TUGAS AKHIR - EC184801

OPTIMISASI PENDETEKSIAN OBJEK KECIL YOLOv7 UNTUK MENDETEKSI OBJEK-OBJEK *AIRBORNE*

Dion Andreas Solang
NRP 0721 19 4000 0039

Dosen Pembimbing

Reza Fuad Rachmadi, S.T., M.T., Ph.D

NIP 19850403201212 1 001

Dr. I Ketut Eddy Purnama, S.T., M.T.
NIP 19690730199512 1 001

Program Studi Strata 1 (S1) Teknik Komputer
Departemen Teknik Komputer
Fakultas Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember

Surabaya
2023

[Halaman ini sengaja dikosongkan]

TUGAS AKHIR - EC184801

OPTIMISASI PENDETEKSIAN OBJEK KECIL YOLOv7 UNTUK MENDETEKSI OBJEK-OBJEK *AIRBORNE*

Dion Andreas Solang
NRP 0721 19 4000 0039

Dosen Pembimbing

Reza Fuad Rachmadi, S.T., M.T., Ph.D

NIP 19850403201212 1 001

Dr. I Ketut Eddy Purnama, S.T., M.T.
NIP 19690730199512 1 001

Program Studi Strata 1 (S1) Teknik Komputer
Departemen Teknik Komputer
Fakultas Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember
Surabaya
2023

[Halaman ini sengaja dikosongkan]

FINAL PROJECT - EC184801

YOLOv7 SMALL OBJECT DETECTION OPTIMIZATION TO DETECT AIRBORNE OBJECTS

Dion Andreas Solang
NRP 0721 19 4000 0039

Advisor

Reza Fuad Rachmadi, S.T., M.T., Ph.D

NIP 19850403201212 1 001

Dr. I Ketut Eddy Purnama, S.T., M.T.
NIP 19690730199512 1 001

Undergraduate Study Program of Computer Engineering

Department of Computer Engineering

Faculty of Faculty of Intelligent Electrical and Informatics Technology

Sepuluh Nopember Institute of Technology

Surabaya

2023

[Halaman ini sengaja dikosongkan]

ABSTRAK

Nama Mahasiswa : Dion Andreas Solang

Judul Tugas Akhir : OPTIMISASI PENDETEKSIAN OBJEK KECIL YOLOv7 UNTUK MENDETEKSI OBJEK-OBJEK *AIRBORNE*

Pembimbing : 1. Reza Fuad Rachmadi, S.T., M.T., Ph.D
2. Dr. I Ketut Eddy Purnama, S.T., M.T.

Pada penelitian ini, kami menunjukan percobaan kami untuk meningkatkan kapabilitas YOLOv7 untuk mendeteksi objek airborne. Objek airborne tampak sangat kecil pada kamera karena jarak yang cukup jauh dari kamera. Oleh karena itu, YOLOv7 harus dioptimisasi untuk dapat mendeteksi objek-objek kecil. Beberapa modifikasi diajukan dan diuji pada penelitian ini. Modifikasi-modifikasi ini meliput perubahan arsitektur (Menambah layer deteksi, mengubah sumber feature-map, dan mengganti layer deteksi menjadi anchor-free), dan pada bag-of-freebies (rekalkulasi anchor, dan augmentasi mosaik). Hingga saat ini, kami menemukan bahwa kombinasi augmentasi mosaik, rekalkulasi anchor, dan mengubah sumber feature-map memberikan skor mAP yang paling tinggi 14,09% dibandingkan dengan YOLOv7 yang biasa (mAP=0%).

Kata Kunci: ***Deteksi Objek Kecil, YOLOv7, Modifikasi Arsitektur, Modifikasi Bag-of-Freebies, Objek Airborne***

[Halaman ini sengaja dikosongkan]

ABSTRACT

Name : Dion Andreas Solang
Title : *YOLOv7 SMALL OBJECT DETECTION OPTIMIZATION TO DETECT AIRBORNE OBJECTS*
Advisors : 1. Reza Fuad Rachmadi, S.T., M.T., Ph.D
2. Dr. I Ketut Eddy Purnama, S.T., M.T.

In this research, we present an attempt to improve the capability of YOLOv7 to detect airborne objects. Airborne objects appear very small on cameras due to their distance to the camera. For that reason, YOLOv7 needs to be optimized to detect small objects. Several modification proposal was made and tested in this research. These modifications include changes in the architecture (Adding extra detection head, modifying feature-map source, and replacing detection head to a detached anchor-free head), and some bag-of-freebies applications (anchor recalculation, mosaic augmentation). At the current state, we found the combination of mosaic augmentation, anchor recalculation, and modifying feature-map source produces the greatest score in mAP, a 14.09% increase compared to the plain YOLOv7 (mAP=0%).

Keywords: ***Small Object Detection, YOLOv7, Architecture Modification, Bag-of-Freebies Modification, Airborne Object***

[Halaman ini sengaja dikosongkan]

DAFTAR ISI

DAFTAR GAMBAR

[Halaman ini sengaja dikosongkan]

DAFTAR TABEL

[Halaman ini sengaja dikosongkan]

BAB I
PENDAHULUAN

1.1 Latar Belakang

Gambar 1.1: Contoh Dataset Objek *Airborne*

Seiring berkembangnya teknologi *autonomous vehicles*, terdapat banyak keinginan untuk mengaplikasikan teknologi tersebut di berbagai bidang. Salah satu aplikasi teknologi ini di bidang komersil adalah *Amazon Prime Air*. *Amazon Prime Air* memanfaatkan *Autonomous Aerial Vehicle* (AAV) untuk melakukan pengantaran barang dari warehouse ke rumah kostumer secara *autonomous* (Amazon, 2022). Untuk melakukan hal ini, AAV yang digunakan harus mempunyai kemampuan penerbangan *autonomous* yang mumpuni.

Salah satu tantangan terbesar dari penerbangan autonomous adalah kemampuan *Sense and Avoid* (SAA) dari AAV. Meskipun di udara terdapat ruang gerak yang luas, tetap terdapat resiko AAV akan menabrak objek di udara. Objek - objek tersebut dapat berupa helikopter, pesawat, burung, misil, dan lain - lain. Objek - objek ini juga sering disebut dengan objek *airborne* ("Airborne Object Tracking Challenge", 2021).

Salah satu sensor yang dapat digunakan untuk melakukan SAA adalah kamera. Kamera memiliki bobot yang cukup ringan, sehingga dapat dibawah oleh AAV. Selain itu, kamera juga memiliki harga yang relatif lebih murah dibandingkan sensor - sensor seperti LiDAR atau Radar.

Dengan memilih kamera sebagai sensor, maka dibutuhkan suatu model computer vision untuk diaplikasikan pada kamera tersebut. Objek - objek *airborne* akan tampak sangat kecil pada kamera seperti yang dapat dilihat pada Gambar 1.1. Beberapa dataset kamera *airborne* yang memiliki resolusi 20482448 pixel, objeknya dapat berukuran 4 (0.00008% luas resolusi) hingga 1000 pixel (0.01% luas resolusi) sehingga terlihat sangat kecil ("Airborne Object Tracking Dataset", 2021). Oleh karena itu, dibutuhkan suatu model yang dapat mendeteksi objek - objek yang sangat kecil sehingga dapat mendeteksi objek *airborne*.

