall* Model 2 memiliki *recall* yang lebih tinggi pada tipe kapal "*Boat*" dan "*Kapal*". Pada tipe kapal *"Cargo"*, *"Container"*, *"Tanker"*, dan "*Barge"* pada kedua model memiliki nilai *recall* yang sama. Rata-rata *recall* pada Model 2 lebih tinggi dibandingkan dengan Model 1, artinya secara rata-rata Model 2 lebih baik dalam mengklasifikasikan jenis kapal yang sesuai dengan *ground truth*. Selanjutnya pada Tabel 4.13 disajikan hasil kinerja deteksi objek yang didapatkan pada metode YOLO-CNN menggunakan sintaks pada Lampiran 11, disajikan *Average Precission* untuk setiap kelas beserta nilai mAP untuk kedua model.

Tabel 4. 13 Nilai *Average Precission Testing* Setiap Kelas

Kelas	***Average Precission*/AP (%)**	
	Model 1	**Model 2**
Boat	12,01	23,66
Cargo	68,25	70,58
Container	66,83	70,52
Tanker	60,10	79,43
Barge	10,00	22,00
Kapal	34,11	36,27
mAP (%)	41,88	50,41

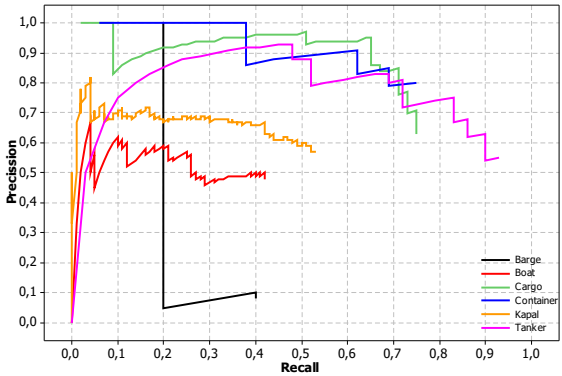
Berdasarkan Tabel 4.13 diperoleh perbandingan *average precission* untuk kedua model, dimana model 2 tetap merupakan model yang terbaik, dengan hasil *average precission* pada setiap kelas memiliki nilai yang lebih tinggi jika dibandingkan dengan model 1, sehingga diperoleh nilai mAP pada model 2 juga lebih tinggi jika dibandingkan dengan model 1, yaitu masing-masing sebesar 50,41% dan 41,88%.



Gambar 4. 12 *Precission Recall Curve* Kedua Model

Berdasarkan Gambar 4.12 dibandingkan *precission recall curve* antara model 1 dan 2. *Precission recall curve* adalah ukuran

yang berguna untuk keberhasilan prediksi ketika kategori kelas sangat tidak seimbang (*imbalanced*). *Precission* merupakan rasio prediksi benar positif dibandingkan dengan keseluruhan hasil yang diprediksi positif. *Recall* merupakan rasio prediksi benar positif dibandingkan dengan keseluruhan data yang benar positif. Dengan *precission* pada sumbu y dan *recall rate* pada sumbu x, didapatkan kurva *precision-recall* (P-R). Kurva P-R menunjukkan *tradeoff* antara presisi dan *recall* untuk *threshold* yang berbeda. Kurva P-R yang sedekat mungkin kekanan atas semakin baik menunjukkan model dan algoritma yang dihasilkan lebih efisien. Seperti yang ditunjukkan pada Gambar 4.13, garis ungu (model 2) paling dekat dengan kanan atas dan melampaui garis hijau (model 1). Oleh karena itu, model 2 model yang menunjukkan kinerja terbaik dari kedua model.



Gambar 4. 13 *Precission Recall Curve* Kedua Model Setiap Kelas

Berdasarkan Gambar 4.13 dibandingkan *precission recall curve* model 2 untuk setiap kelas tipe kapal dalam penelitian. Pada model 2 garis bewarna ungu kelas "*Tanker*" semakin mendekat kekanan atas artinya kelas "*Tanker*" dapat terdeteksi cukup baik dibandingkan dengan lainnya. Garis hitam merupakan garis yang paling jauh dari kanan atas, artinya kelas "*Barge*" mendapatkan hasil deteksi yang kurang baik pada model 2.

(Halaman ini sengaja dikosongkan)

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan analisis dan pembahasan yang telah dilakukan pada bab IV, maka diperoleh kesimpulan yaitu metode yang memiliki kinerja klasifikasi terbaik merupakan model 2 dengan model yang dilakukan *train*ing menggunakan anchors sebanyak 9 anchors melalui proses *k-means*. Hasil ini mendapatkan rata-rata loss pada model 2 lebih kecil daripada model 1 yaitu senilai 0,1978, serta nilai mAP pada model 2 pada data *train*ing lebih tinggi dibandingkan model 1 yaitu masing-masing senilai 95,06% dan 94,85%.Hasil deteksi tipe kapal pada data *testing* diperoleh hasil *average precission* pada model 2 setiap kelas memiliki nilai yang lebih tinggi jika dibandingkan dengan model 1, sehingga diperoleh nilai mAP pada model 2 lebih tinggi dibandingkan model 1, yaitu masing-masing sebesar 50,41% dan 41,88%.

5.2 Saran

Berdasarkan kesimpulan yang diperoleh, dapat dirumuskan saran sebagai pertimbangan penelitian selanjutnya adalah sebagai berikut,

1. Sebaiknya dibandingkan dengan metode deteksi objek lainnya seperti FasterRCNN dan SSD (*Single-Shot Multi-Box Detectors*) untuk mengetahui metode terbaik dalam melakukan deteksi tipe kapal.
2. Dapat digunakan lebih banyak data sehingga "*machine*" dapat belajar pola-pola kapal yang ada di dunia serta model kapal yang ada di Indonesia untuk meningkatkan pengenalan kapal dan juga dapat ditambahkan tahap *preprocessing* augmentasi citra seperti *rotate*, *flip*, dsb sehingga deteksi tipe kapal dapat diaplikasikan pada berbagai jenis kasus.

(Halaman ini sengaja dikosongkan)

DAFTAR PUSTAKA

BAPPENAS. (2016). *Prakarsa Strategis Optimalisasi Pemanfaatan Potensi Kelautan Menuju Terwujudnya Indonesia sebagai Poros Maritim.* https://www.bappenas.go.id/files/8514/6217/9185/Ringkasan_Laporan_Akhir_Poros_Maritim_Final_31032016.pdf.

Bergstra, J., Bardenet, R., Bengio, Y., & Kegl, B. (2011). Algorithms for Hyper-Parameter Optimization. *Conference: Advances in Neural Information Processing Systems*.

Databoks. (2016). *Indonesia Produsen Ikan Laut Kedua Terbesar Dunia*. Retrieved Januari 28, 2020, from https://databoks.katadata.co.id/datapublish/2016/09/28/indonesia-produsen-ikan-laut-kedua-terbesar-dunia

detik.com. (2014). *Meski Punya Alat Kontrol Canggih, RI Sulit Berantas Pencurian Ikan*. Retrieved Januari 24, 2020, from https://finance.detik.com/berita-ekonomi-bisnis/d-2734721/meski-punya-alat-kontrol-canggih-ri-sulit-berantas-pencurian-ikan/komentar?device=desktop

detik.com. (2015). *Kapal Jumbo Pencuri Ikan Dituntut Rp 200 Juta, Satgas: Harusnya Ditenggelamkan*. Retrieved Januari 24, 2020, from https://news.detik.com/berita/2865303/kapal-jumbo-pencuri-ikan-dituntut-rp-200-juta-satgas-harusnya-ditenggelamkan

FAO. (2015). *Expert Workshop to Estimate the Magnitude of Illegal, Unreported and Unregulated Fishing Globally*. Rome.

Gonzalez, R. C., & Woods, R. E. (2008). *Digital Image Processing* (Third Edition ed.). New Jersey: Pearson Education.

Goodfellow, I., Bengio, Y., & Courville, A. (2017). *Deep Learning.* Cambridge: MIT Press.

Hadinata, Y. (2010). *Pelaksanaan Vessel Monitoring System (VMS) di Indonesia.*

Han, J., Kamber, M., & Pei, J. (2011). *Data Mining: Concepts and Techniques.* San Francisco: Morgan Kaufmann.

ImageNet. (2016). *ImageNet*. Retrieved from http://www.image-net.org/

Ioffe, S., & Szegedy, C. (2015, February 11). *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*. Retrieved from arXiv: https://arxiv.org/pdf/1502.03167.pdf

Jaelani, A. Q., & Basuki, U. (2014). Illegal Unreported and Unregulated (IUU) Fishing: Upaya Mencegah dan Memberantas Illegal Fishing dalam Membangun Poros Maritim Indonesia. *SUPREMASI HUKUM, 3*.

Kementrian Kelautan dan Perikanan. (n.d.). *UNDANG-UNDANG REPUBLIK INDONESIA NOMOR 6 TAHUN 1996*. Retrieved Januari 24, 2020, from http://jdih.kkp.go.id/peraturan/uu-1996-06.pdf

Kementrian Luar Negeri. (2008). *UNDANG-UNDANG REPUBLIK INDONESIA NOMOR 17 TAHUN 2008*. Retrieved from https://pih.kemlu.go.id/files/uu_17_tahun_2008.pdf

Krizhevsky, A., Sutskever, I., & Hinton, G. (2012). Imagenet classification with deep convolutional neural networks, In Advances in neural information processing systems.

LeCun, Y., Haffner, P., Bottou, L., & Bengio, Y. (1998). Gradient-Based Learning Applied to Document Recognition. *Proc. of the IEEE, 86*(11).

Li, Y., Zhang, H., Guo, Q., & Li, X. (2018). Machine Learning Methods for Ship Detection in Satellite Images.

LIPI. (2019). *Riset dan Konservasi Jadi Kunci Pemanfaatan Potensi Laut Indonesia*. Retrieved Januari 24, 2020, from http://lipi.go.id/berita/single/Riset-dan-Konservasi-Jadi-Kunci-Pemanfaatan-Potensi-Laut-Indonesia/21606

Liu, Y., Xu, P., Zhang, M., & Guo, Z. (2017). SAR Ship Detection Using Sea-land Segmentation-based Convolutional Neural Network.

Manning, C., Raghavan, P., & Schutze, H. (2009). *An Introduction to Information Retriveral*. Cambridge : Cambridge University Press.

Patterson, J., & Gibson, A. (2017). *Deep Learning: A Practitioner's Approach* (1st ed.). United Syates of America: O'Reilly Media, Inc.

Pratiwi, Y. D. (2016). PERTANGGUNGJAWABAN PIDANA ILLEGAL FISHING KORPORASI DALAM CITA-CITA INDONESIA POROS MARITIM DUNIA. *1* .

Putra, I. S., Wijaya, A. Y., & Soelaiman, R. (2016). Klasifikasi Citra menggunakan Convolutional Neural Network (CNN) pada Caltech 101. *Jurnal Teknik ITS*.

Raykov, T., & Marcoulides, G. A. (2006). *A first course in structural equation modeling* (2nd ed ed.). US: Lawrence Erlbaum Associates Publishers.

Redmon, J., & Farhadi, A. (2018). Yolo: Real-time object detection.

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You Only Look Once: Unified, Real-Time Object Detection. Retrieved from https://arxiv.org/pdf/1506.02640.pdf

Sewak, M., Karim, M. R., & Pujari, P. (2018). *Practical Convolutional Neural Networks*. UK: Packt Publishing Ltd.

Shao, X., Li, H., Lin, H., Kang, X., & Lu, T. (2017). Ship Detection in Optical Satellite Image Based on RX Method and PCAnet.

Syahrani, D. A., Al Musadieq, M., & Darmawan, A. (2017). ANALISIS PERAN KEBIJAKAN ILLEGAL, UNREPORTED, AND UNREGULATED FISHING (IUU) PADA EKSPOR IKAN TUNA DAN UDANG TANGKAP. *Jurnal Administrasi Bisnis (JAB), 45*.

