BAB 2
LANDASAN TEORI

2.1. Landasan Teori

2.1.1 Rambu Lalu Lintas

Di Indonesia, rambu lalu lintas ditetapkan dan dibuat peraturannya oleh Pemerintah. Rambu lalu lintas ada yang berisi larangan, petunjuk, dan perintah. Secara umum, pengertian rambu-rambu lalu lintas adalah salah satu alat perlengkapan di jalan dalam bentuk tertentu, yang memuat lambang, huruf, angka, kalimat dan perpaduan di antaranya, digunakan untuk memberikan peringatan, larangan , perintah dan petunjuk bagi pengguna jalan [2].

2.1.2 Pelanggaran Lalu Lintas

Pengertian dari pelanggaran lalu lintas diterbitkan oleh Direktorat Jenderal Pembinaan Badan Peradilan Umum Departemen Kehakiman edisi pertama tahun 1993 yang berbunyi:

“ Pelanggaran lalu lintas adalah setiap pelanggaran yang dilakukan oleh pemakai jalan baik terhadap rambu-rambu lalu lintas maupun dalam cara mengemudi jalan, orang yang menggunakan kendaraan bermotor maupun pejalan kaki. “

UU No. 22 tahun 2009 membagi dua tindak pidana pelanggaran lalu lintas yaitu:

1. Tindak pidana pelanggaran lalu lintas, yang terdiri dari beberapa jenis pelanggaran, yaitu:
    a. Pelanggaran terhadap alat pemberi isyarat lalu lintas.
    b. Pelanggaran terhadap marka jalan.
    c. Pelanggaran terhadap rambu - rambu lalu lintas.
    d. Pelanggaran terhadap kecepatan maksimum dan minimum.

e. Pelanggaran terhadap peringatan bunyi.
f. Pelanggaran terhadap persyaratan administratif pengemudi dan kendaraan.

2. Tindak pelanggaran angkutan jalan, terdiri dari beberapa jenis pelanggaran yaitu:
   a. Pelanggaran terhadap persyaratan teknis dan layak jalan kendaraan.
   b. Pelanggaran terhadap perizinan.
   c. Pelanggaran terhadap uatan kendaraan (Undang – Undang Nomor 22 Tahun 2009 tentang Lalu Lintas dan Angkutan Jalan BAB VII tentang kendaraan).

Berdasarkan pemaparan di atas, disimpulkan bahwa pelanggaran lalu lintas merupakan perbuatan yang bertentangan dengan apa yang secara tegas terulis dalam UU No. 22 tahun 2009 ssebgai pelanggaran [9].

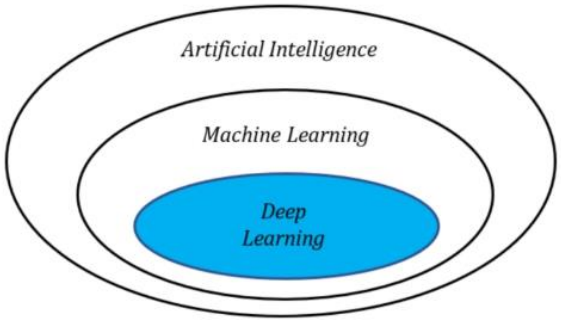
2.1.3 Pelanggaran Rambu Dilarang Parkir

Berdasarkan peraturan daerah provinsi daerah khusus Ibukota Jakarta Nomor 5 tahun 2012 tentang perparkiran, tertera pada pasal 64 ayat 2 yang berbunyi “Apabila setelah jangka waktu 15 (lima belas) menit sejak kendaraan parkir, pengemudi kendaraan tidak memindahkan kendaraannya, pemindahan kendaraan dapat dilakukan oleh petugas yang berwenang di ruang milik jalan atau petugas parkir di luar ruang milik jalan”.

