TESIS

PENINGKATAN PERFORMA *YOU ONLY LOOK ONCE V4* DENGAN *HYBRID HYPERPARAMETER LOSS FUNCTION*

Improved You Only Look Once V4 Performance using Hybrid Hyperparameter Loss Function

NUGRA TASIK ALLO
D082212012

PROGRAM STUDI MAGISTER TEKNIK INFORMATIKA

UNIVERSITAS HASANUDDIN

GOWA

2023

PENGAJUAN TESIS

PENINGKATAN PERFORMA *YOU ONLY LOOK ONCE V4* DENGAN *HYBRID HYPERPARAMETER LOSS FUNCTION*

Tesis
Sebagai Salah Satu Syarat untuk Mencapai Gelar Magister
Program Studi Magister Teknik Informatika

Disusun dan diajukan oleh

NUGRA TASIK ALLO
D082212012

Kepada

FAKULTAS TEKNIK
UNIVERSITAS HASANUDDIN
GOWA
2023

TESIS

PENINGKATAN PERFORMA *YOU ONLY LOOK ONCE V4* DENGAN *HYBRID HYPERPARAMETER LOSS FUNCTION*

NUGRA TASIK ALLO
D082212012

telah dipertahankan di hadapan Panitia Ujian Magister pada 26 Juni 2024 dan dinyatakan telah memenuhi syarat kelulusan

pada

Program Studi Teknik Informatika
Departemen Teknik Informatika
Fakultas Teknik
Universitas Hasanuddin
Gowa

Mengesahkan:

Pembimbing utama,

Prof.Dr.Ir. Indrabayu,ST.,MT.,M.Bus.Sys.,IPM,ASEAN,Eng.
NIP. 197310101998021001

Pembimbing Pendamping,

Dr.Ir. Zahir Zainuddin,M.Sc.
NIP. 196404271989101002

Ketua Program Studi
Magister Teknik Informatika,

Dr.Ir. Zahir Zainuddin,M.Sc.
NIP. 196404271989101002

Dekan Fakultas Teknik
Universitas Hasanuddin,

Prof. Dr.Eng.Ir. Muhammad Isran Ramli,M.T.,IPM., ASEAN.Eng.
NIP. 197309262000121002

PERNYATAAN KEASLIAN TESIS

DAN PELIMPAHAN HAK CIPTA

Yang bertanda tangan di bawah ini

Nama : Nugra Tasik Allo

Nomor mahasiswa : D082212012

Program studi : Magister Teknik Informatika

Dengan ini menyatakan bahwa, tesis yang berjudul "PENINGKATAN PERFORMA YOU ONLY LOOK ONCE V4 DENGAN HYBRID HYPERPARAMETER LOSS FUNCTION" adalah benar karya saya dengan arahan dari komisi pembimbing Prof. Dr. Ir. Indrabayu, ST., MT., M.Bus.Sys., IPM, ASEAN, Eng. dan Dr. Ir. Zahir Zainuddin, M.Sc.. Karya ilmiah ini belum diajukan dan tidak sedang diajukan dalam bentuk apapun kepada perguruan tinggi manapun. Sumber informasi yang berasal atau dikutip dari karya yang diterbitkan maupun tidak diterbitkan dari penulis lain telah disebutkan dalam teks dan dicantumkan dalam Daftar Pustaka tesis ini.

Dengan ini saya melimpahkan hak cipta ini dari karya tulis saya berupa tesis ini kepada Universitas Hasanuddin.

Gowa, 19 Agustus 2024

Yang menyatakan

Nugra Tasik Allo

KATA PENGANTAR

Puji dan Syukur Penulis panjatkan atas Kehadirat Tuhan Yang Maha Esa karena berkat Rahmat dan Karunia-Nya sehingga penulis dapat menyelesaikan Karya Ilmiah berupa Tesis ini yang berjudul "Peningkatan Performa *You Only Look Once V4* dengan *Hybrid Hyperparameter Loss Function*". Penyusunan Tesis ini merupakan salah satu syarat untuk memperoleh gelar Magister Komputer (M.Kom) pada Program Studi Magister Teknik Informatika, Departemen Teknik Informatika, Fakultas Teknik, Universitas Hasanuddin. Penulis ingin menyampaikan ucapan terima kasih yang sebesar-besarnya atas dukungan dan semangat yang diberikan serta membantu Penulis baik secara langsung ataupun tidak langsung dalam menyelesaikan Karya Ilmiah ini. Penulis ingin mengucapkan terima kasih kepada :

1. Tuhan Yesus Kristus karena atas Rahmat dan kasih-Nya sehingga penulis dapat menyelesaikan tesis ini dengan baik.
2. Keluarga penulis yang telah memberikan dukungan dan doanya, sehingga penulis bisa menyelesaikan studi hingga selesai.
3. Bapak Dr. Ir. Zahir Zainuddin, M.Sc. sebagai Kordinator Prodi Magister Teknik Informatika, sekaligun pembimbing pendamping yang selalu memberikan arahan kepada Penulis agar cepat menyelesaikan segala rangkaian proses akademik di Departemen Teknik Informatika.
4. Bapak Prof. Dr. Ir. Indrabayu, ST., MT., M.Bus.Sys., IPM, ASEAN, Eng. selaku pembimbing utama yang telah memberikan arahan dalam menyelesaikan tesis ini.
5. Seluruh bapak dan ibu dosen di Departemen Teknik Informatika yang telah memberikan ilmunya kepada kami mahasiswa untuk menjadi bekal menyusun tesis-tesis kami.
6. Seluruh Staff Administrasi Program Studi Magister Teknik Informatika yang telah memberikan bantuan administrasi selama kuliah di Fakultas Teknik, Universitas Hasanuddin.

7. Pengurus dan Anggota Himpunan Mahasiswa Magister Teknik Informatika yang senantiasa memberikan dukungan dan semangat kepada Penulis.
8. Dan juga kepada teman-teman Angkatan 5 Program Studi Magister Teknik Informatika yang telah mendukung penulis untuk menyelesaikan tesis ini.

Penulis mohon maaf yang sebesar-besarnya apabila terdapat kekurangan dalam penyusunan dan penulisan Tesis ini. Kritik dan Saran yang membangun penulis harapkan dari para Pembaca dan Peneliti serta Akademisi untuk perbaikan dan pembelajaran dikemudian hari, semoga tulisan ini dapat memberikan manfaat yang besar dan seluas-luasnya untuk kemajuan Ilmu pengetahuan di Indonesia pada umumnya dan ilmu dibidang komputer pada khususnya.

Gowa, 16 Agustus 2024

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

ABSTRAK

Loss function merupakan salah satu dari banyaknya *hyperparameter* pada algoritma deteksi objek yang digunakan untuk mengukur sejauh mana prediksi model dari nilai sebenarnya. IoU (*Intersection over Union*) merupakan metrik yang digunakan untuk mengevaluasi prediksi tersebut. *Loss function* akan memberikan umpan balik kepada model untuk memperbaiki *parameter* tersebut sehingga prediksi dan lokalisasi bisa lebih baik. Algoritma YOLOv4 memberikan beberapa opsi untuk *hyperparameter* yaitu GIoU, DIoU, dan CIoU (*default*). Namun, tiap *loss function* tersebut memiliki beberapa limitasi, seperti GIoU yang lambat dan dapat turun menjadi IoU, ataupun CIoU yang sensitif terhadap *bounding box* dan *ground truth* yang rasio boxnya sama, sehingga dapat turun ke DIoU dan mengurangi kecepatan konvergensi. Oleh karena itu, peneliti menawarkan *loss function* baru dimana dapat mengatasi limitasi kedua *loss function* tersebut. Penelitian yang ditawarkan adalah metode *hybrid* GIoU + CIoU. Dengan metode ini GIoU tidak akan turun menjadi IoU dan rasio kedua *box* pada CIoU akan selalu berbeda, sehingga dapat meningkatkan kemampuan GIoU dan CIoU pada formula *loss function* yang baru. Pada simulasi kecepatan konvergensi menunjukkan bahwa usulan *loss function* ini lebih cepat 14% dan dapat meningkatkan mAP sebanyak 2.20% dibanding CIoU loss pada dataset PASCAL VOC 2007 dengan menawarkan akurasi yang baik serta konsisten pada *threshold* tinggi. Hal serupa juga berlaku pada PASCAL VOC 2007 + 2012 yang diberikan *noise*, mAP meningkat sebanyak 0.5%. Terakhir, usulan *loss function* ini dapat mendeteksi lebih banyak objek positif, dan mengurangi *false positive* dibanding dengan CIoU *loss*, karena *loss* ini menawarkan kemampuan lokalisasi dan klasifikasi yang lebih baik.

Kata kunci : *YOLOv4, GIoU, CIoU, loss function,* deteksi objek, lokalisasi

ABSTRACT

The loss function is one of the many hyperparameters in object detection algorithms used to measure how far the model's predictions are from the actual values. IoU (Intersection over Union) is a metric used to evaluate these predictions. The loss function provides feedback to the model to improve these parameters, enhancing the accuracy and localization of predictions. The YOLOv4 algorithm offers several options for hyperparameters, including GIoU, DIoU, and CIoU (default). However, each loss function has limitations, such as GIoU being slow and potentially degrading to IoU, or CIoU being sensitive to bounding boxes and ground truths with the same box ratio, which can degrade to DIoU and reduce convergence speed. Therefore, researchers propose a new loss function to overcome these limitations. The proposed research is a hybrid method combining GIoU and CIoU. With this method, GIoU will not degrade to IoU, and the ratio of the two boxes in CIoU will always differ, thus improving the capabilities of both GIoU and CIoU in the new loss function formula. Simulation of convergence speed shows that this proposed loss function is 14% faster and can increase mAP by 2.20% compared to CIoU loss on the PASCAL VOC 2007 dataset, offering good and consistent accuracy at high thresholds. Similarly, on PASCAL VOC 2007 + 2012 with added noise, mAP increased by 0.5%. Finally, this proposed loss function can detect more positive objects and reduce false positives compared to CIoU loss, as it offers better localization and classification capabilities.

Keywords : *YOLOv4, GIoU, CIoU, loss function, object detection, localization*

BAB I

PENDAHULUAN

I.1 Latar Belakang

Visi komputer adalah suatu bidang dalam ilmu komputer yang berfokus pada pengembangan sistem dan teknologi yang memungkinkan komputer untuk "melihat" dan "memahami" dunia secara visual, seperti halnya manusia. Visi komputer mencakup beberapa aspek seperti pemrosesan citra digital, analisis pola, dan pengenalan objek [1]. Dalam visi komputer, gambar atau video input akan melewati beberapa proses ekstraksi informasi, bagian tertentu pada gambar yang telah diekstraksi akan diolah bergantung dari kasus yang ingin dicapai. Salah satu contohnya yaitu deteksi objek, dimana algoritma akan mengolah keseluruhan input gambar untuk mengidentifikasi lokasi dan objek didalamnya [2].

