BAB II
TINJAUAN PUSTAKA

2.1. Penelitian Terkait

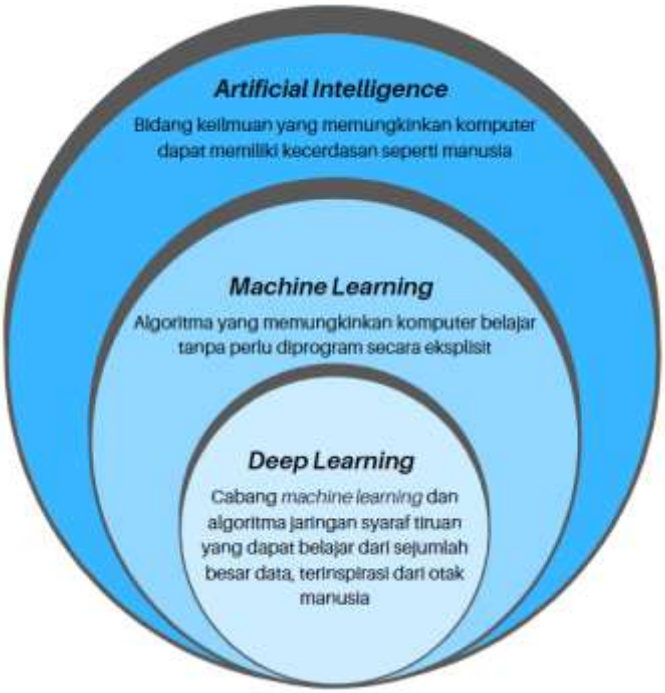
Tabel 2. 1 Penelitian Terkait

No	Penulis	Metode	Hasil Utama
[16]	Sunil Singh, Umang Ahuja dan Monika Sachdeva [16]	YOLOv3 dan faster R-CNN	Tujuan dari penelitian ini untuk menggali teknologi deteksi masker wajah untuk memantau penggunaan masker di tempat-tempat umum selama pandemi COVID-19 menggunakan YOLOv3 dan Faster R-CNN. Hasil percobaan menunjukkan bahwa penerapan dalam kamera pengawas di dunia nyata model dengan algoritma YOLO lebih disukai karena dapat melakukan deteksi dalam satu kali pengambilan gambar (single-shot detection) dan memiliki frame rate yang lebih tinggi daripada Faster R-CNN atau algoritma deteksi objek lainnya.
[17]	Rakkshab Varadharajan Iyer, Priyansh Shashikant Ringe, Kevin Prabhulal Bhensdadiya	YOLOv3, YOLOv5, dan MobileNet-SSD V2	Tujuan dari penelitian ini adalah untuk membandingkan berbagai pendekatan sistematis dalam menganalisis gambar dan menentukan apakah seseorang yang terdeteksi menggunakan masker wajah dengan benar, salah, atau sama sekali tidak menggunakan masker menggunakan tiga algoritma pembelajaran yaitu YOLOv3, YOLOv5, dan MobileNet-SSD V2. Hasil penelitian menunjukan bahwa YOLOv5s merupakan model yang paling cocok untuk situasi real-time dengan nilai akurasi dan fps yang optimal. Meskipun MobileNet-SSD V2 menyediakan kecepatan yang cukup mirip dengan YOLOv5s, namun kurang unggul dalam akurasi.
[18]	Jimin Yu dan Wei Zhang	YOLOv4	penelitian ini membahas mengenai analisis algoritma pengenalan masker wajah dan deteksi

			penggunaan masker yang berstandar berbasis perbaikan pada model YOLO-v4. Hasil dari penelitian menunjukkan bahwa tingkat akurasi pengenalan masker wajah dapat mencapai 98.3% dan kecepatan frame (frame rate) tinggi sebesar 54.57 FPS yang lebih akurat dibandingkan dengan algoritma yang telah ada sebelumnya.
[19]	Madhura	Facemasknet deep learning network	Tujuan dari penelitian ini adalah untuk membantu membatasi penyebaran virus COVID-19 dengan menggunakan teknologi pengenalan masker wajah menggunakan metode deep learning yang disebut "Facemasknet" untuk berhasil menguji apakah seseorang mengenakan masker wajah atau tidak. Hasil penelitian ini berhasil mencapai akurasi sebesar 98.6% sehingga dapat digunakan baik pada gambar diam (still images) maupun pada video secara langsung. Ketika masker dipakai dengan tidak benar contohnya ketika hidung dan mulut hanya sebagian tertutup, sistem ini dapat mendeteksinya.
[20]	Faisal Dharma Adhinata , Diovianto Putra Rakhmadani, Merlinda Wibowo,A khmad Jayadi	DenseNet201	Tujuan dari penelitian ini adalah untuk mengembangkan sistem deteksi wajah yang memakai masker atau tidak memakai masker dengan menggunakan model DenseNet201. Hasil penelitian ini membuktikan bahwa sistem deteksi wajah yang memakai masker atau tidak memakai masker menjadi lebih akurat dengan menggunakan model DenseNet201. Dimana hasil dari beberapa iterasi pelatihan (epochs) hingga 30 menunjukkan bahwa model DenseNet201 menghasilkan akurasi 99% pada data pelatihan. Selanjutnya, dilakukan pengujian pada data vide0 dan model DenseNet201 menghasilkan nilai F-Measure sebesar 0.98, sedangkan model MobileNetV2 hanya menghasilkan nilai F-Measure sebesar 0.67.

2.2. Kecerdasan Buatan

Kecerdasan buatan adalah salah satu bidang ilmu komputer yang mempelajari bagaimana komputer dapat berperilaku cerdas seperti manusia. Kecerdasan buatan dapat digunakan dalam berbagai bidang, seperti bisnis, otomotif, pendidikan, dan kesehatan. Contoh penerapan kecerdasan buatan adalah sistem otomatis dan mobil self-driving[21]. Dalam bidang riset kecerdasan buatan yang saat ini ada, didalamnya ada 2 sub-bidang yang mejadi banyak topik riset beberapa tahun terakhir yaitu *Machine Learning* yang jika dikupas lebih dalam akan bertemu dengan sub paling dalam yakni *Deep Learning* seperti yang terlihat pada struktur riset pada Gambar 2.1 berikut ini.



Gambar 2. 1 Lapisan Riset Kecerdasan Buatan

Machine learning adalah cabang kecerdasan buatan yang mempelajari bagaimana komputer dapat belajar dari data dan menghasilkan prediksi atau keputusan tanpa diprogram secara eksplisit[22]. *Machine learning* bekerja dengan cara mengambil data sebagai input, kemudian memproses data tersebut menggunakan algoritma tertentu untuk menghasilkan model atau prediksi. Adapun penerapan yang popular digunakan saat ini seperti Klasifikasi jaringan internet untuk penghematan, pengalokasian, pembatasan sumber daya internet, Sistem

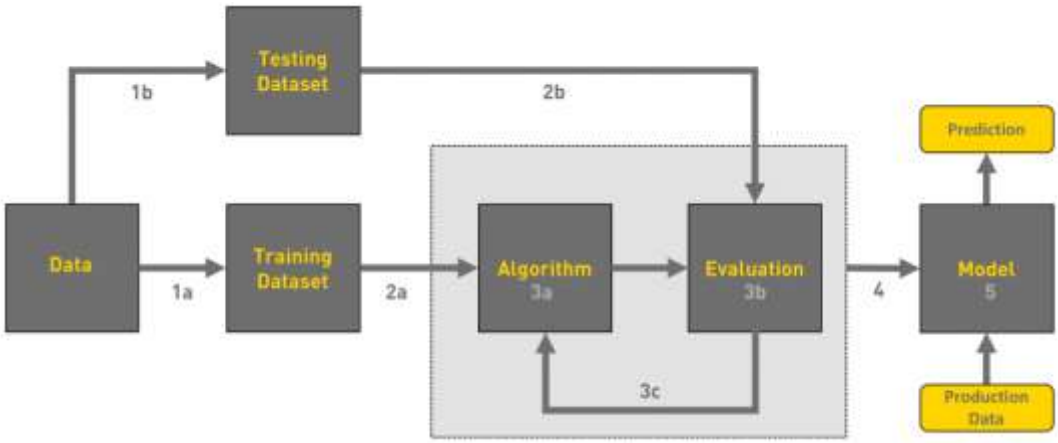
kehadiran siswa, Klasifikasi pada penyakit daun tomat, Alat pendeteksi pengguna masker sebagai upaya pencegahan Covid-19, Proses deteksi web phishing dan lain sebagainya.