Wang, J., & Perez, L. (2017). *The Effectiveness of Data Augmentation in Image Classification using DeepLearning*. Retrieved from https://arxiv.org/pdf/1712.04621.pdf

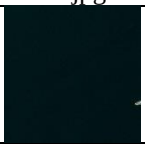
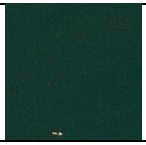
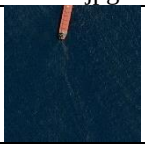
Zhang, A., Lipton, Z. C., Li, M., & Smola, A. J. (2020). *Dive into Deep Learning*. UC Berkeley: Spring.

Zhang, R., Yao, J., Zhang, K., Feng, C., & Zhang, J. (2016). S-CNN-Based Ship Detection from High Resolution Remote Sensing Images.

(Halaman ini sengaja dikosongkan)

LAMPIRAN

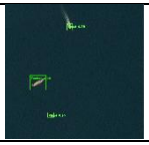
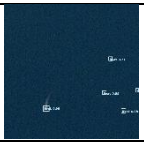
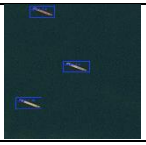
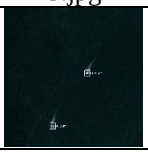
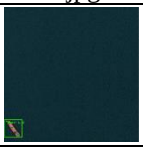
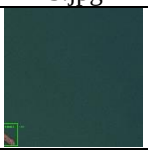
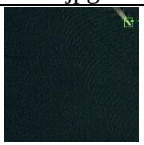
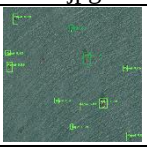
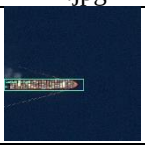
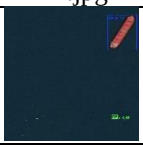
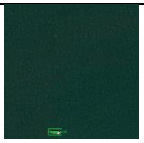
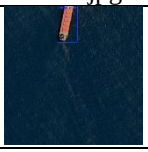
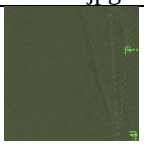
Lampiran 1 Data Gambar Kapal

				
1.jpg	2.jpg	3.jpg	4.jpg	5.jpg
				
6.jpg	7.jpg	8.jpg	9.jpg	10.jpg
				
11.jpg	12.jpg	13.jpg	14.jpg	15.jpg
				
16.jpg	17.jpg	18.jpg	19.jpg	20.jpg
⋮	⋮	⋮	⋮	⋮
				
1456.jpg	1457.jpg	1458.jpg	1459.jpg	1460.jpg
				
1461.jpg	1462.jpg	1463.jpg	1464.jpg	1465.jpg

Lampiran 2 Data *Ground Truth Bounding Box*

Citra	***Ground Truth Bounding Box***
1.jpg	3 0.236328 0.580729 0.123698 0.088542 5 0.329427 0.822917 0.054688 0.036458 5 0.468099 0.170573 0.048177 0.044271
2.jpg	0 0.314453 0.777344 0.045573 0.036458 0 0.743490 0.657552 0.036458 0.033854 0 0.887370 0.792318 0.037760 0.037760 0 0.989583 0.639323 0.018229 0.033854 0 0.791667 0.404297 0.031250 0.032552
3.jpg	2 0.724609 0.249349 0.207031 0.055990
4.jpg	1 0.183594 0.738281 0.179688 0.096354 1 0.529297 0.470052 0.175781 0.091146 1 0.281250 0.061849 0.169271 0.089844
5.jpg	5 0.331380 0.390625 0.097656 0.208333 5 0.350911 0.342448 0.069010 0.143229 5 0.386719 0.343099 0.065104 0.162760 5 0.414714 0.319010 0.076823 0.127604
⋮	⋮
1461.jpg	5 0.037760 0.062500 0.072917 0.039062
1462.jpg	5 0.114583 0.745443 0.085938 0.024740 5 0.214844 0.744141 0.085938 0.016927 5 0.306641 0.740885 0.084635 0.020833 5 0.401693 0.739583 0.095052 0.020833 5 0.421875 0.682292 0.075521 0.023438 5 0.416016 0.695312 0.076823 0.018229 5 0.335938 0.695964 0.088542 0.019531 5 0.200521 0.695964 0.177083 0.029948 5 0.098958 0.629557 0.039062 0.105469 5 0.066406 0.635417 0.036458 0.104167 5 0.065755 0.487630 0.029948 0.069010 1 0.332031 0.248698 0.187500 0.028646 1 0.054036 0.182943 0.042969 0.136719
1463.jpg	1 0.451172 0.132812 0.141927 0.263021
1464.jpg	5 0.718750 0.827474 0.263021 0.076823
1465.jpg	5 0.944010 0.965495 0.052083 0.069010 5 0.912760 0.322917 0.046875 0.072917

Lampiran 3 Hasil Deteksi Kapal

				
1.jpg	2.jpg	3.jpg	4.jpg	5.jpg
				
6.jpg	7.jpg	8.jpg	9.jpg	10.jpg
				
11.jpg	12.jpg	13.jpg	14.jpg	15.jpg
				
16.jpg	17.jpg	18.jpg	19.jpg	20.jpg
⋮	⋮	⋮	⋮	⋮
				
1456.jpg	1457.jpg	1458.jpg	1459.jpg	1460.jpg
				
1461.jpg	1462.jpg	1463.jpg	1464.jpg	1465.jpg

Lampiran 4 Sintaks Eksplorasi Data

```
from skimage import io
import matplotlib.pyplot as plt
```

```
from IPython.display import Image
#Mengimport Gambar
Image(filename='D:/sampel/obj/Image/17.jpg',width=300,height=300
)
#Mendapatkan Histogram
image = io.imread('D:/sampel/obj/Image/17.jpg')
_ = plt.hist(image[:, :, 0].ravel(), bins = 256, color = 'red', alpha = 0.5)
_ = plt.hist(image[:, :, 1].ravel(), bins = 256, color = 'Green', alpha =
0.5)
_ = plt.hist(image[:, :, 2].ravel(), bins = 256, color = 'Blue', alpha =
0.5)
_ = plt.xlabel('Intensity Value')
_ = plt.ylabel('Count')
_ = plt.legend(['Red_Channel', 'Green_Channel', 'Blue_Channel'])
plt.show()
```

Lampiran 5 Sintaks Mengubah Format Data dari YOLO txt ke VOC xml *Python*

```
import os
import xml.etree.cElementTree as ET
from PIL import Image

ANNOTATIONS_DIR_PREFIX = "annotations"

DESTINATION_DIR = "converted_labels"

CLASS_MAPPING = {
   '0': 'name'
   # Add your remaining classes here.
}


def create_root(file_prefix, width, height):
   root = ET.Element("annotations")
   ET.SubElement(root, "filename").text =
"{}.jpg".format(file_prefix)
   ET.SubElement(root, "folder").text = "images"
   size = ET.SubElement(root, "size")
   ET.SubElement(size, "width").text = str(width)
   ET.SubElement(size, "height").text = str(height)
   ET.SubElement(size, "depth").text = "3"
```

```
    return root


def create_object_annotation(root, voc_labels):
    for voc_label in voc_labels:
        obj = ET.SubElement(root, "object")
        ET.SubElement(obj, "name").text = voc_label[0]
        ET.SubElement(obj, "pose").text = "Unspecified"
        ET.SubElement(obj, "truncated").text = str(0)
        ET.SubElement(obj, "difficult").text = str(0)
        bbox = ET.SubElement(obj, "bndbox")
        ET.SubElement(bbox, "xmin").text = str(voc_label[1])
        ET.SubElement(bbox, "ymin").text = str(voc_label[2])
        ET.SubElement(bbox, "xmax").text = str(voc_label[3])
        ET.SubElement(bbox, "ymax").text = str(voc_label[4])
    return root


def create_file(file_prefix, width, height, voc_labels):
    root = create_root(file_prefix, width, height)
    root = create_object_annotation(root, voc_labels)
    tree = ET.ElementTree(root)
    tree.write("{}/{}.xml".format(DESTINATION_DIR, file_prefix))


def read_file(file_path):
    file_prefix = file_path.split(".txt")[0]
    image_file_name = "{}.jpg".format(file_prefix)
    img = Image.open("{}/{}".format("images", image_file_name))
    w, h = img.size
    with open(file_path, 'r') as file:
        lines = file.readlines()
        voc_labels = []
        for line in lines:
            voc = []
            line = line.strip()
            data = line.split()
            voc.append(CLASS_MAPPING.get(data[0]))
            bbox_width = float(data[3]) * w
            bbox_height = float(data[4]) * h
            center_x = float(data[1]) * w
```

```
            center_y = float(data[2]) * h
            voc.append(center_x - (bbox_width / 2))
            voc.append(center_y - (bbox_height / 2))
            voc.append(center_x + (bbox_width / 2))
            voc.append(center_y + (bbox_height / 2))
            voc_labels.append(voc)
        create_file(file_prefix, w, h, voc_labels)
    print("Processing complete for file: {}".format(file_path))


def start(dir_name):
    if not os.path.exists(DESTINATION_DIR):
        os.makedirs(DESTINATION_DIR)
    for filename in os.listdir(ANNOTATIONS_DIR_PREFIX):
        if filename.endswith('txt'):
            read_file(filename)
        else:
            print("Skipping file: {}".format(filename))


if __name__ == "__main__":
    start()
```

Lampiran 6 Sintaks Mengubah Format Data dari pascal VOC xml ke COCO json *Python*

```
import sys
import os
import json
import xml.etree.ElementTree as ET
import glob

START_BOUNDING_BOX_ID = 0
PRE_DEFINE_CATEGORIES = None

def get(root, name):
    vars = root.findall(name)
    return vars


def get_and_check(root, name, length):
    vars = root.findall(name)
```

```
    if len(vars) == 0:
        raise ValueError("Can not find %s in %s." % (name, root.tag))
    if length > 0 and len(vars) != length:
        raise ValueError(
            "The size of %s is supposed to be %d, but is %d."
            % (name, length, len(vars))
        )
    if length == 1:
        vars = vars[0]
    return vars


def get_filename_as_int(filename):
    try:
        filename = filename.replace("\\", "/")
        filename = os.path.splitext(os.path.basename(filename))[0]
        return int(filename)
    except:
        raise ValueError("Filename %s is supposed to be an integer." %
(filename))


def get_categories(xml_files):
    """Generate category name to id mapping from a list of xml files.

    Arguments:
        xml_files {list} -- A list of xml file paths.

    Returns:
        dict -- category name to id mapping.
    """
    classes_names = []
    for xml_file in xml_files:
        tree = ET.parse(xml_file)
        root = tree.getroot()
        for member in root.findall("object"):
            classes_names.append(member[0].text)
    classes_names = list(set(classes_names))
    classes_names.sort()
    return {name: i for i, name in enumerate(classes_names)}
```

```
def convert(xml_files, json_file):
    json_dict = {"images": [], "type": "instances", "annotations": [],
"categories": []}
    if PRE_DEFINE_CATEGORIES is not None:
        categories = PRE_DEFINE_CATEGORIES
    else:
        categories = get_categories(xml_files)
    bnd_id = START_BOUNDING_BOX_ID
    for xml_file in xml_files:
        tree = ET.parse(xml_file)
        root = tree.getroot()
        path = get(root, "path")
        if len(path) == 1:
            filename = os.path.basename(path[0].text)
        elif len(path) == 0:
            filename = get_and_check(root, "filename", 1).text
        else:
            raise ValueError("%d paths found in %s" % (len(path),
xml_file))
        ## The filename must be a number
        image_id = get_filename_as_int(filename)
        size = get_and_check(root, "size", 1)
        width = int(get_and_check(size, "width", 1).text)
        height = int(get_and_check(size, "height", 1).text)
        image = {
            "file_name": filename,
            "height": height,
            "width": width,
            "id": image_id,
        }
        json_dict["images"].append(image)
        ## Currently we do not support segmentation.
        #  segmented = get_and_check(root, 'segmented', 1).text
        #  assert segmented == '0'
        for obj in get(root, "object"):
            category = get_and_check(obj, "name", 1).text
            if category not in categories:
                new_id = len(categories)
                categories[category] = new_id
            category_id = categories[category]
```

```
            bndbox = get_and_check(obj, "bndbox", 1)
            xmin = int(get_and_check(bndbox, "xmin", 1).text) - 1
            ymin = int(get_and_check(bndbox, "ymin", 1).text) - 1
            xmax = int(get_and_check(bndbox, "xmax", 1).text)
            ymax = int(get_and_check(bndbox, "ymax", 1).text)
            assert xmax > xmin
            assert ymax > ymin
            o_width = abs(xmax - xmin)
            o_height = abs(ymax - ymin)
            ann = {
                "area": o_width * o_height,
                "iscrowd": 0,
                "image_id": image_id,
                "bbox": [xmin, ymin, o_width, o_height],
                "category_id": category_id,
                "id": bnd_id,
                "ignore": 0,
                "segmentation": [],
            }
            json_dict["annotations"].append(ann)
            bnd_id = bnd_id + 1

    for cate, cid in categories.items():
        cat = {"supercategory": "none", "id": cid, "name": cate}
        json_dict["categories"].append(cat)

    os.makedirs(os.path.dirname(json_file), exist_ok=True)
    json_fp = open(json_file, "w")
    json_str = json.dumps(json_dict)
    json_fp.write(json_str)
    json_fp.close()


if __name__ == "__main__":
    import argparse

    parser = argparse.ArgumentParser(
        description="Convert Pascal VOC annotation to COCO format."
    )
    parser.add_argument("xml_dir", help="Directory path to xml
files.", type=str)
```

```
    parser.add_argument("json_file", help="Output COCO format json
file.", type=str)
    args = parser.parse_args()
    xml_files = glob.glob(os.path.join(args.xml_dir, "*.xml"))

    # If you want to do train/test split, you can pass a subset of xml
files to convert function.
    print("Number of xml files: {}".format(len(xml_files)))
    convert(xml_files, args.json_file)
    print("Success: {}".format(args.json_file))
```