Deep learning dan *machine learning* merupakan pendekatan yang paling umum digunakan untuk visi komputer. Dalam beberapa tahun terakhir, *deep learning* telah memberikan kemajuan yang pesat dalam bidang visi komputer dengan mengalahkan *machine learning,* hal ini karena *deep learning* memungkinkan model komputasi menggunakan beberapa lapisan pemrosesan untuk belajar dengan meniru cara otak manusia memahami berbagai informasi, dengan demikian secara implisit *deep learning* dapat mempelajari struktur data yang rumit dalam skala yang besar [3]. Ambisi untuk menciptakan sistem yang mensimulasikan otak manusia menjadi dorongan awal dalam pengembangan *neural network*. Pada tahun 1943, McCulloch dan Pitts [4] mencoba memahami bagaimana otak dapat menghasilkan pola yang sangat kompleks dengan menggunakan sel-sel yang saling terhubung, disebut neuron. Model ciptaan McCulloch dan Pitts disebut model *MCP*, telah memberikan kontribusi penting dalam pengembangan *artificial neural network* atau jaringan saraf tiruan. *Neural network* yang menggunakan banyak lapisan *layer,* disebut sebagai *deep neural network,* atau *deep learning*, dari sinilah asal muasal istilah *deep learning* [5].

Deep learning bekerja dengan memasukkan data ke dalam jaringan syaraf tiruan. Setiap lapisan dalam jaringan syaraf tiruan melakukan transformasi pada data dan kemudian mengirimkan hasilnya ke lapisan berikutnya. Setiap lapisan menggunakan berbagai jenis fungsi matematika untuk menghitung nilai output yang sesuai dengan setiap input. Ketika data masuk ke dalam jaringan syaraf tiruan, nilai output yang dihasilkan oleh setiap lapisan diubah sedemikian rupa sehingga jaringan syaraf tiruan dapat mempelajari pola atau informasi yang ada dalam data tersebut. Jika jaringan syaraf tiruan sudah cukup banyak belajar dari data yang diberikan, maka jaringan syaraf tiruan dapat memprediksi hasil yang tepat untuk data baru yang belum pernah dilihat sebelumnya [6].

YOLO (*You Only Look Once*) merupakan salah satu algoritma deteksi objek secara *realtime* yang dibuat oleh Joseph Redmon, dkk pada tahun 2016 [7]. YOLO menjadi populer karena mampu memberikan *fps* (*frame per second*) yang tinggi, algoritma YOLO bekerja dengan menggunakan satu *convolitional network* yang memproses seluruh gambar sekaligus, bukan dengan memproses gambar secara terpisah untuk setiap objek. Hingga saat ini, algoritma YOLO telah mengalami peningkatan yang pesat setelah pertama kali diperkenalkan, saat penelitian ditulis YOLO sudah sampai ke versi 8 [8].

Dalam [7] YOLO membagi gambar *input* kedalam *S* x *S* bagian (*grid*), jika bagian tengah dari objek berada pada sebuah *grid*, maka *grid* tersebut akan bertanggungjawab untuk memprediksi lokasi dan kelas objek dalam sebuah gambar. Untuk setiap sel pada *grid* tengah, YOLO akan memprediksi beberapa *bounding box* yang kemungkinan mengandung objek. *Bounding box* tersebut didefinisikan dengan lima parameter: koordinat *x*, *y*, lebar (*width*), tinggi (*height*), dan *confidence score*. *Confidende score* merepresentasikan seberapa yakin model dalam mendeteksi objek, dapat dihitung dengan IoU (*Intersection over Union*) yaitu selisih dari *bounding box* hasil prediksi dengan *ground truth.* IoU selanjutnya akan dievaluasi dengan mengukur selisihnya, selanjutnya akan dioptimalkan menggunakan *loss function*.

Loss function merupakan salah satu elemen penting dalam *deep learning*. *Loss function* digunakan untuk mengukur seberapa baik model bekerja dalam memprediksi *output* yang benar. Setelah melakukan perhitungan pada loss function, jaringan syaraf tiruan mengoptimalkan parameter-parameter dalam model untuk meminimalkan nilai loss function tersebut [7]. Berbagai pengembangan telah dilakukan untuk meningkatkan akurasi dari model, seperti mengganti *probabilistic loss* L1-*norm* dan L2-*norm* menjadi IoU-*loss* [9] karena L1-*norm* dan L2-*norm* tidak dapat menghitung rasio kedua *bounding box*, menyebabkan tidak akuratnya prediksi [11] [12]. Namun, dalam [12] mengatakan bahwa menggunakan IoU sebagai *loss function* akan menyebabkan tidak mampunya model mengoptimalisasi kedua *bounding box* yang tidak tumpang tindih (*overlap*), sehingga dikembangkannya *loss* berbasis IoU bernama GIoU (*Generalized Intersection over Union*). GIoU mengatasi limitasi IoU dengan menambahkan *gradient* pada proses perhitungannya. GIoU mampu mengoptimalkan kedua *bounding box* yang tidak *overlap,* namun memerlukan komputasi yang lama, khususnya jika *bounding box* sejajar horizontal atau vertikal [13]. Pada tahun yang sama, DIoU (*Distance Intersection over Union*) dan CIoU (*Complete Intersection over Union*) datang dengan perhitungan *loss* yang beda, yaitu menggunakan *euclidean distance* untuk koordinat tengah *bounding box* yang tidak *overlap*. CIoU merupakan peningkatan dari DIoU dengan menambahkan perhitungan aspek rasio. Namun, keduanya masih memiliki limitasi, dimana ketika kedua koordinat tengah di posisi yang sama, DIoU akan turun menjadi IoU. Di sisi lain, ketika kedua aspek rasio *bounding box* sama, CIoU akan turun menjadi DIoU [14].

Berdasarkan uraian di atas, peneliti mengusulkan sebuah metode perhitungan *loss function* yang menggabungkan GIoU dengan CIoU. Penggabungan kedua *loss function* ini diharapkan dapat meningkatkan akurasi dan mengatasi limitasi GIoU yaitu komputasi horizontal/vertikal yang lama dengan *euclidean distance,* turunnya GIoU ke IoU dan limitasi CIoU yang berganti ke DIoU karena aspek rasio yang sama.

I.2 Rumusan Masalah

Berdasarkan latar belakang di atas, maka rumusan masalah pada penelitian ini adalah:

1. Bagaimana menggabungkan GIoU dan CIoU untuk mengatasi limitasi kedua *loss function* tersebut?
2. Bagaimana meningkatkan akurasi dari model algoritma menggunakan gabungan GIoU dan CIoU?
3. Bagaimana cara mengukur performa dari *loss function* yang ditawarkan?

I.3 Tujuan Penelitian

Tujuan yang ingin dicapai dari penelitian ini adalah sebagai berikut:

1. Menggabungkan GIoU dan CIoU untuk mengatasi limitasi dari kedua *loss function* tersebut
2. Meningkatkan akurasi dari model algoritma menggunakan gabungan GIoU dan CIoU
3. Mengetahui cara mengukur performa dari *loss function* yang ditawarkan.

I.4 Manfaat Penelitian

Manfaat yang dapat diperoleh dari penelitian ini adalah sebagai berikut:

1. Meningkatkan akurasi dari algoritma YOLO
2. Menjadi *literature* dalam memahami *loss funtion* pada algoritma YOLO
3. Menjadi referensi ilmiah bagi para mahasiswa atau peneliti lain pada bidang penelitian yang serupa

I.5 Batasan Masalah

Adapun batasan masalah dalam penelitian ini yaitu:

1. Penelitian ini hanya difokuskan untuk memodifikasi *loss function*
2. Proses *training* dilakukan menggunakan *local GPU*
3. Tolak ukur pertama perbandingan performa model menggunakan mAP *threshold* 50 sampai 95 dengan *step* 5 dan *fps*.

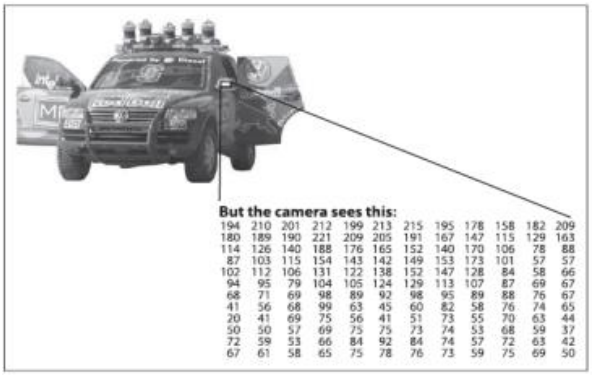
BAB II

KAJIAN LITERATUR DAN METODE PENYELESAIAN MASALAH

II.1 Kajian Literatur

II.1.1 Visi Komputer

Sebagai manusia kita dapat dengan mudah mengenali objek pada sebuah gambar, atau mengetahui berapa jumlah orang dalam grup foto sambil mengenali wajah-wajah mereka. Namun, bagaimana jika sebuah komputer dapat "melihat" serta "mengenali" apa yang manusia lihat? Inilah yang disebut dengan *computer vision*, atau visi komputer. Menurut buku [1] oleh Richard Szeliski, visi komputer adalah sebuah disiplin ilmu yang mencakup teknik dan metode untuk memproses, menganalisis, dan memahami data visual dari dunia nyata untuk menghasilkan informasi yang dapat dipahami oleh komputer demi mencapai suatu tujuan. Visi komputer bertujuan untuk mengembangkan algoritma dan teknologi yang dapat membantu komputer memahami dan menginterpretasikan data visual seperti gambar atau video, dan menghasilkan informasi yang bermanfaat dari data tersebut, seperti "adakah kucing dalam gambar tersebut?" atau "berapa kecepatan mobil pada video tersebut?". Visi komputer melibatkan berbagai teknik dan metode seperti pemrosesan citra, pengenalan pola, analisis statistik, *deep learning* dan *machine learning*. Visi komputer digunakan dalam berbagai bidang, termasuk pengolahan citra medis, pengawasan video, otomatisasi industri, atau pengenalan wajah.