2.1.4 Deep Learning

Deep Learning adalah bagian dari jaringan saraf tiruan (JST). Dipublikasikan oleh Rina Dechter pada tahun 1986 dan menurut Google trends algoritma ini terus berkembang dan mulai luas dikenal pada tahun 2014 [10]. *Deep learning* adalah sebuah metode pembelajaran terhadap data yang bertujuan untuk membuat representasi (abstraksi) data secara bertingkat menggunakan sejumlah lapisan pengolahan data. Hal penting dari *deep*

learning adalah representasi data tersebut tidak dibuat secara eksplisit oleh manusia tetapi dihasilkan oleh sebuah algoritma pembelajaran [11].



Gambar 2.1 Deep learning sebagai cabang ilmu Artificial Intelligence dan Machine Learning

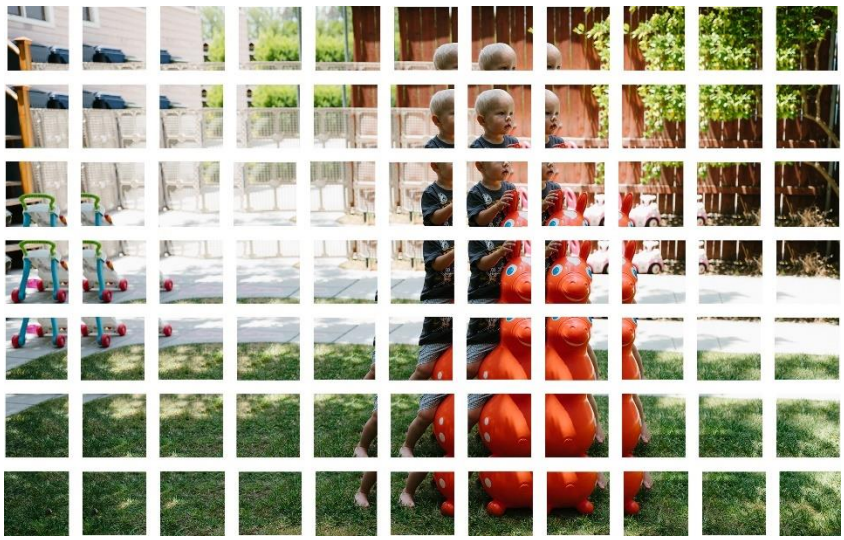
2.1.5 *Convolutional Neural Network* (CNN)

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 ditargetkan, tujuan dari metode ini adalah pengelompokan suatu data menuju data yang sudah ada [12].

Jaringan konvolusi adalah jaringan khusus yang dimiliki CNN, citra masukan diolah pada lapisan ini berdasarkan filter yang sudah ditentukan. Hasil dari setiap lapisan ini berupa sebuah pola dari beberapa bagian citra yang nanti akan diklasifikasikan. Di bawah ini adalah cara kerja CNN [13]:

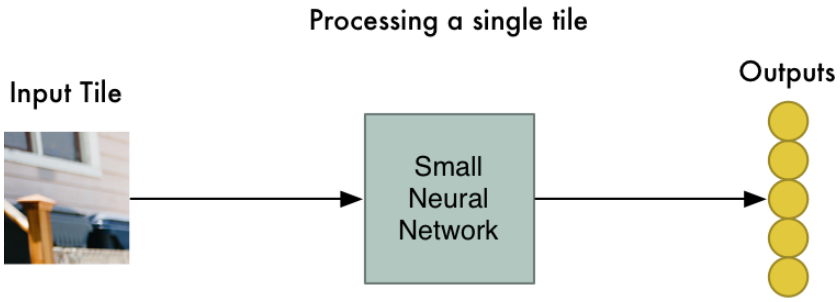
1. Citra masukan

2. Membagi citra menjadi citra yang lebih kecil



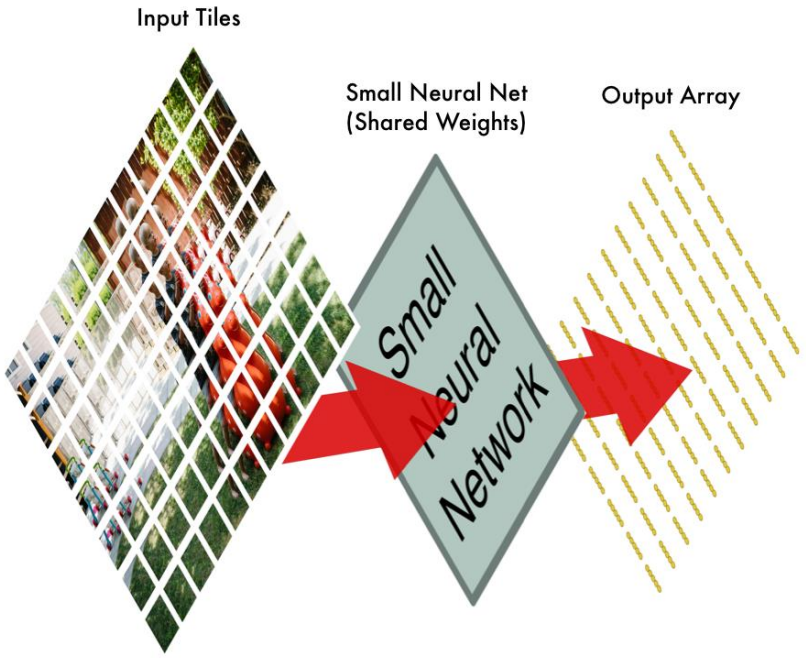
Dari citra masukan yang ada, hasil dari konvolusinya adalah berupa 77 citra yang lebih kecil dengan konvolusi yang sama.

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



Proses ini diulang sebanyak 77 kali pada setiap citra kecil yang sudah dibagi di proses sebelumnya. Setiap citra kecil akan memiliki bobot *neural network* yang sama dengan citra aslinya atau disebut sebagai *weights sharing*. Jika ada yang tampak menarik dalam setiap citranya, maka bagian itu akan ditandai sebagai *object of interest*.

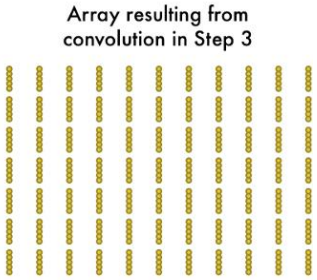
4. Menyimpan hasil dari setiap citra kecil ke array baru



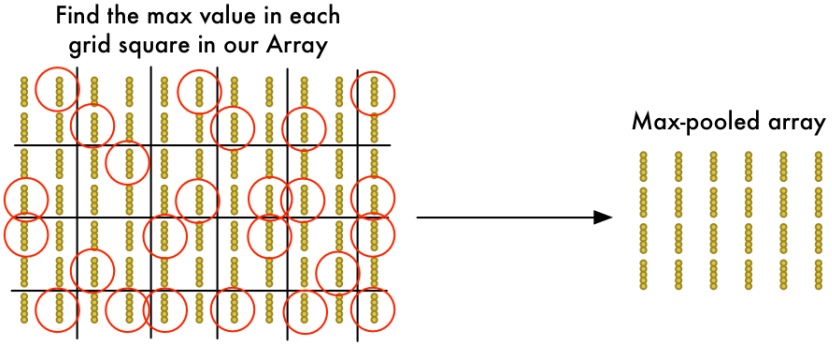
Menyimpan hasil dari pemrosesan setiap bagian citra ke dalam *grid* dengan pengaturan yang sama dengan citra aslinya.

5. Downsampling

hasil dari langkah keempat adalah array yang sudah memetakan bagian mana saja dari citra aslinya yang paling menraik. Tetapi array tersebut masih cukup besar.