Lampiran 7 Sintaks Mengubah Format Data dari pascal VOC xml ke csv *Python*

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


def xml_to_csv(path):
    xml_list = []
    for xml_file in glob.glob(path + '/*.xml'):
        tree = ET.parse(xml_file)
        root = tree.getroot()
        for member in root.findall('object'):
            bbx = member.find('bndbox')
            xmin = int(bbx.find('xmin').text)
            ymin = int(bbx.find('ymin').text)
            xmax = int(bbx.find('xmax').text)
            ymax = int(bbx.find('ymax').text)
            label = member.find('name').text

            value = (root.find('filename').text,
                    int(root.find('size')[0].text),
                    int(root.find('size')[1].text),
                    label,
                    xmin,
                    ymin,
                    xmax,
                    ymax
                    )
```

```
            xml_list.append(value)
    column_name = ['filename', 'width', 'height',
                   'class', 'xmin', 'ymin', 'xmax', 'ymax']
    xml_df = pd.DataFrame(xml_list, columns=column_name)
    return xml_df


def main():
    datasets = ['train', 'dev', 'test']
    for ds in datasets:
        image_path = os.path.join(os.getcwd(), ds, 'annotations')
        xml_df = xml_to_csv(image_path)
        xml_df.to_csv('labels_{}.csv'.format(ds), index=None)
        print('Successfully converted xml to csv.')


main()
```

Lampiran 8 Sintaks *K-means Anchor Box Python*

Import Module
import json import os import math import matplotlib.pyplot as plt import numpy as np import seaborn as sns;
Setting Directory Input

sns.set() # for plot styling
os.chdir("D:/KmeanAnchorBox/bdd100k")
#1
#file_json = open('D:/sampel/data/coco/output1.json')
 	
#data = json.loads(file_json.read())
#for anotasi in data['annotations']:
# print (anotasi) 	
 	
*train*_path = "D:/sampel/data/coco/output4.json"
 	
with open(*train*_path,"r") as ftr:
    trlabel = json.load(ftr)
    anotasi = trlabel["annotations"]

```
    w, h = [], []
    for ind2 in range(len(anotasi)):
        try:
            a = anotasi[ind2]["bbox"]  # traffic sign
            x1, y1, x2, y2 = list(a.values())
            width = abs(x1 - x2)
            height = abs(y1 - y2)
            w.append(width)
            h.append(height)
        except:
            pass
w = np.asarray(w)
h = np.asarray(h)

x = [w, h]
x = np.asarray(x)
x = x.transpose()
```

K-means Output

```
##########################################   K- Means
##########################################

from sklearn.cluster import K-means

k-means3 = K-means(n_clusters=9)
k-means3.fit(x)
y_k-means3 = k-means3.predict(x)

##########################################
centers3 = k-means3.cluster_centers_

yolo_anchor_average = []
for ind in range(9):
    yolo_anchor_average.append(np.mean(x[y_k-means3 == ind],
axis=0))

yolo_anchor_average = np.array(yolo_anchor_average)

plt.scatter(x[:, 0], x[:, 1], c=y_k-means3, s=2, cmap='viridis')
plt.scatter(yolo_anchor_average[:, 0], yolo_anchor_average[:, 1],
c='red', s=50);
yoloV3anchors = yolo_anchor_average
```

```
yoloV3anchors[:, 0] = yolo_anchor_average[:, 0] / 1280 * 608
yoloV3anchors[:, 1] = yolo_anchor_average[:, 1] / 720 * 608
yoloV3anchors = np.rint(yoloV3anchors)
fig, ax = plt.subplots()
for ind in range(9):
    rectangle = plt.Rectangle((304 - yoloV3anchors[ind, 0] / 2, 304 -
yoloV3anchors[ind, 1] / 2), yoloV3anchors[ind, 0],
                              yoloV3anchors[ind, 1], fc='b', edgecolor='b',
fill=None)
    ax.add_patch(rectangle)
ax.set_aspect(1.0)
plt.axis([0, 608, 0, 608])
plt.show()
yoloV3anchors.sort(axis=0)
print("Your custom anchor boxes are {}".format(yoloV3anchors))

F = open("YOLOV3_BDD_Anchors.txt", "w")
F.write("{}".format(yoloV3anchors))
F.close()
```

Lampiran 9 Sintaks Pelatihan Model YOLO-CNN Google Colab

Cloning dan *Building* Darknet
clone darknet repo !git clone https://github.com/AlexeyAB/darknet # change makefile to have GPU and OPENCV enabled %cd darknet !sed -i 's/OPENCV=0/OPENCV=1/' Makefile !sed -i 's/GPU=0/GPU=1/' Makefile !sed -i 's/CUDNN=0/CUDNN=1/' Makefile # verify CUDA !/usr/local/cuda/bin/nvcc –version # make darknet (build) !make
Download pretrained YOLOv3 weights
get yolov3 pre*train*ed coco dataset *weight*s !wget https://pjreddie.com/media/files/yolov3.*weight*s

Import Module

# define helper functions 	
def imShow(path):
  import cv2
  import matplotlib.pyplot as plt
  %matplotlib inline
 	
  image = cv2.imread(path)
  height, width = image.shape[:2]
  resized_image = cv2.resize(image,(3*width, 3*height), interpolation = cv2.INTER_CUBIC)
 	
  fig = plt.gcf()
  fig.set_size_inches(18, 10)
  plt.axis("off")
  plt.imshow(cv2.cvtColor(resized_image, cv2.COLOR_BGR2RGB))
  plt.show()
 	
# use this to upload files 	
def upload():
  from google.colab import files
  uploaded = files.upload()
  for name, data in uploaded.items():
    with open(name, 'wb') as f:
      f.write(data)
      print ('saved file', name)
 	
# use this to download a file 	
def download(path):
  from google.colab import files
  files.download(path)
 	
# This stops 'Run all' at this cell by causing an error 	
assert False

Menghubungkan Google Drive dengan Google Colab
%cd .. from google.colab import drive drive.mount('/content/gdrive') # this creates a symbolic link so that now the path /content/gdrive/My\ Drive/ is equal to /mydrive !ln -s /content/gdrive/My\ Drive/ /mydrive

```
!ls /mydrive
# LOCAL MACHINE DOWNLOAD
# if you get an error first run then run it again and it should work
download('predictions.jpg')

# GOOGLE DRIVE DOWNLOAD
# note that I can change what the image name is saved as (I am saving
it as detection1.jpg)
!cp predictions.jpg /mydrive/images/detection1.jpg
```

Mengkonfigurasi File Training YOLOv3

```
# this is where my zip is stored (I created a yolov3 folder where I will
get my required files from)
!ls /mydrive/yolov3

# copy the .zip file into the root directory of cloud VM
!cp /mydrive/yolov3/obj.zip ../

# unzip the zip file and its contents should now be in /darknet/data/obj
!unzip ../obj.zip -d data/

# download cfg to google drive and change its name
!cp cfg/yolov3.cfg /mydrive/yolov3/yolov3_custom50.cfg

# to download to local machine (change its name to
yolov3_custom.cfg once you download)
download('cfg/yolov3.cfg')

# upload the custom .cfg back to cloud VM from Google Drive
!cp /mydrive/yolov3/yolov3_custom.cfg ./cfg

# upload the custom .cfg back to cloud VM from local machine
(uncomment to use)
#%cd cfg
#upload()
#%cd ..

# upload the obj.names and obj.data files to cloud VM from Google
Drive
!cp /mydrive/yolov3/obj.names ./data
!cp /mydrive/yolov3/obj.data  ./data
```

```
# upload the obj.names and obj.data files to cloud VM from local
machine (uncomment to use)
#%cd data
#upload()
#%cd ..

# upload the generate_train.py script to cloud VM from Google Drive
!cp /mydrive/yolov3/generate_train.py ./

# upload the generate_train.py script to cloud VM from local machine
(uncomment to use)
#upload()

!python generate_train.py

# verify train.txt can be seen in our darknet/data folder
!ls data/
```

Upload Weights Training ke Google Colab

```
# upload pretrained convolutional layer weights
!wget http://pjreddie.com/media/files/darknet53.conv.74
```

Train YOLOv3

```
# train your custom detector
!./darknet detector train data/obj.data cfg/yolov3_custom.cfg
darknet53.conv.74 -dont_show
```

Melanjutkan Train YOLOv3 di Google Colab yang Berhenti

```
!./darknet detector train data/obj.data cfg/yolov3_custom.cfg
/mydrive/yolov3/backup/yolov3_custom_last.weights -dont_show
```

Menampilkan *Loss Chart*

```
imShow('chart.png')
```

Mendapatkan *Output* Objek Deteksi dalam txt File

```
# need to set our custom cfg to test mode
%cd cfg
!sed -i 's/batch=64/batch=1/' yolov3_custom.cfg
!sed -i 's/subdivisions=16/subdivisions=1/' yolov3_custom.cfg
%cd ..
```

!./darknet detector test data/obj.data cfg/yolov3_custom.cfg /mydrive/yolov3/backup/yolov3_custom_last.*weight*s -dont_show -ext_output < data/*train*.txt > resulttest.txt -thresh 0.1
Mendapatkan *Output* Objek Deteksi dalam jpg File
1. The first step is to clone the repository. The link is: https://github.com/vincentgong7/VG_AlexeyAB_darknet.git """ !git clone https://github.com/vincentgong7/VG_AlexeyAB_darknet.git """**2. Next, we move to the weight folder and decompress the weight file.**""" cd VG_AlexeyAB_darknet cd weights/ """***We use the 7z command to decompress the weight file.***""" !7z x yolov4.weights.7z.001 """***Come back to the repository root folder.***""" cd .. """**3. "make"**""" !make """**4. Run the sample command** After that, you can check the exported files in folder: ***VG_AlexeyAB_darknet/output/*** """ !./darknet detector batch cfg/obj.data cfg/yolov4-obj.cfg weights/yolov4-obj_final.weights io_folder sample_imgs/ output/ -out output/result.json -ext_output > output/result.txt