Deep learning merupakan sub-bidang dari machine learning yang menggunakan jaringan saraf tiruan untuk memproses data dan menghasilkan prediksi atau keputusan. *Deep learning* memungkinkan mesin untuk belajar secara mandiri dari data yang besar dan kompleks, sehingga dapat menghasilkan hasil yang lebih akurat dan kompleks[23]. Dalam beberapa kasus, *deep learning* dapat menghasilkan hasil yang lebih baik daripada *machine learning* konvensional. Namun, *deep learning* juga membutuhkan lebih banyak data dan sumber daya komputasi yang lebih besar daripada *machine learning* konvensional. Oleh karena itu, *deep learning* lebih cocok untuk masalah yang kompleks dan membutuhkan tingkat akurasi yang tinggi, seperti pengenalan wajah dan suara, serta dalam mobil *self-driving*.

2.2.1. Pembelajaran Mesin

Machine learning adalah cabang ilmu komputer yang mempelajari bagaimana membuat sistem yang dapat belajar dari data dan pengalaman, tanpa harus diprogram secara eksplisit. Machine learning dapat digunakan untuk berbagai tujuan, seperti klasifikasi, regresi, klastering, rekomendasi, deteksi anomali, pengenalan pola, dan lain-lain[24].

Cara kerja machine learning secara umum adalah sebagai berikut: pertama, sistem menerima input berupa data yang memiliki fitur atau atribut tertentu. Kedua, sistem memproses data tersebut dengan menggunakan algoritma atau model yang telah dipilih atau dibuat sebelumnya. Ketiga, sistem menghasilkan output berupa prediksi, rekomendasi, atau keputusan yang didasarkan pada data input dan model yang digunakan. Keempat, sistem mengevaluasi kinerja output dengan menggunakan metrik atau ukuran tertentu, seperti akurasi, presisi, recall, f1-score, dan lain-lain. Kelima, sistem melakukan pembelajaran atau penyesuaian terhadap model berdasarkan hasil evaluasi, sehingga model dapat meningkatkan kinerjanya untuk data input selanjutnya[24]. Untuk memahami bagaimana pembelajaran mesin bekerja, dapat dilihat pada Gambar 2.2 dibawah ini.



Gambar 2. 2 Alur Kerja Machine Learning

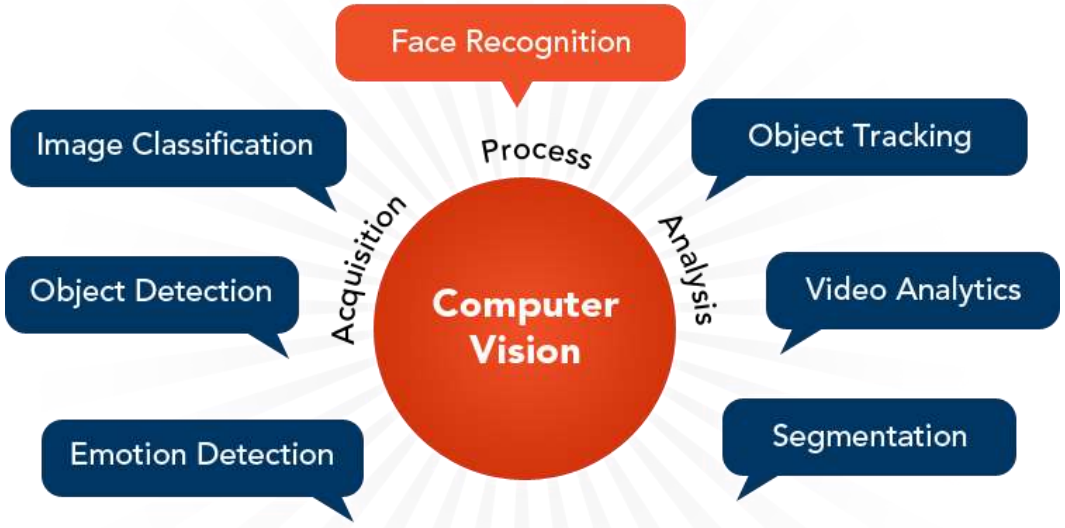
2.2.2. Deep Learning

Deep learning adalah sub-bidang dari *machine learning* yang menggunakan jaringan saraf tiruan untuk memproses data dan menghasilkan prediksi atau keputusan. *Deep learning* memungkinkan mesin untuk belajar secara mandiri dari data yang besar dan kompleks, sehingga dapat menghasilkan hasil yang lebih akurat dan kompleks[23]. Dalam *deep learning*, jaringan saraf tiruan terdiri dari beberapa lapisan yang saling terhubung dan terdiri dari banyak neuron. Setiap neuron menerima input dari neuron sebelumnya, memproses input tersebut, dan mengirimkan output ke neuron berikutnya. Dalam proses pelatihan, jaringan saraf tiruan akan belajar dari data dan menyesuaikan bobot antar-neuron untuk menghasilkan hasil yang lebih akurat[25].

Deep learning menjadi metode yang efektif beberapa tahun terakhir ini karena memiliki keuntungan seperti mampu memproses data yang besar dan kompleks, sehingga dapat menghasilkan hasil yang lebih akurat dan kompleks, pengimplementasian yang fleksibel dan luas dan dapat belajar secara mandiri dari data yang besar dan kompleks[26], [27]. Namun metode ini bukan metode yang sempurna karena membutuhkan banyak sumber daya komputasi sehingga biaya *development* yang dibutuhkan lebih mahal, selain itu Model yang dihasilkan sulit untuk diinterpretasikan, sehingga sulit untuk menjelaskan bagaimana model tersebut membuat keputusan[26]. Kemudian meski mampu belajar dalam data yang besar, kualitas dari data yang digunakan perlu sangat diperhatikan dan akan sangat tidak efektif bila menggunakan data dengan jumlah yang sangat kecil[28].

2.2.3. Computer Vision

Computer Vision adalah salah satu bidang yang lebih spesifik ada Machine Learning yang mengkaji kemampuan mesin (dalam hal ini computer) dalam melihat, mendeteksi (*detection*) hingga mengenali (*recognition*) dari apa yang terlihat oleh komputer, seperti mendeteksi bentuk, warna, maupun kontras yang terdapat dalam sebuah objek baik berupa citra diam atau gambar dan citra bergerak atau video[29]. *Computer vision* dapat digunakan dalam berbagai aplikasi, seperti pengenalan wajah, identifikasi objek, dan analisis citra medis[30].berikut adalah sub-bidang dari *computer vision* yang saat ini banyak diteliti oleh kalangan luas.



Gambar 2. 3 Bidang-bidang dalam Computer Vision

Deteksi objek dan klasifikasi citra: Bidang ini mempelajari bagaimana mesin dapat mendeteksi objek dalam gambar dan mengklasifikasikannya ke dalam kategori tertentu. Teknik-teknik seperti deep learning dan convolutional neural network (CNN) sering digunakan dalam bidang ini[31], [32]. Pengenalan wajah: Bidang ini mempelajari bagaimana mesin dapat mengenali wajah manusia dalam gambar atau video. Teknik-teknik seperti Fisherface dan artificial neural network (ANN) sering digunakan dalam bidang ini[33]. Dan Pengolahan citra digital: Bidang ini mempelajari bagaimana mesin dapat memproses citra digital untuk meningkatkan kualitas atau mendapatkan informasi dari citra tersebut. Teknik-teknik seperti objek detection dan tracking menggunakan deep learning sering digunakan dalam bidang ini[32].