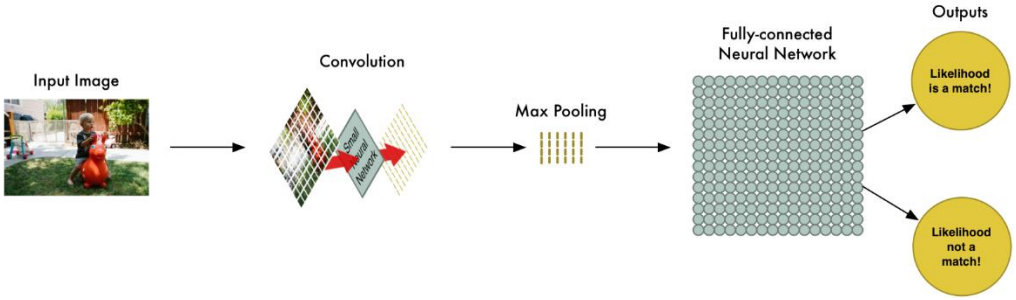


Untuk mengurangi ukuran array, dilakukan *downsample* menggunakan algoritma *max polling.* Array yang ada akan dilihat dengan ukuran kernel 2x2 dan akan tetap menyimpan ukuran piksel paling besar di setaip kernel.



6. Membuat prediksi

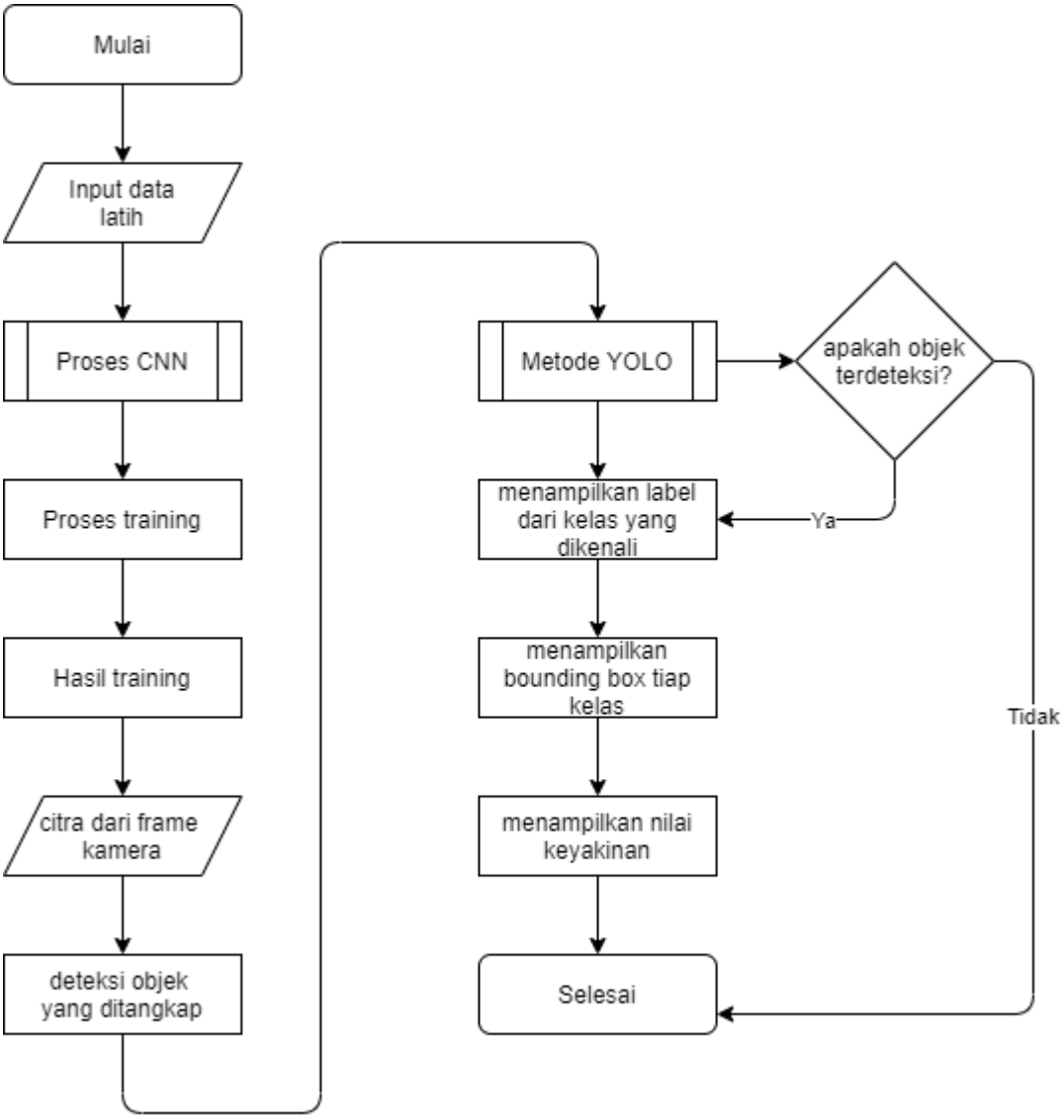
Setelah merubah citra yang berukuran besar menjadi array kecil, berikutnya adalah menggunakan array kecil sebagai masukkan untuk *neural network.* Masukkan ini adalah proses terakhir untuk menentukan citra tersebut cocok atau tidak. Langkah ini diberi nama *fully connected network* untuk membedakan dengan langkah konvolusi.