```
!zip -r "/content/VG_AlexeyAB_darknet/output/output.zip"
"/content/VG_AlexeyAB_darknet/output"
```

Lampiran 10 Sintaks Mengubah Format Data dari *Output* YOLO Menjadi Partisi Beberapa File txt *Python*

Import Module

```
import os
import re
```

Setting *Directory* dan *File Input*

```
# make sure that the cwd() in the beginning is the location of the
python script (so that every path makes sense)
os.chdir(os.path.dirname(os.path.abspath(__file__)))

IN_FILE = 'D:/mAP-master/defaultanchorbox/resulttestDef01.txt'


# change directory to the one with the files to be changed
parent_path = os.path.abspath(os.path.join(os.getcwd(), os.pardir))
parent_path = os.path.abspath(os.path.join(parent_path, os.pardir))
DR_PATH = os.path.join(parent_path, 'input','detection-
resultstestDef01')
#print(DR_PATH)
os.chdir(DR_PATH)

SEPARATOR_KEY = 'Enter Image Path:'
IMG_FORMAT = '.jpg'
```

Generate Output Partisi File

```
outfile = None
with open(IN_FILE) as infile:
    for line in infile:
        if SEPARATOR_KEY in line:
            if IMG_FORMAT not in line:
                break
            # get text between two substrings (SEPARATOR_KEY and
IMG_FORMAT)
            image_path = re.search(SEPARATOR_KEY + '(.*)' +
IMG_FORMAT, line)
            # get the image name (the final component of a image_path)
            # e.g., from 'data/horses_1' to 'horses_1'
```

```
            image_name = os.path.basename(image_path.group(1))
            # close the previous file
            if outfile is not None:
                outfile.close()
            # open a new file
            outfile = open(os.path.join(DR_PATH, image_name + '.txt'),
'w')
        elif outfile is not None:
            # split line on first occurrence of the character ':' and '%'
            class_name, info = line.split(':', 1)
            confidence, bbox = info.split('%', 1)
            # get all the coordinates of the bounding box
            bbox = bbox.replace(')','') # remove the character ')'
            # go through each of the parts of the string and check if it is a
digit
            left, top, width, height = [int(s) for s in bbox.split() if s.lstrip('-
').isdigit()]
            right = left + width
            bottom = top + height
            outfile.write("{} {} {} {} {} {}\n".format(class_name,
float(confidence)/100, left, top, right, bottom))
```

Lampiran 11 Sintaks Mendapatkan Nilai Evaluasi mAP *Python*

Import Module
import glob import json import os import shutil import operator import sys import argparse import math import numpy as np
Setting Directory input
MINOVERLAP = 0.5 # *default* value (defined in the PASCAL VOC2012 challenge) parser = argparse.ArgumentParser() parser.add_argument('-na', '--no-animation', help="no animation is shown.", action="store_true")

```
parser.add_argument('-np', '--no-plot', help="no plot is shown.",
action="store_true")
parser.add_argument('-q', '--quiet', help="minimalistic console
output.", action="store_true")
# argparse receiving list of classes to be ignored (e.g., python main.py
--ignore person book)
parser.add_argument('-i', '--ignore', nargs='+', type=str, help="ignore a
list of classes.")
# argparse receiving list of classes with specific IoU (e.g., python
main.py --set-class-iou person 0.7)
parser.add_argument('--set-class-iou', nargs='+', type=str, help="set
IoU for a specific class.")
args = parser.parse_args()

'''
    0,0 ------> x (width)
     |
     |  (Left,Top)
     |      *_________
     |     |         |
           |         |
     y     |_________|
  (height)            *
                (Right,Bottom)
'''

# if there are no classes to ignore then replace None by empty list
if args.ignore is None:
    args.ignore = []

specific_iou_flagged = False
if args.set_class_iou is not None:
    specific_iou_flagged = True

# make sure that the cwd() is the location of the python script (so that
every path makes sense)
os.chdir(os.path.dirname(os.path.abspath(__file__)))

GT_PATH = os.path.join(os.getcwd(), 'input', 'ground-truth-test')
DR_PATH = os.path.join(os.getcwd(), 'input', 'detection-
resultstestDef01')
```

```
# if there are no images then no animation can be shown
IMG_PATH = os.path.join(os.getcwd(), 'input', 'images-optional-test')
if os.path.exists(IMG_PATH):
    for dirpath, dirnames, files in os.walk(IMG_PATH):
        if not files:
            # no image files found
            args.no_animation = True
else:
    args.no_animation = True
```

Generate OpenCV dan Matplotlib

```
# try to import OpenCV if the user didn't choose the option --no-
animation
show_animation = False
if not args.no_animation:
    try:
        import cv2
        show_animation = True
    except ImportError:
        print("\"opencv-python\" not found, please install to visualize the
results.")
        args.no_animation = True

# try to import Matplotlib if the user didn't choose the option --no-plot
draw_plot = False
if not args.no_plot:
    try:
        import matplotlib.pyplot as plt
        draw_plot = True
    except ImportError:
        print("\"matplotlib\" not found, please install it to get the
resulting plots.")
        args.no_plot = True
```

Menghitung *Log Average Miss Rate*

```
def log_average_miss_rate(prec, rec, num_images):
    """
        log-average miss rate:
            Calculated by averaging miss rates at 9 evenly spaced FPPI
points
            between 10e-2 and 10e0, in log-space.
```

```
        output:
                lamr | log-average miss rate
                mr | miss rate
                fppi | false positives per image

        references:
            [1] Dollar, Piotr, et al. "Pedestrian Detection: An Evaluation
of the
                State of the Art." Pattern Analysis and Machine Intelligence,
IEEE
                Transactions on 34.4 (2012): 743 - 761.
    """

    # if there were no detections of that class
    if prec.size == 0:
        lamr = 0
        mr = 1
        fppi = 0
        return lamr, mr, fppi

    fppi = (1 - prec)
    mr = (1 - rec)

    fppi_tmp = np.insert(fppi, 0, -1.0)
    mr_tmp = np.insert(mr, 0, 1.0)

    # Use 9 evenly spaced reference points in log-space
    ref = np.logspace(-2.0, 0.0, num = 9)
    for i, ref_i in enumerate(ref):
        # np.where() will always find at least 1 index, since min(ref) =
0.01 and min(fppi_tmp) = -1.0
        j = np.where(fppi_tmp <= ref_i)[-1][-1]
        ref[i] = mr_tmp[j]

    # log(0) is undefined, so we use the np.maximum(1e-10, ref)
    lamr = math.exp(np.mean(np.log(np.maximum(1e-10, ref))))

    return lamr, mr, fppi

"""
 throw error and exit
```

```
"""
def error(msg):
    print(msg)
    sys.exit(0)

"""
 check if the number is a float between 0.0 and 1.0
"""
def is_float_between_0_and_1(value):
    try:
        val = float(value)
        if val > 0.0 and val < 1.0:
            return True
        else:
            return False
    except ValueError:
        return False
```

Menghitung *Mean Average Precision*

```
"""
 Calculate the AP given the recall and precision array
    1st) We compute a version of the measured precision/recall curve
with
         precision monotonically decreasing
    2nd) We compute the AP as the area under this curve by numerical
integration.
"""
def voc_ap(rec, prec):
    """
    --- Official matlab code VOC2012---
    mrec=[0 ; rec ; 1];
    mpre=[0 ; prec ; 0];
    for i=numel(mpre)-1:-1:1
            mpre(i)=max(mpre(i),mpre(i+1));
    end
    i=find(mrec(2:end)~=mrec(1:end-1))+1;
    ap=sum((mrec(i)-mrec(i-1)).*mpre(i));
    """
    rec.insert(0, 0.0) # insert 0.0 at begining of list
    rec.append(1.0) # insert 1.0 at end of list
    mrec = rec[:]
    prec.insert(0, 0.0) # insert 0.0 at begining of list
```

```
    prec.append(0.0) # insert 0.0 at end of list
    mpre = prec[:]
    """
     This part makes the precision monotonically decreasing
        (goes from the end to the beginning)
        matlab: for i=numel(mpre)-1:-1:1
                    mpre(i)=max(mpre(i),mpre(i+1));
    """
    # matlab indexes start in 1 but python in 0, so I have to do:
    #     range(start=(len(mpre) - 2), end=0, step=-1)
    # also the python function range excludes the end, resulting in:
    #     range(start=(len(mpre) - 2), end=-1, step=-1)
    for i in range(len(mpre)-2, -1, -1):
        mpre[i] = max(mpre[i], mpre[i+1])
    """
     This part creates a list of indexes where the recall changes
        matlab: i=find(mrec(2:end)~=mrec(1:end-1))+1;
    """
    i_list = []
    for i in range(1, len(mrec)):
        if mrec[i] != mrec[i-1]:
            i_list.append(i) # if it was matlab would be i + 1
    """
     The Average Precision (AP) is the area under the curve
        (numerical integration)
        matlab: ap=sum((mrec(i)-mrec(i-1)).*mpre(i));
    """
    ap = 0.0
    for i in i_list:
        ap += ((mrec[i]-mrec[i-1])*mpre[i])
    return ap, mrec, mpre


"""
 Convert the lines of a file to a list
"""
def file_lines_to_list(path):
    # open txt file lines to a list
    with open(path) as f:
        content = f.readlines()
    # remove whitespace characters like `\n` at the end of each line
```

```
    content = [x.strip() for x in content]
    return content

"""
 Draws text in image
"""
def draw_text_in_image(img, text, pos, color, line_width):
    font = cv2.FONT_HERSHEY_PLAIN
    fontScale = 1
    lineType = 1
    bottomLeftCornerOfText = pos
    cv2.putText(img, text,
            bottomLeftCornerOfText,
            font,
            fontScale,
            color,
            lineType)
    text_width, _ = cv2.getTextSize(text, font, fontScale, lineType)[0]
    return img, (line_width + text_width)

"""
 Plot - adjust axes
"""
def adjust_axes(r, t, fig, axes):
    # get text width for re-scaling
    bb = t.get_window_extent(renderer=r)
    text_width_inches = bb.width / fig.dpi
    # get axis width in inches
    current_fig_width = fig.get_figwidth()
    new_fig_width = current_fig_width + text_width_inches
    propotion = new_fig_width / current_fig_width
    # get axis limit
    x_lim = axes.get_xlim()
    axes.set_xlim([x_lim[0], x_lim[1]*propotion])

"""
 Draw plot using Matplotlib
"""
def draw_plot_func(dictionary, n_classes, window_title, plot_title,
x_label, output_path, to_show, plot_color, true_p_bar):
    # sort the dictionary by decreasing value, into a list of tuples
```

```
    sorted_dic_by_value = sorted(dictionary.items(),
key=operator.itemgetter(1))
    # unpacking the list of tuples into two lists
    sorted_keys, sorted_values = zip(*sorted_dic_by_value)
    #
    if true_p_bar != "":
        """
         Special case to draw in:
            - green -> TP: True Positives (object detected and matches
ground-truth)
            - red -> FP: False Positives (object detected but does not
match ground-truth)
            - pink -> FN: False Negatives (object not detected but present
in the ground-truth)
        """
        fp_sorted = []
        tp_sorted = []
        for key in sorted_keys:
            fp_sorted.append(dictionary[key] - true_p_bar[key])
            tp_sorted.append(true_p_bar[key])
        plt.barh(range(n_classes), fp_sorted, align='center',
color='crimson', label='False Positive')
        plt.barh(range(n_classes), tp_sorted, align='center',
color='forestgreen', label='True Positive', left=fp_sorted)
        # add legend
        plt.legend(loc='lower right')
        """
         Write number on side of bar
        """
        fig = plt.gcf() # gcf - get current figure
        axes = plt.gca()
        r = fig.canvas.get_renderer()
        for i, val in enumerate(sorted_values):
            fp_val = fp_sorted[i]
            tp_val = tp_sorted[i]
            fp_str_val = " " + str(fp_val)
            tp_str_val = fp_str_val + " " + str(tp_val)
            # trick to paint multicolor with offset:
            # first paint everything and then repaint the first number
            t = plt.text(val, i, tp_str_val, color='forestgreen', va='center',
fontweight='bold')
```