YOLOv7 merupakan model state-of-the-art untuk melakukan pendeteksian objek secara

real-time. YOLOv7 memiliki akurasi tertinggi dari semua model pendeteksi objek dengan kecepatan deteksi 30 FPS (yang terpublikasi) pada GPU Nvidia V100. Terdapat versi scaled dari YOLOv7 yang memiliki jumlah parameter yang lebih kecil dan dapat diaplikasikan pada device edge computing (Wang et al., 2022). Oleh karena itu, YOLOv7 ini cocok untuk digunakan pada AAV di mana dibutuhkan suatu pendeteksi objek yang real-time.

1.2 Rumusan Masalah

YOLOv7 bukan merupakan model deteksi objek umum sehingga YOLOv7 tidak didesain untuk melakukan deteksi objek kecil seperti objek-objek *airborne*. Oleh karena itu, dibuatlah rumusan masalah seperti berikut:

- Apa solusi yang dapat diaplikasikan pada YOLOv7 agar kemampuan deteksi objek *airborne*-nya dapat dioptimalisasi?

1.3 Tujuan

Adapun tujuan dari tugas akhir ini adalah untuk menemukan solusi untuk mengoptimisasi kemampuan YOLOv7 mendeteksi objek airborne.

1.4 Batasan Masalah

Optimisasi kemampuan deteksi objek kecil hanya akan dilakukan dengan memodifikasi YOLOv7. Modifikasi yang diaplikasikan tidak boleh menyebabkan YOLOv7 untuk tidak dapat melakukan pendeteksian secara *real time*. Target pengaplikasian model ini adalah untuk AAV dengan *computational resource* yang terbatas sehingga model hasil modifikasi harus cukup ringan untuk hal tersebut.

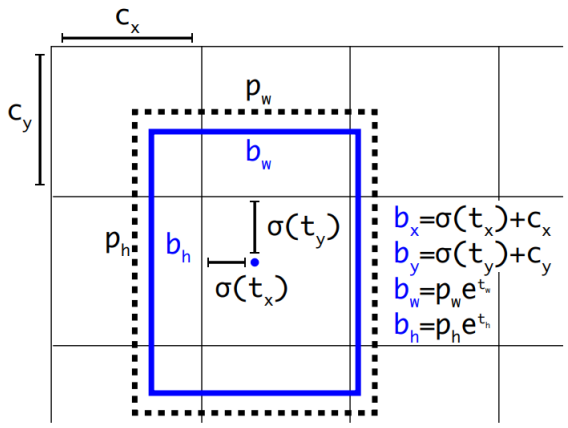
BAB II
TINJAUAN PUSTAKA

2.1 Teori Dasar

2.1.1 Arsitektur Famili YOLO

Arsitektur famili YOLO pada dasarnya terbagi akan 3 bagian yaitu *head*, *neck*, dan *backbone*. Setiap bagian ini mempunyai fungsi masing-masing. Berikut adalah penjelasan fungsi dan cara kerja dari ketiga bagian tersebut.

Layer Head YOLO dan *Anchor Box*

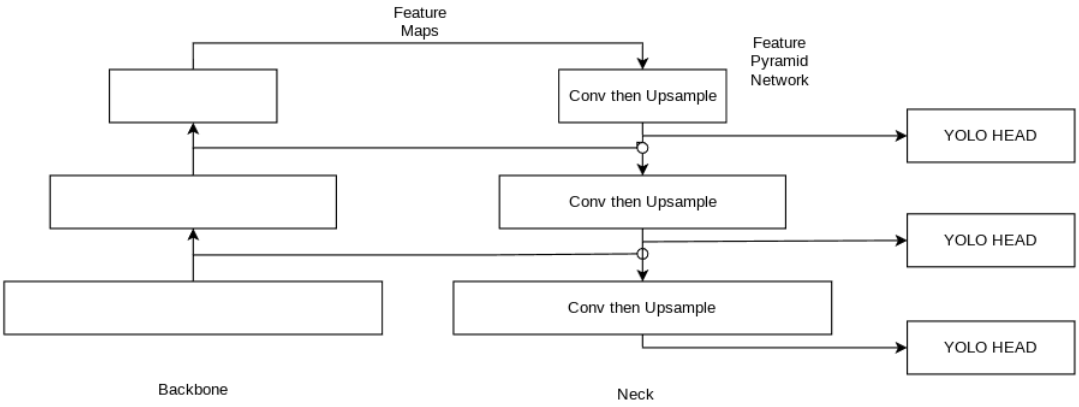


Gambar 2.1: Prediksi *Anchor Box* dan *offset* dari koordinat latis (Redmon & Farhadi, 2018)

Arsitektur famili YOLO yang dipublikasikan setelah YOLOv2 terus menggunakan *anchor box* untuk melakukan deteksi (Redmon & Farhadi, 2016; Redmon & Farhadi, 2018; Bochkovskiy et al., 2020; Wang et al., 2020; Jocher et al., 2022; Wang et al., 2021; Wang et al., 2022). *Anchor boxes* merupakan beberapa *Bounding Box* yang telah terdefinisikan. Arsitektur YOLO akan memprediksi probabilitas *anchor box* berada pada suatu koordinat latis beserta dengan *offset anchor box* tersebut untuk menepatkan *anchor box* pada objek yang dideteksi. Penggunaan *anchor box* ini dapat meningkatkan akurasi deteksi karena *neural network* hanya perlu mencari titik tengah objek dan *error* dimensi *boudning box* dengan menggunakan *offset* (Redmon & Farhadi, 2018). Hal ini lebih sederhana daripada mencari titik-titik *bounding box* secara independen sehingga lebih mudah untuk dipelajari oleh *neural network*.

Prediksi *bounding boxes* terjadi di bagian *head* dari arsitektur YOLO. Bagian *head* dari YOLO akan mengambil beberapa hasil *upsampling* yang terjadi pada *neck* YOLO, dan kemudian melakukan prediksi *anchor boxes* dari hasil tersebut. Hasil prediksi *Head* YOLO pada suatu tingkatan *upsampling* berupa tensor dengan ukuran $N \times N \times [A \times (4 + 1 + C)]$ dengan N sebagai dimensi hasil *upsampling*-nya, A sebagai jumlah *anchor boxes* untuk *scaling* tersebut, dan C sebagai jumlah kelas prediksi. Angka 4 merepresentasikan 4 *offset* b_x, b_y, b_w, b_h seperti pada Gambar 2.1 dan angka 1 merepresentasikan *objectness score* dari prediksi *bounding box*.

Neck YOLO



Gambar 2.2: *Feature Pyramid Network* pada *Neck* YOLO

Neck dari YOLO merupakan *layer-layer* dimana *head* YOLO mengambil fitur untuk dilakukan deteksi *bounding box*. Pada YOLOv3 Redmon and Farhadi (2018), arsitektur *neck* dibuat menyerupai *Feature Pyramid Network* (FPN) seperti pada Gambar 2.2. Pada versi-versi YOLO selanjutnya, bentuk *neck* ini tidak banyak berubah dan pada dasarnya tetap mempertahankan bentuk *pyramid*-nya.

Penaikkan tingkatan *pyramid* dari FPN merupakan *upsampling* dari *feature map* yang dihasilkan *backbone*. Output tiap tingkatan pada FPN di *neck* inilah yang diinputkan pada *head* YOLO. Melakukan prediksi pada tingkatan *upsampling* yang berbeda-beda dapat membuat *neural network* mendapatkan lebih banyak informasi semantik dan informasi yang lebih detail sehingga dapat lebih akurat dalam mendeteksi objek besar maupun kecil.