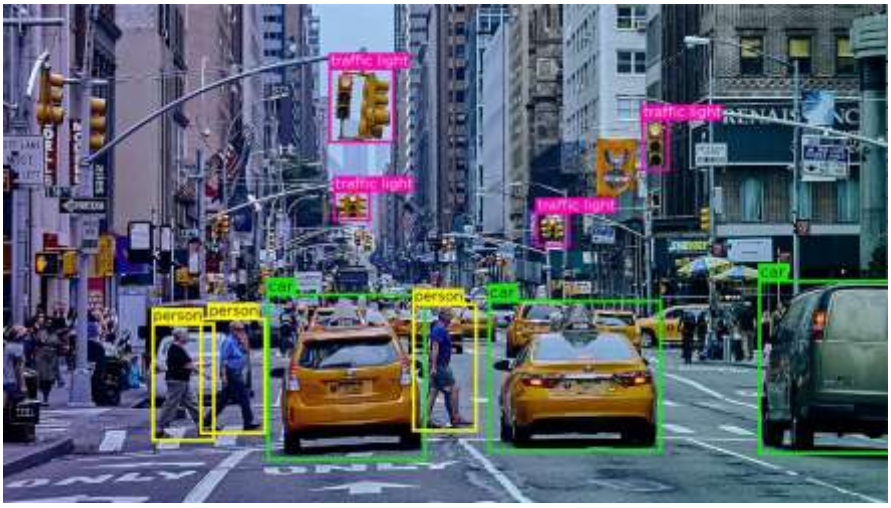
2.2.3.1.Face Recognition

Sebelum memahami definisi dari rekognisi wajah, perlu untuk mengetahui secara jelas apa yang disebut dengan wajah. Definisi wajah menurut kamus besar bahasa indonesia merupakan bagian dari kepala; roman muka; muka. Wajah atau muka adalah bagian depan dari kepala pada manusia meliputi wilayah dari dahi hingga dagu. Bagian-bagian yang termasuk wilayah wajah yaitu rambut, dahi, alis, mata, hidung, pipi, mulut, bibir, gigi, kulit, dan dagu

Face Recognition atau pengenalan wajah merupakan sebuah kegiatan alami manusia yang kemudian ditransformasikan ke dalam sebuah bentuk keilmuan baru dalam *Computer Vision* untuk membedakan identitas manusia berdasarkan ciri tertentu agar lebih mudah diidentifikasi. Wajah merupakan objek yang tidak mudah meninggalkan traces dalam berbagai media sehingga sulit dicuri dan memiliki kemungkinan keamanan yang tinggi[34].

2.2.3.2.Anotasi Data

Tahap anotasi sendiri merupakan tahap untuk mengekstraksi fitur berupa pemberikan sebuah kotak pembatas yang disebut *Bounding Box* pada objek yang berasal dari *frame* pada data latih. Tahap ekstraksi fitur ini penting agar objek dapat dikenali dengan memperhatikan ciri khas pada sebuah objek. Tahap ini merupakan tahap dimana objek akan diberikan pembatas dan diberikan nilai label pada setiap objek yang akan dideteksi[35]. *Bounding box* ini juga akan muncul pada saat pendeteksian. Contoh dari anotasi gambar dapat dilihat pada Gambar 2.1.



Gambar 2. 4 Contoh Anotasi Data

Nilai dari hasil anotasi nantinya akan disimpan dalam file yang diberi nama sama dengan data latih yang dianotasi. File berisi nomor *class* dan koordinat objek pada setiap *frame*, dimana satu baris terdapat satu objek dengan format sebagai berikut.

<class_id><x_center><y_center><width><height>

Dimana setiap komponen diatas mewakili :

a. <class_id> merupakan nomor objek / *class* dengan nilai awal 0 sampai class 32
b. <x_center>, <y_center>, <width> dan <height> merupakan nilai desimal relatif dari koordinat tengah, lebar dan tinggi dari bounding box, nilai biasanya berupa 0.0 sampai dengan 1.0
c. Koordinat tengah dari *bounding box* didapat dari perhitungan absolut tinggi dan lebar bounding box dibagi 2, kemudian ditambah koordinat kiri atas bounding box.
d. Nilai <x_center> dan <y_center> didapat dari koordinat tengah dibagi lebar dan tinggi dari gambar berurutan.
e. <width> dan <height> dihitung dari lebar dan tinggi bounding box dibagi lebar dan tinggi dari gambar berurutan.

Untuk dapat menghitung nilai dari x_center,y_center,width dan height digunakan persamaan berikut.

$$\text{<x\_center>} = \frac{x}{tinggi\_gambar} \quad \textbf{(2.1)}$$

$$\text{<y\_center>} = \frac{y}{lebar\_gambar} \quad \textbf{(2.2)}$$

$$\text{<width>} = \frac{lebar\_bbox}{lebar\_gambar} \quad \textbf{(2.3)}$$

$$\text{<height>} = \frac{tinggi\_bbox}{tinggi\_gambar} \quad \textbf{(2.4)}$$

2.2.3.3.Preprocessing

Preprocessing merupakan tahap awal pada pengolahan data citra dalam *computer vision*. *Preprocessing* dilakukan untuk mempersiapkan data citra sebelum masuk ke tahap pengolahan dan analisis lebih lanjut. Tahap *preprocessing* meliputi beberapa operasi seperti *resizing*. *Preprocessing* sangat penting dalam pengolahan data citra karena dapat mempengaruhi kualitas hasil pengolahan dan analisis lebih lanjut. Dengan melakukan tahap *preprocessing* yang tepat, data citra dapat dipersiapkan secara optimal untuk tahap selanjutnya dalam pengolahan dan analisis data.

Resizing adalah salah satu teknik dalam *computer vision* yang sering digunakan untuk mengubah ukuran suatu gambar. Tujuan dari proses *resizing* sebelum masuk pada pemrosesan

citra lebih lanjut adalah untuk menstandarkan ukuran gambar input dan rasio aspek agar sesuai dengan persyaratan input model. Ukuran yang digunakan pada penelitian ini adah 416x416 sesuai dengan acuan dari penelitian sebelumnya[36].

Adapun tahapan dari algoritma bilinear interpolation adalah sebagai berikut

1. Hitung *weighted average color* sehubungan dengan jarak ke piksel kiri atas dan kanan atas
2. Menghitung *weighted average color* dengan jarak ke piksel kiri bawah dan kanan bawah
3. Hitung rata-rata tertimbang dengan sehubungan dengan jarak y dari dua warna yang dihasilkan

Pada penelitian ini resolusi awal gambar adalah sebesar 352x640 dan akan diubah menggunakan algoritma *Bilinear Interpolation* menjadi ukuran 416x416 dengan memperhitungkan nilai piksel yang berdekatan. Tabel 2.2 menggambarkan sebuah matriks dari sebuah gambar.

Tabel 2. 2 Matriks sebuah citra

(x,y)	0	1	2
0	a	b	c
1	d	e	f
2	g	h	i

Kemudian dilakukan *bilinear interpolation* dengan menyisipkan kolom baru dengan nilai yang belum diketahui atau disimbolkan x seperti pada tabel 2.3.

Tabel 2. 3 Penyisipan nilai

X,y	0	1	2	3	4
0	a	x	b	x	c
1	d	x	e	x	f
2	g	x	h	x	i

Kemudian setiap nilai x dihitung dengan menambahkan dua nilai dari pixel tetangga dan membagi 2 seperti pada Persamaan 2.1 untuk menghitung nilai x pada koordinat (1,0) berikut

$$Bilinear_{(1,0)} = \frac{a+b}{2} \quad \textbf{(2.5)}$$

Keterangan :

a = nilai piksel pada titik (0,0)

b = nilai piksel pada titik (2,0)

Maka didapat nilai pixel seperti pada tabel 2.4

Tabel 2. 4 Hasil Penyisipan nilai piksel pada kolom

X,y	0	1	2	3	4
0	a	(a+b)/2	b	(b+c)/2	c
1	d	(d+e)/2	e	(e+f)/2	f
2	g	(g+h)/2	h	(h+i)/2	i

Kemudian dilakukan juga perhiungan yang sama seperti pada kolom di setiap baris yang ada seperti pada tabel 2.5.