```
            plt.text(val, i, fp_str_val, color='crimson', va='center',
fontweight='bold')
            if i == (len(sorted_values)-1): # largest bar
                adjust_axes(r, t, fig, axes)
    else:
        plt.barh(range(n_classes), sorted_values, color=plot_color)
        """
         Write number on side of bar
        """
        fig = plt.gcf() # gcf - get current figure
        axes = plt.gca()
        r = fig.canvas.get_renderer()
        for i, val in enumerate(sorted_values):
            str_val = " " + str(val) # add a space before
            if val < 1.0:
                str_val = " {0:.2f}".format(val)
            t = plt.text(val, i, str_val, color=plot_color, va='center',
fontweight='bold')
            # re-set axes to show number inside the figure
            if i == (len(sorted_values)-1): # largest bar
                adjust_axes(r, t, fig, axes)
    # set window title
    fig.canvas.set_window_title(window_title)
    # write classes in y axis
    tick_font_size = 12
    plt.yticks(range(n_classes), sorted_keys, fontsize=tick_font_size)
    """
     Re-scale height accordingly
    """
    init_height = fig.get_figheight()
    # comput the matrix height in points and inches
    dpi = fig.dpi
    height_pt = n_classes * (tick_font_size * 1.4) # 1.4 (some spacing)
    height_in = height_pt / dpi
    # compute the required figure height
    top_margin = 0.15 # in percentage of the figure height
    bottom_margin = 0.05 # in percentage of the figure height
    figure_height = height_in / (1 - top_margin - bottom_margin)
    # set new height
    if figure_height > init_height:
        fig.set_figheight(figure_height)
```

```
    # set plot title
    plt.title(plot_title, fontsize=14)
    # set axis titles
    # plt.xlabel('classes')
    plt.xlabel(x_label, fontsize='large')
    # adjust size of window
    fig.tight_layout()
    # save the plot
    fig.savefig(output_path)
    # show image
    if to_show:
        plt.show()
    # close the plot
    plt.close()
```

Setting Directory Output

```
"""
 Create a ".temp_files/" and "output/" directory
"""
TEMP_FILES_PATH = ".temp_files"
if not os.path.exists(TEMP_FILES_PATH): # if it doesn't exist
already
    os.makedirs(TEMP_FILES_PATH)
output_files_path = "outputtestDef01"
if os.path.exists(output_files_path): # if it exist already
    # reset the output directory
    shutil.rmtree(output_files_path)

os.makedirs(output_files_path)
if draw_plot:
    os.makedirs(os.path.join(output_files_path, "classes"))
if show_animation:
    os.makedirs(os.path.join(output_files_path, "images",
"detections_one_by_one"))

"""
 ground-truth
     Load each of the ground-truth files into a temporary ".json" file.
     Create a list of all the class names present in the ground-truth
(gt_classes).
```

```
"""
# get a list with the ground-truth files
ground_truth_files_list = glob.glob(GT_PATH + '/*.txt')
if len(ground_truth_files_list) == 0:
    error("Error: No ground-truth files found!")
ground_truth_files_list.sort()
# dictionary with counter per class
gt_counter_per_class = {}
counter_images_per_class = {}

gt_files = []
for txt_file in ground_truth_files_list:
    #print(txt_file)
    file_id = txt_file.split(".txt", 1)[0]
    file_id = os.path.basename(os.path.normpath(file_id))
    # check if there is a correspondent detection-results file
    temp_path = os.path.join(DR_PATH, (file_id + ".txt"))
    if not os.path.exists(temp_path):
        error_msg = "Error. File not found: {}\n".format(temp_path)
        error_msg += "(You can avoid this error message by running
extra/intersect-gt-and-dr.py)"
        error(error_msg)
    lines_list = file_lines_to_list(txt_file)
    # create ground-truth dictionary
    bounding_boxes = []
    is_difficult = False
    already_seen_classes = []
    for line in lines_list:
        try:
            if "difficult" in line:
                class_name, left, top, right, bottom, _difficult =
line.split()
                is_difficult = True
            else:
                class_name, left, top, right, bottom = line.split()
        except ValueError:
            error_msg = "Error: File " + txt_file + " in the wrong
format.\n"
            error_msg += " Expected:
<class_name><left><top><right><bottom> ['difficult']\n"
            error_msg += " Received: " + line
```

```
            error_msg += "\n\nIf you have a <class_name> with spaces
between words you should remove them\n"
            error_msg += "by running the script \"remove_space.py\" or
\"rename_class.py\" in the \"extra/\" folder."
            error(error_msg)
        # check if class is in the ignore list, if yes skip
        if class_name in args.ignore:
            continue
        bbox = left + " " + top + " " + right + " " +bottom
        if is_difficult:
            bounding_boxes.append({"class_name":class_name,
"bbox":bbox, "used":False, "difficult":True})
            is_difficult = False
        else:
            bounding_boxes.append({"class_name":class_name,
"bbox":bbox, "used":False})
            # count that object
            if class_name in gt_counter_per_class:
                gt_counter_per_class[class_name] += 1
            else:
                # if class didn't exist yet
                gt_counter_per_class[class_name] = 1

            if class_name not in already_seen_classes:
                if class_name in counter_images_per_class:
                    counter_images_per_class[class_name] += 1
                else:
                    # if class didn't exist yet
                    counter_images_per_class[class_name] = 1
                already_seen_classes.append(class_name)


    # dump bounding_boxes into a ".json" file
    new_temp_file = TEMP_FILES_PATH + "/" + file_id +
"_ground_truth.json"
    gt_files.append(new_temp_file)
    with open(new_temp_file, 'w') as outfile:
        json.dump(bounding_boxes, outfile)

gt_classes = list(gt_counter_per_class.keys())
# let's sort the classes alphabetically
```

```
gt_classes = sorted(gt_classes)
n_classes = len(gt_classes)
#print(gt_classes)
#print(gt_counter_per_class)

"""
 Check format of the flag --set-class-iou (if used)
    e.g. check if class exists
"""
if specific_iou_flagged:
    n_args = len(args.set_class_iou)
    error_msg = \
        '\n --set-class-iou [class_1] [IoU_1] [class_2] [IoU_2] [...]'
    if n_args % 2 != 0:
        error('Error, missing arguments. Flag usage:' + error_msg)
    # [class_1] [IoU_1] [class_2] [IoU_2]
    # specific_iou_classes = ['class_1', 'class_2']
    specific_iou_classes = args.set_class_iou[::2] # even
    # iou_list = ['IoU_1', 'IoU_2']
    iou_list = args.set_class_iou[1::2] # odd
    if len(specific_iou_classes) != len(iou_list):
        error('Error, missing arguments. Flag usage:' + error_msg)
    for tmp_class in specific_iou_classes:
        if tmp_class not in gt_classes:
                error('Error, unknown class \'" + tmp_class + '\". Flag
usage:' + error_msg)
    for num in iou_list:
        if not is_float_between_0_and_1(num):
            error('Error, IoU must be between 0.0 and 1.0. Flag usage:' +
error_msg)

"""
 detection-results
     Load each of the detection-results files into a temporary ".json"
file.
"""
# get a list with the detection-results files
dr_files_list = glob.glob(DR_PATH + '/*.txt')
dr_files_list.sort()

for class_index, class_name in enumerate(gt_classes):
```

```
    bounding_boxes = []
    for txt_file in dr_files_list:
        #print(txt_file)
        # the first time it checks if all the corresponding ground-truth
files exist
        file_id = txt_file.split(".txt",1)[0]
        file_id = os.path.basename(os.path.normpath(file_id))
        temp_path = os.path.join(GT_PATH, (file_id + ".txt"))
        if class_index == 0:
            if not os.path.exists(temp_path):
                error_msg = "Error. File not found:
{}\n".format(temp_path)
                error_msg += "(You can avoid this error message by
running extra/intersect-gt-and-dr.py)"
                error(error_msg)
        lines = file_lines_to_list(txt_file)
        for line in lines:
            try:
                tmp_class_name, confidence, left, top, right, bottom =
line.split()
            except ValueError:
                error_msg = "Error: File " + txt_file + " in the wrong
format.\n"
                error_msg += " Expected:
<class_name><confidence><left><top><right><bottom>\n"
                error_msg += " Received: " + line
                error(error_msg)
            if tmp_class_name == class_name:
                #print("match")
                bbox = left + " " + top + " " + right + " " +bottom
                bounding_boxes.append({"confidence":confidence,
"file_id":file_id, "bbox":bbox})
                #print(bounding_boxes)
    # sort detection-results by decreasing confidence
    bounding_boxes.sort(key=lambda x:float(x['confidence']),
reverse=True)
    with open(TEMP_FILES_PATH + "/" + class_name + "_dr.json",
'w') as outfile:
        json.dump(bounding_boxes, outfile)
```

Menghitung Average Precision Setiap Kelas

```
"""
 Calculate the AP for each class
"""
sum_AP = 0.0
ap_dictionary = {}
lamr_dictionary = {}
# open file to store the output
with open(output_files_path + "/output.txt", 'w') as output_file:
    output_file.write("# AP and precision/recall per class\n")
    count_true_positives = {}
    for class_index, class_name in enumerate(gt_classes):
        count_true_positives[class_name] = 0
        """
         Load detection-results of that class
        """
        dr_file = TEMP_FILES_PATH + "/" + class_name + "_dr.json"
        dr_data = json.load(open(dr_file))

        """
         Assign detection-results to ground-truth objects
        """
        nd = len(dr_data)
        tp = [0] * nd # creates an array of zeros of size nd
        fp = [0] * nd
        for idx, detection in enumerate(dr_data):
            file_id = detection["file_id"]
            if show_animation:
                # find ground truth image
                ground_truth_img = glob.glob1(IMG_PATH, file_id + ".*")
                #tifCounter = len(glob.glob1(myPath,"*.tif"))
                if len(ground_truth_img) == 0:
                    error("Error. Image not found with id: " + file_id)
                elif len(ground_truth_img) > 1:
                    error("Error. Multiple image with id: " + file_id)
                else: # found image
                    #print(IMG_PATH + "/" + ground_truth_img[0])
                    # Load image
                    img = cv2.imread(IMG_PATH + "/" +
ground_truth_img[0])
                    # load image with draws of multiple detections
```

```
                img_cumulative_path = output_files_path + "/images/" +
ground_truth_img[0]
                if os.path.isfile(img_cumulative_path):
                    img_cumulative = cv2.imread(img_cumulative_path)
                else:
                    img_cumulative = img.copy()
                # Add bottom border to image
                bottom_border = 60
                BLACK = [0, 0, 0]
                img = cv2.copyMakeBorder(img, 0, bottom_border, 0, 0,
cv2.BORDER_CONSTANT, value=BLACK)
            # assign detection-results to ground truth object if any
            # open ground-truth with that file_id
            gt_file = TEMP_FILES_PATH + "/" + file_id +
"_ground_truth.json"
            ground_truth_data = json.load(open(gt_file))
            ovmax = -1
            gt_match = -1
            # load detected object bounding-box
            bb = [ float(x) for x in detection["bbox"].split() ]
            for obj in ground_truth_data:
                # look for a class_name match
                if obj["class_name"] == class_name:
                    bbgt = [ float(x) for x in obj["bbox"].split() ]
                    bi = [max(bb[0],bbgt[0]), max(bb[1],bbgt[1]),
min(bb[2],bbgt[2]), min(bb[3],bbgt[3])]
                    iw = bi[2] - bi[0] + 1
                    ih = bi[3] - bi[1] + 1
                    if iw > 0 and ih > 0:
                        # compute overlap (IoU) = area of intersection / area of
union
                        ua = (bb[2] - bb[0] + 1) * (bb[3] - bb[1] + 1) + (bbgt[2]
- bbgt[0]
                                + 1) * (bbgt[3] - bbgt[1] + 1) - iw * ih
                        ov = iw * ih / ua
                        if ov > ovmax:
                            ovmax = ov
                            gt_match = obj

