BAB 2
LANDASAN TEORI

2.1 Lalu Lintas dan Pelanggaran Lalu Lintas

Lalu lintas biasanya terdiri dari pengguna jalan termasuk pejalan kaki, hewan yang dikendarai, kendaraan, bus, dan alat angkut lainnya, baik secara sendiri - sendiri atau bersama - sama saat berada di jalan umum untuk tujuan perjalanan. Lalu lintas dapat diartikan sebagai suatu kesatuan yang terdiri atas lalu lintas dan angkutan beserta jaringan dan prasarananya, pengguna jalan serta pengelolanya sebagaimana yang tercantum pada Pasal 1 Undang - Undang Nomor 22 tahun 2009 tentang Lalu Lintas dan Angkutan Jalan [9]. Peraturan Lalu lintas dibutuhkan agar aliran lalu lintas tetap tertib dan tidak terhambat. Namun yang menjadi alasan utamanya ialah karena terlalu banyaknya pengguna jalan dan kendaraan yang mencoba menempati jalan yang terbilang terbatas. Ketidakmampuan pengguna jalan untuk mempertahankan kecepatan dan mengikuti jarak di jalan raya menjadi hal lain yang membuat lalu lintas tidak lancar sebagaimana mestinya.

Sementara Undang - undang Lalu lintas dibuat dengan dua tujuan. Undang - undang dibuat untuk menangani pelanggaran yang sifatnya bergerak seperti kecepatan kendaraan, pelanggaran rambu Lalu lintas, merubah jalur sembarangan dan sebagainya. Tujuan lain dibentuknya Undang - undang Lalu lintas adalah menangani pelanggaran yang sifatnya tidak bergerak untuk memastikan kendaraan yang dipakai sesuai standar atau sesuai dengan kondisi hukum. Contoh dari pelanggaran ini seperti pendaftaran kendaraan dan mesin kendaraan yang sesuai.

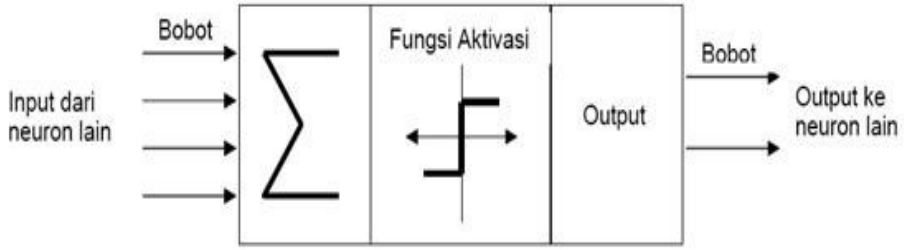
Penggunaan helm saat mengendarai kendaraan sepeda motor merupakan salah satu perlengkapan kendaraan bermotor yang diwajibkan dalam peraturan Indonesia. Helm yang dimaksudkan yaitu helm yang memiliki Standar Nasional Indonesia seperti yang tercantum pada Pasal 106 (8) Undang - Undang Nomor 22 tahun 2009 tentang Lalu Lintas dan Angkutan Jalan [9]. Peraturan ini berlaku baik untuk pengemudi maupun penumpang sepeda motor.

Gambar 2.1 Penumpang motor yang tidak mengenakan helm

2.2 *Artificial Neural Network*

Artificial Neural Network (Jaringan Syaraf Tiruan) adalah istilah yang berasal dari jaringan syaraf biologis yang membangun struktur otak manusia [10]. Seperti otak manusia yang memiliki neuron yang saling terhubung satu sama lain, ANN juga memiliki neuron yang saling terhubung satu sama lain di berbagai lapisan jaringan. Neuron-neuron ini disebut sebagai node. ANN sangat cocok untuk masalah dimana data latih banyak noise, data sensor yang kompleks seperti *input* dari kamera dan mikrofon. Proses yang dilakukan oleh ANN berupa *supervised learning,* yaitu dengan menggunakan contoh berupa data masukan dan keluaran. ANN dikonfigurasi sesuai tujuan, contohnya pengenalan pola atau memprediksi, kemudian dituntaskan melalui proses pembelajaran [11].

Struktur dari ANN berawal dari masukan yang diterima oleh neuron dengan bobot nilai dari setiap masukan yang ada. Selanjutnya nilai masukan dihitung oleh fungsi perambatan dan diproses untuk membandingkan hasil perhitungan dengan *threshold* (nilai ambang) dari setiap neuron oleh fungsi aktivasi. Aktivasi neuron dikatakan aktif jika hasilnya masih dibawah *threshold.* Kemudian neuron akan memberikan nilai keluaran ke semua neuron yang terhubung dengannya [9].