Tabel 2. 5 Hasil penyisipan nilai piksel pada baris

X,y	0	1	2	3	4
0	a	(a+b)/2	b	(b+c)/2	c
1	(a+d)/2	x	(b+e)/2	x	(c+f)/2
2	d	(d+e)/2	e	(e+f)/2	f
3	(d+g)/2	x	(e+h)/2	x	(f+i)/2
4	g	(g+h)/2	h	(h+i)/2	i

Masih ada nilai x pada beberapa koordina *pixel* yang belum terisi, misal pada titik (1,1) dapat dihitung dengan Persamaan 2.2.

$$Bilinear_{(1,1)} = \frac{a+b+d+e}{4} \quad \textbf{(2.6)}$$

Keterangan :

a = nilai piksel pada titik (0,0)

b = nilai piksel pada titik (2,0)

d = nilai piksel pada titik (0,2)

e = nilai piksel pada titik (2,2)

Maka dihasilkan gambar nilai *pixel* pada tabel dibawah ini setelah dilakukan proses resize dengan menggunakan metode *bilinear interpolation*.

Tabel 2. 6 Hasil Akhir Citra *Bilinear Interpolation*

X,y	0	1	2	3	4
0	a	(a+b)/2	b	(b+c)/2	c
1	(a+d)/2	(a+b+d+e)/4	(b+e)/2	(b+c+e+f)/4	(c+f)/2
2	d	(d+e)/2	e	(e+f)/2	f
3	(d+g)/2	(d+e+g+h)/4	(e+h)/2	(e+f+h+i)/4	(f+i)/2
4	g	(g+h)/2	h	(h+i)/2	i

2.2.4. Augmentasi Data

Augmentasi data ditujukan untuk menangani minimnya jumlah dataset yang digunakan dalam melatih sebuah model pembelajaran mesin atau *Deep learning* sekalipun. Untuk menanggulangi hal tersebut, strategi berupa teknik augmentasi data digunakan untuk memperkaya dataset yang ada. Metode yang digunakan dalam penelitian ini adalah *mosaic augmentation* yang secara *default* digunakan pada banyak model YOLO. Dibawah ini adalah cara kerja dari *mosaic augmentation*.

1. Diambil 4 gambar dari dataset secara acak, dimana setiap gambar sudah dianotasi dan diberi label sehingga terdapat *bounding box*

Gambar 2. 5 Contoh Gambar Yang Telah Dianotasi

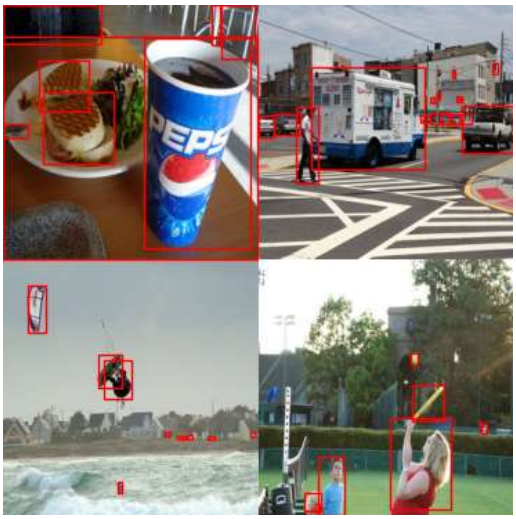
2. Setiap gambar yang tadi terpilih diubah ukurannya menjadi 416x416

Gambar 2. 6 Perubahan Ukuran Gambar

3. Gabungkan semua gambar menjadi satu gambar. untuk melakukan hal ini, dibuat sebuah tensor nol dengan bentuk 3×512×512 dan tumpang susunkan setiap gambar pada setiap sudut tensor yang baru. Proses tadi menghasilkan ketidakteraturan *bounding box*, maka bounding box yang tidak pada tempatnya perlu di pindahkan ke tempat yang seharusnya.

Gambar 2. 7 Penyatuan gambar menjadi pola Mosaik

4. Ubah posisi *bounding box* dengan cara berkut
   a. *Bounding box* untuk gambar kiri atas tidak perlu dipindahkan
   b. *Bounding box* pada gambar kanan atas harus dipindahkan 256 ke kanan
   c. *Bounding box* di kiri bawah harus dipindahkan 256 piksel ke bawah
   d. *Bounding box* di kanan bawah harus dipindahkan 256 piksel ke bawah dan ke kanan



Gambar 2. 8 Hasil Pemindahan posisi Bounding Box

5. Kemudian, mengambil potongan acak dari gambar yang telah disatukan untuk mendapatkan gambar Mosaik akhir seperti pada gambar 2.5, cara menentukan titik potongan dapat dilakukan. dengan menghitung dengan Persamaan 2.3 dan 2.4 sebagai berikut.
   a. *Top Left Corner* (x, y) of *Cutout*:

      $$x, y = random\_point(\text{dim } \_sqrt(0.5 * current\_dim) - \text{dim } \_sqrt) \quad \textbf{(2.7)}$$

      Keterangan :

      Random_point = titik acak yang dipilih sebagai lokasi *cutout*

      Dim_sqrt = akar kuadrat dari dimensi gambar akhir

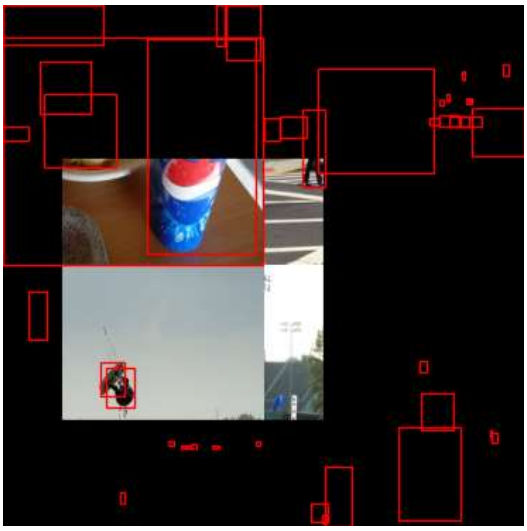
      Current_dim = dimensi gabungan saat ini

   b. *Bottom Right Corner* (x, y) of *Cutout*:

      $$x, y = x + 256 \quad \textbf{(2.8)}$$

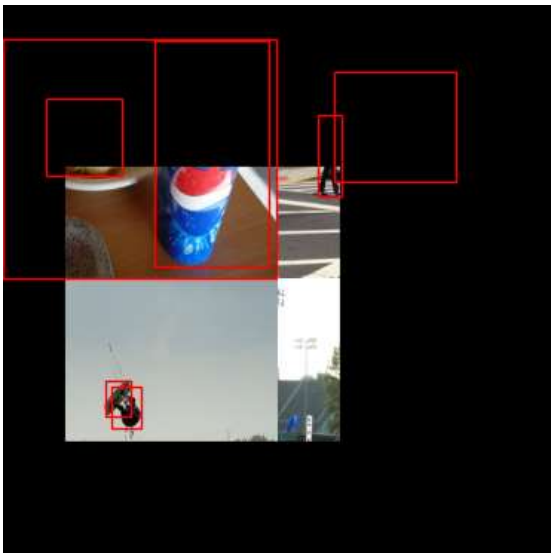
      Keterangan :

      x, y = titik x,y dari *top left corner*



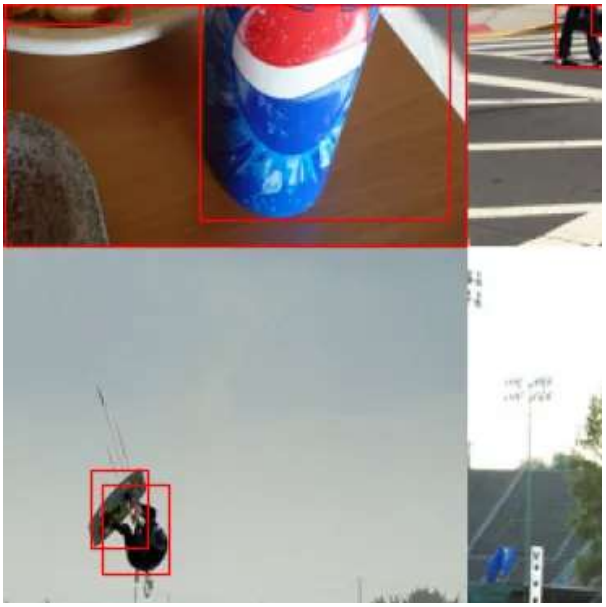
Gambar 2. 9 Proses Cutout

6. Menghapus *bounding box* diluar area *cutout* dengan meghapus semua *bounding box* yang berada diluar area *cutout*, hanya sisakan yang berada didalam area *cutout* meski kecil.