            # assign detection as true positive/don't care/false positive
            if show_animation:
```

```
            status = "NO MATCH FOUND!" # status is only used in
the animation
        # set minimum overlap
        min_overlap = MINOVERLAP
        if specific_iou_flagged:
            if class_name in specific_iou_classes:
                index = specific_iou_classes.index(class_name)
                min_overlap = float(iou_list[index])
        if ovmax >= min_overlap:
            if "difficult" not in gt_match:
                    if not bool(gt_match["used"]):
                        # true positive
                        tp[idx] = 1
                        gt_match["used"] = True
                        count_true_positives[class_name] += 1
                        # update the ".json" file
                        with open(gt_file, 'w') as f:
                                f.write(json.dumps(ground_truth_data))
                        if show_animation:
                            status = "MATCH!"
                    else:
                        # false positive (multiple detection)
                        fp[idx] = 1
                        if show_animation:
                            status = "REPEATED MATCH!"
        else:
            # false positive
            fp[idx] = 1
            if ovmax > 0:
                status = "INSUFFICIENT OVERLAP"

        """
         Draw image to show animation
        """
        if show_animation:
            height, widht = img.shape[:2]
            # colors (OpenCV works with BGR)
            white = (255,255,255)
            light_blue = (255,200,100)
            green = (0,255,0)
            light_red = (30,30,255)
```

```
            # 1st line
            margin = 10
            v_pos = int(height - margin - (bottom_border / 2.0))
            text = "Image: " + ground_truth_img[0] + " "
            img, line_width = draw_text_in_image(img, text, (margin,
v_pos), white, 0)
            text = "Class [" + str(class_index) + "/" + str(n_classes) +
"]: " + class_name + " "
            img, line_width = draw_text_in_image(img, text, (margin +
line_width, v_pos), light_blue, line_width)
            if ovmax != -1:
                color = light_red
                if status == "INSUFFICIENT OVERLAP":
                    text = "IoU: {0:.2f}% ".format(ovmax*100) + "<
{0:.2f}% ".format(min_overlap*100)
                else:
                    text = "IoU: {0:.2f}% ".format(ovmax*100) + ">=
{0:.2f}% ".format(min_overlap*100)
                    color = green
                img, _ = draw_text_in_image(img, text, (margin +
line_width, v_pos), color, line_width)
            # 2nd line
            v_pos += int(bottom_border / 2.0)
            rank_pos = str(idx+1) # rank position (idx starts at 0)
            text = "Detection #rank: " + rank_pos + " confidence:
{0:.2f}% ".format(float(detection["confidence"])*100)
            img, line_width = draw_text_in_image(img, text, (margin,
v_pos), white, 0)
            color = light_red
            if status == "MATCH!":
                color = green
            text = "Result: " + status + " "
            img, line_width = draw_text_in_image(img, text, (margin +
line_width, v_pos), color, line_width)

            font = cv2.FONT_HERSHEY_SIMPLEX
            if ovmax > 0: # if there is intersections between the
bounding-boxes
                bbgt = [ int(round(float(x))) for x in
gt_match["bbox"].split() ]
```

```
cv2.rectangle(img,(bbgt[0],bbgt[1]),(bbgt[2],bbgt[3]),light_blue,2)

cv2.rectangle(img_cumulative,(bbgt[0],bbgt[1]),(bbgt[2],bbgt[3]),ligh
t_blue,2)
                cv2.putText(img_cumulative, class_name,
(bbgt[0],bbgt[1] - 5), font, 0.6, light_blue, 1, cv2.LINE_AA)
            bb = [int(i) for i in bb]
            cv2.rectangle(img,(bb[0],bb[1]),(bb[2],bb[3]),color,2)

cv2.rectangle(img_cumulative,(bb[0],bb[1]),(bb[2],bb[3]),color,2)
            cv2.putText(img_cumulative, class_name, (bb[0],bb[1] - 5),
font, 0.6, color, 1, cv2.LINE_AA)
            # show image
            cv2.imshow("Animation", img)
            cv2.waitKey(20) # show for 20 ms
            # save image to output
            output_img_path = output_files_path +
"/images/detections_one_by_one/" + class_name + "_detection" +
str(idx) + ".jpg"
            cv2.imwrite(output_img_path, img)
            # save the image with all the objects drawn to it
            cv2.imwrite(img_cumulative_path, img_cumulative)

        #print(tp)
        # compute precision/recall
        cumsum = 0
        for idx, val in enumerate(fp):
            fp[idx] += cumsum
            cumsum += val
        cumsum = 0
        for idx, val in enumerate(tp):
            tp[idx] += cumsum
            cumsum += val
        #print(tp)
        rec = tp[:]
        for idx, val in enumerate(tp):
            rec[idx] = float(tp[idx]) / gt_counter_per_class[class_name]
        #print(rec)
        prec = tp[:]
        for idx, val in enumerate(tp):
```

```
            prec[idx] = float(tp[idx]) / (fp[idx] + tp[idx])
        #print(prec)

        ap, mrec, mprec = voc_ap(rec[:], prec[:])
        sum_AP += ap
        text = "{0:.2f}%".format(ap*100) + " = " + class_name + " AP "
#class_name + " AP = {0:.2f}%".format(ap*100)
        """
         Write to output.txt
        """
        rounded_prec = [ '%.2f' % elem for elem in prec ]
        rounded_rec = [ '%.2f' % elem for elem in rec ]
        output_file.write(text + "\n Precision: " + str(rounded_prec) + "\n
Recall :" + str(rounded_rec) + "\n\n")
        if not args.quiet:
            print(text)
        ap_dictionary[class_name] = ap

        n_images = counter_images_per_class[class_name]
        lamr, mr, fppi = log_average_miss_rate(np.array(prec),
np.array(rec), n_images)
        lamr_dictionary[class_name] = lamr

        """
         Draw plot
        """
        if draw_plot:
            plt.plot(rec, prec, '-o')
            # add a new penultimate point to the list (mrec[-2], 0.0)
            # since the last line segment (and respective area) do not affect
the AP value
            area_under_curve_x = mrec[:-1] + [mrec[-2]] + [mrec[-1]]
            area_under_curve_y = mprec[:-1] + [0.0] + [mprec[-1]]
            plt.fill_between(area_under_curve_x, 0, area_under_curve_y,
alpha=0.2, edgecolor='r')
            # set window title
            fig = plt.gcf() # gcf - get current figure
            fig.canvas.set_window_title('AP ' + class_name)
            # set plot title
            plt.title('class: ' + text)
            #plt.suptitle('This is a somewhat long figure title', fontsize=16)
```

```
            # set axis titles
            plt.xlabel('Recall')
            plt.ylabel('Precision')
            # optional - set axes
            axes = plt.gca() # gca - get current axes
            axes.set_xlim([0.0,1.0])
            axes.set_ylim([0.0,1.05]) # .05 to give some extra space
            # Alternative option -> wait for button to be pressed
            #while not plt.waitforbuttonpress(): pass # wait for key display
            # Alternative option -> normal display
            #plt.show()
            # save the plot
            fig.savefig(output_files_path + "/classes/" + class_name +
".png")
            plt.cla() # clear axes for next plot

    if show_animation:
        cv2.destroyAllWindows()

    output_file.write("\n# mAP of all classes\n")
    mAP = sum_AP / n_classes
    text = "mAP = {0:.2f}%".format(mAP*100)
    output_file.write(text + "\n")
    print(text)

"""
 Draw false negatives
"""
if show_animation:
    pink = (203,192,255)
    for tmp_file in gt_files:
        ground_truth_data = json.load(open(tmp_file))
        #print(ground_truth_data)
        # get name of corresponding image
        start = TEMP_FILES_PATH + '/'
        img_id =
tmp_file[tmp_file.find(start)+len(start):tmp_file.rfind('_ground_truth.j
son')]
        img_cumulative_path = output_files_path + "/images/" + img_id
+ ".jpg"
        img = cv2.imread(img_cumulative_path)
```

```
        if img is None:
            img_path = IMG_PATH + '/' + img_id + ".jpg"
            img = cv2.imread(img_path)
        # draw false negatives
        for obj in ground_truth_data:
            if not obj['used']:
                bbgt = [ int(round(float(x))) for x in obj["bbox"].split() ]

cv2.rectangle(img,(bbgt[0],bbgt[1]),(bbgt[2],bbgt[3]),pink,2)
        cv2.imwrite(img_cumulative_path, img)

# remove the temp_files directory
shutil.rmtree(TEMP_FILES_PATH)
```

Menghitung Hasil Deteksi

```
"""
 Count total of detection-results
"""
# iterate through all the files
det_counter_per_class = {}
for txt_file in dr_files_list:
    # get lines to list
    lines_list = file_lines_to_list(txt_file)
    for line in lines_list:
        class_name = line.split()[0]
        # check if class is in the ignore list, if yes skip
        if class_name in args.ignore:
            continue
        # count that object
        if class_name in det_counter_per_class:
            det_counter_per_class[class_name] += 1
        else:
            # if class didn't exist yet
            det_counter_per_class[class_name] = 1
#print(det_counter_per_class)
dr_classes = list(det_counter_per_class.keys())


"""
 Plot the total number of occurences of each class in the ground-truth
"""
```

```
if draw_plot:
    window_title = "ground-truth-info"
    plot_title = "ground-truth\n"
    plot_title += "(" + str(len(ground_truth_files_list)) + " files and " +
str(n_classes) + " classes)"
    x_label = "Number of objects per class"
    output_path = output_files_path + "/ground-truth-info.png"
    to_show = False
    plot_color = 'forestgreen'
    draw_plot_func(
        gt_counter_per_class,
        n_classes,
        window_title,
        plot_title,
        x_label,
        output_path,
        to_show,
        plot_color,
        '',
        )

"""
 Write number of ground-truth objects per class to results.txt
"""
with open(output_files_path + "/output.txt", 'a') as output_file:
    output_file.write("\n# Number of ground-truth objects per class\n")
    for class_name in sorted(gt_counter_per_class):
        output_file.write(class_name + ": " +
str(gt_counter_per_class[class_name]) + "\n")

"""
 Finish counting true positives
"""
for class_name in dr_classes:
    # if class exists in detection-result but not in ground-truth then there
are no true positives in that class
    if class_name not in gt_classes:
        count_true_positives[class_name] = 0
#print(count_true_positives)

"""
```

```
 Plot the total number of occurences of each class in the "detection-
results" folder
"""
if draw_plot:
    window_title = "detection-results-info"
    # Plot title
    plot_title = "detection-results\n"
    plot_title += "(" + str(len(dr_files_list)) + " files and "
    count_non_zero_values_in_dictionary = sum(int(x) > 0 for x in
list(det_counter_per_class.values()))
    plot_title += str(count_non_zero_values_in_dictionary) + "
detected classes)"
    # end Plot title
    x_label = "Number of objects per class"
    output_path = output_files_path + "/detection-results-info.png"
    to_show = False
    plot_color = 'forestgreen'
    true_p_bar = count_true_positives
    draw_plot_func(
        det_counter_per_class,
        len(det_counter_per_class),
        window_title,
        plot_title,
        x_label,
        output_path,
        to_show,
        plot_color,
        true_p_bar
        )

"""
 Write number of detected objects per class to output.txt
"""
with open(output_files_path + "/output.txt", 'a') as output_file:
    output_file.write("\n# Number of detected objects per class\n")
    for class_name in sorted(dr_classes):
        n_det = det_counter_per_class[class_name]
        text = class_name + ": " + str(n_det)
        text += " (tp:" + str(count_true_positives[class_name]) + ""
        text += ", fp:" + str(n_det - count_true_positives[class_name]) +
")\n"
```

```
        output_file.write(text)