Backbone YOLO

Backbone dari YOLO merupakan bagian yang mengekstrak fitur dari citra yang diinputkan. Hasil ekstraksi fitur ini akan diinputkan pada *neck* yang kemudian akan di*upsampling* olehnya. Model-model YOLO dapat menggunakan *feature extractor* dari model-model klasifikasi citra sebagai *backbone*-nya. Sebagai contoh, salah satu varian YOLO, YOLO-Z menggunakan DenseNet sebagai *backbone*-nya sedangkan arsitektur YOLO dasarnya, YOLOv5 menggunakan *backbone* YOLOv5v7.0 (Benjumea et al., 2021).

2.1.2 YOLOv7

YOLOv7 merupakan pendeteksi objek *real time* dengan skor akurasi tertinggi pada dataset COCO di tahun 2022. Pada YOLOv7, dilakukan beberapa perubahan untuk meningkatkan akurasi dan kecepatan deteksinya. Perubahan-perubahan tersebut dilakukan pada arsitekturnya dan pada *bag-of-freebies*-nya.

Perubahan arsitektur dilakukan pada *backbone*. YOLOv7 menggunakan *Extended Efficient Layer Aggregation Network* (E-ELAN) sebagai *backbone*, berbeda dengan leluhurnya YOLOv4 yang menggunakan CSP-Darknet. E-ELAN merupakan arsitektur *neural network* yang efisien karena E-ELAN didesain dengan mengontrol *gradient path* terpanjang yang terpendek. Karena efisiensinya, arsitektur E-ELAN ini dapat meningkatkan kecepatan deteksi dan akurasi. (Wang et al., 2022)

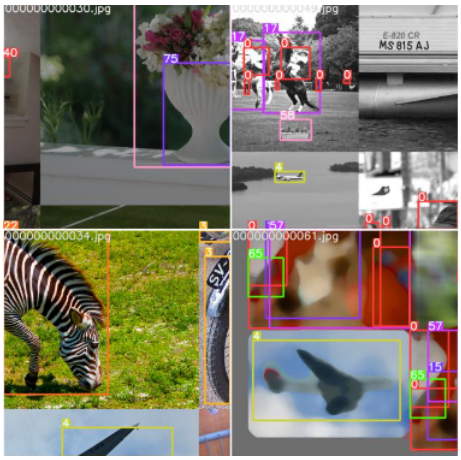
Bag-of-freebies merupakan kumpulan metode peningkatan akurasi yang tidak meningkatkan *cost inferrence* (Bochkovskiy et al., 2020). Pada YOLOv7, ditambahkan beberapa *bag-of-freebies* yang dapat dilatih seperti *re-parameterized convolution* dan *extra auxilary head* di tengah-tengah *neural network*. Selain kedua itu, YOLOv7 juga menambahkan *trainable bag-of-freebies* dari YOLOR seperti EMA, *Implicit Knowledge*, dan *conv-bn topology Batch Normalization* (Wang et al., 2022).

2.2 Rekalkulasi *Anchor*

Anchor box dari model-model *pre-trained* YOLO pada umumnya mengoptimisasi *anchor box* modelnya pada dataset COCO. Ukuran *anchor box* yang akan digunakan pada model YOLO dapat dikonfigurasikan agar lebih sesuai dengan dataset yang akan digunakan untuk melatih model YOLO. Penyesuaian ini dapat meningkatkan IoU(*Intersection Over Union*) prediksi model dengan *ground truth* sehingga meningkatkan akurasi.

Penyesuaian anchor box dapat dilakukan pada saat sebelum training atau pada saat training. Penyesuaian anchor box sebelum training dapat dilakukan dengan cara mengkonfigurasi secara manual tiap ukuran *anchor box* atau dengan menggunakan algoritma *clustering*. Penggunaan algoritma *clustering* akan lebih baik karena setiap ukuran *anchor box*-nya disesuaikan dengan pengelompokan-pengelompokan ukuran *bounding box* natural yang terdapat pada dataset. Untuk penyesuaian saat training, dapat digunakan algoritma dari Zhong et al. (2018). Algoritma ini akan mengoptimisasi ukuran-ukuran anchor bukan hanya berdasarkan dataset, namun berdasarkan kemampuan dari neural network pendeteksi objeknya. Untuk melakukan hal tersebut, algoritma ini akan memanfaatkan back propagation localization loss untuk rekalkulasi anchor.

2.3 Augmentasi Mosaik



Gambar 2.3: Contoh Augmentasi Data Mosaik (Jocher et al., 2022)

Augmentasi mosaik merupakan teknik augmentasi yang baru dikenalkan pada YOLOv4. Teknik augmentasi ini akan memilih 4 gambar dari dataset, memotong gambar-gambar tersebut dan menggabungkannya secara acak pada satu gambar seperti pada Gambar 2.3. Hasil dari penggabungan itu membuat gambar terlihat seperti mosaik. Teknik augmentasi ini mampu meningkatkan akurasi model (Bochkovskiy et al., 2020).

2.4 Penelitian Terkait

2.4.1 YOLO-Z

YOLO-Z merupakan arsitektur famili YOLO yang modifikasi dari YOLOv5 (Benjumea et al., 2021). Modifikasi-modifikasi yang dilakukan meliputi pergantian *backbone*, *neck*, dan jumlah *anchor Backbone* dari YOLOv5r5.0 menjadi DenseNet yang di-*downscale*. *Neck* dari YOLO-Z juga diganti dari PanNet menjadi FPN dan biFPN tergatung pada *scale* YOLO-Z yang digunakan.

Modifikasi pada YOLO-Z didesain untuk mendeteksi objek kecil untuk tujuan melakukan deteksi *cone* yang nampak jauh pada lintasan *autonomous racing* secara *real time* (lihat Gambar 2.4). Modifikasi-modifikasi dibuktikan dapat meningkatkan kemampuan pendeteksian objek kecil (Benjumea et al., 2021). Oleh karena itu, untuk meningkatkan kemampuan mendeteksi objek kecil YOLOv7, beberapa modifikasi yang dilakukan YOLO-Z pada YOLOv5 dapat diaplikasikan.

Gambar 2.4: Contoh Objek *Cone* yang Terlihat Jauh dari Kamera

2.4.2 exYOLO

exYOLO merupakan hasil modifikasi arsitektur YOLOv3 (Xiao, 2021). Pada exYOLO, dilakukan modifikasi *neck* dengan menambahkan suatu *Receptive Field Block* sebelum penggabungan *feature map* yang akan diupsampling. Modifikasi-modifikasi ini membuat exYOLO memiliki skor mAP yang lebih tinggi daripada YOLOv3 pada dataset PASCAL VOC 2007.

2.4.3 Implementasi YOLOv4-tiny pada *Autonomous Surface Vehicle*

Aditya et al. (2022) menggunakan model modifikasi YOLOv4-tiny pada *Autonomous Surface Vehicle*(ASV) mereka. YOLOv4-tiny sebenarnya hanya menggunakan 2 layer head, namun yang diimplementasikan pada ASV adalah model YOLOv4-tiny yang ditambahkan 1 layer head lagi sehingga menggunakan total sebanyak 3 layer head. Perubahan ini memberikan peningkatan pada skor mAP dan memberikan kemampuan modelnya untuk mendeteksi objek yang lebih jauh.