Gambar 2.2 Struktur ANN

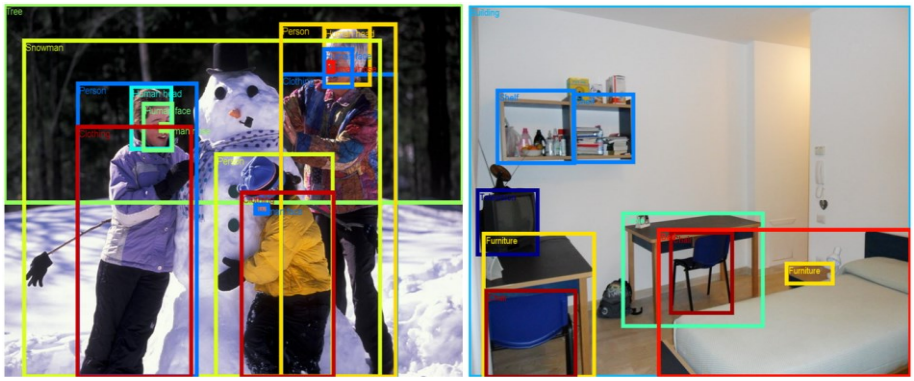
2.3 *Computer Vision*

Computer Vision adalah bidang yang mengajarkan komputer untuk mengevaluasi dan memahami visual. Dapat juga diartikan sebagai kemampuan mesin untuk menerima dan menganalisis data visual. Proses yang dilakukan bisa berupa *extracting* (ekstraksi), *understanding* (memahami), *learning* (mempelajari) dari sebuah atau banyak gambar [12]. Tujuan utamanya adalah memungkinkan komputer untuk melakukan tugas yang sama seperti manusia dengan tingkat akurasi yang sama. *Computer Vision* sangat terkait dengan pembelajaran mesin dan deteksi objek. Dengan menggunakan gambar dari kamera dan video serta dilakukannya proses pembelajaran, mesin secara tepat dapat mengidentifikasi dan mengklasifikasikan objek. *Computer Vision* dapat membantu masyarakat karena dapat menganalisis dan memahami lingkungan secara mendalam dan efektif.

2.3.1 Bounding Box

Bounding box membantu untuk menandai sebuah objek yang ingin dikenali pada gambar membuatnya dapat diketahui untuk *Computer Vision* [13]. *Bounding box* digunakan untuk mendeteksi objek pada berbagai bidang seperti mobil *self-driving*, drone, kamera pengintai / cctv dan, robot serta semua jenis sistem yang menggunakan *Computer Vision*.

Saat komputer digunakan untuk mengklasifikasi gambar atau pengenalan gambar untuk mendeteksi probabilitas suatu objek, *bounding box* dapat memvisualisasikan gambar dengan menetapkan pembatas pada objek [13]. *Bounding box* bekerja dengan cara menghasilkan koordinat yang merupakan lokasi suatu objek pada gambar dan memberi pembatas serta label pada objek tersebut.



Gambar 2.3 Implementasi Bounding Box

2.3.2 Image Resizing

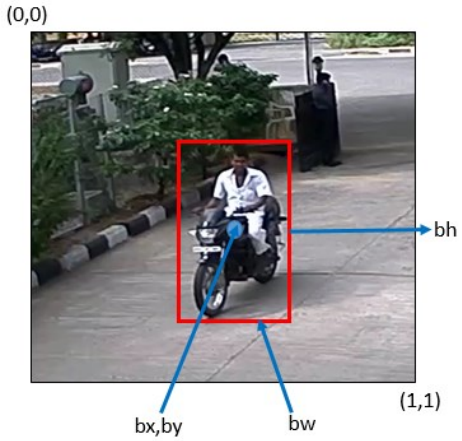
Image Resizing atau perubahan ukuran citra dilakukan untuk memperkecil ukuran citra agar komputer mampu memproses citra dengan cepat [14]. Gambar bitmap menyimpan serangkaian pixel dimana jika dilakukan *Resizing* akan mulai terlihat satu persatu pixel yang membentuk gambar. Hal ini dapat mudah terlihat pada tepi gambar. Efeknya membuat gambar tidak jelas karena banyak informasi warna yang hilang pada saat *Resizing*. Berbeda halnya dengan gambar vektor saat dilakukan *Resizing*, gambar vektor akan digambar ulang menggunakan rumus matematika, sehingga gambar yang dihasilkan sehalus gambar aslinya. Hal ini dikarenakan gambar vektor tidak berdasarkan pada pixel, melainkan menggunakan rumus matematika untuk menggambar garis dan kurva yang dapat digabungkan untuk membuat gambar.

2.3.3 Object Localization

Klasifikasi merupakan pengenalan suatu objek yang terdapat pada frame, apakah pada frame tertentu terdapat suatu atau beberapa objek. Lokalisasi sendiri merupakan proses pencarian objek yang terdapat pada suatu frame. Untuk bisa mencari objek yang diinginkan, maka perlu dilakukan klasifikasi terlebih dahulu.

Apabila suatu objek yang akan dideteksi berupa mobil, output yang dihasilkan berupa *softmax* yang berisi klasifikasi objek yang dideteksi dan output *bounding box* / kotak pembatas yang dapat dinotasikan sebagai parameter

bx,by,bh,bw. Parameter – parameter tersebut merupakan nilai relatif antara 0 sampai 1 [5].



Gambar 2.4 *Object Localization*

Jika output didefinisikan sebagai y, maka:

$$y = [Pc,bx,by,bh,bw,Cn]$$

Dimana :

Pc = klasifikasi objek, jika ada suatu objek maka bernilai 1

bx,by = koordinat tengah dari bounding box

bh = tinggi *bounding box*

bw = lebar *bounding box*

Cn = *class probability* objek ke-n

2.3.4 Deteksi Objek

Deteksi Objek merupakan teknik yang berkaitan dengan mendeteksi suatu objek dari *class* tertentu (seperti manusia, bangunan, atau mobil) dalam gambar atau video digital [15]. Mesin akan memproses gambar atau video dan akan menghasilkan keluaran setiap objek yang terdeteksi beserta label yang sesuai dengan *class*. Mesin juga akan menampilkan *bounding box* (kotak pembatas) pada objek yang terdeteksi dan akan menampilkan nilai akurasi dari pendeteksian tersebut. Metode deteksi objek dikategorikan menjadi tiga kelas utama: deteksi berdasarkan gerakan, deteksi berdasarkan fitur penampilan, dan deteksi berbasis *convolutional neural network* (CNN).