"""
 Draw log-average miss rate plot (Show lamr of all classes in
decreasing order)
"""
if draw_plot:
    window_title = "lamr"
    plot_title = "log-average miss rate"
    x_label = "log-average miss rate"
    output_path = output_files_path + "/lamr.png"
    to_show = False
    plot_color = 'royalblue'
    draw_plot_func(
        lamr_dictionary,
        n_classes,
        window_title,
        plot_title,
        x_label,
        output_path,
        to_show,
        plot_color,
        ""
        )
```

Memvisualisasikan mAP Plot

```
"""
 Draw mAP plot (Show AP's of all classes in decreasing order)
"""
if draw_plot:
    window_title = "mAP"
    plot_title = "mAP = {0:.2f}%".format(mAP*100)
    x_label = "Average Precision"
    output_path = output_files_path + "/mAP.png"
    to_show = True
    plot_color = 'royalblue'
    draw_plot_func(
        ap_dictionary,
        n_classes,
        window_title,
        plot_title,
        x_label,
```

```
        output_path,
        to_show,
        plot_color,
        """
        )
```

Lampiran 12 *Hyperparameter Custom* YOLOv3 Default Anchor Box (yolov3_custom_default.cfg)

```
[net]
# Testing
#batch=1
#subdivisions=1
# Training
batch=64
subdivisions=16
width=416
height=416
channels=3
momentum=0.9
decay=0.0005
angle=0
saturation = 1.5
exposure = 1.5
hue=.1

learning_rate=0.001
burn_in=1000
max_batches = 12000
policy=steps
steps=9600, 10800
scales=.1,.1

[convolutional]
batch_normalize=1
filters=32
size=3
stride=1
pad=1
activation=leaky

# Downsample
```

```
[convolutional]
batch_normalize=1
filters=64
size=3
stride=2
pad=1
activation=leaky

[convolutional]
batch_normalize=1
filters=32
size=1
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
filters=64
size=3
stride=1
pad=1
activation=leaky

[shortcut]
from=-3
activation=linear

# Downsample

[convolutional]
batch_normalize=1
filters=128
size=3
stride=2
pad=1
activation=leaky

[convolutional]
batch_normalize=1
```

```
filters=64
size=1
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
filters=128
size=3
stride=1
pad=1
activation=leaky

[shortcut]
from=-3
activation=linear

[convolutional]
batch_normalize=1
filters=64
size=1
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
filters=128
size=3
stride=1
pad=1
activation=leaky

[shortcut]
from=-3
activation=linear

# Downsample

[convolutional]
```

```
batch_normalize=1
filters=256
size=3
stride=2
pad=1
activation=leaky

[convolutional]
batch_normalize=1
filters=128
size=1
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
filters=256
size=3
stride=1
pad=1
activation=leaky

[shortcut]
from=-3
activation=linear

[convolutional]
batch_normalize=1
filters=128
size=1
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
filters=256
size=3
stride=1
pad=1
```

```
activation=leaky

[shortcut]
from=-3
activation=linear

[convolutional]
batch_normalize=1
filters=128
size=1
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
filters=256
size=3
stride=1
pad=1
activation=leaky

[shortcut]
from=-3
activation=linear

[convolutional]
batch_normalize=1
filters=128
size=1
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
filters=256
size=3
stride=1
pad=1
activation=leaky
```

```
[shortcut]
from=-3
activation=linear


[convolutional]
batch_normalize=1
filters=128
size=1
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
filters=256
size=3
stride=1
pad=1
activation=leaky

[shortcut]
from=-3
activation=linear

[convolutional]
batch_normalize=1
filters=128
size=1
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
filters=256
size=3
stride=1
pad=1
activation=leaky
```

```
[shortcut]
from=-3
activation=linear

[convolutional]
batch_normalize=1
filters=128
size=1
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
filters=256
size=3
stride=1
pad=1
activation=leaky

[shortcut]
from=-3
activation=linear

[convolutional]
batch_normalize=1
filters=128
size=1
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
filters=256
size=3
stride=1
pad=1
activation=leaky
```

```
[shortcut]
from=-3
activation=linear

# Downsample

[convolutional]
batch_normalize=1
filters=512
size=3
stride=2
pad=1
activation=leaky

[convolutional]
batch_normalize=1
filters=256
size=1
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
filters=512
size=3
stride=1
pad=1
activation=leaky

[shortcut]
from=-3
activation=linear


[convolutional]
batch_normalize=1
filters=256
size=1
stride=1
pad=1
```

```
activation=leaky

[convolutional]
batch_normalize=1
filters=512
size=3
stride=1
pad=1
activation=leaky

[shortcut]
from=-3
activation=linear


[convolutional]
batch_normalize=1
filters=256
size=1
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
filters=512
size=3
stride=1
pad=1
activation=leaky

[shortcut]
from=-3
activation=linear


[convolutional]
batch_normalize=1
filters=256
size=1
stride=1
```

```
pad=1
activation=leaky

[convolutional]
batch_normalize=1
filters=512
size=3
stride=1
pad=1
activation=leaky

[shortcut]
from=-3
activation=linear

[convolutional]
batch_normalize=1
filters=256
size=1
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
filters=512
size=3
stride=1
pad=1
activation=leaky

[shortcut]
from=-3
activation=linear


[convolutional]
batch_normalize=1
filters=256
size=1
stride=1
```

```
pad=1
activation=leaky

[convolutional]
batch_normalize=1
filters=512
size=3
stride=1
pad=1
activation=leaky

[shortcut]
from=-3
activation=linear

[convolutional]
batch_normalize=1
filters=256
size=1
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
filters=512
size=3
stride=1
pad=1
activation=leaky

[shortcut]
from=-3
activation=linear

[convolutional]
batch_normalize=1
filters=256
size=1
stride=1
pad=1
```

```
activation=leaky

[convolutional]
batch_normalize=1
filters=512
size=3
stride=1
pad=1
activation=leaky

[shortcut]
from=-3
activation=linear

# Downsample

[convolutional]
batch_normalize=1
filters=1024
size=3
stride=2
pad=1
activation=leaky

[convolutional]
batch_normalize=1
filters=512
size=1
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
filters=1024
size=3
stride=1
pad=1
activation=leaky

[shortcut]
```

```
from=-3
activation=linear

[convolutional]
batch_normalize=1
filters=512
size=1
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
filters=1024
size=3
stride=1
pad=1
activation=leaky

[shortcut]
from=-3
activation=linear

[convolutional]
batch_normalize=1
filters=512
size=1
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
filters=1024
size=3
stride=1
pad=1
activation=leaky

[shortcut]
from=-3
```

```
activation=linear

[convolutional]
batch_normalize=1
filters=512
size=1
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
filters=1024
size=3
stride=1
pad=1
activation=leaky

[shortcut]
from=-3
activation=linear

######################

[convolutional]
batch_normalize=1
filters=512
size=1
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
size=3
stride=1
pad=1
filters=1024
activation=leaky

[convolutional]
```

```
batch_normalize=1
filters=512
size=1
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
size=3
stride=1
pad=1
filters=1024
activation=leaky

[convolutional]
batch_normalize=1
filters=512
size=1
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
size=3
stride=1
pad=1
filters=1024
activation=leaky

[convolutional]
size=1
stride=1
pad=1
filters=33
activation=linear


[yolo]
mask = 6,7,8
```

```
anchors = 10,13,  16,30,  33,23,  30,61,  62,45,  59,119,  116,90,
156,198,  373,326
classes=6
num=9
jitter=.3
ignore_thresh = .7
truth_thresh = 1
random=1


[route]
layers = -4

[convolutional]
batch_normalize=1
filters=256
size=1
stride=1
pad=1
activation=leaky

[upsample]
stride=2

[route]
layers = -1, 61

[convolutional]
batch_normalize=1
filters=256
size=1
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
size=3
stride=1
pad=1
filters=512
```

```
activation=leaky

[convolutional]
batch_normalize=1
filters=256
size=1
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
size=3
stride=1
pad=1
filters=512
activation=leaky

[convolutional]
batch_normalize=1
filters=256
size=1
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
size=3
stride=1
pad=1
filters=512
activation=leaky

[convolutional]
size=1
stride=1
pad=1
filters=33
activation=linear
```

```
[yolo]
mask = 3,4,5
anchors = 10,13,  16,30,  33,23,  30,61,  62,45,  59,119,  116,90,
156,198,  373,326
classes=6
num=9
jitter=.3
ignore_thresh = .7
truth_thresh = 1
random=1

[route]
layers = -4

[convolutional]
batch_normalize=1
filters=128
size=1
stride=1
pad=1
activation=leaky

[upsample]
stride=2

[route]
layers = -1, 36

[convolutional]
batch_normalize=1
filters=128
size=1
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
size=3
stride=1
```

```
pad=1
filters=256
activation=leaky

[convolutional]
batch_normalize=1
filters=128
size=1
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
size=3
stride=1
pad=1
filters=256
activation=leaky

[convolutional]
batch_normalize=1
filters=128
size=1
stride=1
pad=1
activation=leaky


[convolutional]
batch_normalize=1
size=3
stride=1
pad=1
filters=256
activation=leaky


[convolutional]
size=1
stride=1
```

```
pad=1
filters=33
activation=linear


[yolo]
mask = 0,1,2
anchors = 10,13,  16,30,  33,23,  30,61,  62,45,  59,119,  116,90,
156,198,  373,326
classes=6
num=9
jitter=.3
ignore_thresh = .7
truth_thresh = 1
random=1
```

Lampiran 13 *Hyperparameter Custom* YOLOv3 dengan *K-means* (yolov3_custom_kmean.cfg)

```
[net]
# Testing
#batch=1
#subdivisions=1
# Training
batch=64
subdivisions=16
width=416
height=416
channels=3
momentum=0.9
decay=0.0005
angle=0
saturation = 1.5
exposure = 1.5
hue=.1

learning_rate=0.001
burn_in=1000
max_batches = 12000
policy=steps
steps=9600, 10800
scales=.1,.1
```

```
[convolutional]
batch_normalize=1
filters=32
size=3
stride=1
pad=1
activation=leaky

# Downsample

[convolutional]
batch_normalize=1
filters=64
size=3
stride=2
pad=1
activation=leaky

[convolutional]
batch_normalize=1
filters=32
size=1
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
filters=64
size=3
stride=1
pad=1
activation=leaky

[shortcut]
from=-3
activation=linear

# Downsample
```

```
[convolutional]
batch_normalize=1
filters=128
size=3
stride=2
pad=1
activation=leaky

[convolutional]
batch_normalize=1
filters=64
size=1
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
filters=128
size=3
stride=1
pad=1
activation=leaky

[shortcut]
from=-3
activation=linear

[convolutional]
batch_normalize=1
filters=64
size=1
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
filters=128
size=3
stride=1
```

```
pad=1
activation=leaky

[shortcut]
from=-3
activation=linear

# Downsample

[convolutional]
batch_normalize=1
filters=256
size=3
stride=2
pad=1
activation=leaky

[convolutional]
batch_normalize=1
filters=128
size=1
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
filters=256
size=3
stride=1
pad=1
activation=leaky

[shortcut]
from=-3
activation=linear

[convolutional]
batch_normalize=1
filters=128
size=1
```

```
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
filters=256
size=3
stride=1
pad=1
activation=leaky

[shortcut]
from=-3
activation=linear

[convolutional]
batch_normalize=1
filters=128
size=1
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
filters=256
size=3
stride=1
pad=1
activation=leaky

[shortcut]
from=-3
activation=linear

[convolutional]
batch_normalize=1
filters=128
size=1
stride=1
```

```
pad=1
activation=leaky

[convolutional]
batch_normalize=1
filters=256
size=3
stride=1
pad=1
activation=leaky

[shortcut]
from=-3
activation=linear


[convolutional]
batch_normalize=1
filters=128
size=1
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
filters=256
size=3
stride=1
pad=1
activation=leaky