Gambar 1 Ilustrasi Cara Visi Komputer Melihat Objek

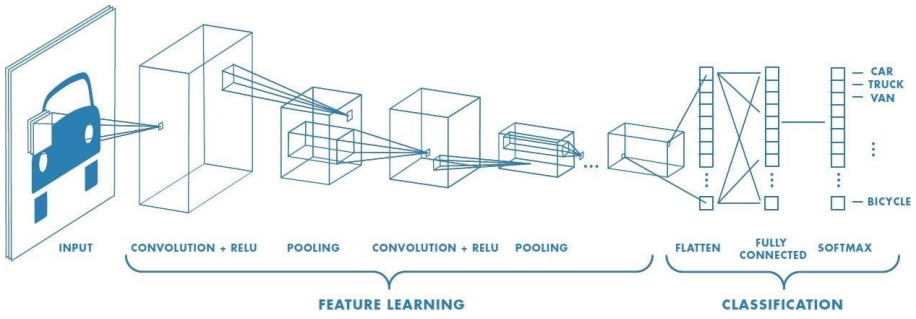
Dari Gambar 1, kita sebagai manusia melihat objek sebagai bentuk dan garis-garis. Namun bagi komputer hanyalah sekumpulan angka-angka. Gambar tersebut bisa jadi masih sulit untuk dikenali oleh komputer, karena "penglihatan" yang terbatas pada 2 dimensi, sedangkan objek tersebut adalah 3 dimensi pada dunia nyata. Hal lain yang bisa menjadi tantangan komputer untuk mengenali objek adalah *noise, blur,* resolusi yang rendah, dan *over/underexposure*[15]. Tantangan-tanganan tersebut terus menjadi bidang penelitian yang dilakukan untuk mencari teknik perbaikan lebih baik.

II.1.2 Konsep *Convolutional Neural Network*

Metode dalam penelitian *deep dearning* yang sering digunakan dalam pemrosesan gambar adalah *Convolutional Neural Network* (CNN). CNN termasuk dalam jenis *deep neural detwork* atau *deep* learning karena memiliki kedalaman jaringan yang tinggi, dan sering diterapkan pada data citra. *Neural network* yang pertama dikembangkan bernama *NeoCognitron* oleh Kunihiko Fukushima untuk pengenalan pola [16]. Konsep dari Kunihiko ini lalu dikembangkan oleh Yann LeChun dengan nama *LeNet* untuk pengenalan angka dan tulisan tangan [17]. Selama tahun-tahun awalnya, *deep learning* kurang diminati dibanding dengan *machine learning* karena terbatasnya kemampuan komputer. Deep learning mulai menjadi populer pada akhir tahun 2000-an dan awal 2010-an ketika teknologi komputasi mulai berkembang pesat dan jumlah data yang tersedia semakin banyak. Pada tahun 2012, terjadi titik balik dalam sejarah *deep learning* ketika sebuah tim dari University of Toronto memenangkan kompetisi

ImageNet dengan menggunakan model deep neural network yang disebut CNN berhasil mengalahkan *machine learning* [18].

Langkah awal pada struktur CNN adalah tahap *convolution*, di mana *filter* dengan ukuran tertentu digunakan untuk menghitung fitur-fitur pada gambar. Jumlah *filter* yang digunakan akan bervariasi tergantung pada berapa banyak fitur yang ingin dihasilkan. Setelah itu, gambar yang telah melalui tahap *convolution* akan melewati *activation function*, yang biasanya menggunakan fungsi ReLU (*Rectifier Linear Unit*), ada juga *Sigmoid*, Tanh, *softmax*, dan lain-lain. Setelah keluar dari tahap *activation funtion*, gambar akan melewati tahap *pooling* untuk mengurangi dimensi *feature map*. *Pooling* biasanya menggunakan operasi *max pooling* (mengambil nilai tertinggi), atau *average pooling* (mengambil rata-rata). Proses ini akan diulang beberapa kali sampai didapatkan *feature map* yang cukup untuk masuk ke dalam *fully connected neural network* dan dihasilkan *output class* (prediksi)[19].



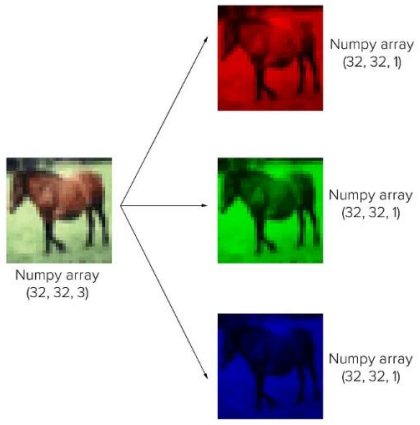
Gambar 2 Ilustrasi Cara Kerja CNN

Dilihat pada Gambar 2 dapat dilihat bahwa tahap awal dari arsitektur CNN adalah *convolutional process*. Seperti yang dijelaskan sebelumnya, tahap ini melakukan proses konvolusi, dari *layer* sebelumnya.

II.1.2.1 *Filter*

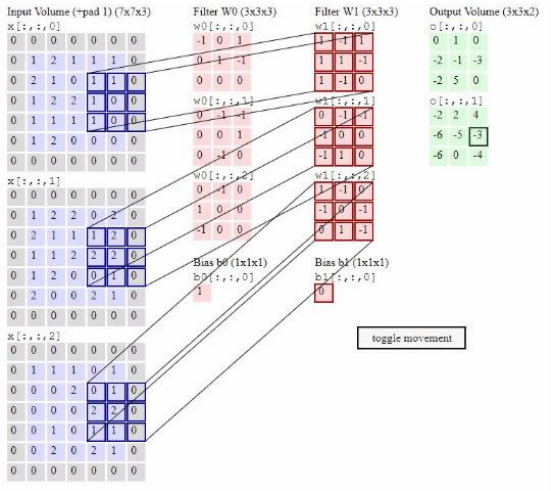
Gambar diatas adalah RGB (*red*, *green*, dan *blue*), dengan *size* 32x32 *pixel*, dimana jika dibaca oleh CNN menjadi multidimensional *array* 32x32x3 (3 adalah total *channel* warna pada gambar) pada Gambar 3. Proses *convolution* sederhananya adalah proses mencari *feature map* objek pada gambar, bisa garis, sudut atau tekstur. Proses

convolution terdiri dari neuron-neuron yang tersusun sedemikian rupa membentuk *filter* yang akan bergeser pada ketiga *channel* gambar untuk mencari *feature map*.



Gambar 3 Ilustrasi Diproses CNN Mengolah Gambar

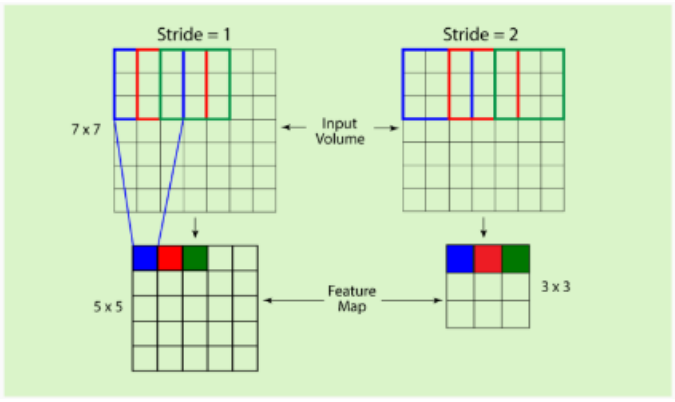
Filter atau biasa disebut *kernel* adalah suatu *matrix* yang lebih kecil dari ukuran input gambar, *filter* akan mendeteksi fitur-fitur dengan bergeser ke seluruh gambar secara bertahap. Proses pergeseran *filter* inilah yang disebut sebagai *convolution* [20]. Setiap pergeseran akan dilakukan operasi *dot* antara input dan *filter* tersebut sehingga menghasilkan output *feature map* [21]. Semakin banyak jumlah *layer convolution* dan *filter* didalamnya, maka semakin banyak juga informasi dan perulangan komputasi yang dilakukan [22]. Namun, perlu diperhatikan bahwa, tidak selamanya jumlah *filter* yang banyak akan memberikan *output* yang lebih baik, tetapi kadang menyebabkan model menjadi *overfitting* [23]. Ilustrasi *filter* dapat dilihat pada Gambar 4.



Gambar 4 Ilustrasi Cara Kerja *Filter* dalam Proses *Convolution*

II.1.2.2 *Stride*

Stride merupakan salah satu *hyperparameter* didalam CNN, dimana fungsinya adalah menentukan berapa jumlah pergeseran *filter* ketika melakukan proses *convolution*. Karena *convolution* adalah pencarian fitur menggunakan *filter*, maka *stride* adalah penentu seberapa banyak pergeseran *filter* dalam sebuah gambar [25]. Penentuan *stride* memberikan dampak bagi *output*, jika nilai *stride* kecil (paling kecil 1), maka proses *convolution* akan memakan banyak waktu, namun mendapatkan lebih banyak informasi dari gambar.

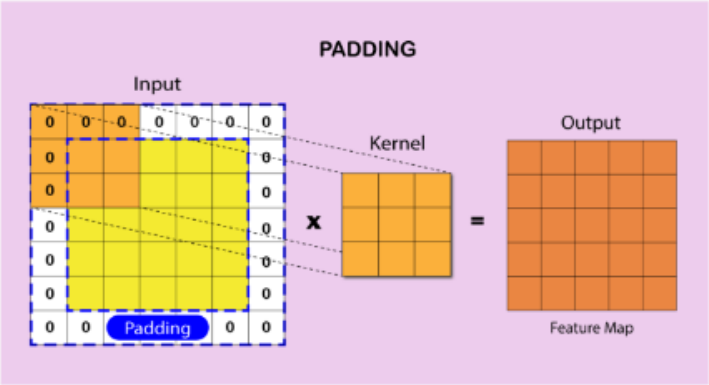


Gambar 5 Ilustrasi *Stride* Dalam Proses *Convolution*

Gambar di atas menunjukkan dengan cukup jelas peran *stride* dalam melakukan proses *convolution*. *Stride* 1 berarti *filter* bergerak sebanyak 1 *pixel* setiap melakukan proses *convolution* ke seluruh gambar. *Stride* 2 berarti *filter* bergerak sebanyak 2 *pixel*. Perlu dimengerti bahwa *stride* kecil menghasilkan *feature map* yang besar dengan lebih banyak informasi yang diperoleh. Sehingga penentuan nilai *stride* bergantung pada kasus tertentu dan juga karakteristik dari gambar. Semakin kompleks dan detail gambar *input*, maka lebih cocok menggunakan *stride* kecil. Ilustrasi *stride* dapat dilihat pada Gambar 5.

II.1.2.3 *Padding*

Dalam CNN, *padding* merujuk pada proses menambahkan nilai-nilai ke tepi gambar (*matrix*) sebelum melakukan proses *convolution*. Hal ini dimaksudkan agar *filter* memiliki lebih banyak ruang untuk beroperasi hingga ke tepi-tepi gambar [27]. Operasi *convolution* biasanya mengasilkan *feature map* dengan dimensi yang kecil, sebelum masuk ke proses *convolution* berikutnya, *padding* akan mengatur dimensi *feature map* sama dengan *input* sebelumnya.