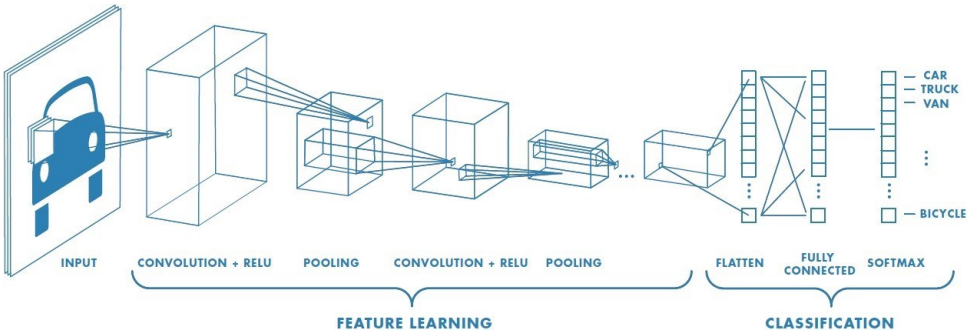
Deteksi Berdasarkan Fitur penampilan memfokuskan pada fitur atau rupa bentuk dari objek, seperti warna, bentuk, arah tampilan objek dan sebagainya. Maka dari itu pemilihan fitur sangat berpengaruh besar pada keseluruhan kinerja sistem dalam melakukan deteksi.

Mendeteksi objek bergerak artinya mengenali gerakan fisik suatu objek di tempat atau wilayah tertentu. Cara kerjanya dengan melakukan segmentasi diantara objek yang bergerak dengan area yang tidak berubah. Objek yang bergerak tersebut dapat dilacak dan dianalisis gerakannya dengan cara mencari objek target yang bergerak dari setiap frame pada video atau hanya pada saat objek menunjukkan pergerakan pertamanya pada video [16].

Sedangkan pada Deteksi berbasis *Convolutional Neural Network* (CNN) yang merupakan sebuah *neural network* yang terdiri dari beberapa lapisan berbeda seperti lapisan *input layer* (masukan), setidaknya ada satu *hidden layer* (lapisan tersembunyi), dan *output layer* (lapisan keluaran) [17]. *Hidden layer* bertugas seperti filter yang menerima masukan awal, kemudian merubahnya menggunakan fitur tertentu, lalu memindahkannya ke lapisan berikutnya [17]. Perubahan yang dilakukan pada setiap *hidden layer* berbeda - beda, misalkan satu layer mengidentifikasi objek berwarna hijau, kemudian *hidden layer* berikutnya mungkin menyimpulkan bahwa objek itu berupa daun. Kesimpulannya, semakin banyak layer yang ada, objek yang dideteksi bisa beragam.

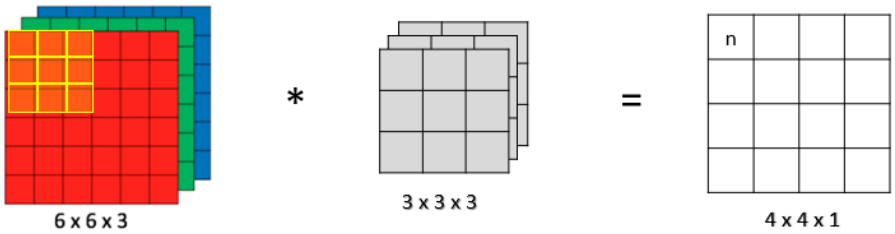
2.4 CNN (Convolutional Neural Network)

Arsitektur CNN secara umum terbagi menjadi *Feature Extraction Layer* dan *Fully-Connected Layer* [18]. Setiap bagiannya terdiri atas lebar (*width*), tinggi (*height*) dan *channel*. Lebar dan tinggi mewakili dimensi citra sedangkan *channel* mewakili kedalaman citra seperti 3 *channel* pada RGB (Red Green Blue) ataupun jumlah filter yang dilakukan saat konvolusi. Citra yang menjadi masukan akan diproses pada *layer* CNN berdasarkan dimensi filter dan jumlah filternya.



Gambar 2.5 Convolutional Neural Network

Operasi Konvolusi dilakukan pada *convolutional layer* dengan mengalikan setiap filter dengan citra asli dimulai dari posisi kiri atas. Hasil tersebut kemudian ditambahkan dan akan menghasilkan satu nilai baru. Selanjutnya filter akan bergeser ke horizontal lalu vertikal sebanyak *stride* [18]. Antara input dan nilai dari filter menghasilkan sebuah output (*activation map*).