2.1.6 *You Only Look Once* (YOLO)

Model pertama dari YOLO dijelaskan pertama kali oleh Joseph Redmon, pada tahun 2015 jurnal yang berjudul *You Only Look Once: Unified, Real-Time Object Detection.* Tercatat bahwa Ross Girshick, pengembang dari R-CNN, merupakan penulis dan contributor dalam jurnal tersebut. Pendekatannya melibatkan jaringan saraf tunggal yang dilatih *end-to-end* menggunakan gambar sebagai input dan memprediksi *bounding box* dan label kelas untuk setiap *bounding box*. Teknik ini menawarkan akurasi prediksi yang lebih rendah, meskipun beroperasi pada 45 *fps* hingga 155 *fps* dengan optimasi kecepatan dari versi model ini.

Model ini bekerja dengan langkah awal membagi gambar input menjadi beberapa sel grid, dimana setiap sel bertanggung jawab untuk memprediksi *bounding box* jika *bounding box* yang ditengah termasuk didalamnya. Setiap sel grid memprediksi *bounding box* melibatkan koordinat x,y lebar, tinggi dan nilai *confidence*. Sebuah prediksi kelas juga berdasarkan dari setiap sel. Sebagai contoh, sebuag gambar dibagi menjadi 7x7 grid dan setiap sel dalam grid dapat memprediksi dua *bounding box*, menghasilkan 98 prediksi *bounding box* yang diusulkan. Kelas probabilitas dan *bounding box* dengan *confidence* lalu digabungkan menjadi sebuah *bounding box* akhir dan label kelas. Berikut adalah flowchart proses YOLO [14].



Gambar 2.2 Flowchart proses YOLO

1. Input citra latih

Tahap pertama adalah memasukkan citra yang akan dilatih

2. Proses CNN

Pada bagian ini citra yang sudah dimasukkan akan dilakukan proses CNN yang meliputi pemecahan citra, *weights sharing*, pembentukan hasil array, *downsampling*.

3. Proses training

Disini adalah tahap pelatihan citra untuk dapat mendeteksi objek yang ditargetkan.

4. Hasil Training

Keluaran dari proses training adalah *weight* yang dapat digunakan dalam sistem untuk mendeteksi objek.

5. Citra dari frame kamera

Citra yang terdapat di dalam kamera akan digunakan sebagai data uji.

6. Deteksi objek yang ditangkap

Disini adalah tahap dimana sistem berusaha mengenali objek yang dideteksi berdasarkan hasil training yang didapatkan.