[shortcut]
from=-3
activation=linear

[convolutional]
batch_normalize=1
filters=128
size=1
stride=1
```

```
pad=1
activation=leaky

[convolutional]
batch_normalize=1
filters=256
size=3
stride=1
pad=1
activation=leaky

[shortcut]
from=-3
activation=linear

[convolutional]
batch_normalize=1
filters=128
size=1
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
filters=256
size=3
stride=1
pad=1
activation=leaky

[shortcut]
from=-3
activation=linear

[convolutional]
batch_normalize=1
filters=128
size=1
stride=1
pad=1
```

```
activation=leaky

[convolutional]
batch_normalize=1
filters=256
size=3
stride=1
pad=1
activation=leaky

[shortcut]
from=-3
activation=linear

# Downsample

[convolutional]
batch_normalize=1
filters=512
size=3
stride=2
pad=1
activation=leaky

[convolutional]
batch_normalize=1
filters=256
size=1
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
filters=512
size=3
stride=1
pad=1
activation=leaky

[shortcut]
```

```
from=-3
activation=linear


[convolutional]
batch_normalize=1
filters=256
size=1
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
filters=512
size=3
stride=1
pad=1
activation=leaky

[shortcut]
from=-3
activation=linear


[convolutional]
batch_normalize=1
filters=256
size=1
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
filters=512
size=3
stride=1
pad=1
activation=leaky
```

```
[shortcut]
from=-3
activation=linear


[convolutional]
batch_normalize=1
filters=256
size=1
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
filters=512
size=3
stride=1
pad=1
activation=leaky

[shortcut]
from=-3
activation=linear

[convolutional]
batch_normalize=1
filters=256
size=1
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
filters=512
size=3
stride=1
pad=1
activation=leaky
```

```
[shortcut]
from=-3
activation=linear


[convolutional]
batch_normalize=1
filters=256
size=1
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
filters=512
size=3
stride=1
pad=1
activation=leaky

[shortcut]
from=-3
activation=linear


[convolutional]
batch_normalize=1
filters=256
size=1
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
filters=512
size=3
stride=1
pad=1
activation=leaky
```

```
[shortcut]
from=-3
activation=linear

[convolutional]
batch_normalize=1
filters=256
size=1
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
filters=512
size=3
stride=1
pad=1
activation=leaky

[shortcut]
from=-3
activation=linear

# Downsample

[convolutional]
batch_normalize=1
filters=1024
size=3
stride=2
pad=1
activation=leaky

[convolutional]
batch_normalize=1
filters=512
size=1
stride=1
pad=1
```

```
activation=leaky

[convolutional]
batch_normalize=1
filters=1024
size=3
stride=1
pad=1
activation=leaky

[shortcut]
from=-3
activation=linear

[convolutional]
batch_normalize=1
filters=512
size=1
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
filters=1024
size=3
stride=1
pad=1
activation=leaky

[shortcut]
from=-3
activation=linear

[convolutional]
batch_normalize=1
filters=512
size=1
stride=1
pad=1
activation=leaky
```

```
[convolutional]
batch_normalize=1
filters=1024
size=3
stride=1
pad=1
activation=leaky

[shortcut]
from=-3
activation=linear

[convolutional]
batch_normalize=1
filters=512
size=1
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
filters=1024
size=3
stride=1
pad=1
activation=leaky

[shortcut]
from=-3
activation=linear

######################

[convolutional]
batch_normalize=1
filters=512
size=1
stride=1
pad=1
```

```
activation=leaky

[convolutional]
batch_normalize=1
size=3
stride=1
pad=1
filters=1024
activation=leaky

[convolutional]
batch_normalize=1
filters=512
size=1
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
size=3
stride=1
pad=1
filters=1024
activation=leaky

[convolutional]
batch_normalize=1
filters=512
size=1
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
size=3
stride=1
pad=1
filters=1024
activation=leaky
```

```
[convolutional]
size=1
stride=1
pad=1
filters=33
activation=linear


[yolo]
mask = 6,7,8
anchors = 11,20,  24,38,  31,50,  42,65,  53,82,  60,85,  88,108,
100,163,  141,168
classes=6
num=9
jitter=.3
ignore_thresh = .7
truth_thresh = 1
random=1


[route]
layers = -4

[convolutional]
batch_normalize=1
filters=256
size=1
stride=1
pad=1
activation=leaky

[upsample]
stride=2

[route]
layers = -1, 61



[convolutional]
```

```
batch_normalize=1
filters=256
size=1
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
size=3
stride=1
pad=1
filters=512
activation=leaky

[convolutional]
batch_normalize=1
filters=256
size=1
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
size=3
stride=1
pad=1
filters=512
activation=leaky

[convolutional]
batch_normalize=1
filters=256
size=1
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
```

```
size=3
stride=1
pad=1
filters=512
activation=leaky

[convolutional]
size=1
stride=1
pad=1
filters=33
activation=linear


[yolo]
mask = 3,4,5
anchors = 11,20,  24,38,  31,50,  42,65,  53,82,  60,85,  88,108,
100,163,  141,168
classes=6
num=9
jitter=.3
ignore_thresh = .7
truth_thresh = 1
random=1

[route]
layers = -4

[convolutional]
batch_normalize=1
filters=128
size=1
stride=1
pad=1
activation=leaky

[upsample]
stride=2

[route]
layers = -1, 36
```

```
[convolutional]
batch_normalize=1
filters=128
size=1
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
size=3
stride=1
pad=1
filters=256
activation=leaky

[convolutional]
batch_normalize=1
filters=128
size=1
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
size=3
stride=1
pad=1
filters=256
activation=leaky

[convolutional]
batch_normalize=1
filters=128
size=1
stride=1
pad=1
activation=leaky
```

```
[convolutional]
batch_normalize=1
size=3
stride=1
pad=1
filters=256
activation=leaky

[convolutional]
size=1
stride=1
pad=1
filters=33
activation=linear

[yolo]
mask = 0,1,2
anchors = 11,20,  24,38,  31,50,  42,65,  53,82,  60,85,  88,108,
100,163,  141,168
classes=6
num=9
jitter=.3
ignore_thresh = .7
truth_thresh = 1
random=1
```

Lampiran 14 Sintaks *Confusions Matrix* Google Colab

Cloning dan *Building* Darknet ke Google Colab
!python –version !ls !cd /content !rm -fr darknet !git clone https://github.com/AlexeyAB/darknet/ % cd darknet !sed -i 's/OPENCV=0/OPENCV=1/g' Makefile !sed -i 's/GPU=0/GPU=1/g' Makefile !sed -i 's/CUDNN=0/CUDNN=1/g' Makefile !apt update !apt-get install libopencv-dev

```
%cd darknet
!sed -i 's/GPU=0/GPU=1/g' Makefile
!cat Makefile
!make

# verify CUDA
!/usr/local/cuda/bin/nvcc –version

# make darknet (build)
!make
```

Import Module ke Google Colab

```
# define helper functions
def imShow(path):
  import cv2
  import matplotlib.pyplot as plt
  %matplotlib inline

  image = cv2.imread(path)
  height, width = image.shape[:2]
  resized_image = cv2.resize(image,(3*width, 3*height), interpolation
= cv2.INTER_CUBIC)

  fig = plt.gcf()
  fig.set_size_inches(18, 10)
  plt.axis("off")
  plt.imshow(cv2.cvtColor(resized_image, cv2.COLOR_BGR2RGB))
  plt.show()

# use this to upload files
def upload():
  from google.colab import files
  uploaded = files.upload()
  for name, data in uploaded.items():
    with open(name, 'wb') as f:
      f.write(data)
      print ('saved file', name)

# use this to download a file
def download(path):
  from google.colab import files
  files.download(path)
```

```
#Import Module Confusion Matrix
  import numpy as np
  import progressbar
  import sklearn

# This stops 'Run all' at this cell by causing an error
assert False
```

Menghubungkan Google Drive ke Google Colab dan Cek Daftar *File*

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

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

# this is where my zip is stored (I created a yolov3 folder where I will
get my required files from)
!ls /mydrive/yolov3
```

Unzip Data Citra dengan Label

```
# copy the .zip file into the root directory of cloud VM
!cp /mydrive/yolov3/obj.zip ../

# unzip the zip file and its contents should now be in /darknet/data/obj
!unzip ../obj.zip -d data/
```

Configuring Files

```
# upload the custom .cfg back to cloud VM from Google Drive
!cp /mydrive/yolov3/yolov3_custom.cfg ./cfg

# upload the custom .cfg back to cloud VM from local machine
(uncomment to use)
#%cd cfg
#upload()
#%cd ..

# upload the obj.names and obj.data files to cloud VM from Google
Drive
!cp /mydrive/yolov3/obj.names ./data
```

```
!cp /mydrive/yolov3/obj.data  ./data

# upload the obj.names and obj.data files to cloud VM from local
machine (uncomment to use)
#%cd data
#upload()
#%cd ..

# upload the generate_train.py script to cloud VM from Google Drive
!cp /mydrive/yolov3/generate_train.py ./

# upload the generate_train.py script to cloud VM from local machine
(uncomment to use)
#upload()
!python generate_train.py

# verify train.txt can be seen in our darknet/data folder
!ls data/
```

Cloning dan *Building* Confusion Matrix Repository ke Google Colab

```
# Confusion Matrix

# clone darknet repo
!git clone https://github.com/whynotw/YOLO_metric.git
# change makefile to have GPU and OPENCV enabled
%cd YOLO_metric
!sed -i 's/OPENCV=0/OPENCV=1/' Makefile
!sed -i 's/GPU=0/GPU=1/' Makefile
!sed -i 's/CUDNN=0/CUDNN=1/' Makefile
!sed -i 's/CUDNN_HALF=0/CUDNN_HALF=1/' Makefile

# make darknet (builds darknet so that you can then use the darknet
executable file to run or train object detectors)
!make
```

Generate Prediksi Label YOLOv3

```
# upload the generate_train.py and generate_test.py script to cloud
VM from Google Drive
!cp /content/YOLO_metric/modulized/save_label_as_yolo_format.py
./

!python save_label_as_yolo_format.py
```

Generate Confusion Matrix
Berikut untuk CEK Directory sudah benar apa belum import cv2 img_src = cv2.imread('/mydrive/datatrain/1.jpg',0) print img_src # upload the generate_train.py and generate_test.py script to cloud VM from Google Drive !cp /content/YOLO_metric/modulized/compare_simple.py ./ !python compare_simple.py

BIODATA PENULIS

Penulis dilahirkan di Tulungagung, 26 Mei 1997 dengan nama lengkap Adam Fahmi Fandisyah namun biasa dipanggil Adam.Penulis merupakan anak pertama dari dua bersaudara oleh pasangan suami istri Nurhadi dan Farida Masviyah. Pendidikan formal yang ditempuh oleh penulis antara lain SDN 2 Tanjungsari (2004-2010), SMPN 2 Tulungagung (2010-2013), dan MAN 2 Tulungagung (2013-2016). Setelah lulus, penulis diterima sebagai mahasiswa Departemen Statistika ITS dengan NRP 1316100065 yang kemudian berganti menjadi 06211640000065. Selama masa perkuliahan, penulis aktif dalam kegiatan organisasi, kepanitiaan, dan pelatihan. Organisasi kampus yang pernah di ikuti oleh penulis adalah Professional Statistics Himpunan Mahasiswa Statistika ITS (PSt HIMASTA-ITS) sebagai staf periode 2017-2018 dan Manajer PR pada periode 2018-2019. Selain itu, penulis menjadi staf ahli Media FORSIS-ITS 39/40. Penulis juga aktif dalam kepanitiaan yang diadakan oleh HIMASTA-ITS seperti Pekan Raya Statistika (PRS) 2018 sebagai Sie Medfo. Penulis juga aktif dalam menjadi surveyor seperti surveyor LIPI, Dinas Pariwisata, dsb. Bagi pembaca yang ingin berdiskusi, memberikan saran, dan kritik mengenai Tugas Akhir ini untuk perbaikan kedepannya dapat disampaikan melalui *e-mail* adamfahmifandisyah@gmail.com.