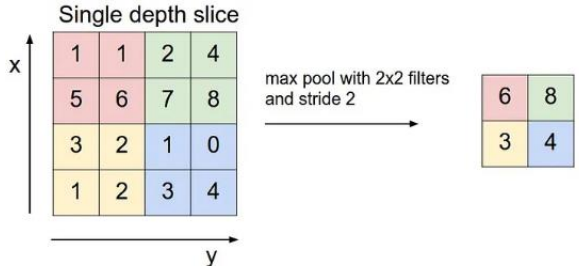


Gambar 6 Ilustrasi *Padding*

Padding yang paling umum digunakan adalah *zero padding* seperti contoh gambar di atas. *Zero padding* menambakan nilai nol pada tepi gambar, pada [28] menunjukkan bahwa *zero padding* membuat proses *convolution* dapat menyimpan informasi lebih baik dibanding *reflection padding*, *replicate padding*, dan *circular padding*. Ilustrasi *padding* dapat dilihat pada Gambar 6.

II.1.2.4 *Pooling Layer*

Setelah melewati proses diatas, selanjutnya akan dilakukan proses *pooling* untuk mengurangi ukuran *matrix* [29]. Dalam arsitektur model CNN, lapisan *pooling* dapat ditempatkan di antara lapisan *convolution* secara berurutan untuk secara bertahap mengurangi ukuran volume *output* pada *feature map*. Dengan demikian, jumlah parameter dan perhitungan di dalam jaringan dapat dikurangi untuk menghindari terjadinya *overfitting*. Lapisan *pooling* bekerja pada setiap tumpukan *feature map* dan mengurangi ukurannya dengan menggunakan *filter* umumnya berukuran 2x2 yang diterapkan dengan *step* sebanyak dua. *Pooling* umumnya ada 2 jenis, yaitu *max pooling* dan *average pooling*. Nilai yang diambil pada *max pooling* adalah nilai tertinggi, sedangkan *average pooling* mengambil nilai rata-ratanya, selanjutnya nilai-nilai hasil *pooling* ini akan diproses pada lapisan *convolutional* selanjutnya. Ilustrasi *max pooling* dapat dilihat pada Gambar 7.

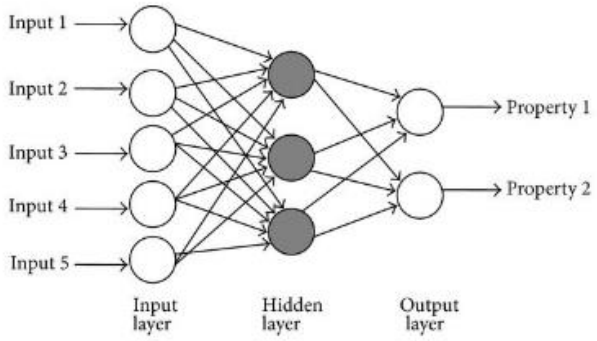


Gambar 7 Ilustrasi *Max Pooling* dari 4x4 Menjadi 2x2 *pixel*

II.1.2.5 *Fully-Connected Layer*

Setelah melalui proses *feature learning* atau *feature extraction*, tahap selanjutnya adalah melakukan klasifikasi akhir dari *feature map* [29]. Namun, sebelum diklasifikasikan, *feature map* yang masih berbentuk multidimensional *array* harus di-*flatten* atau *reshape* menjadi sebuah *vector* tunggal sebelum masuk ke *fully-connected layer*. Pada dasarnya *fully-connected layer* memiliki persamaan cara kerja dengan *multi-layer perceptron*, algoritma yang digunakan di *layer* ini yaitu *feedforward* dan

backpropagation. Tahapan pertama yaitu melakukan *feedforward* lalu mencari nilai error, selanjutnya eror tersebut akan dikurangi dengan mengubah *weight* melalui proses *backpropagation* [30]. Ilustrasi *fully-connected layer* dapat dilihat pada Gambar 8.



Gambar 8 Ilustrasi Proses *Fully-Connected Layer*

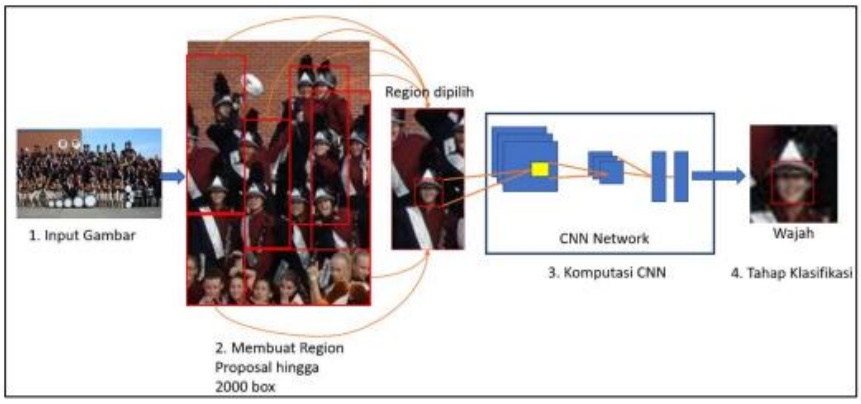
II.1.3 Algoritma Deteksi dan Lokalisasi Objek

Deteksi dan lokalisasi objek adalah tugas dalam bidang *computer vision* yang bertujuan untuk mendeteksi keberadaan objek pada gambar dan memperkirakan posisi objek tersebut. Tugas ini sering dilakukan untuk aplikasi seperti pengenalan wajah, deteksi kendaraan, atau pengenalan plat nomor kendaraan. Dalam deteksi objek, objek ditemukan dengan cara memindai gambar secara sistematis dan menemukan bagian-bagian gambar yang memiliki karakteristik yang cocok dengan objek yang ingin dideteksi, seperti bentuk atau warna. Dalam lokalisasi objek, setelah objek terdeteksi, batas-batas objek ditemukan dengan cara memperkirakan posisi dan ukuran objek dalam gambar, seperti *bounding box* dan *confidence score.* Algoritma deteksi dan lokalisasi objek secara garis besar dibagi menjadi 2, yaitu *Region Proposal-based Framework* dan *Regression/Classification-based Framework* [31].

II.1.3.1 *Region Proposal-based Framework*

Region Proposal-based Framework atau sering disebut *two-stage classification*, disebut 2 tahap karena algoritma ini membedakan proses deteksi dan klasifikasi. Tahap deteksi membuat *Region of Interest* (ROI) terlebih dahulu, dimana area ini diprediksi memiliki banyak objek dibandingkan dengan *background*-nya [31]. Pada tahap ini,

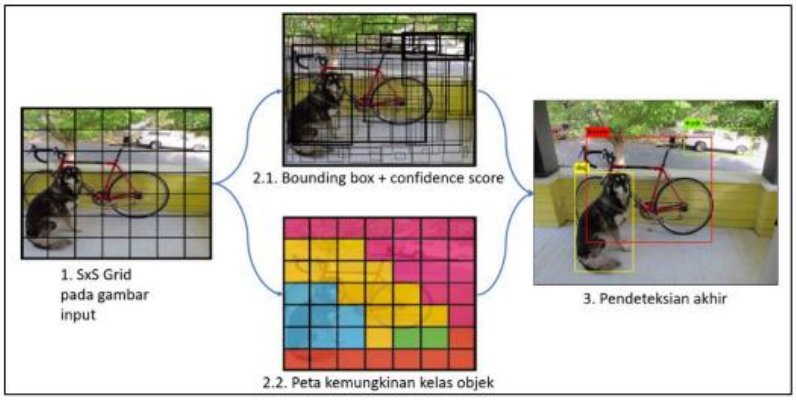
terdapat beberapa metode seperti *Selective Search* dan *Region Proposal Network* (RPN) untuk menghasilkan daerah-daerah ROI yang kemungkinan tinggi memiliki objek. Metode ini mencari tekstur atau warna yang berbeda dengan *background*. Tahap kedua yaitu melakukan klasifikasi terhadap semua ROI, semakin banyak ROI yang dideteksi maka probabilitas *region proposal* tersebut memiliki objek semakin besar [32]. Keuntungan dari menggunakan *two-stage classification* adalah kemampuan untuk mendeteksi objek dengan variasi ukuran, posisi, dan orientasi yang berbeda-beda dalam *input* gambar, namun memiliki kompleksitas dan waktu yang lebih lama karena harus memproduksi *region proposal frames*. Salah satu implementasi dari *two-stage classification* adalah Faster-RCNN. Gambar 9 di bawah ini memperlihatkan cara kerja dari *two-stage classification*.



Gambar 9 Cara Kerja Faster-RCNN dengan *Region Proposal-based Framework*

II.1.3.2 *Regression/Classification-based Framework*

Algoritma ini sering juga disebut sebagai *single-stage classification*, karena melakukan deteksi dan klasifikasi secara bersamaan. Deteksi dan klasifikasi objek dilakukan dilakukan secara langsung pada *input* gambar dengan memberikan *bounding box* dan *confidence score.* Algoritma ini dikenal karena kemampuannya yang cepat dalam mendeteksi objek dengan komputasi yang relatif lebih kecil dibanding *two-stage classification*. Contoh implementasi dari algoritma ini adalah *Single Shot Detector* (SSD) dan *You Only Look Once* (YOLO). Cara kerja YOLO sebagai *single-stage classification* dapat dilihat pada Gambar 10.



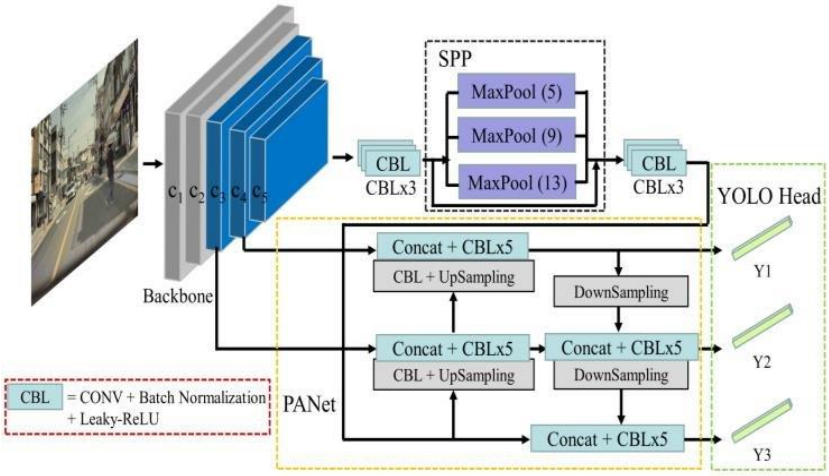
Gambar 10 Cara Kerja YOLO dengan *Regression/Classification-based Framework*

II.1.4 Astitektur YOLOv4

You Only Look Once merupakan salah satu algoritma yang menggunakan *regression/classification-based framework* atau lebih dikenal dengan sebuatan *single-stage classification*. Sebelumnya, YOLOv1 dikembangkan pada tahun 2016, dimana menggunakan 24 *convolutional layers* dan 2 *fully-connected layers*. YOLOv1 dapat mendeteksi objek dengan membuat *S x S grid* pada input, tiap *grid* bertanggungjawab pada satu objek dengan memprediksi *bounding box* dan *confidence score*-nya. Metode ini akan membuat tiap *bounding box* memiliki nilai untuk merepresentasikan nilai dan ukurannya pada objek yang dideteksi. *Bounding box* yang diprediksi akan di-*filter* menggunakan *Non-Max Suppression* (NMS) untuk mengurangi deteksi yang redundan (tumpang tindih) pada objek yang sama [7]. Pada tahun yang sama, YOLOv2 membawa perkembangan yang signifikan dari versi sebelumnya. YOLOv1 memiliki lebih banyak eror pada lokalisasi dibandingkan dengan Fast-RCNN dan *recall* yang rendah dibanding dengan *two-stage classification*. Oleh karena itu, YOLOv2 fokus untuk meningkatkan *recall* dan lokalisasi dengan tetap memperhatankan akurasinya dengan tetap menggunakan *mean of squared error* (MSE) sebagai *loss function* [33]. Dua tahun setelahnya Joseph Redmon kembali merilis YOLO versi ketiga dengan tujuan meningkatkan kecepatan dan akurasi algoritmanya. Salah satu peningkatan pada YOLOv3 yaitu penggunaan arsitektur CNN baru yang dinamakan Darknet-53 (53 *convolutional layers*), dikhususkan untuk tugas deteksi objek. Peningkatan lainnya dari

YOLOv3 adalah *predicted box* dengan skala dan rasio yang berbeda-beda, tidak seperti versi sebelumnya yang memiliki ukuran yang sama sehingga menjadi hambatan bagi algoritma untuk mendeteksi objek dengan ukuran berbeda [34]. YOLOv3 tetap menggunakan *loss* yang sama dengan dua versi sebelumnya, yaitu MSE [35].