7. Metode YOLO

Data masukan yang ada akan di proses dalam metode YOLO ini yang meliputi *resize*, pembagian grid, penentuan letak *bounding box*, dan menghitung nilai keyakinan.

8. Menampilkan label dari kelas yang dikenali

Jika objek dapat dikenali dan sudah diketahui kelasnya maka berikutnya adalah menampilkan label berdasarkan kelas yang dikenali.

9. Menampilkan *bounding box* setiap kelas

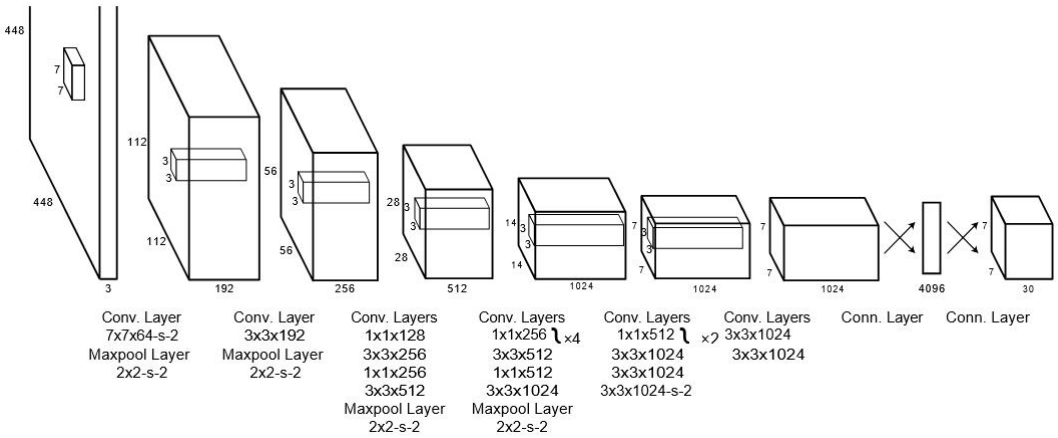
Setelah mendapatkan label dari kelas yang terdeteksi proses berikutnya adalah menampilkan *bounding box* di objek yang terdeteksi.

10. Menampilkan nilai keyakinan

Tahap terakhir adalah menampilkan nilai keyakinan dari objek yang terdeteksi berdasarkan proses YOLO yang sudah dilakukan.

2.1.6.1 Arsitektur YOLO

Jaringan arsitektur YOLO terinspirasi dari GoogLeNet sebagai klasifikasi citra. Memiliki 24 lapisan konvolusi yang diikuti dengan dua lapisan yang berhubungan penuh. YOLO menggunakan lapisan pengungaran berukuran 1 x 1, kemudian diikuti dengan lapisan konvolusi berukuran 3 x 3 [15]. Berikut adalah gambar dari arsitektur YOLO:



Gambar 2.3 Arsitektur YOLO

2.1.6.2 YOLOv3

YOLOv3 adalah versi yang diperbarui dari versi sebelumnya yaitu YOLOv1 dan YOLOv3. Pembaruan pertama adalah *multi-label classification*, menggunakan *logistic classifier* untuk mengkalkulasi kemungkinan dari suatu objek dari label yang sudah dibuat.

Pembaruan kedua adalah penggunaan dari prediksi kotak pembatas (*bounding box*). Berhubungan dengan skor objektivitas kotak pembatas jangkar yang tumpang tindih dengan objek aslinya. Mengabahaikan jangkar lainnya yang tumpang tindih dengan objek aslinya dari *treshold* yang sudah ditentukan.

Pembaruan ketiga adalah membuat penggunaan prediksi menggunakan konsep fitur jaringan piramida. YOLOv3 memprediksi 3 kotak dengan skala yang berbeda, lalu melakukan ekstraksi fitur dari setiap skala tersebut. Hasil prediksi dari jaringan merupakan 3D tensor yang mengkodekan kotak pembatas, nilai objektivitas dan prediksi kelas [16].