Gambar 2. 10 Penghapusan Bounding box diluar area cutout

7. Mengatur ukuran semua *bounding box* pada area *cutout*



Gambar 2. 11 Hasil Pengaturan ukuran Bounding Box

2.3. Convolutional Neural Network

Convolutional Neural Network merupakan salah satu algoritma *deep learning* yang dikembangkan dari *Multilayer Percepton* (MLP), dirancang untuk mengolah data dua dimensi. Algoritma ini digunakan untuk klasifikasi data yang sudah diberi label menggunakan metode *supervised learning* dengan cara kerjanya yang memerlukan data yang dilatih dan memiliki variable yang sudah di ditargetkan, tujuan dari metode ini adalah pengelompokan suatu data yang sudah ada.

Jaringan konvolusi adalah jaringan khusus yang dimiliki CNN, citra masukan diolah pada lapisan ini berdsarkan filter yang sudah ditentukan. Hasil dari setiap lapisan ini berupa sebuah pola dari beberapa bagian citra yang nanti akan diklasifikasikan. Berikut adalah cara kerja dari CNN.

1. Membaca citra masukan

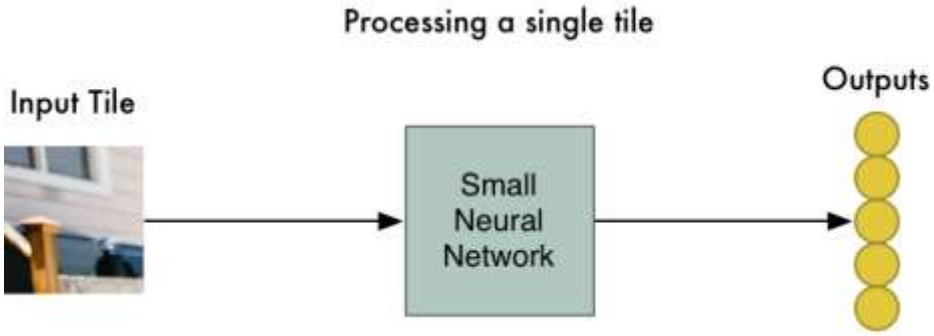
Gambar 2. 12 Contoh Citra Masukan

2. Membagi citra masukan menjadi bagian yang lebih kecil

Gambar 2. 13 Hasil Pembagian Citra

Dari citra masukan diatas, didapatkan hasil konvolusinya berupa 77 citra yang leih kecil dengan konvolusi yang sama.

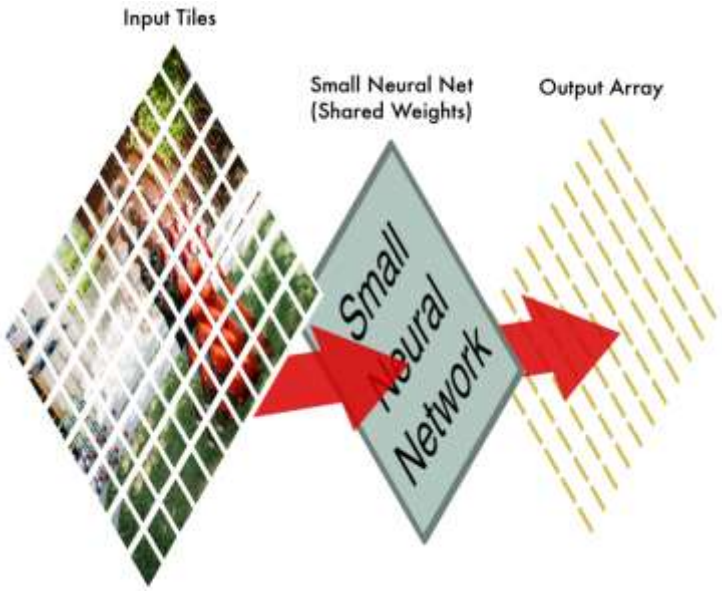
3. Memasukkan *small neural network* ke setiap gambar yang telah dibagi



Gambar 2. 14 Pemrosesan pada Small neural network

proses ini akan diulang sebanyak 77 kali pada setiap citra yang telah dibagi kedalam bentuk yang lebih kecil. Setiap citra kecil tadi nantinya memiliki bobot *neural network* yang sama dengan citra aslinya atau disebut dengan *weights sharing*. Jika ada yang tampak menarik pada setiap citranya, maka bagian tersebut akan dianggap sebagai *object of interest*.

4. Menyimpan hasil dari setiap citra kecil ke *array* baru

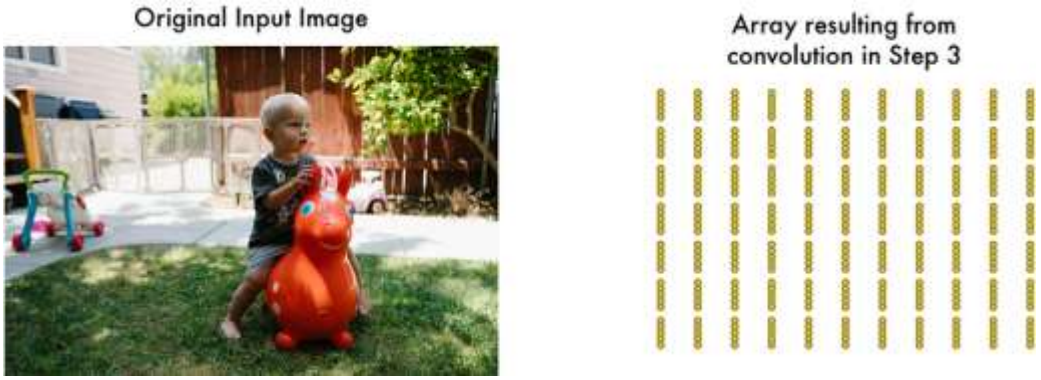


Gambar 2. 15 Penyimpanan hasil dari citra ke array

Menyimpan hasil dari pemrosesan setiap bagian citra ke dalam *grid* dengan pengaturan yang sama dengan citra aslinya.

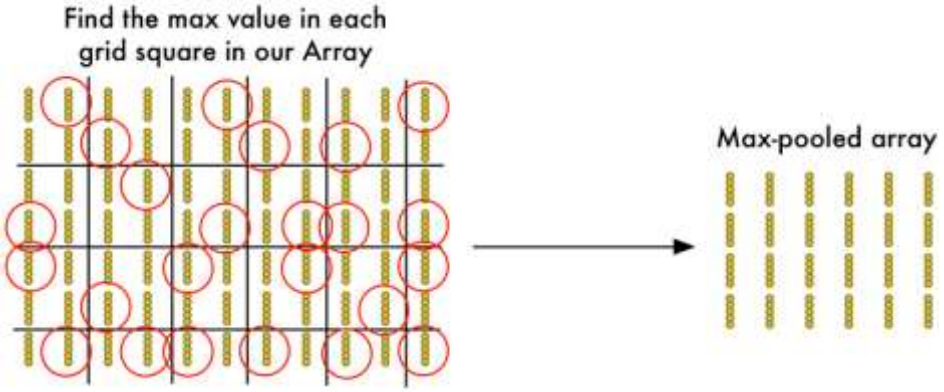
5. Melakukan downsampling

   Hasil dari Langkah sebelumnya adalah *array* yang sudah memetakan bagian mana saja dari citra asli yang paling menari atau disebut *object of interest* tadi. Tetapi ukuran *array* tersebut masih terlalu besar.