YOLOv4 sudah tidak dikembangkan lagi oleh J. Redmon tetapi dilanjutkan oleh Alexey Bochkovskiy. Peningkatan utama dari YOLOv4 terletak pada arsitektur yang digunakan yaitu CSPDarknet53 (*Cross Stage Partial Network*), menambahkan SPP *block* (*Spatial Pyramid Pooling*) setelah *convolution layer* dan sebelum *output layer*. Peningkatan lainnya yaitu modifikasi pada arsitektur agar dapat melatih data pada *single* GPU dengan memperkenalkan dua metode augmentasi data *Mosaic* dan SAT (*Self-Adversarial Training*), *hyperparameter* yang lebih optimal, dan memodifikasi SAM, PAN, dan *Cross mini-Batch Normalization* [36].



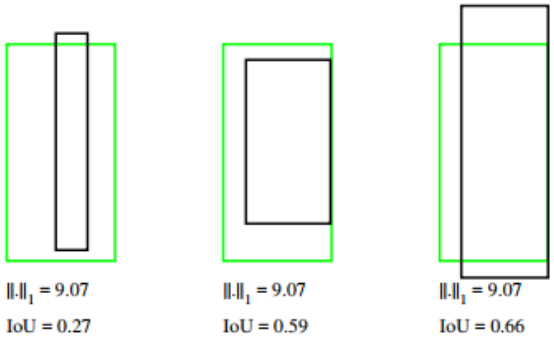
Gambar 11 Arsitektur YOLOv4

Pada Gambar 11 dapat dilihat bahwa struktur model YOLOv4 terdiri dari CSPDarknet53 sebagai *backbone* untuk mengekstraksi fitur, SPPNet sebagai *spatial pooling* dalam *convolutional*, PANet untuk meningkatkan kemampuan *network* mengintegrasi gambar dalam berbagai skala, dan tiga YOLO sebagai *head*. Sesuai namanya, CSPDarknet53 memiliki 53 total *convolutional layer* dengan ukuran 1x1 dan 3x3, setiap *convolutional layer* terhubung dengan *batch normalization* dan *Leaky-*

ReLU activation. Tiga YOLO sebagai *head* dengan ukuran 19x19, 38x38, dan 76x76 digunakan untuk menghubungkan *feature map* dari skala yang berbeda untuk mendeteksi objek dengan berbagai ukuran [37].

II.1.5 *Loss Function* pada YOLOv4

Berbeda dengan YOLO versi sebelumnya, YOLOv4 menggunakan *loss function* yang berbeda dengan menerapkan konsep IoU, karena *loss function* sebelumnya menganggap setiap koordinat *bounding box* sebagai variabel independen. Namun hal ini berbanding terbalik dengan integritas objek itu sendiri.



Gambar 12 Perbedaan Nilai Antara MSE Dengan IoU

Mari perhatikan Gambar 12 di atas, asumsikan jika jarak antara *bounding box* (hijau) dan *ground truth* (hitam) berada pada hasil akhir, SSE memiliki nilai yang berbeda dengan IoU-nya, hal ini membuat SSE tidak mempertahankan karakteristik dari IoU sebagai *evaluation metric* model. SSE hanya menghitung selisih antara *bounding box* dan *ground truth* secara absolut, dan tidak memperhatikan hubungan antara kedua *box* seperti posisi dan ukurannya [12]. Oleh karena itu, peneliti menggunakan IoU sebagai *loss function* [36]. Beberapa tahun terakhir, IoU *loss* juga mengalami perkembangan seperti GIoU (*Generalized Intersection over Union*) *loss* yang menggunakan *gradient* untuk *bounding box* prediksi yang tidak *overlap* dengan *ground truth* [12], DIoU (*Distance Intersection over Union*) yang menggunakan *euclidean distance* dibandingkan *gradient* pada GIoU, dan CIoU (*Complete Intersection over Union*) yang menambahkan perhitungan rasio *bounding box* pada DIoU. CIoU menjadi *loss function* pada YOLOv4 karena dapat memperoleh performa

yang baik pada *convergence speed* untuk kedua *box* yang tidak *overlap*, dan akurasi yang lebih tinggi dibanding dengan *loss* berbasis IoU lainnya [13], [36]. *Loss function* pada YOLOv4 dibagi menjadi tiga bagian, yaitu *confidence loss, classficication loss* dan *localization loss*, rumusnya dapat dilihat pada Gambar 13 di bawah.

$$Loss = \lambda_1 L_{conf} + \lambda_2 L_{cla} + \lambda_3 L_{loc} \quad (1)$$

$$L_{conf} = -\sum (Obj_i \ln(p_i) + (1 - Obj_i)\ln(1 - p_i)) \quad (2)$$

$$L_{cla} = -\sum_{i \epsilon Box} \sum_{j \epsilon class} (O_{ij} \ln(p_{ij}) + (1 - O_{ij})\ln(1 - p_{ij})) \quad (3)$$

$$L_{loc} = 1 - IOU(A, B) + \frac{d_{AB}^2 (A_{ctr}, B_{ctr})}{l^2} + \alpha v \quad (4)$$

Gambar 13 *Loss Function* pada YOLOv4

Loss function pada YOLOv4 didefinisikan pada (1), dengan λ sebagai koefisien penyeimbang untuk mengatasi masalah ketidakseimbangan antara *penalty term box* dengan *penalty term* perbedaan luas. Dalam perhitungan CIoU loss, *penalti term box* mengukur seberapa jauh pusat *bounding box* dari *ground truth*, sementara *penalty term* perbedaan luas mengukur perbedaan antara luas *bounding box* yang diprediksi dan *ground truth*-nya. Pada bagian (2), Obj_i mengidikasikan apakah ada objek pada *bounding box i* yang diprediksi, hasilnya yaitu 0 untuk tidak ada, dan 1 untuk ada. p_i merujuk pada probabilitas jika pada *box* prediksi benar-benar ada objek menggunakan fungsi *sigmoid*. Pada (3), O_{ij} dan p_{ij} menunjukkan apakah ada objek kelas *j* dan probabilitasnya pada *box* prediksi *i*. Sedangkan pada (4) merujuk pada CIoU sebagai algoritma untuk menghitung *localization offset loss*, αv adalah perhitungan aspek rasio *bounding box*, dan (A_{ctr}, B_{ctr}) adalah perhitungan *euclidean distance* untuk titik tengah kedua *box* [37]. Semenjak *loss function* merupakan *hyperparameter* dalam *deep learning*, maka pada algoritma aslinya, YOLOv4 juga memberikan opsi lain seperti MSE, GIoU, dan DIoU. Jika ada kesalahan dalam mengeja *loss functionnya*, maka YOLOv4 akan secara *default* memilih IoU *loss* [38].

II.1.5.1 *Generalized Intersection over Union*

GIoU *loss* [12] hadir untuk mengatasi kekurangan dari IoU *loss*, yaitu ketika kedua *box* tidak *overlap* maka nilai IoU menjadi nol, sehingga model tidak dapat dioptimalisasi. GIoU bekerja dengan menambahkan *box* ketiga (C) yaitu *box* yang mengapit *bounding box* dan *ground truth*. Nilai dari *box C* akan menjadi *gradient* untuk perhitungan *loss*.

IoU dapat dihitung dengan:

$$IoU = \frac{|\ A \cap B\ |}{|\ A \cup B\ |} \tag{5}$$

Dimana, $A \cap B$ adalah nilai *area* atau gabungan antara *bounding box* (A) dan *ground truth* (B). $A \cup B$ adalah *intersection* atau irisan antara A dan B. Sedangkan IoU *loss* dapat diperoleh dengan:

$$L_{IoU} = 1 - IoU \tag{6}$$

Perlu diperhatikan bahwa ketika *bounding box* dan *ground truth* tidak *overlap*, maka rumus yang digunakan yaitu L_{IoU}, dan untuk mengoptimalisasi model yang *overlap* digunakan rumus *IoU*. Hal ini juga berlaku untuk *GIoU*, *DIoU*, *CIoU*, dan *loss*-nya. *Loss* hanya akan dipakai ketika $IoU = 0$.

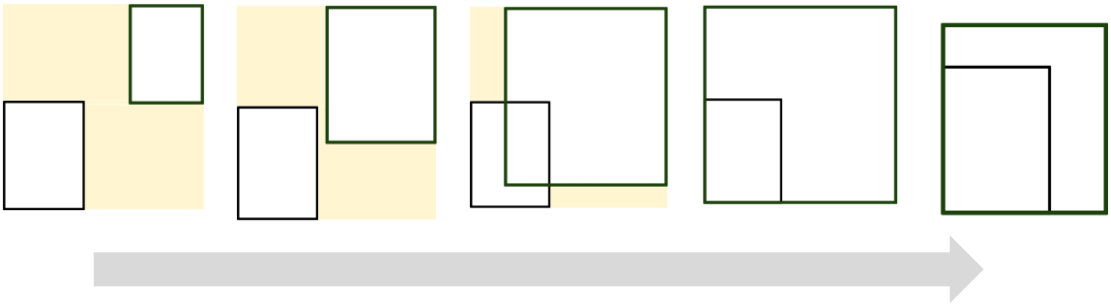
GIoU dapat dihitung menggunakan rumus:

$$GIoU = IoU - \frac{|\ C\ /\ A \cup B\ |}{|\ C\ |} \tag{7}$$

C merupakan *box* ketiga yang mengapit *box* A dan B. Sedangkan untuk *loss* dari GIoU dapat dihitung menggunakan rumus:

$$L_{GIoU} = 1 - GioU \tag{8}$$

Kelemahan dari GIoU adalah ketika *gradient* atau nilai dari *C* sama dengan nol, maka GIoU akan turun menjadi IoU. Kelemahan lainnya yaitu ketika posisi kedua *box* sejajar horizontal atau vertikal, kecepatan dari *convergence* akan memakan waktu yang lebih lama. Berikut ilustrasi cara kerja dari GIoU:



Gambar 14 Ilustrasi Cara Kerja *Generalized Intersection over Union*

Gambar 14 mengilustrasikan kotak hitam sebagai *ground truth*, kotak hijau sebagai *prediction / bounding box*, dan kotak kuning sebagai *C* atau *gradient*. *Bounding box* akan terus membesar seiring *gradient* berkurang, ketika sudah 100% menutupi *ground truth* (*gradient* = 0), GIoU akan turun menjadi IoU untuk mengoptimasi prediksi.