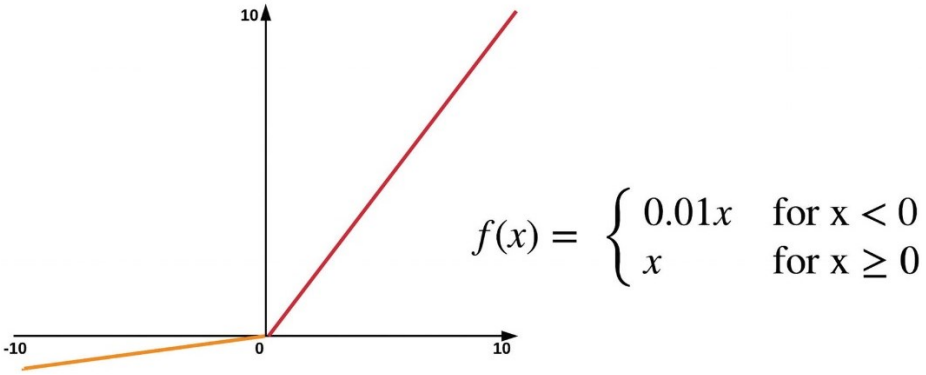


Gambar 2.6 Proses Konvolusi

Setiap filter diberi bias dimana bisa bernilai 1 lalu diaplikasikan Fungsi aktivasi non-linearitas seperti LeakyReLU (rectified liner unit) di mana untuk nilai negatif ***x***, hasil fungsinya didefinisikan sebagai komponen linear ***x*** yang sangat kecil [19]. Maka proses *convolutional layer* akan menghasilkan bobot (*weights*) yang berguna untuk proses pendeteksian dengan nilai berupa

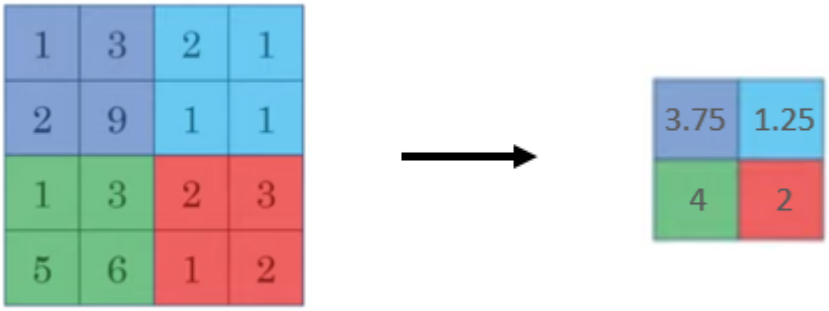
$$weights = f[l] \; x \; f[l] \; x \; nc[l-1] \; x \; nc[l] \qquad (2.1)$$

Dimana *weights* terdiri dari dimensi filter (lebar dan tinggi) pada suatu *layer l*, jumlah filter pada *layer* sebelumnya dan jumlah filter pada *layer* yang sama dengan dimensi filter.



Gambar 2.7 Grafik LeakyReLU

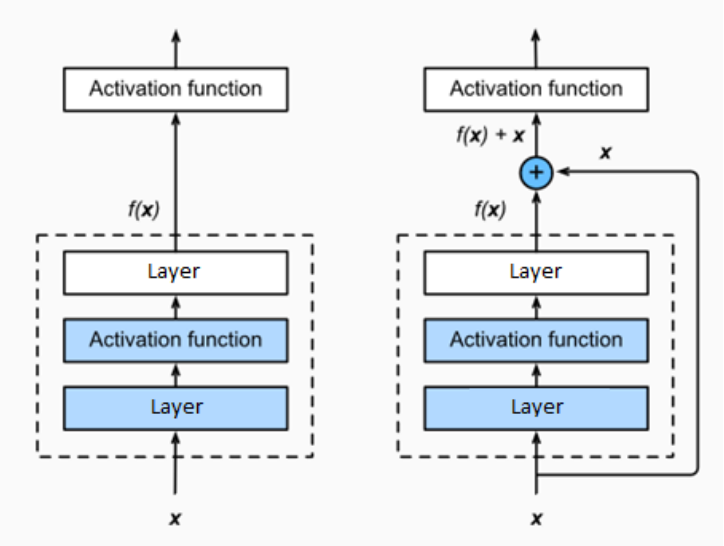
Pada proses CNN juga terdapat *pooling layer* untuk mengurangi ukuran dimensi citra yang fungsinya untuk mempercepat proses komputasi [18]. *Pooling layer* juga dapat membuat beberapa fitur yang terdeteksi sedikit lebih optimal. *Average pooling* merupakan salah satu metode *pooling* yang populer digunakan. Langkah yang dilakukan adalah mengambil rata - rata dari setiap wilayah yang terkena filter / *mask.*



Gambar 2.8 Proses Average Pooling

2.5 *Residual Block*

Neural Network yang memiliki banyak lapisan sangat sulit dilatih karena banyaknya proses perubahan citra. *Residual Block* melakukan melewatkan koneksi dari suatu *layer* menuju fungsi aktivasi dan melanjutkan proses ke *layer* berikutnya yang lebih dalam [20].



Gambar 2.9 Proses *Residual Block*

Gambar 2.9 menunjukkan proses *Residual Block* dimana jika sebelumnya terdapat nilai ***x*** kemudian melakukan komputasi dari layer ke fungsi aktivasi, kemudian ke layer selanjutnya dan menghasilkan nilai ***f(x).*** Jika hal ini dilakukan terus menerus maka dimensi input akan mengecil. Pada proses *Residual Block* dilakukan *"shortcut"* yang mana alih – alih melakukan proses secara linear, *Residual Block* melakukan pemotongan untuk mempercepat proses ke layer yang lebih dalam. Maka *activation map* tidak akan mengecil karena proses pemotongan.