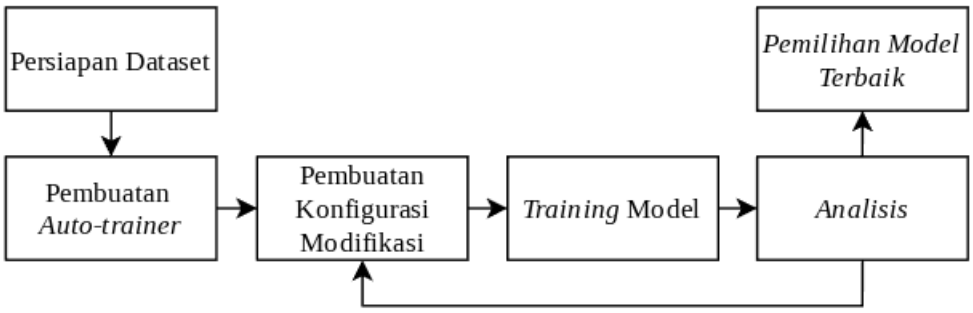
BAB III
METODOLOGI

3.1 Metode Pencarian Solusi Optimisasi

Untuk mencari solusi optimisasi pendeteksian objek kecil YOLOv7 yang terbaik, akan dilakukan penambahan atau perubahan *bag-of-freebies*, *bag-of-specials*, dan arsitektur YOLOv7. Setiap modifikasi-modifikasi itu akan diaplikasikan secara independen dan kombinatif. Yang dimaksud dengan kombinatif adalah modifikasi-modifikasi akan dikombinasikan menjadi 1 model YOLOv7.

Setiap kombinasi modifikasi, independen atau tidak, akan diuji kemampuannya mendeteksi objek *airborne*. Solusi optimisasi terbaik akan ditentukan berdasarkan metrik AP_{50}. Subbab 3.2 akan membahas tentang kandidat modifikasi-modifikasi yang dapat dilakukan.

Tahapan pencarian solusi optimisasi sendiri dapat dibagi menjadi enam tahap. Tahap-tahap tersebut adalah Persiapan Dataset, Pembuatan *Auto-trainer*, Pembuatan Konfigurasi Modifikasi, *Training Model*, Analisis, dan Pemilihan Model Terbaik. Urutan pengerjaan dari tahap-tahap ini dapat dilihat pada Gambar 3.1.



Gambar 3.1: Urutan Pengerjaan Penelitian

Pada tahap persiapan dataset, akan dilakukan pengunduhan dataset dari "Airborne Object Tracking Dataset" (2021). Gambar-gambar pada dataset ini kemudian akan di-*sampling* dan didistribusikan menjadi dataset *training*, validasi, dan pengujian. Pada pendistribusian dataset ini juga akan dilakukan *balancing* antar kelas dan dataset positif negatif. *Balancing* dilakukan agar tidak ada kelas yang mendominasi pada dataset.

Selanjutnya, di tahap pembuatan *auto-trainer*, akan dilakukan pembuatan program yang dapat dengan otomatis membangun dan melatih *neural network* modifikasi YOLOv7. Pembuatan *auto-trainer* ini dilakukan agar proses-proses pengerjaan tahapan-tahapan selanjutnya menjadi dapat dilakukan dengan lebih mudah.

Setelah itu, akan dilakukan pembuatan konfigurasi-konfigurasi modifikasi. Konfigurasi modifikasi YOLOv7 akan dibuat agar dapat diinputkan pada *auto-trainer*. Konfigurasi modifikasi akan berisi kombinasi modifikasi-modifikasi yang ada pada subbab 3.2.

Tahapan selanjutnya adalah *Training* model. Pada tahap ini, konfigurasi-konfigurasi mod-

ifikasi pada tahapan sebelumnya akan dibangun dan kemudian dilatih. Model akan dilatih *from scratch*. Dengan kata lain, model tidak akan dilatih dengan menggunakan *pre-trained weights*. Hasil dari tahap ini adalah *weights neural network* modifikasi YOLOv7, histori pelatihannya, dan metrik-metriknya pada dataset uji.

Pada tahap analisis, hasil dari tahap sebelumnya akan dianalisis untuk mencari tahu performa model-model hasil modifikasi. Analisis juga dilakukan untuk menemukan *gap* kandidat modifikasi lain yang masih bisa dieksplorasi untuk meningkatkan kemampuan pendeteksian objek kecil YOLOv7. Ketika suatu kandidat modifikasi seperti itu ditemukan, maka akan dilakukan kembali pembuatan konfigurasi modifikasi untuk menguji kandidat modifikasi baru tersebut.

Tahapan terakhir adalah pemilihan model terbaik. Pada tahapan ini akan dipilih model yang memiliki performa terbaik dari antara model-model hasil modifikasi lainnya. Pemilihan model akan dilakukan dengan berdasarkan pada skor mAP tertinggi. Untuk mempertahankan solusi optimisasi yang dapat melakukan deteksi secara *real time*, model yang akan dipilih adalah model yang dapat melakukan deteksi yang cukup cepat pada *edge* GPU.

3.2 Kandidat Modifikasi

3.2.1 Augmentasi Mosaik

Seperti yang dibahas pada subbab 2.3, augmentasi mosaic pada dataset mampu meningkatkan akurasi deteksi objek-objek kecil dari model. Oleh karena itu, akan dilakukan eksperimen menambahkan augmentasi mosaik pada YOLOv7 untuk melihat apakah augmentasi ini akan meningkatkan akurasi, khususnya pada dataset objek kecil.

3.2.2 Rekalkulasi *Anchor*

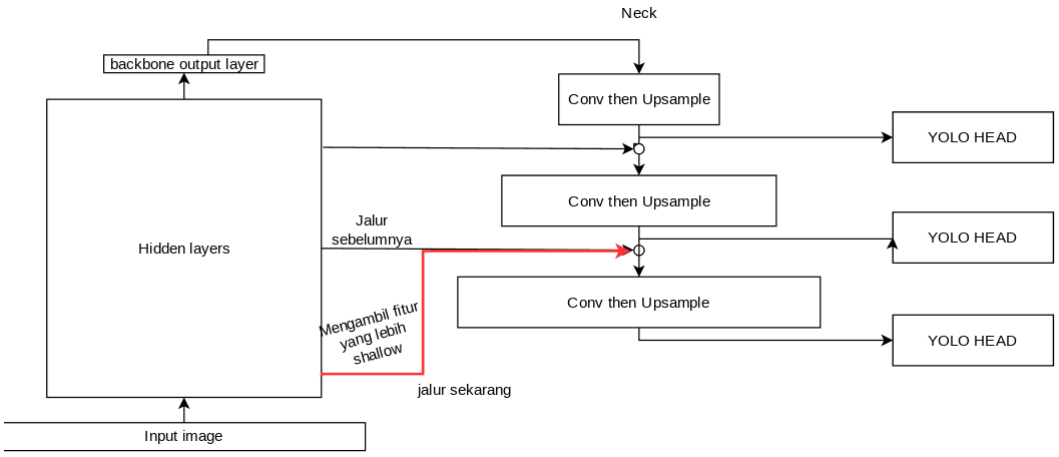
Anchor yang disediakan pada kode implementasi dari YOLOv7 merupakan anchor yang dikalkulasi untuk mengoptimisasi deteksi pada dataset COCO2017. Dengan pertimbangan bahwa distribusi dataset yang akan digunakan pada penelitian ini berbeda dari objek general di COCO2017, maka anchor harus direkalkulasi. Dengan melakukan rekalkulasi anchor, tiap layer head pada YOLOv7 akan dengan lebih mudah mem-*fit* objek-objek yang ada pada gambar.