II.1.5.2 *Distance Intersection over Union*

DIoU [13] hadir setahun setelah GIoU dipublikasikan, DIoU memperkenalkan metode baru yang lebih cepat untuk *box* yang tidak *overlap* menggunakan *euclidean distance*. DIoU terbukti lebih cepat mempertemukan kedua *box* yang tidak *overlap* dengan menggunakan koordinat tengah kedua *box* sebagai perhitungan *euclidean*.

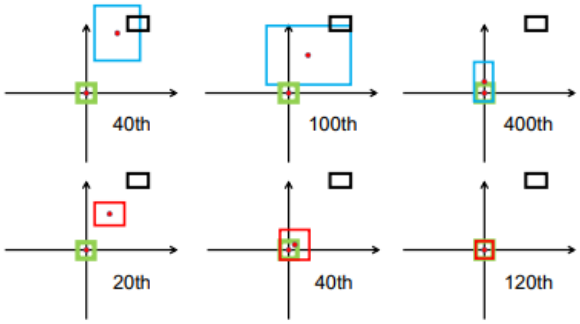
DIoU dapat dihitung menggunakan rumus:

$$DIoU = \frac{p^2(\boldsymbol{b},\boldsymbol{b}^{\boldsymbol{gt}})}{c^2} \tag{9}$$

Dimana *p* adalah rumus euclidean distance untuk koordinat tengah *bounding box* ($\boldsymbol{b}$) dan *ground truth* ($\boldsymbol{b}^{\boldsymbol{gt}}$), *c* adalah garis diagonal kedua *box*. *c* berguna sebagai *normalization* agar nilai akhir selalu di antara 0 dan 1.

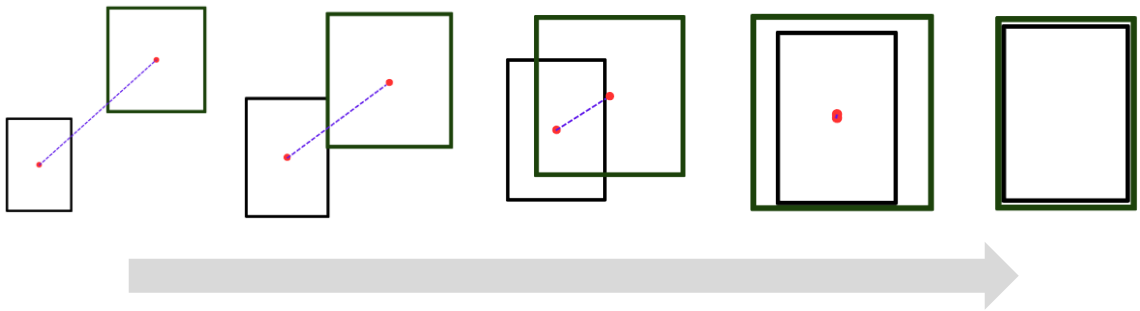
Sedangkan *loss*-nya dapat dihitung dengan:

$$L_{DIoU} = 1 - DioU \quad (10)$$



Gambar 15 Perbandingan *Convergence Speed* antara GIoU dan DIoU

Dari Gambar 15 diatas dapat dilihat bahwa DIoU hanya membutuhkan 120 iterasi untuk membuat *bounding box* optimal dengan *ground truth*-nya, sedangkan GIoU membutuhkan lebih dari 400 iterasi. Kelemahan dari DIoU yaitu performa yang kurang dan ketika koordinat tengah kedua *box* bertemu, DIoU akan turun menjadi IoU. Berikut ilustrasi cara kerja DIoU:



Gambar 16 Ilustrasi Cara Kerja *Distance Intersection over Union*

Gambar 16 mengilustrasikan kedua box sama seperti GIoU, namun tidak menggunakan *gradient* untuk menambah ukuran *bounding box*, melainkan *euclidean distance* untuk mendekatkan kedua *box*. Cara ini terbukti lebih cepat dari GIoU, namun

DIoU juga tetap akan menjadi IoU jika kedua koordinat tengan (merah) berada di posisi yang sama.

II.1.5.3 Complete Intersection over Union

CIoU dirilis pada paper yang sama dengan DIoU [13] dengan memberikan peningkatan pada performa dan kecepatan dengan menghitung tiga faktor berbeda yaitu *overlap area, central point distance* dan *aspect ratio*.

Rumus dari CIoU dapat dihitung dengan:

$$CIoU = \frac{p^2(\boldsymbol{b},\boldsymbol{b}^{\boldsymbol{gt}})}{c^2} + \alpha v \tag{11}$$

Dimana α adalah nilai *trade-off* positif, dan *v* menghitung konsistensi dari aspek rasio.

$$v = \frac{4}{\pi^2}\left(\arctan\frac{w^{gt}}{h^{gt}} - \arctan\frac{w}{h}\right)^2 \tag{12}$$

$$\alpha = \frac{v}{(1 - IoU) + v} \tag{13}$$

Sehingga *loss* dari CIoU dapat dituliskan menjadi:

$$L_{CIoU} = 1 - CioU \tag{14}$$

CIoU dapat mempertahankan kecepatan dari *convergence* sekaligus meningkatkan performa dari deteksi model, hal inilah yang membuat CIoU menjadi *default loss* untuk YOLOv4, YOLOv5, YOLOv6, dan YOLOv7. CIoU mengadopsi cara kerja *euclidean distance* pada DIoU, namun dengan performa yang lebih baik karena menambahkan perhitungan aspek rasio. Jika aspek rasio kedua box sama, CIoU akan turun menjadi DIoU, hal ini merupakan kelemahan dari CIoU.

II.1.6 *Evalutation Metric*

Evaluation metric atau tolak ukur evaluasi adalah sebuah satuan ukur kuantitatif yang digunakan untuk mengevaluasi performa suatu model. Pada *computer vision*¸ *evaluation metric* yang umum digunakan yaitu:

II.I.6.1 *Average Precision*

Average precision merupakan satuan ukur pada NLP dan *computer vision* dengan menghitung rata-rata *precision* dari keseluruhan *recall* pada level yang berbeda. *Precision* merupakan hasil bagi antara *true positive* dengan keseluruhan hasil prediksi yang positif (*true positive* dan *false negative*). Sedangkan *recall* adalah hasil bagi antara *true positive* dengan *true positive* ditambah *false negative*[39].

$$Precision = \frac{True\ Positive}{True\ Positive + False\ Positive} \quad (15)$$

$$Recall = \frac{True\ Positive}{True\ Positive + False\ Negative} \quad (16)$$

True positive adalah *instance* yang bernilai positif berhasil diprediksi oleh model sebagai positif, *false negative* adalah *instance* yang sebenarnya positif, namun dideteksi sebagai negatif, *false positive* adalah *instance* yang negatif namun dideteksi posifit oleh model, dan *true negative* adalah *instance* negatif berhasil dideteksi model sebagai *negatif*. *Mean average precision* atau mAP adalah *evaluation metric* yang paling umum digunakan pada algoritma deteksi dan lolakisasi, mAP didapatkan dengan menghitung rata-rata *average precision* tiap kelas pada *threshold* tertentu.

II.1.6.2 *Detection Speed*

Salah satu *evaluation metric* yang paling banyak digunakan pada *computer vision* adalah *detection speed*, atau waktu yang diperlukan model untuk memuat informasi *weight*, kemudian menerima *input* hingga mendeteksi dan melokalisasi semua objek pada *frame*.

$$DS = t_{akhir} - t_{awal} \tag{17}$$

Pada rumus di atas, t_{akhir} merupakan waktu dalam satuan detik yang diperoleh ketika sebuah *frame* selesai diproses, sedangkan t_{awal} adalah waktu sebelum *frame* diolah oleh model. Perlu diketahui bahwa waktu yang diperlukan untuk memproses sebuah *input* berbeda-beda tergantung dari jumlah objek, resolusi, dan *feature map* pada *frame* tersebut. Oleh karena itu, harus dilakukan pencarian nilai rata-ratanya dengan rumus:

$$\overline{DS} = \frac{\sum DS}{\sum frame} \tag{18}$$

Rumus 18 akan mencari total dari seluruh *detection speed* dalam detik, lalu kemudian membaginya dengan penjumlahan seluruh *frame* yang berhasil diproses. Pada *input* video, *detection speed* akan direpresentasikan dengan *frame per second* (*fps*) yang menunjukkan kemampuan model untuk memprediksi dan melokalisasi *frame* berkelanjutan. Semakin tinggi *fps*, maka model akan dianggap baik (*realtime*), berikut rumus untuk mencari *fps*:

$$FPS = \frac{1}{\overline{DS}} \tag{19}$$

II.2 Kajian Pustaka

Berikut merupakan ringkasan penelitian terkait usulan penelitian yang diperoleh dari beberapa referensi berupa jurnal.

1. Joseph Redmon dan Ali Farhadi [34], dengan judul *YOLOv3: An Incremental Improvement*. Penelitian ini dilakukan untuk memberikan peningkatan dari versi YOLO sebelumnya yaitu akurasi dan kecepatan komputasi yang rendah. YOLOv3 mengusung arsitektur baru bernama Darknet53 dengan 53 *convolutional layer*. Sama seperti versi pertama dan kedua, YOLOv3 tetap

menggunakan MSE sebagai *loss function* untuk kasus *bounding box regression* karena mudah untuk dioptimalisasi. Performa dari YOLOv3 terbukti lebih baik dengan mampu mengolah dimensi 320 x 320 pada 22 *ms* dan 28.2 mAP seperti SSD namun tiga kali lebih cepat. YOLOv3 mampu memberikan 57.9 AP_{50} dengan 51 *ms* pada GPU Titan X, dibanding dengan RetinaNet dengan kecepatan 198 *ms* pada 57.5 $AP_{50.}$

2. Jiahui Yu, Yuning Jiang, dkk [40], dengan judul *UnitBox: An Advanced Object Detection Network*. Penelitian ini berawal dari masalah jaringan CNN yang menganggap variabel yang ada pada *box* (*x, y, witdh,* dan *height*) sebagai variabel independen, yang akan diregresi menggunakan NMS satu per satu mengakibatkan kurangnya *localization*. Oleh karena itu, penelitian ini menggunakan IoU sebagai *loss function* karena *evaliuation metric* pada deteksi objek juga menggunakan IoU. Dengan Iou sebagai *loss function,* keempat variabel *box* diolah sebagai satu unit, menghasilkan tidak hanya *convergence* yang cepat, tetapi *localization* yang lebih akurat dibanding MSE.