2.6 Fully Connected Layer

Output berupa *activation map* merupakan hasil yang masih berupa dimensi lebar (*width*), tinggi (*height*) dan *channel.* Maka dilakukan perubahan (*flatten*) agar output menjadi vektor sehingga data dapat diklasifikasikan secara linear [18]. Output diubah menjadi 1 x 1 dimensi dimana jumlahnya didapat dari perkalian antara lebar (*width*), tinggi (*height*) dan *channel* ditambah jumlah bias. Proses ini akan berlanjut pada *softmax* yang menghitung probabilitas dari setiap *class* yang memungkinkan menentukan *class* dari input citra yang diberikan. Jika dimisalkan a = [-0.21, 0.47, 1.72], dan output y adalah y = softmax(a).

$$y = \frac{e^{a_i}}{\sum_j e^{a_j}} \tag{2.2}$$

$$y = \left[\frac{e^{-21}}{e^{-21}+e^{47}+e^{1.72}}, \frac{e^{47}}{e^{-21}+e^{47}+e^{1.72}}, \frac{e^{1.72}}{e^{-21}+e^{47}+e^{1.72}}\right]$$

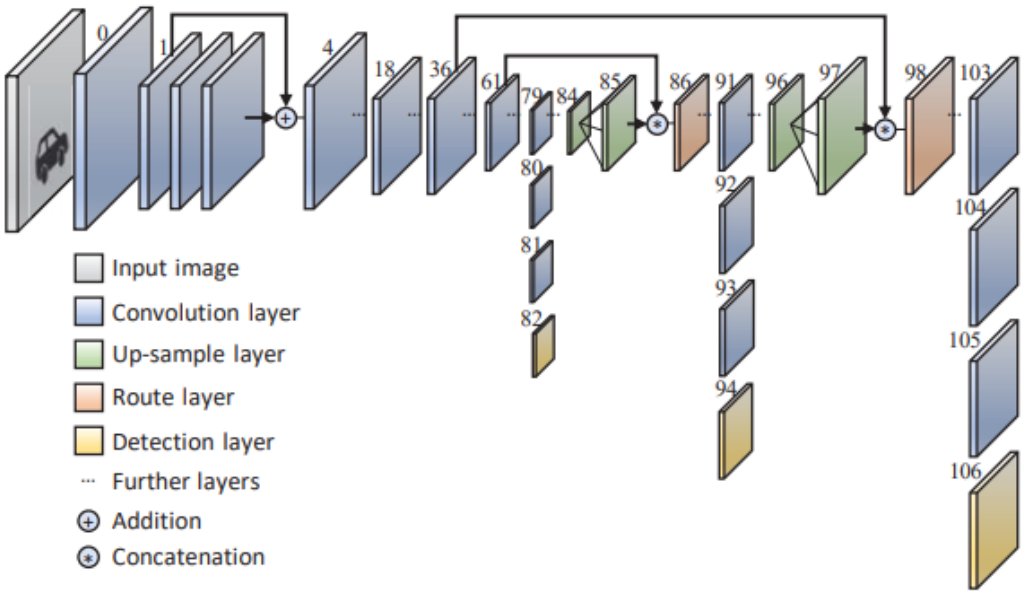
$$y = \left[\frac{e^{-21}}{8}, \frac{e^{47}}{8}, \frac{e^{1.72}}{8}\right]$$

$$y = [0.1, 0.2, 0.7]$$

Vektor diatas menunjukan probabilitas 10%, 20% dan 70%.

2.7 YOLO (You Only Look Once)

YOLO merupakan salah satu dari metode pendeteksian objek berbasis berbasis *convolutional neural network* (CNN). Arsitektur YOLO bekerja "*Only Look Once*" atau hanya melihat sekali pada suatu gambar atau frame sehingga hanya memerlukan satu *forward propagation* pada *neural network* untuk melakukan prediksi. Satu CNN secara bersamaan memprediksi beberapa *bounding box* sehingga proses yang dilakukan pada gambar hanya sekali untuk melihat objek apa saja yang ada dan letak objek tersebut [5].