3.2.3 Menggunakan *Localization Loss Extended IoU*

Extended IoU (EIoU) merupakan salahsatu modifikasi dari IoU yang dibuat untuk menyelesaikan permasalahan *vanishing gradient* pada IoU (Peng & Yu, 2021). Hal yang menyebabkan IoU bermasalah dengan *vanishing gradient* adalah nilai dari IoU yang selalu menjadi 0 ketika 2 *bounding box* tidak beririsan. Permasalahan ini diselesaikan oleh EIoU dengan memberikan nilai negatif untuk bounding box yang tidak beririsan, sehingga neural network dapat mengoptimasi loss function $-EIoU$. Teknik konveksikasi juga dapat dilakukan dengan mengoptimasi $(1 - EIoU)^2$

YOLOv7 sendiri menggunakan CIoU sebagai localization loss. CIoU juga merupakan suatu solusi dari permasalahan *vanishing gradient*. CIoU menambahkan suku jarak antar bounding box dan kecocokan *aspect ratio* antar boundingbox pada IoU. Keunggulan EIoU daripada CIoU adalah EIoU akan bertingkah seperti IoU ketika bounding box beririsan. Karena metrik-metrik yang digunakan untuk mengukur kemampuan deteksi dilakukan berdasarkan IoU, dianggap akan lebih baik jika loss yang digunakan sama seperti metriknya (Peng & Yu, 2021).

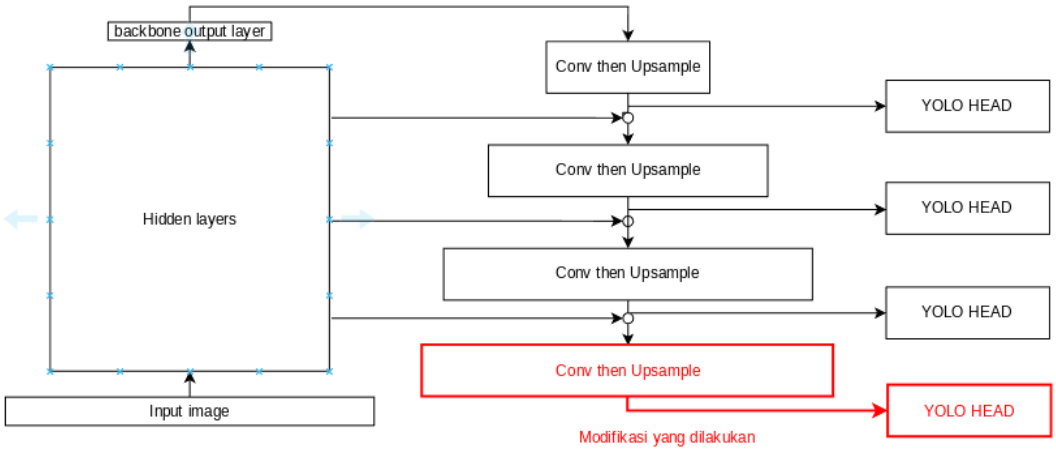
3.2.4 Modifikasi Koneksi Neck pada Backbone



Gambar 3.2: Menggunakan *feature map* dari layer yang lebih di belakang

Seperti pada penelitian-penelitian terkait di subbab 2.4, modifikasi *neck* dapat dilakukan untuk meningkatkan akurasi pendeteksian objek kecil. Modifikasi koneksi *neck* ke *backbone* dapat dilakukan dengan memindahkan sumber feature map untuk beberapa layer neck lebih jauh ke belakang seperti pada Gambar 3.2. Pemindahan sumber feature map ke belakang dapat dilakukan untuk mengantisipasi fitur objek kecil yang bisa saja hilang ketika layer neural network semakin dalam. Dengan memindahkannya lebih ke belakang, YOLOv7 akan melakukan deteksi dengan memanfaatkan fitur yang abstraksi yang lebih rendah tetapi sedikit informasi yang hilang.

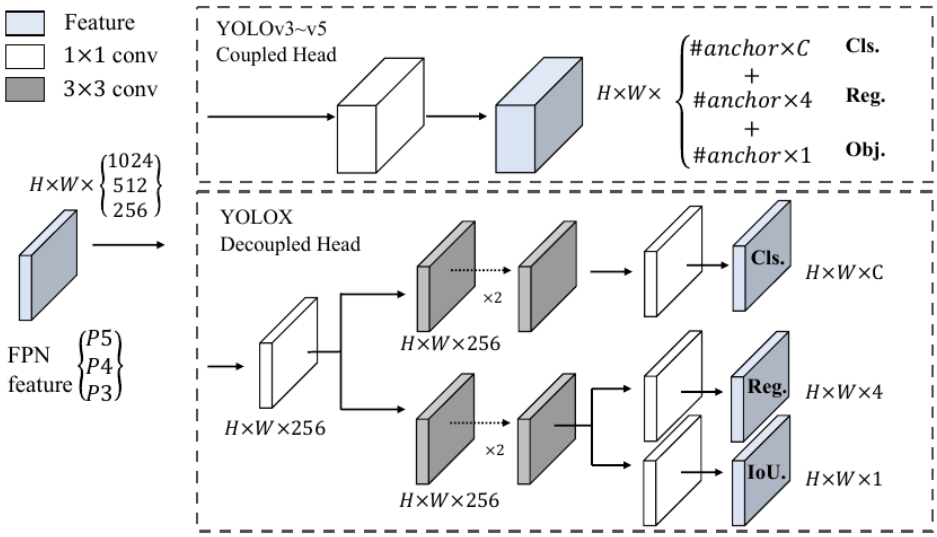
3.2.5 Penambahan YOLO Head



Gambar 3.3: Penambahan *Layer* Head pada YOLO

Penambahan YOLO head dapat membuat YOLOv7 melakukan deteksi pada skala yang lebih banyak. Hal ini akan berpengaruh pada kemampuan pendeteksian objek kecil. Dengan melakukan pendeteksian pada skala yang lebih banyak, YOLOv7 dapat mendeteksi objek yang besar maupun kecil.

3.2.6 Mengganti YOLO Head menjadi *Decoupled Anchor-free Head*



Gambar 3.4: *Decoupled Anchor-free Head* pada YOLOX (Ge et al., 2021)

Pada YOLOX, *coupled anchor head* seperti pada arsitektur YOLO pada umumnya diganti dengan *decoupled anchor-free head* (Ge et al., 2021). Keuntungan dari model anchor-free adalah kita tidak perlu mendefinisikan *anchor* sebelum melakukan training sehingga mengurangi beberapa proses tuning heuristik seperti rekalkulasi anchor. Memakai *head* yang anchor free juga memberi kompleksitas proses training dan pendekodean output model. Ge et al., 2021 melaporkan peningkatan pada akurasi dan pengurangan parameter sehingga mempercepat lama deteksi. Oleh karena hal-hal tersebut, mencoba memakai *head anchor-free* di YOLOv7 baik untuk dicoba.

3.3 Dataset

3.3.1 Sumber Dataset

Dataset untuk objek-objek *airborne* didapatkan dari "Airborne Object Tracking Dataset" (2021) dan dihost pada suatu server AWS S3 Bucket. Dataset ini berisi video-video monokromatik penerbangan UAV. Terdapat 4 kelas pada dataset ini yaitu pesawat, helikopter, burung, dan *other*. Distribusi dataset training dan uji dapat dilihat pada tabel 3.1 sedangkan distribusi kelas dari dataset dapat dilihat pada Tabel 3.2.