3. Alexey Bochkovskiy, Chien-Yao Wang, dan Hong-Yuan. M. Liao [36], dengan judul *YOLOv4: Optimal Speed and Accuracy of Object Detection*. Peningkatan utama dari YOLOv4 adalah perubahan arsitektur menjadi CSPDarknet54 dengan 54 *convolutional layer,* SPPNet sebagai *spatial pooling,* PANet mengintegrasi gambar dalam berbagai skala, dan tiga YOLO sebagai *head* dengan ukuran 19x19, 38x38, dan 76x76 digunakan untuk menghubungkan *feature map* dari skala yang berbeda untuk mendeteksi objek dengan berbagai ukuran. Penelitian ini juga mengganti MSE ke salah satu peningkatan IoU *loss function* (CIoU). Hasil dari penelitian ini menunjukkan bahwa YOLOv4 sangat cepat dengan berjalan pada 45 *fps,* dan mengalahkan algoritma deteksi lainnya seperti DPN dan R-CNN.

4. Hamid Rezatofighi, Nathan Tsoi, dkk [12], dengan judul *Generalized Intersection over Union: A Metric and A Loss for Bounding Box Regression.*

Penelitian ini merupakan peningkatan dari *loss function* IoU dengan mengusulkan *evaluation metric* dan *loss metric* baru dengan nama GIoU. Limitasi IoU pada YOLOv4 yaitu model tidak dapat dioptimalisasi ketika *bounding box* dan *ground truth* tidak saling *overlap*. GIoU bekerja dengan menambah *box* ketiga sebagai *gradient* untuk memperbesar ukuran *bounding box* hingga menyentuh *ground truth*. Hasil penelitian ini membuktikan GIoU dapat mengalahkan MSE dan IoU pada *dataset* PASCAL VOC dan MS COCO.

5. Zhaohui Zheng, Ping Wang, Wei Liu, dkk [13], dengan judul *Distance-Iou Loss: Faster and Better Learning for Bounding Box Regression.* Penelitian ini merupakan mengusulkan *evaluation* dan *loss metric* baru bernama DIoU, sekaligus mengatasi kekurangan GIoU yang membutuhkan komputasi lebih dalam mempertemukan kedua *non overlapping box*. Penelitian ini menawarkan metode *loss* baru, yaitu mendekatkan kedua *box* menggunakan *euclidean distance*. Penelitian ini menunjukkan bahwa DIoU mengalahkan GIoU dalam hal kecepatan *convergence*, sekaligus meningkatkan mAP dibanding IoU dan GIoU *loss*.

6. Zhaohui Zeng, Ping Wang, Dongwei Ren, dkk [41], dengan judul *Enhancing Geometric Factors in Model Learning and Inference for Object Detection and Instancec Segmentation.* Penelitian ini merupakan peningkatan pada DIoU oleh *author* yang sama [13]. *Evaluation* dan *loss metric* ini bernama CIoU, meskipun telah disinggung pada paper yang sama dengan DIoU, namun pada paper ini CIoU diperkenalkan kembali dengan Cluster NMS. CIoU menambahkan satu faktor perhitungan baru yang dinilai penting dalam meningkatkan performa model, yaitu aspek rasio. Sehingga CIoU adalah versi lengkap untuk melakukan regresi *bounding box*. Hasil dari penelitian ini menunjukkan bahwa CIoU mengalahkan MSE, GIoU, dan DIoU pada AP dan AP_{75} di COCO *dataset*. Performa ini membuat CIoU menjadi *default loss* untuk YOLOv4.

II.3 Metode Penyelesaian Masalah

II.3.1 *State of The Art*

Tabel 1 *State of The Art*

No	Judul Karya Ilmiah, Nama,Tahun Terbit dan Penerbit	Objek dan Permasalahan	Metode Penyelesaian	Kinerja	Korelasi < = >
1	**Judul usulan** : Peningkatan Performa Deteksi Model *You Only Look Once V4* Menggunakan Kombinasi *Loss Function* **Pengusul** : Nugra Tasik Allo	**Objek** : Menggunakan *mask dataset* dan *racoon* dataset **Permasalahan** : *Loss function* yang ditawarkan YOLOv4 masih sulit mendeteksi objek pada lingkungan yang kompleks	Menggabungkan *loss function* GIoU dan CIoU untuk meningkatkan performa sekaligus memperbaiki limitasi kedua *loss function*	Hipotesis : Performa *localization* dari model dapat meningkat sehingga dapat mendeteksi objek pada lingkungan yang kompleks	

No	Judul Karya Ilmiah, Nama,Tahun Terbit dan Penerbit	Objek dan Permasalahan	Metode Penyelesaian	Kinerja	Korelasi < = >
2	**Judul :** *YOLOv3: An Incremental Improvement* [34] **Nama :** Joseph Redmon, Ali Farhadi **Tahun Terbit :** 2018 **Penerbit :** arXiv	**Objek** : COCO *dataset* **Permasalahan** : MSE sebagai *loss function* pada YOLOv1 sampai YOLOv3 tidak dapat mendeteksi objek kecil	Menggunakan FPN (*Feature Pyramid Network*) untuk dapat mendeteksi objek kecil.	Dapat mendeteksi objek dengan berbagai bentuk dan ukuran, sekaligus meningkatkan performa deteksi objek kecil dibandingkan YOLOv2	<

No	Judul Karya Ilmiah, Nama,Tahun Terbit dan Penerbit	Objek dan Permasalahan	Metode Penyelesaian	Kinerja	Korelasi < = >
3	**Judul** : *UnitBox: An Advanced Object Detection Network* [40] **Nama** : Jiahui Yu, Yuning Jiang,dkk **Tahun Terbit** : 2016 **Penerbit** : arXiv	**Objek** : Menggunakan *face detection task* **Permasalahan** : MSE selain tidak dapat mendeteksi objek kecil, juga memberikan *localization* yang lemah karena MSE tidak cocok dengan *evaluation metric* IoU	Untuk menyesuaikan *loss function* dan *evaluation metric* pada model CNN, peneliti menawarkan IoU menjadi *loss function*.	IoU sebagai *loss metric* memberikan hasil yang jauh lebih baik pada *localization*, dapat mendeteksi objek dengan bentuk dan skala yang bervariasi, dan *converge* lebih cepat.	<

No	Judul Karya Ilmiah, Nama,Tahun Terbit dan Penerbit	Objek dan Permasalahan	Metode Penyelesaian	Kinerja	Korelasi < = >
4	**Judul :** *YOLOv4: Optimal Speed and Accuracy of Object Detection* [36] **Nama :** Alexey Bochkovskiy, Chen-Yao Wang, dan Hong-Yuan M. Liao **Tahun Terbit :** 2020 **Penerbit :** arXiv	**Objek** : COCO *dataset* **Permasalahan** : YOLOv3 yang masih menggunakan MSE dinilai belum cukup unduk mendeteksi objek kecil dan tumpang tindih, selain itu kecepatan dari YOLOv3 dinilai masih kurang jika dalam kasus *realtime*.	Menggunakan arsitektuk *convolution* yang lebih dalam (53 *layer*), PANet untuk mengintegrasi gambar pada berbagai skala, dan menggunakan *localization loss* terbaru yaitu CIoU.	Kombinasi dari arsitektur dan model yang berbeda menghasilkan 65.7% AP_{50} pada COCO dataset dengan ~65 *fps* secara *realtime*.	=

No	Judul Karya Ilmiah, Nama,Tahun Terbit dan Penerbit	Objek dan Permasalahan	Metode Penyelesaian	Kinerja	Korelasi < = >
5	**Judul :** *Generalized Intersection over Union: A Metric and A Loss for Bounding Box Regression* [12] **Nama :** Hamid Rezatofighi, Nathan Tsoi, dkk. **Tahun Terbit :** 2019 **Penerbit :** CVPR	**Objek** : COCO dan PASCAL VOC *dataset* **Permasalahan** : Ketika *bounding box* dan *ground truth* tidak *overlapping*, maka nilai IoU akan menjadi nol, sehingga tidak dapat dioptimalisasi.	Menambahkan *box* baru (C) sebagai *gradient*. *Gradient* akan menambah ukuran *bounding box* hingga *overlap* dengan *ground truth*.	GIoU mengungguli MSE dan IoU *loss* di AP dan AP_{75} pada PASCAL VOC dan COCO *dataset*. Selain lebih baik pada performa, GIoU juga mengungguli keduanya pada kecapatan *convergence*.	=

No	Judul Karya Ilmiah, Nama,Tahun Terbit dan Penerbit	Objek dan Permasalahan	Metode Penyelesaian	Kinerja	Korelasi < = >
6	**Judul :** *Distance-IoU Loss: Faster and Better Learning for Bounding Box Regression* [13] **Nama :** Zhaohui Zheng, Ping Wang, dkk **Tahun Terbit :** 2019 **Penerbit :** arXiv	**Objek** : PASCAL VOC *dataset* **Permasalahan** : GIoU memiliki 2 limitasi yang utama, yaitu ketika *gradient* menjadi nol, GIoU akan turun menjadi IoU, dan ketika posisi kedua *box* yang tidak *overlap* horizontal/vertikal, maka kecepatan *convergence*-nya akan lama.	Menggunakan *euclidean distance* untuk meng-*converge* kedua *box* yang tidak *overlap*.	DIoU mengungguli GIoU dalam hal kecepatan *convergence*. Selain lebih cepat, DIoU juga memiliki AP dan AP_{75} lebih tinggi dibanding IoU dan GIoU pada PASCAL VOC.	=

No	Judul Karya Ilmiah, Nama,Tahun Terbit dan Penerbit	Objek dan Permasalahan	Metode Penyelesaian	Kinerja	Korelasi < = >
7	**Judul :** *Enhancing Geometric Factors in Model Learning and Inference for Object Detection and Instance Segmentation* [41] **Nama :** Zhaohui Zheng, Ping Wang, dkk **Tahun Terbit :** 2021 **Penerbit :** arXiv	**Objek** : COCO *dataset* **Permasalahan** : DIoU masih kurang dalam performa *localization*	CIoU menambahkan satu faktor pengukuran *loss* yang baru, yaitu *aspect ratio* dari *bounding box*	Hasil dari penelitian ini menunjukkan bahwa CIoU mengalahkan MSE, GIoU, dan DIoU pada AP dan AP_{75} di COCO *dataset* dengan memberikan *localization* yang lebih baik. Kinerja ini membuat CIoU menjadi *default loss function* untuk beberapa versi YOLO.	=