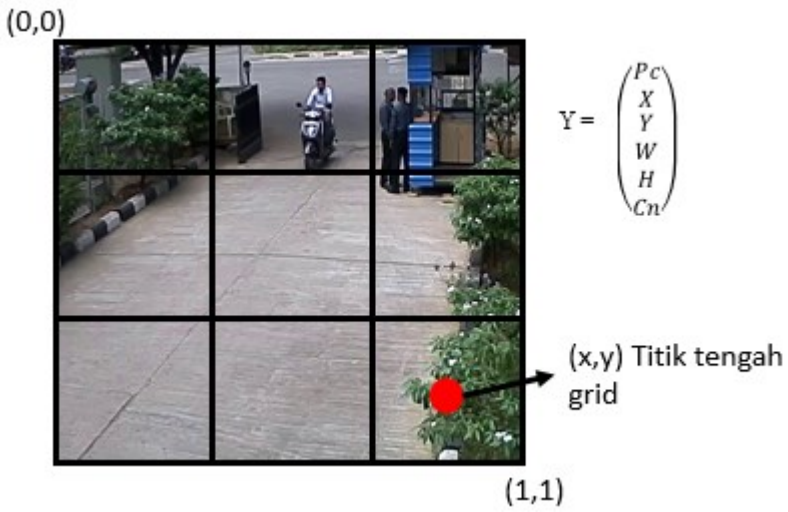
Gambar 2.10 Arsitektur YOLO

YOLO menggunakan 75 *Convolutional Layer* dengan *Residual Block* dan juga *upsampling layer* dengan total 106 layer. Pada Convolutional layer digunakan *stride* sebanyak 2 dengan ukuran filter 32, 64 dan 128 untuk melakukan *downsample* pada data masukan [21]. Sedangkan proses deteksi memiliki 3 layer deteksi menggunakan *stride* 32, 16, dan 8 [21]. Pada layer deteksi pertama memiliki stride 32 dan digunakan filter 1 x 1 ditugaskan untuk mendeteksi objek yang lebih besar. *Stride* 8 digunakan untuk deteksi objek kecil dan *stride* 16 untuk objek yang berukuran sedang.

Proses yang dilakukan YOLO network ialah melakukan *forward propagation* dari data masukan dimisalkan 100 x 100 x 3 menjadi S x S x (B * (5 + Cn)). YOLO membagi gambar menjadi S x S grid. Setiap bagiannya berkewajiban untuk memprediksi 3 *bounding box*, kotak yang mencakupi objek yang dideteksi [5].

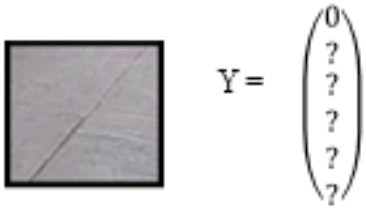
Gambar 2.11 Gambar dibagi menjadi SxS bagian

Tiap grid cell terdapat *bounding box* (B) yang berisi 5 nilai, yaitu lokasi koordinat x diasumsikan berdasar baris (x), koordinat y berdasarkan kolom (y), ukuran dan nilai *confidence* (X,Y,W,H,Cn) terhadap *bounding box* yang ada serta nilai Pc untuk setiap grid yang menandakan adanya suatu objek dalam grid [5].



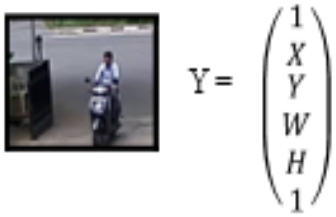
Gambar 2.12 Parameter setiap bounding box

Jika tidak pada suatu grid tidak terdapat objek maka nilai Pc akan bernilai 0, dan parameter lainnya akan dihiraukan karena tidak adanya objek maka tidak ada *bounding box* yang perlu ditentukan.



Gambar 2.13 Grid yang tidak memiliki objek beserta parameternya

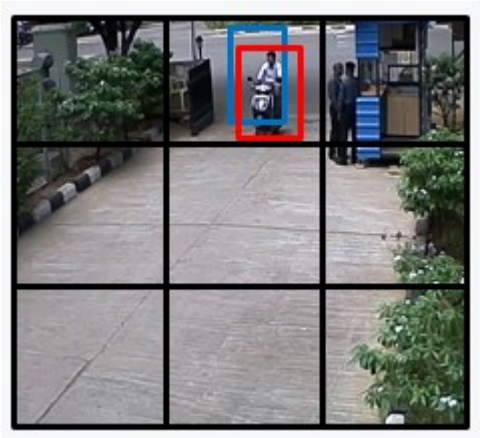
Sebelum menyatakan nilai Y, maka sangat penting bagi metode YOLO untuk mengambil titik tengah pada objek yang ingin dideteksi dan menyematkan titik tengah tersebut terhadap grid.



Gambar 2.14 Grid yang memiliki objek beserta parameternya

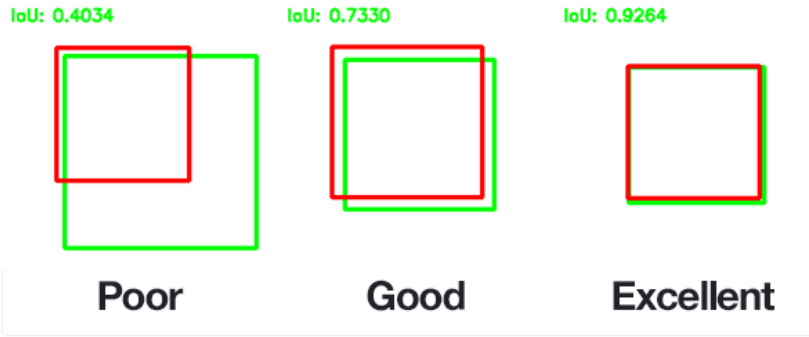
Sementara nilai X,Y,H,W merupakan nilai relatif terhadap grid. Maka Untuk mendeteksi pengendara motor yang tidak mengenakan helm, maka maka nilai keluaran akan berupa 3 x 3 x 18.

Nilai *confidence* didapat dari perhitungan IOU. IOU (Intersection Over Union) merupakan ukuran tumpang tindih antara dua *bounding box*. Pada Computer Vision digunakan untuk mendeteksi objek dengan benar. Saat pelatihan memprediksi objek dengan *bounding box*, maka *bounding box* yang diprediksi dibandingkan dengan *bounding box* dari *ground-truth* [22].