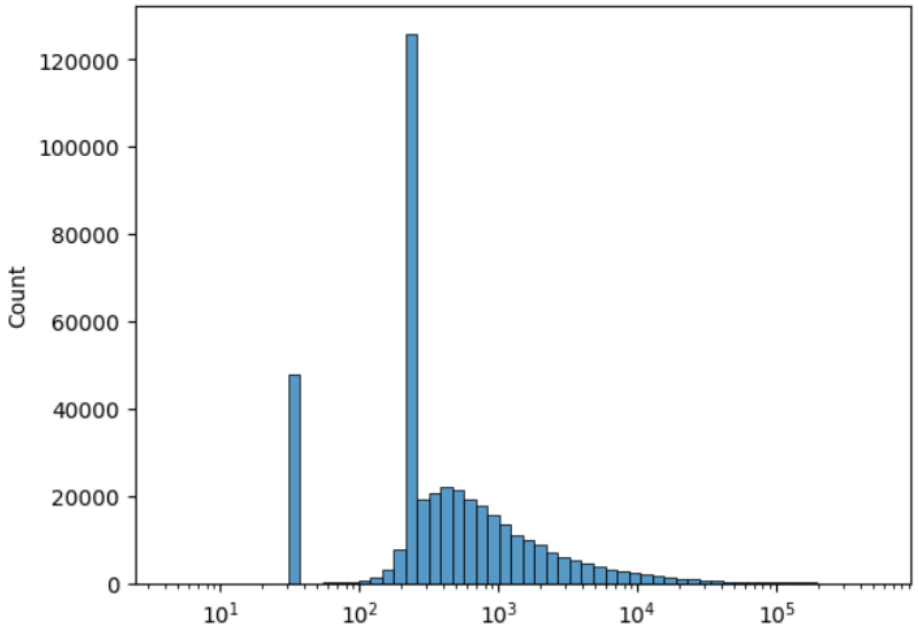
Tabel 3.1: Distribusi Dataset Training dan Test

Pembagian Dataset	Ukuran (TB)	Sekuen penerbangan UAV	Jumlah Gambar	Jumlah Label
Training	11,3	4154	4975765	2891891
Validation + Test	2.1	789	943852	496075
Total	13,4	4943	5919617	3387966

Tabel 3.2: Distribusi Kelas Dataset

Pembagian	Total Objek	Pesawat	Helikopter	Burung	*Other*
Training	2,89 juta	0,79 juta	1,22 juta	0,33 juta	0,54 juta
Validation + Test	0,50 juta	0,13 juta	0,17 juta	0,06 juta	0,14 juta
Total	3,39 juta	0,92 juta	1,39 juta	0,39 juta	0,69 juta

Kebanyakan *bounding box* pada dataset berukuran sangat kecil. Distribusi dari luas *boudning box* dapat dilihat pada gambar 3.5. Perhatikan pada gambar tersebut, sumbu x menggunakan skala logaritmik. Sebagai referensi, luas dari tiap frame itu sekitar $5 \times 10^1 6px^2$.



Gambar 3.5: Penambahan *Layer* Head pada YOLO

3.3.2 Sampling Dataset

Karena jumlah dataset pada "Airborne Object Tracking Dataset" (2021) berukuran sangat besar, dan keterbatasan *computational resource*, hanya sebagian dari dataset tersebut akan digunakan untuk *training* dan *test*. Akan diambil total 700 gambar dari dataset dengan pembagian sesuai dengan Tabel 3.3

Tabel 3.3: Distribusi Sampling Dataset

Pembagian	Total	Presentase				
	Gambar	Pesawat	Helikopter	Burung	Drone	Negatif
Training	400	23,75%	23,75%	23,75%	23,75%	5%
Validation	100	20%	20%	20%	20%	20%
Test	200	20%	20%	20%	20%	20%

3.4 Instrumen dan *Setup* Eksperimen

Untuk melaksanakan eksperimen ini, akan digunakan suatu komputer yang dilengkapi dengan GPU Nvidia RTX 2080 Ti yang memiliki kapasitas 11GB VRAM. Oleh karena keterbatasan ini, Melatih model-model YOLOv7 besar seperti W6, E6, dan E2E yang dimodifikasi menjadi sangat sulit, apalagi pada skala input sesuai dengan dimensi dataset. Oleh karena hal ini, arsitektur yang akan dipilih sebagai *baseline* modifikasi adalah YOLOv7 dengan ukuran normal karena model tersebut adalah model terbesar yang mampu di-*train* pada RTX 2080 Ti dengan input size 1600x1600 dan batch size 1.

Selain itu, jumlah dataset yang akan digunakan juga akan dibatasi menjadi 400 seperti pada subbab 3.3 untuk menghemat waktu training. Pada pilot test, ditemukan bahwa dibutuhkan sekitar 20 jam untuk men-*train* model sebanyak 300 epoch pada dataset dengan 400 gambar jika menggunakan GPU RTX 2080 Ti.

BAB IV
HASIL DAN PEMBAHASAN

Pada bab ini, akan dipaparkan pengaruh modifikasi-modifikasi yang dilakukan pada YOLOv7.

4.1 Performa Awal

Untuk mengukur pengaruh dari modifikasi-modifikasi yang dilakukan pada YOLOv7, maka hal pertama yang harus dilakukan adalah mengukur performa YOLOv7 tanpa segala modifikasi yang diajukan pada bab 3.2. Arsitektur YOLOv7 *plain* ini di-*train* pada 400 sampel data dari "Airborne Object Tracking Dataset" (2021) dengan 300 epoch dan batch size 1. Dengan aturan tersebut, ditemukan bahwa model *plain* tidak mampu untuk mendeteksi objek apapun pada dataset uji, dengan kriteria "terdeteksi" $IoU > 0.5$ (mAP@.5 = 0).

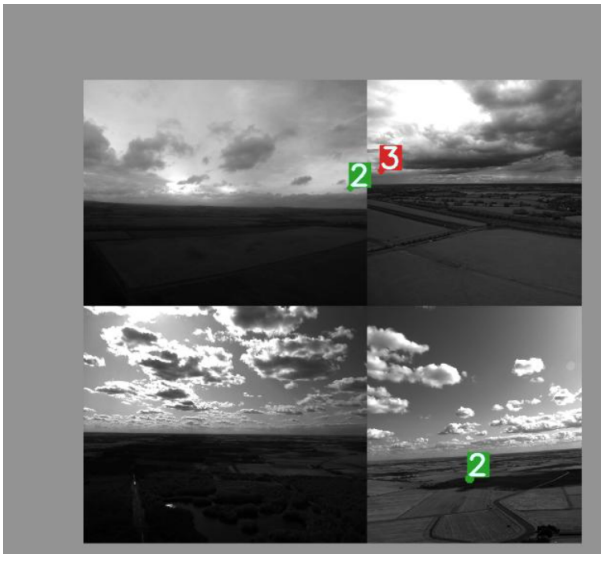
Untuk keperluan komparasi dengan performa-performa dari modifikasi pada YOLOv7, model *plain* ini akan selanjutnya disebut sebagai `YOLOv7-plain`.

4.2 Pengaruh Augmentasi Mosaic dan Rekalkulasi Anchor

Terdapat 3 modifikasi yang diujikan pada bagian ini, yaitu `YOLOv7-plain` yang ditambahkan augmentasi mosaic, `YOLOv7-plain` yang direkalkulasi anchor, dan `YOLOv7-plain` yang ditambahkan augmentasi mosaic dan rekalkulasi anchor.