Gambar 2. 16 proses Downsampling

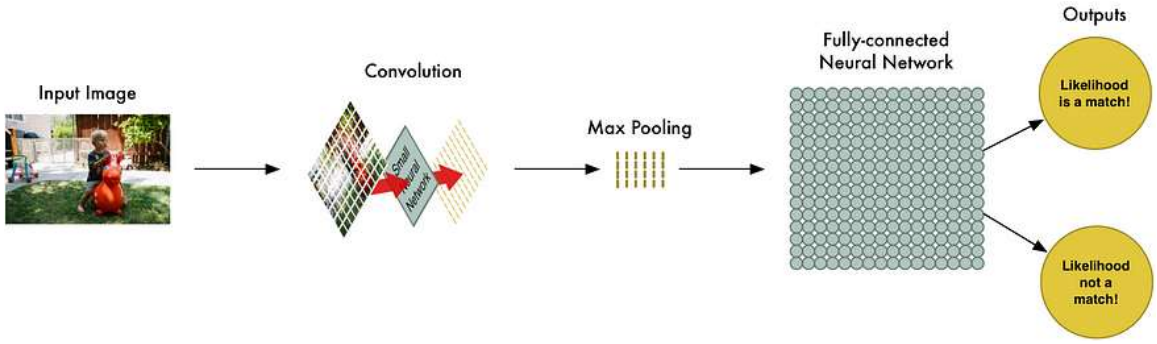
Untuk mengurangi ukuran *array* tersebut, maka perlu dilakukan *downsample* menggunakan algoritma *max pooling*. *array* yang ada akan dilihat dengan ukuran kernel 2x2 dan akan tetap menyimpan ukuran piksel paling besar pada setiap kernelnya.



Gambar 2. 17 *Max Pooling Array*

6. Membuat prediksi

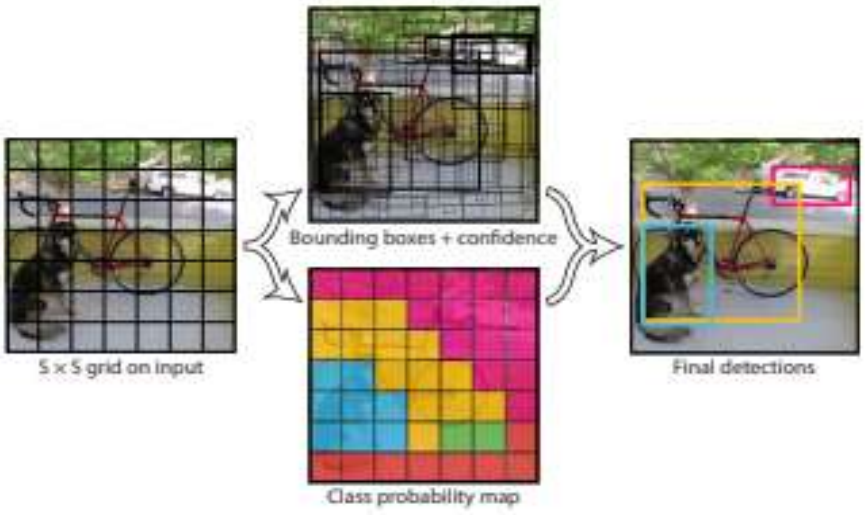
   Setelah membuat citra yang berukuran besar menjadi *array* kecil, berikutnya adalah menggunakan *array* kecil sebagai masukkan untuk *neural network*. Masukkan nilai ini adalah proses akhir untuk menentukan apakah citra tersebut memiliki kecocokan atau tidak. Langkah ini disebut *fully connected network* untuk membedakan dengan Langkah konvolusi sebelumnya.



Gambar 2. 18 Proses Prediksi

2.4. YOLO

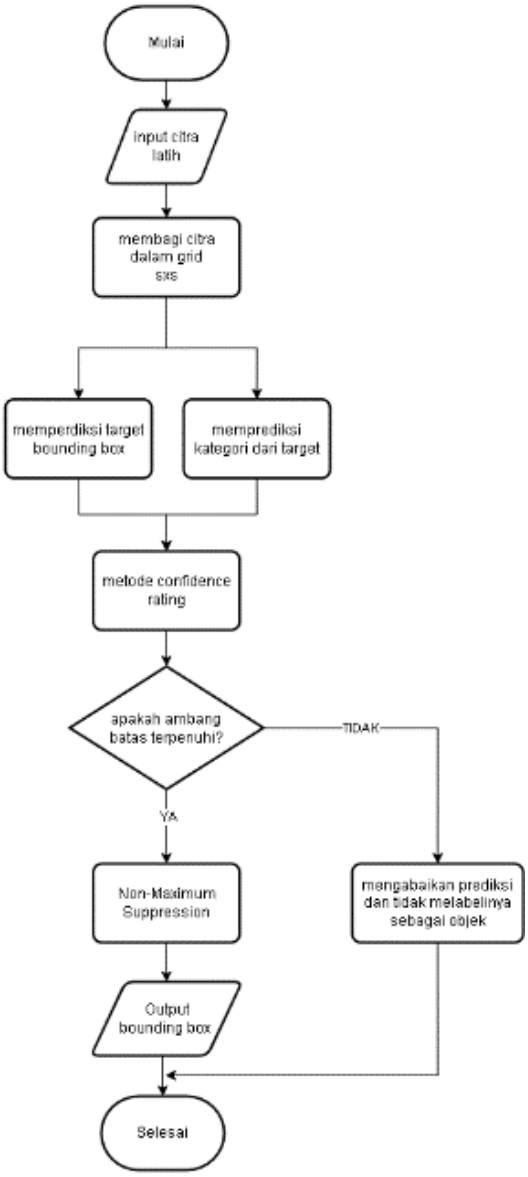
YOLO (*You Only Look Once*) adalah salah satu algoritma yang digunakan untuk pengenalan objek yang dibuat sebagai suksesor dari berbagai algoritma pendeteksi yang telah ada sebelumnya. YOLO menggunakan *Convolutional Neural Network* dalam proses deteksi objek. Adapun 3 tahap pendeteksian dengan menggunakan YOLO sebagai berikut; pertama-tama citra yang menjadi masukan akan diubah ukurannya, kemudian *Single Convolutional*



Gambar 2. 19 Gambaran pemrosesan YOLO

Network akan dijalankan pada citra masukkan yang telah disesuaikan ukurannya dan terakhir akan diterapkan *threshold* yang mengacu pada nilai *confidence*.

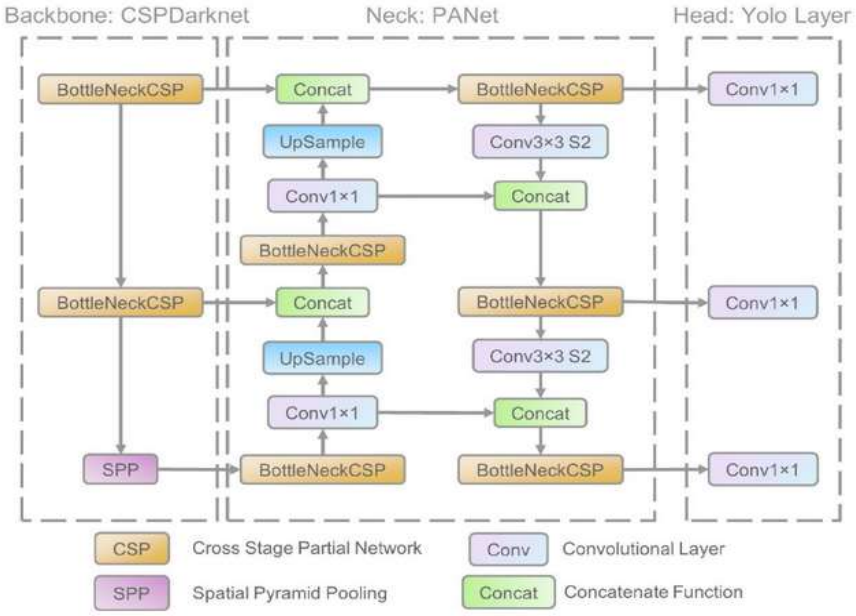
YOLO membagi citra masukkan menjadi *grid* berukuran S x S. Setiap *grid* yang terbentuk akan bertanggung jawab untuk memprediksi keberadaan objek, *bounding box* dan nilai *confidence*. Nilai *confidence* akan memprediksikan kehadiran sebuah objek. Setiap *bounding box* memiliki beberapa parameter yaitu *x*, *y*, *w*, *h*, dan *confidence score*. Misalkan terdapat sebuah gambar yang dibagi kedalam grid 7x7 yang mana setiap sel terdiri dari prediksi dua *bounding box*, sehingga total terdapat 98 prediksi *bounding box* yang diusulkan. Kelas probabilitas dan *bounding box* dengan *confidence* lalu digabungkan menjadi sebuah *bounding box* akhir dan label kelas. Proses kerja YOLO dapat dilihat pada *flowchart* pada gambar 2.20.