Gambar 2.15 Ilustrasi IOU

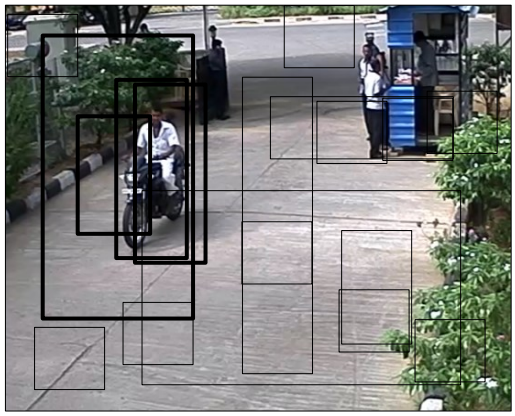
Bounding box yang diprediksi dapat dinilai dengan benar ketika IOU lebih besar dari ambang batas atau 0.5 sebagai contoh. Dalam kasus lain ambang batas bisa lebih besar dari 0.5 untuk hasil yang lebih akurat.



Gambar 2.16 Perbandingan Ukuran IOU

Setiap *bounding box* secara pararel melakukan proses identifikasi untuk mengetahui kategori *class* objek yang ada. Proses ini memberikan nilai probabilitas pada semua *class* yang dianggap memungkinkan. Contoh *class* berupa nohelm atau

pengendara motor tanpa helm. YOLO juga memperlihatkan "*confidence score*" yang menyampaikan seberapa pasti sebuah *bounding box* benar - benar mencakup suatu objek. Gambar 2.11 menunjukkan banyaknya *bounding box* pada satu masukan.

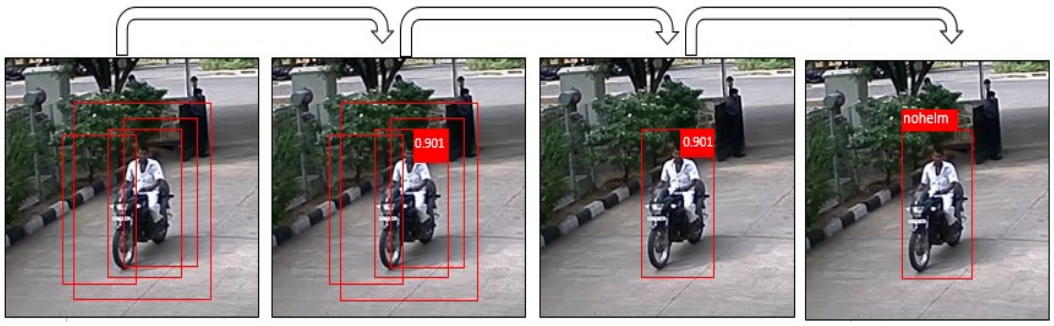


Gambar 2.17 *Bounding box*

Nilai dari *confidence score* dan nilai prediksi *class* dihitung menjadi suatu nilai akhir yang memberi informasi probabilitas bahwa *bounding box* telah memiliki jenis objek tertentu. Dengan demikian, pada contoh diatas ada 3 x 3 = 9 bagian dan setiap bagian memprediksi 3 *bounding box* dengan menggunakan *anchor box*, artinya ada sebanyak 27 *bounding box* pada gambar.

Sebagian besar *bounding box* akan memiliki *confidence score* yang rendah, maka yang diambil ialah yang memiliki nilai akhir diatas *threshold* atau ambang batas. Namun pada pendeteksian akan ditemukan banyak *bounding box* pada satu objek. Untuk mengatasi hal ini maka YOLO melakukan proses *Non-Max Suppression* [5].

Non-max suppression merupakan teknik menghilangkan *bounding box* yang sama yang tumpang tindih pada objek yang sama dengan memilih *bounding box* yang paling cocok dengan objek [5].



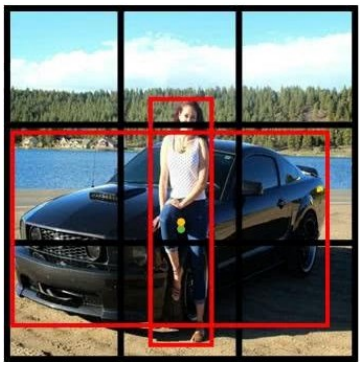
Gambar 2.18 *Non-Max Suppression*

Setelah menghapus semua *bounding box* yang memiliki probabilitas prediksi kurang dari ambang batas maka untuk setiap *class* diambil *bounding box* dengan probabilitas prediksi yang paling besar. Lalu *bounding box* yang memiliki IOU lebih besar dari ambang batas akan dihapus [22].

Gambar 2.19 Hasil Akhir

2.8 *Anchor Box*

Selama pelatihan diperlukan *Anchor box* yang memiliki koordinat dan ukuran yang sudah ditetapkan karena CNN tidak memperkirakan posisi sebuah bounding box secara mutlak. Ketika menggunakan *Anchor box*, CNN dapat memprediksi semua objek sekaligus. Menggunakan beberapa *Anchor box* dapat mendeteksi objek yang tumpang tindih. Dimensi dari *Anchor box* dimungkinkan cocok untuk objek yang memiliki bentuk yang berbeda seperti manusia yang tumpang tindih dengan mobil.