2.1.7 YOLO sebagai Deteksi Objek

Deteksi objek merupakan tugas dari computer vision yang melibatkan satu atau lebih objek dalam suatu gambar dan mengklasifikasikan setiap objek yang ada dalam gambar. Ini adalah sebuah tugas yang menantaang bagi computer vision karena membutuhkan keberhasilan lokalisasi objek untuk menemukan dan membuat bounding box di sekitar objek dalam sebuah gambar, dan klasifikasi objek untuk memprediksi kelas yang benar dari object yang sudah terlokalisasi. You Only Look Once atau yang disebut sebagai

YOLO merupakan keluarga dari model end-to-end deep learning model yang dibuat untuk mendeteksi objek secara cepat, dikembangkan oleh Joseph Redmon dijelaskan pada tahun 2015 dengan jurnal berjudul You Only Look Once: Unified, Real-Time Object Detection.

Pendekatannya meliputi single deep convolutional neural network (versi asli dari GoogleNet, diperbarui dan disebut DarkNet berdasarkan VGG) yang membagi input menjadi sel grid dan setiap sel secara langsung memprediksi sebuah bounding box dan klasifikasi objek. Hasilnya adalah jumlah kandidat bounding box dalam jumlah besar yang tergabung menjadi prediksi akhir melalui langkah post-processing. Ada tiga variasi utama dalam pendekatan, pada saat penulisan jurnalnya terdapat YOLOv1, YOLOv2, dan YOLOv3. Versi pertama ditujukan sebagai arsitektur umum, dimana versi keduanya dihilangkan desainnya dan digunakan untuk anchor box yang sudah ditentukan sebelumnya untuk meningkatkan bounding box, dan versi ketiga menghilangkan lebih banyak model aksitektur dan proses latihan. Walaupun akurasi dari setiap model cukup dekat tapi tidak sebaik Region-Based Convolutional Neural Network (R-CNN), algoritma ini popular untuk deteksi objek karena kecepatan deteksinya, banyak diperagakan dalam real-time video atau input kamera [14].

2.1.8 Pengujian *Confusion Matrix*

Confusion matrix merupakan suatu metode yang digunakan untuk menghitung akurasi pada konsep *data minin confusion matrix* sebagaimana digambarkan pada tabel dibawah yang menyatakan jumlah data uji yang benar diklasifikasikan dan jumlah data yang salah diklasifikasikan [17].

Table 2.1 pengujian *confusion matrix*

		Prediksi	
		Positif / 1	Negatif / 0
Actual	Positif / 1	TP	FN
	Negatif / 0	FP	TN
	TOTAL	TP+FP	FN+TN

Dari tabel diatas dijelaskan sebagai berikut:

1. *True Positive* (TP) merupakan jumlah *record* dari kelas positif yang benar diklasifikasikan sebagai sebagai positif.
2. *False Negative* (FN) merupakan jumlah *record* dari kelas positif yang salah diklasifikasikan sebagai kelas negatif.
3. *False Positive* (FP) merupakan jumlah *record* dari kelas negatif yang salah diklasifikasikan sebagai kelas positif.
4. *True Negative* (TN) merupakan jumlah *record* dari kelas negative yang benar diklasifikasikan sebagai kelas negative.