Gambar 2. 20 Flowchart Algoritma YOLO

2.4.1. Arsitektur YOLO

Jaringan arsitektur dari YOLO terdiri dari 3 bagian yaitu *backbone* CSPDarkNet53 sebagai pengekstraksi fitur, *Neck* PANet sebagai agreagator fitur dan *Head* YOLO Layer yang akan melakukan prediksi terhadap setiap objek yang ada pada sebuah citra. YOLO memilik arsitektur yang terinspirasi dari GoogleNet, memiliki 24 lapisan konvolusi dengan 2 *fully conncected layer*. YOLO menggunakan lapisan konvolusi 1x1 kemudian diikuti dengan lapisan konvolusi 3x3. Berikut merupakan gambaran dari arsitektur YOLO pada gambar 2.21.



Gambar 2. 21 Arsitektur YOLO v-5

2.4.2. YOLO v-5

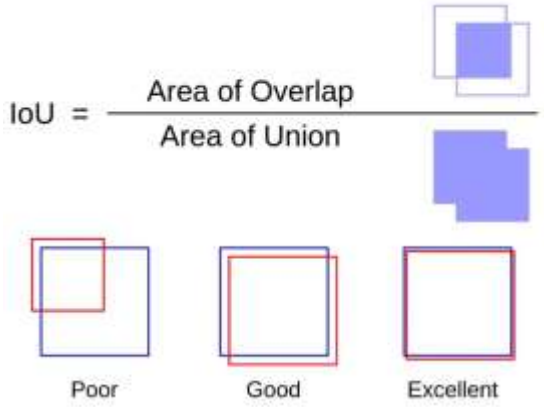
Joseph Redmon memperkenalkan struktur *anchor box* pada YOLOv2. Dalam pendekatan ini, algoritma *k-means clustering* digunakan untuk memilih 5 *anchor box* terbaik yang mendekati *ground truth bounding box* pada dataset COCO yang terdiri dari 80 kelas. Penggunaan *anchor box* ini secara *default* memberikan beberapa keuntungan, seperti mengurangi waktu pelatihan dan meningkatkan akurasi jaringan.

Namun, ada tantangan ketika menerapkan *anchor box default* ini pada dataset custom yang tidak termasuk dalam 80 kelas COCO. *Anchor box* tidak dapat dengan cepat beradaptasi dengan *bounding box ground truth* dari dataset custom. Sebagai contoh, dataset jerapah lebih memilih *anchor box* yang lebih tipis dan lebih tinggi daripada kotak berbentuk persegi. Untuk mengatasi masalah ini, para peneliti *computer vision* biasanya menjalankan algoritma *k-means clustering* pada dataset unik untuk mendapatkan *anchor box* terbaik yang sesuai dengan data.

Dalam upaya untuk mengatasi kendala tersebut, Glenn Jocher mengusulkan integrasi proses pemilihan *anchor box* ke dalam YOLOv5. Dalam pendekatan ini, jaringan tidak perlu mempertimbangkan dataset apa pun sebagai input. Sebaliknya, jaringan secara otomatis "mempelajari" *anchor box* terbaik untuk dataset tersebut selama proses pelatihan. Dengan demikian, jaringan dapat secara dinamis menyesuaikan anchor box berdasarkan karakteristik unik dari dataset yang digunakan, tanpa memerlukan konfigurasi manual.

2.4.3. Intersection over Union

Intersection over Union (IoU) berfungsi untuk menghitung level presisi *bounding box*. Setidaknya 2 *bounding box* diperlukan dalam perhitungan IoU. Perhitungan IoU adalah perhitungan antara *ground-truth* dan *predicted box*.



Gambar 2. 22 Perhitungan IoU

Untuk melakukan perhitungan nilai irisan dan gabungan dari rumus pada Gambar 2.22, maka perhitungannya adalah sebagai berikut.

$$I\_H = \min(W1,W2) - \max(X1,X2) \quad \textbf{(2.9)}$$

$$I\_W = \min(H1,H2) - \max(Y1,Y2) \quad \textbf{(2.10)}$$

$$\text{Area of Overlap} = I\_H * I\_W \quad \textbf{(2.11)}$$

$$\text{Area of Union} = [(W_1 - X_1) * (H_1 - Y_1)] + [(W_2 - X_2) * (H_2 - Y_2)] - \text{Area of Overlap} \quad \textbf{(2.12)}$$

$$\text{IoU} = \text{Area of Overlap} / \text{Area of Union} \quad \textbf{(2.13)}$$

Keterangan :

X_1,X_2 = Koordinat x dari *bounding box*

Y_1,Y_2 = Koordinat y dari *bounding box*

W_1,W_2= Lebar dari *bounding box*

H_1,H_2 = Tinggi dari *bounding box*

I_H = Nilai Tinggi Irisan

I_W = Nilai Lebar Irisan

Nilai IoU ini akan digunakan sebagai nilai *confidence score* atau nilai kepercayaan yang muncul Bersama dengan *bounding box* ketika pengujian.

2.4.4. Non-Maximum Suppression (NMS)

Algoritma dapat menemukan beberapa deteksi objek yang sama. *non-maximum suppression* adalah teknik yang digunakan algoritma untuk mendeteksi objek hanya satu kali. Pertimbangkan contoh di mana algoritma mendeteksi tiga *bounding box* untuk objek yang sama. Jika banyak *bounding box* yang *overlap* antara satu dengan yang lainnya, maka diambil yang memiliki *confidence* tertinggi. Jika *Intersection over Union* (IoU) > *Threshold* yang ditentukan, maka akan dianggap overlap dan akan dihilangkan atau dilakukan *suppression*. Jadi, NMS digunakan untuk memfilter prediksi objek yang berlebihan dan memastikan bahwa setiap objek hanya diwakili oleh satu *bounding box* saja.

Untuk melihat *bounding box* mana yang nantinya akan dihilangkan pada proses *suppression* maka kita perlu melihat *value* dari setiap *bounding box*. Setiap *bounding box* memiliki format seperti pada persamaan 2.14 berikut.