4.2.1 Augmentasi Mosaic

Proses melakukan augmentasi mosaic cukup *straightforward*, augmentasi ini dilakukan pada beberapa data training. Contoh hasil augmentasi dapat dilihat pada gambar 4.1



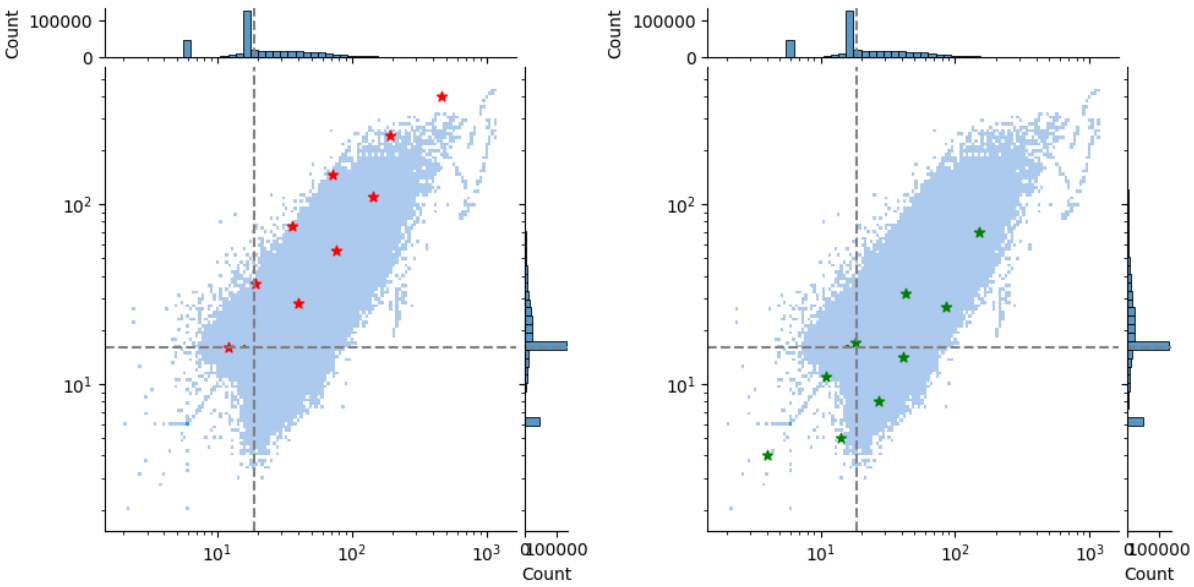
Gambar 4.1: Contoh Augmentasi Mosaic pada Dataset Training

4.2.2 Rekalkulasi Anchor

Rekalkulasi anchor dilakukan dengan mengklaster data training ke 9 centroid menggunakan algoritma k-means. Sembilan centroid tersebut digunakan sebagai anchor, 3 untuk tiap head pada arsitektur YOLO (terdapat 3 head). Persebaran anchor sebelum dan sesudah direkalkulasi dapat dilihat pada Tabel 4.1 dan Gambar 4.2

Tabel 4.1: Titik Hasil Rekalkulasi Anchor

Layer Head	Anchor Lama	Hasil Rekalkulasi Anchor
1	[12,16], [19,36], [40,28]	[4,4], [14,5], [11,11]
2	[36,75], [76,55], [72,146]	[27,8], [18,17], [41,14]
3	[142,110], [192,243], [459,401]	[43,32], [86,27], [149,70]



Gambar 4.2: Persebaran Anchor. Kiri: Anchor Lama. Kanan: Anchor Hasil Rekalkulasi

Jika kita memperhatikan persebaran anchor sebelum dan sesudah direkalkulasi pada Gambar 4.2, dapat kita lihat bahwa anchor hasil rekalkulasi lebih mencakup seluruh persebaran dataset daripada anchor lama. 8 dari 9 anchor lama bertempat di kuadran pertama dari garis median(garis putus-putus). Hal ini berarti 8 anchor tersebut hanya mampu mendeteksi sekitar 25% dari objek-objek pada dataset. Sedangkan, anchor hasil rekalkulasi menempatkan anchor di setiap kuadran.

4.2.3 Performa Augmentasi Mosaik dan Rekalkulasi Anchor

Performa dari tiap modifikasi dapat dilihat pada tabel 4.2. Pada tabel tersebut, terlihat bahwa YOLOv7 mampu untuk mendeteksi beberapa objek pada dataset uji ketika diberi augmentasi mosaic pada data train dan direkalkulasi anchornya.

Tabel 4.2: Performa Modifikasi Augmentasi Mosaic dan Rekalkulasi Anchor

No	Modifikasi	mAP@50
0	**yolov7-plain**	0%
1	**mosaic**	0%
2	**rekalkulasi anchor**	0%
3	**mosaic + rekalkulasi anchor**	**11,2%**
	Peningkatan	**+11,2%**

Hanya modifikasi nomor 3 yang mampu melakukan deteksi, maka modifikasi tersebut dijadikan baseline untuk modifikasi-modifikasi lainnya. Untuk mempermudah komparasi penambahan modifikasi-modifikasi selanjutnya, maka model ini akan disebut sebagai `YOLOv7-base`.

4.3 Pengaruh Penggantian *Box Loss Function*

Dengan menggunakan `YOLOv7-base` sebagai baseline, Box Loss function dari YOLOv7 diganti menjadi EIoU. Telah juga dilakukan percobaan menggunakan convexciation pada EIoU. Hasil dari pengujian dapat dilihat pada Tabel 4.3

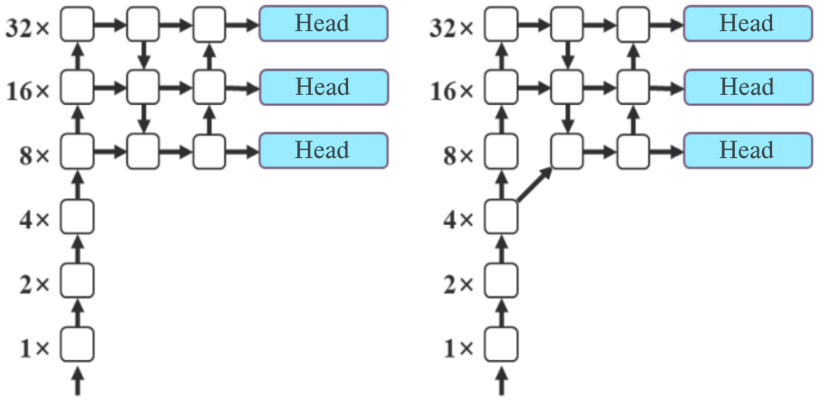
Tabel 4.3: Performa Penggantian Loss Function

No	Modifikasi	mAP@50
0	**yolov7-base + CIoU (original)**	**11,2%**
1	**yolov7-base + EIoU**	0%
2	**yolov7-base + EIoU + Covexication**	4,92%
	Peningkatan	**-6,28%**

Ternyata, meskipun EIoU memiliki performa lebih baik daripada CIoU ketika diaplikasikan pada Faster-RCNN+ResNet50 dengan dataset VOC2007 dan COCO2017, EIoU tidak mampu untuk meningkatkan AP deteksi YOLOv7 pada dataset "Airborne Object Tracking Dataset" (2021).