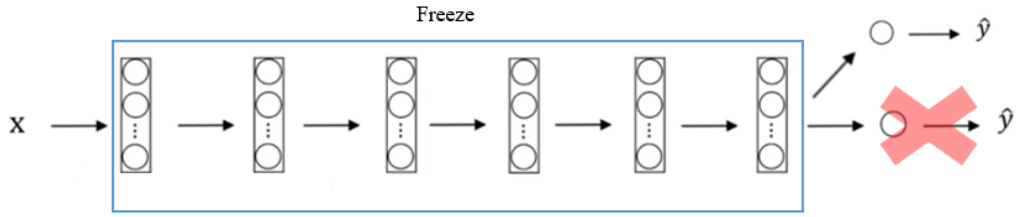
Gambar 2.20 Prediksi Objek Yang Tumpang Tindh

Untuk setiap anchor box akan dihitung IOU. Jika IOU lebih besar dari ambang batas, maka nilai IOU yang tertinggi yang dipilih sebagai *bounding box* hasil prediksi. Jika IOU kurang dari ambang batas maka *anchor box* akan mengira bahwa tidak ada objek dan tidak dipakai sebagai contoh pelatihan [22].

2.9 Transfer Learning

Transfer Learning adalah salah satu metode yang mempersingkat waktu pelatihan namun tetap mendapatkan model yang akurat [24]. *Transfer learning* adalah suatu teknik atau metode yang memanfaatkan model yang sudah dilatih terhadap suatu dataset untuk menyelesaikan permasalahan lain yang serupa dengan cara menggunakannya sebagai *starting point*, memodifikasi dan mengupdate parameternya sehingga sesuai dengan dataset yang baru. Proses ini dapat mempercepat pelatihan dengan menggunakan bobot pra-latih yang sebelumnya telah dilatih pada *Neural Network* dari awal dan melakukan pelatihan dengan objek baru yang akan dikenali.

Transfer learning bekerja dengan mengganti *softmax* yang ada pada bobot pra-latih yang digunakan dan menciptakan *softmax* baru yang sesuai dengan objek yang akan dikenali. Sedangkan layer yang lainnya pada keadaan "*freeze*" walaupun terdapat juga beberapa bobot pra-latih yang dapat memilih *layer* yang akan digunakan [24]. Maka proses pelatihan hanya melibatkan parameter *softmax* baru dan data masukan.



Gambar 2.21 Ilustrasi Proses *Transfer Learning*

2.10 Darknet

Darknet merupakan framework *neural network* yang berbasis *open-source* yang ditulis dalam bahasa pemrograman C dan CUDA. Darknet dapat mendukung komputasi CPU dan GPU [25]. Darknet dapat dipasang hanya dengan dua dependensi opsional yaitu OpenCV jika pengguna menginginkan lebih banyak jenis gambar yang ingin digunakan dan CUDA jika melakukan komputasi dengan GPU. Framework ini menampilkan YOLO yang bisa mengklasifikasikan gambar pada 40-90 FPS. Darknet menampilkan informasi saat memuat file konfigurasi (.cfg file) dan bobot (.weight file) kemudian mengklasifikasikan gambar dan menampilkannya bersama label yang teridentifikasi. Darknet juga dapat menampilkan informasi saat pelatihan untuk menghasilkan bobot baru yang akan digunakan untuk mendeteksi objek yang ditentukan. Berikut contoh implementasinya.

```
./darknet detector train obj_resume.data yolo-obj.cfg
backup/darknet53.conv.74 -dont_show
```

```
Learning Rate: 0.001, Momentum: 0.9, Decay: 0.0005

Region 82 Avg IOU: 0.293486, Class: 0.443128, Obj: 0.680025,
No Obj: 0.532993, .5R: 0.142857, .75R: 0.000000,  count: 7

Region 94 Avg IOU: 0.203765, Class: 0.575399, Obj: 0.361495,
No Obj: 0.513690, .5R: 0.000000, .75R: 0.000000,  count: 4

Region 106 Avg IOU: 0.148327, Class: 0.769327, Obj: 0.539390,
No Obj: 0.469288, .5R: 0.000000, .75R: 0.000000,  count: 1
```

2.11 OpenCV

OpenCV adalah library yang dapat digunakan untuk menganalisis data dari kamera dari sebuah alat atau komputer atau apapun yang dapat menangkap gambar. Terdapat lebih dari 2500 algoritma yang teroptimalisasi yang mencakup serangkaian komputasi Computer Vision maupun algoritma pembelajaran mesin [26].

Algoritma ini dapat digunakan untuk mengidentifikasi objek, mengklasifikasikan kegiatan manusia pada video, melacak objek bergerak, mengekstraksi model objek 3D, menemukan gambar yang serupa dari sebuah database dan lainnya. OpenCV diutamakan didesain untuk digunakan dengan bahasa pemrograman Python, tetapi tersedia juga untuk C++ dan Java. Terdapat versi CV2 dan CV3 dimana CV2 untuk Python2 dan CV3 untuk Python3.

OpenCV mencakup beberapa library umum, salah satunya adalah *imgproc* yang mencakup image *filtering*, transformasi gambar geometris (merubah ukuran, *affine*, pemetaan berbasis tabel generik), konversi ruang warna, histogram dan sebagainya.