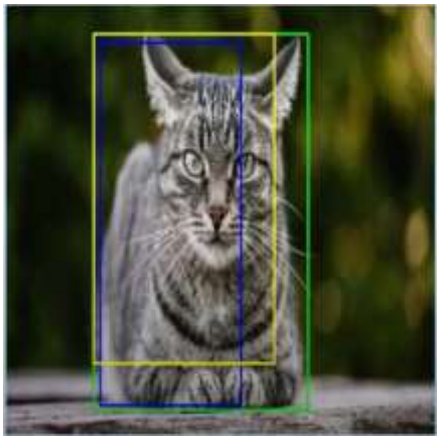
$$Bbox = [x_1, y_{1,} x_{2,} y_2, class, confidence] \quad \textbf{(2.14)}$$

Jika diasumsikan dari dari ketiga bounding box pada gambar 2.23 dengan nilai sebagai berikut

green_box = [x1,y1,x2,y2,"Cat",0.9]

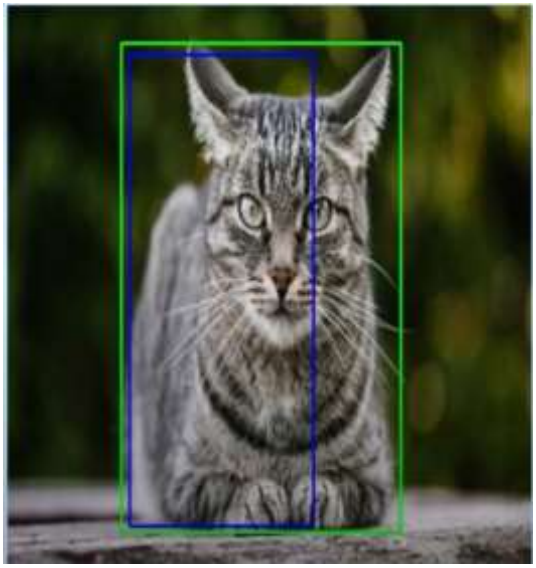
yellow_box = [x5,y5,x6,y6,"Cat",0.75]

blue_box = [x3,y3,x4,y4,"Cat",0.85]



Gambar 2. 23 Prediksi Bounding box sebelum suppression

dan jika diasumsikan nilai *threshold* yang ditentukan sebesar 0.8, maka *yellow_box* yang memiliki *confidence score* lebih kecil dari nilai *threshold* akan dihilangkan sehingga hanya tersisa *bounding box* yang memiliki nilai sama dengan 0.8 atau lebih dari 0.8. Sehingga menghasilkan sebuah citra seperti pada gambar 2.20



Gambar 2. 24 Hasil proses suppression

2.4.5. Loss Function

Untuk melihat apakah model dapat secara akurat memprediksi objek dan mengukur kinerja model, maka dibutuhkan sebuah metode untuk membantu hal tesebut, maka dari itu dibutuhkan adanya perhitungan Loss Function YOLO. Terdapat 3 aspek loss yang akan dihitung dalam

perhitungan *Loss Function* YOLO ini, *Bounding Box*, *Confidence* dan *Classification*. Berikut adalah persamaan 2.15 dari *Loss Function* YOLO secara keseluruhan.

$$loss_{i,j} = loss_{i,j}^{x,y,w,h} + loss_{i,j}^{c} + loss_{i,j}^{p} \quad \textbf{(2.15)}$$

Bounding Box loss digunakan untuk menghitung perbedaan antara prediksi *bounding box* yang dihasilkan oleh model dengan *bounding box* yang sebenarnya pada gambar. Tujuan dari *Bounding Box loss* adalah untuk meminimalkan perbedaan antara prediksi *bounding box* dan *bounding box* yang sebenarnya, sehingga model dapat menghasilkan deteksi objek yang lebih akurat. Bounding Box loss pada YOLO menggunakan *Mean Squared Error* (MSE) sebagai metrik untuk menghitung perbedaan antara prediksi bounding box dan bounding box yang sebenarnya. MSE menghitung perbedaan antara nilai prediksi dan nilai sebenarnya, kemudian mengkuadratkan selisih tersebut dan menjumlahkan seluruh nilai selisih yang dihasilkan. Adapun untuk mendapatkan *Loss Function* dari *Bounding Box Loss* dalam persamaan 2.15 di tuliskan dengan $loss_{i,j}^{x,y,w,h}$ adalah dengan persamaan 2.16 berikut.

$$\begin{aligned} loss_{i,j}^{x,y,w,h} = \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} L_{i,j}^{obj} \left[\left(x_{i,j} - \hat{x}_{i,j}\right)^2 + \left(y_{i,j} - \hat{y}_{i,j}\right)^2 \right] \\ + \sum_{i=0}^{s^2} \sum_{j=0}^{B} \left[(\sqrt{w_{i,j}} - \sqrt{\hat{w}_{i,j}})^2 + (\sqrt{h_{i,j}} - \sqrt{\hat{h}_{i,j}})^2 \right] \end{aligned} \quad \textbf{(2.16)}$$

Sedangkan *Confidence loss* digunakan untuk menghitung perbedaan antara skor kepercayaan yang diprediksi dan skor kepercayaan yang sebenarnya untuk setiap kotak pembatas dalam sebuah gambar. Skor kepercayaan mewakili probabilitas bahwa sebuah objek ada di dalam kotak pembatas. Fungsi *confidence loss* digunakan untuk memberikan nilai pada model karena memprediksi nilai *confidence* yang rendah untuk objek yang benar dan nilai *confidence* yang tinggi untuk objek yang salah. Fungsi *confidence loss* dihitung menggunakan *binary cross-entropy*, yang mengukur perbedaan antara probabilitas yang diprediksi dengan probabilitas sebenarnya. Persamaan 2.17 menunjukkan rumus *confidence loss*.

$$\begin{aligned} loss_{i,j}^{c} = \sum_{i=0}^{S^2} \sum_{j=0}^{B} L_{i,j}^{obj} \left(IoU - \hat{C}_{i,j}\right)^2 \\ + \lambda_{noobj} \sum_{i=0}^{S^2} \sum_{j=0}^{B} L_{i,j}^{obj} \left(0 - \hat{C}_{i,j}\right)^2 \end{aligned} \quad \textbf{(2.17)}$$

Sedangkan bagian loss terakhir adalah classification loss, Jika sebuah objek terdeteksi, maka *loss* pada klasifikasi di setiap sel adalah *squared error* dari estimasi probabilitas suatu *class* dan probabilitas suatu *class* yang sebenarnya, untuk menghitung classification loss dapat dilihat pada Persamaan2.18 berikut.

$$loss_{i,j}^{p} = \sum_{i=0}^{s^2} L_{i,j}^{obj} \sum_{c \in classes} (p_{i,j}(c) - \hat{p}_{i,j}(c))^2 \quad \textbf{(2.18)}$$

Keterangan :

$L_{i,j}^{obj} = \begin{cases} 1 \text{ jika } c_{i,j} = 1 \\ 0 \end{cases}$

$L_{i,j}^{noobj} = \begin{cases} 1 \text{ jika } max_{i,j} \text{dari IoU} < \text{Threshold dan } c_{i,j} = 0 \\ 0 \end{cases}$

$\sum_{j=0}^{B}$ = Jumlah loss dari setiap anchor box

$\sum_{i=0}^{s^2}$ = Jumlah loss dari SxS grid

$L_{i,j}^{obj}$ = Indikasi objek dengan nilai $0/1$

λ_{coord} = Konstanta untuk mengatur loss pada prediksi bounding box dengan nilai 5

λ_{noobj}
= Konstanta untukmengatur loss pada confidence untuk grid tanpa objek dengan nilai 0.5

$x_{i,j}, y_{i,j}, w_{i,j}, h_{i,j}$ = nilai bounding box dari hasil prediksi

$\hat{x}_{i,j}, \hat{y}_{i,j}, \hat{w}_{i,j}, \hat{h}_{i,j}$ = nilai bounding box dari data masukan

$\hat{C}_{i,j}$ = nilai confidence objek pada dari data masukan

$P_{i,j}(c)$ = probabilitas terdapat class hasil prediksi

$\hat{p}_{i,j}(c)$ = probabilitas terdapat class sebenarnya

2.5. Python

Bahasa python merupakan Bahasa pemrograman yang termasuk dalam kategori *interpreted high-level programing* yang bisa digunakan untuk berbagai tujuan. Dalam penelitian ini Pyhton digunakan unntuk memproses gambar input, dan digunakan untuk menjalankan proses training model YOLO v-5 dan melakukan pengujian, dibutuhkan beberapa library yang diperlukan agar packages YOLO dapat berjalan. Beberapa library yang diperlukan gitpython, CUDA (bersifat opsional bagi yang memiliki GPU Nvidia), matplotlib, numpy, opencv-python, Pillow, PyYAML scipy, torch, torchvision, tqdm, dan ultralytics.