4.4 Pengaruh Perubahan Koneksi *Neck-Backbone*

Untuk modifikasi ini, koneksi *neck-backbone* yang diubah adalah koneksi layer neck yang memberikan feature pada *head* pertama yang awalnya terkoneksi dengan skala 8 dari backbone, dipindahkan ke skala 4. Hal ini diilustrasikan pada Gambar 4.3.



Gambar 4.3: Modifikasi Koneksi Neck. Kiri : Sebelum. Kanan : Sesudah.

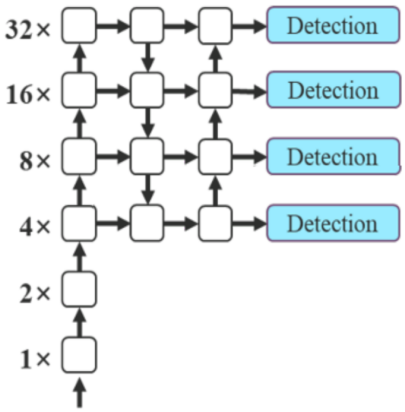
Tabel 4.4: Performa Modifikasi Koneksi *Neck-Backbone*

No	Modifikasi	mAP@50
0	**yolov7-base**	11,2%
1	**yolov7-base + modifikasi neck-backbone**	**14,09%**
	Peningkatan	**+2,98%**

Perbandingan performa modifikasi ini dengan `YOLOv7-base` dapat dilihat pada tabel 4.4. Terlihat bahwa modifikasi ini berhasil meningkatkan skor mAP@50 dari `YOLOv7-base` sebesar 2,98%. Untuk mempermudah perbandingan dengan modifikasi lain, model hasil modifikasi ini akan disebut `YOLOv7-moveconnection`

4.5 Pengaruh Penambahan *Head*

Untuk modifikasi ini, pada skala 4 backbone, dipasangkan suatu layer head tambahan. Ilustrasi penambahan layer ini dapat dilihat pada gambar 4.4.



Gambar 4.4: Penambahan Layer Head

Tabel 4.5: Performa Penambahan Layer Head

No	Modifikasi	mAP@50
0	**yolov7-base**	**11,2%**
1	**yolov7-base + head tambahan**	5,19%
	Peningkatan	**-6%**

Seperti yang dapat dilihat pada tabel 4.5, penambahan modifikasi ini memberi performa yang lebih buruk dibandingkan `YOLOv7-base`. Padahal, modifikasi penambahan layer head dan `YOLOv7-moveconnection` dua-duanya menggunakan fitur pada skala 4. Alasan untuk hal ini akan diinvestiagsi dengan melihat output dari tiap skala pada model penambahan head dan `YOLOv7-moveconnection`.

4.6 Pengaruh Pengubahan *Head* menjadi *Anchor-Free*

Penggantian *head* menjadi *decoupled anchor-free head* membuat model tidak mampu mendeteksi apapun pada dataset uji (mAP=0%).

Tabel 4.6: Performa *Anchor-free Head*

No	Modifikasi	mAP@50
0	**yolov7-base (anchor head)**	**11,2%**
1	**yolov7-base + (anchor-free head)**	0%

[Halaman ini sengaja dikosongkan]

BAB V
PENUTUP

5.1 Kesimpulan Sementara

Berdasarkan hasil pengujian yang dilakukan, dapat disimpulkan bahwa: Dari himpunan modifikasi-modifikasi yang diajukan pada penelitian ini, didapatkan kombinasi modifikasi yang paling optimal. Modifikasi tersebut penambahan augmentasi mosaik, rekalkulasi anchor, dan pemindahan koneksi neck-backbone. Dengan modifikasi itu, skor mAP@50 dari YOLOv7 meningkat sebanyak 14,09%.

5.2 Diskusi

Modifikasi-modifikasi yang diaplikasikan pada penelitian ini hingga saat ini masih dibatasi pada modifikasi yang tidak secara signifikan mempengaruhi *latency*. Beberapa modifikasi seperti melakukan partisi pada gambar, dan melakukan deteksi di tiap partisinya. Dengan metode ini, objek yang dipelajari akan terlihat lebih besar sehingga lebih mudah untuk dideteksi. Akan tetapi, latency pendeteksian untuk tiap gambar akan meningkat sebanyak jumlah partisi kali lipat. Untuk mengatasi hal tersebut, kita dapat melakukan down-scaling pada model yang digunakan. Modifikasi jenis ini perlu dicoba untuk menentukan jumlah partisi dan down-scaling model yang tepat untuk mempertahankan kecepatan deteksi yang *real-time*.

[Halaman ini sengaja dikosongkan]

DAFTAR PUSTAKA

Aditya, N., Indaryo, A., Solang, D., Santiung, M., Atmaja, H., Azis, A., Januar, F., Javanica, F., Kautaman, G., Kazaksti, E., Nugraha, D., Permadani, R., Ramadhan, R., Ramadhan, R., Damayanti, Z., Valentia, M., Sundana, P., Shodiq, F., Waisnawa, R., … Dikairono, R. (2022). Roboboat 2022: Technical design report Barunastra ITS roboboat team.

Airborne object tracking challenge. (2021). Retrieved September 1, 2022, from https://www.aicrowd.com/challenges/airborne-object-tracking-challenge

Airborne object tracking dataset. (2021). Retrieved September 1, 2022, from https://registry.opendata.aws/airborne-object-tracking

Amazon. (2022). *Amazon prime air prepares for drone deliveries*. Retrieved September 1, 2022, from https://www.aboutamazon.com/news/transportation/amazon-prime-air-prepares-for-drone-deliveries

Benjumea, A., Teeti, I., Cuzzolin, F., & Bradley, A. (2021). Yolo-z: Improving small object detection in yolov5 for autonomous vehicles.

Bochkovskiy, A., Wang, C.-Y., & Liao, H.-Y. M. (2020). Yolov4: Optimal speed and accuracy of object detection.

Ge, Z., Liu, S., Wang, F., Li, Z., & Sun, J. (2021). Yolox: Exceeding yolo series in 2021.

Jocher, G., Chaurasia, A., Stoken, A., Borovec, J., NanoCode012, Kwon, Y., Michael, K., TaoXie, Fang, J., imyhxy, & et al. (2022). Ultralytics/yolov5: V7.0 - yolov5 sota realtime instance segmentation. https://doi.org/10.5281/zenodo.7347926

Peng, H., & Yu, S. (2021). A systematic iou-related method: Beyond simplified regression for better localization. *CoRR*, *abs/2112.01793*. https://arxiv.org/abs/2112.01793

Redmon, J., & Farhadi, A. (2016). Yolo9000: Better, faster, stronger.

Redmon, J., & Farhadi, A. (2018). Yolov3: An incremental improvement.

Wang, C.-Y., Bochkovskiy, A., & Liao, H.-Y. M. (2020). Scaled-YOLOv4: Scaling cross stage partial network.

Wang, C.-Y., Bochkovskiy, A., & Liao, H.-Y. M. (2022). YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors.

Wang, C.-Y., Yeh, I.-H., & Liao, H.-Y. M. (2021). You only learn one representation: Unified network for multiple tasks.

Xiao, J. (2021). exYOLO: A small object detector based on YOLOv3 object detector. *Procedia Computer Science*, *188*, 18–25. https://doi.org/10.1016/j.procs.2021.05.048

Zhong, Y., Wang, J., Peng, J., & Zhang, L. (2018). Anchor box optimization for object detection.

[Halaman ini sengaja dikosongkan]