II.3.2 Metode yang Diusulkan

Sebagaimana pada Tabel *State of the Art* telah dijabarkan beberapa topik mengenai peningkatan dari IoU sebagai *bounding box regression loss function* pada algoritma YOLOv4 sebagai *single-stage classification* yaitu GIoU, DIoU, dan CIoU.Pada *framework* Darknet, YOLOv4 menggunakan CIoU sebagai *default loss,* dengan tetap memberikan opsi menggunakan IoU, MSE, GIoU dan DIoU. Tiap *loss function* memiliki kelebihan dan kekurangan masing-masing, semenjak *loss function* merupakan *hyperparameter* pada *deep learning*, membuat pemilihan *loss function* bergantung pada kasus yang ingin diselesaikan. Tiap *loss* memiliki limitasi-limitasi yang akan diperbaiki, yaitu:

Tabel 2 Limitasi GIoU, DIoU, dan CIoU

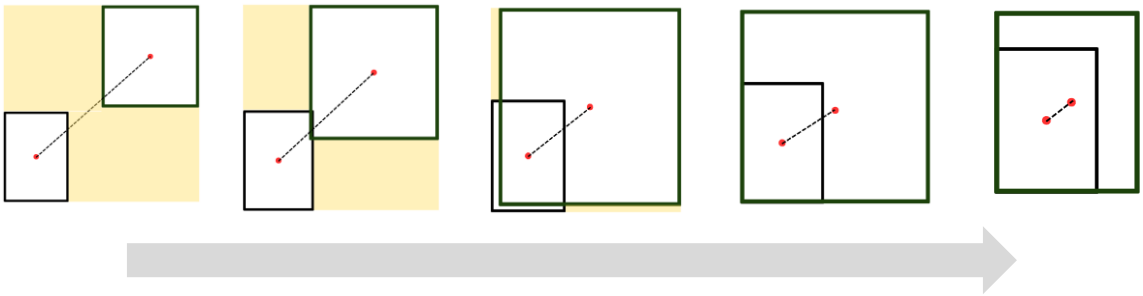
Jenis *loss*	Limitasi
GIoU	- Komputasi yang lama untuk kedua *box* dengan posisi sejajar horizontal / vertikal - Ketika *bounding box* sudah menutupi *ground truth* sepenuhnya sehingga membuat *gradient* menjadi 0, GIoU akan turun menjadi IoU.
DIoU	- Ketika kedua *center coordinate* dari *box* berada pada posisi yang sama, DIoU turun menjadi IoU.
CIoU	- CIoU akan bekerja sangat baik jika kedua aspek rasio *box* berbeda, namun jika sama maka CIoU akan turun menjadi DIoU.

Didasari dari limitasi-limitasi di atas, peneliti ingin mengusulkan *loss function* yang dapat meng-*cover* limitasi tersebut dengan menggabungkan GIoU dengan CIoU sebagai *loss function* yang baru. Rumus dari *loss function* yang ditawarkan yaitu:

$$Proposed = \frac{\frac{|C / A \cup B|}{|C|} + \frac{p^2(\boldsymbol{b},\boldsymbol{b}^{\boldsymbol{gt}})}{c^2} + \alpha v}{\beta}, \text{ where } \beta = (0,8,2) \qquad (20)$$

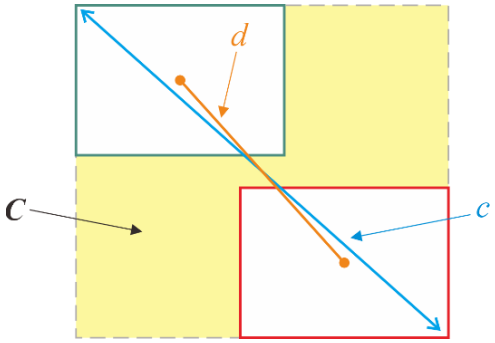
$$L_{Proposed} = 1 - Proposed \qquad (21)$$

Berikut ilustrasi cara kerja jadi *loss function* yang ditawarkan:



Gambar 17 Ilustrasi Cara Kerja *Proposed Loss*

Loss function yang ditawarkan akan menambah ukuran *bounding box* sembari menggunakan *euclidean distace* pada kasus kedua *box* yang tidak *overlap*. Diharapkan *loss* ini dapat mengatasi limitasi *loss function* sebelumnya karena 1) *convergence* pada posisi horizontal/vertikal akan lebih cepat karena dibantu *euclidean distance*, 2) *bounding box* akan terus bertambah besar sehingga CIoU tidak akan pernah turun menjadi DIoU, membuatnya bekerja lebih efisien, dan 3) ketika *gradient* sama dengan nol, akan dilanjutkan oleh CIoU.



Gambar 18 Properti yang Dihitung pada *Proposed Loss*

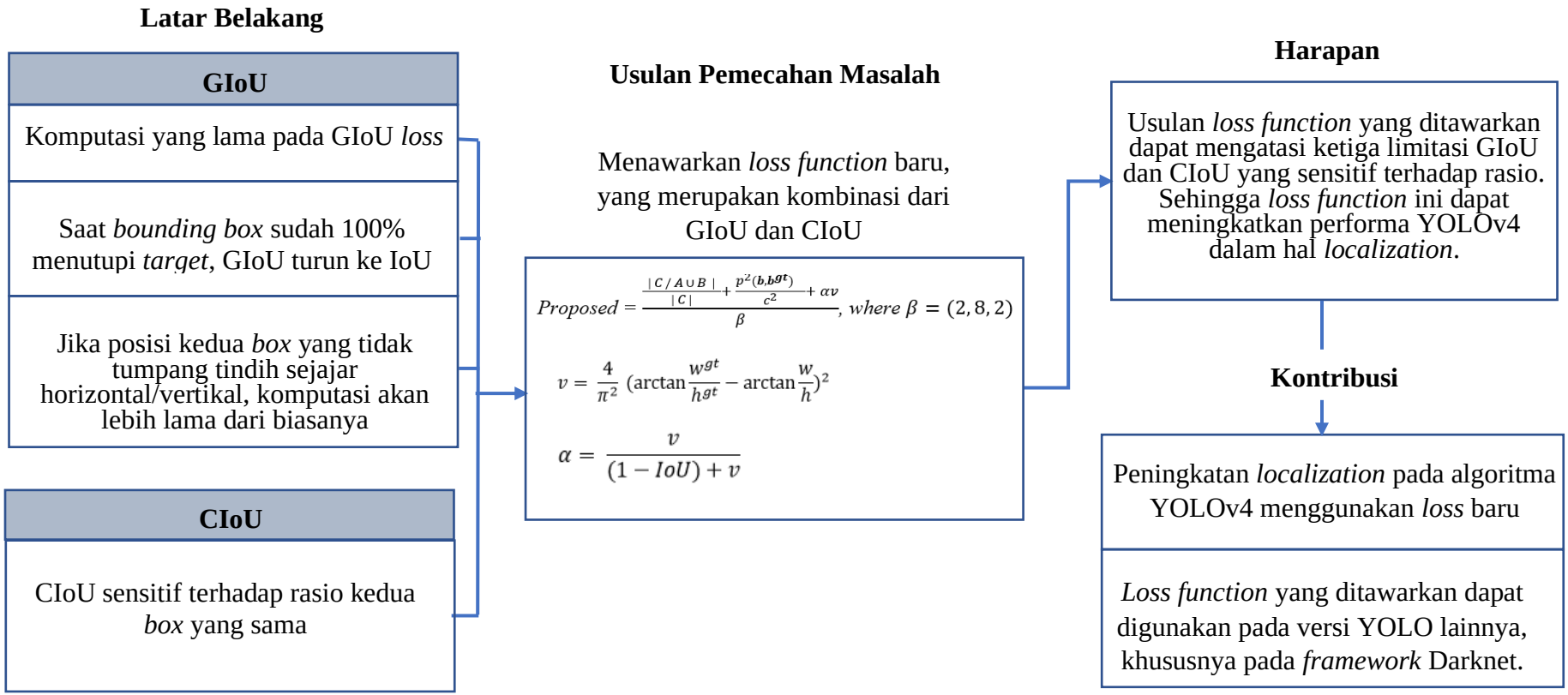
Gambar 18 di atas menunjukkan properti-properti yang digunakan oleh *proposed loss* dalam bekerja. *C* merupakan luas dari box ketiga yang mengapit box prediksi dan box *ground-truth,* box C ini akan menjadi batas untuk menutupi dan juga memberikan ukuran yang lebih informatif tentang ketidaksamaan antara box *ground truth* dan box prediksi. Selanjutnya *d* merupakan jarak titik tengah antara kedua box, kedua titik ini yang akan digunakan dalam perhitungan *Euclidean distance* untuk menarik box prediksi mendekati box *ground truth*. Terakhir, *c* merupakan garis diagonal antara titik kiri atas dan kanan bawah dari box *C* yang berguna untuk menormalisasikan hasil perhitungan dari *d*. Pada penelitian [42] disebutkan bahwa pada proses *training data*, *gradient* yang kecil lebih efektif dalam mencapai *local minimum* atau error terkecil untuk sebuah *loss function*. Dengan menggabungkan GIoU loss dan CIoU loss menjadi sebuah *loss function,* nilai *gradient* yang dihasilkan perlu dipekecil sehingga β diperlukan untuk melakukan tugas tersebut. Pada Rumus 20, dapat dilihat bahwa β yang digunakan berjarak 2 sampai 8 dengan selisih 2. Nilai-nilai tersebut akan memberikan efek yang berbeda pada *gradient* yang dihasilkan. Untuk mengevaluasi nilai tersebut, akan dilakukan beberapa simulasi untuk mendapatkan nilai yang optimal untuk mengecilkan *gradient* dari metode yang diusulkan.

II.3 Target Hipotesis Penelitian

Penelitian ini menawarkan *loss function* baru yang merupakan gabungan dari GIoU dan CIoU untuk menangani tugas *localization* pada YOLOv4. *Loss function* ini juga bisa diterapkan pada YOLOv4 dan YOLOv7 pada *framework* Darknet dengan memodifikasi *config file*. Penelitian ini diharapkan dapat meningkatkan perfoma model karena *loss function* ini dapat menangani limitasi dari GIoU dan CIoU secara bersamaan.

II.5 Kerangka Pikir Penelitian

Kerangka pikir penelitian merupakan dasar penelitian atau struktur berpikir yang disusun dari fakta-fakta dan observasi untuk memahami hubungan antara informasi yang diberikan.



Gambar 19 Kerangka Pikir Penelitian