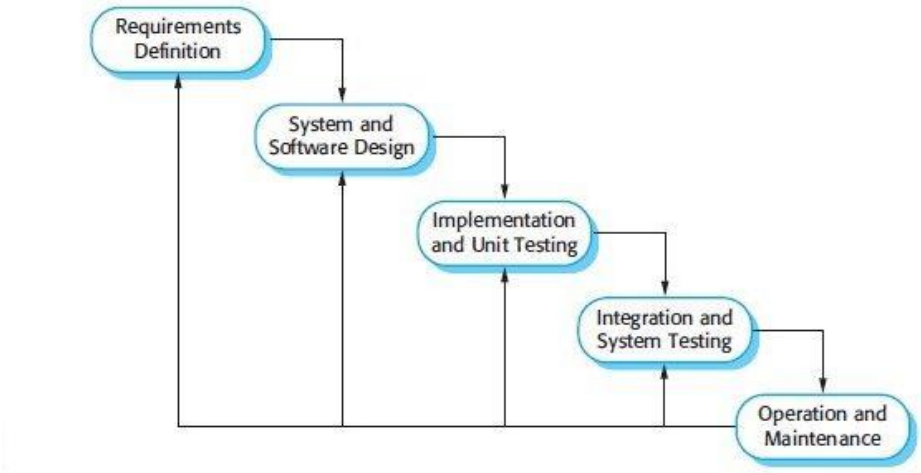
Berikut adalah rumus perhitungan akurasi:

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} x\ 100\%$$

2.1.9 Metode *Waterfall*

Metode Waterfall ini sebenarnya adalah *Linear Sequential Model,* yang sering juga disebut dengan *classic life cycle* atau model Waterfall. Metode ini muncul pertama kali sekitar tahun 1970 sehingga sering dianggap kuno, tetapi merupakan model/ metode yang paling banyak dipakai dalam *Software Engineering.* Metode ini melakukan pendekatan secara sistematis dan urut mulai dari level kebutuhan sistem lalu menuju ke tahap analisis, desain, coding, testing/verification, dan maintenance. Disebut dengan waterfall karena tahap sebelumnya dan berjalan berurutan. Sebagai contoh tahap desai harus menunggu selesainya tahap sebelumnya yaitu tahap *requirement* [18].

Menurut Pressman dan Sommerville tahun 2010, metode ini terdiri dari beberapa step, seperti ditunjukkan pada gambar berikut.



Gambar 2.4 Metode *Waterfall*

2.1.10 Python

Bahasa pemrograman Python adalah bahasa pemrograman yang dibuat oleh Guido van Rossum dari Amsterdam, Belanda. Pada awalnya, pembuatan bahasa pemrograman ini adalah untuk bahasa skrip tingkat tinggi pada sistem operasi terdistribusi Amoeba. Bahasa pemrograman ini menjadi umum digunakan untuk kalangan *engineer* seluruh dunia dalam pembuatan perangkat lunaknya, bahkan perusahaan besar seperti Google, NASA, Instagram, Youtube, dan Spotify menggunakan Python sebagai pembuat perangkat lunak komersial. Python banyak digunakan untuk membuat berbagai macam program, seperti program CLI, GUI (desktop), aplikasi Mobile, Web, IoT, game, program hacking, dan sebagainya [19].

Berikut adalah kelebihan dari bahasa pemrograman Python:

1. Python merupakan bahasa pemrograman ke-3 paling popular di dunia.
2. Python relative lebih mudah dipelajari dan digunakan dibandingkan bahasa pemrograman lain. Sintaks sederhana, mudah dibaca, dan diingat karena filosofi Python sendiri menekankan pada aspek kemudahan dibaca

(*readability*). Kode python mudah ditulis dan mudah dibaca, sehingga lebih mudah diperbaiki jika ada kesalahan.

3. Python merupakan bahasa multifungsi. Dengan Python, dapat dengan mudah mengembangkan sebuah produk, baik itu web, *software*, aplikasi web, maupun game, robotika, data mining, sampai dengan kecerdasan bauta. Bahasa ini juga dapat membuat aplikasi berbasis desktop maupun berbasis mobile.
4. Penulisan kode lebih efisien dibandingkan bahas lain seperti C, C++, ataupun Java. Untuk melakukan dengan lima baris kode dengan bahasa lain, dengan Python bisa jadi hanya diperlukan satu baris kode saja. Hal ini menyebabkan pembuatan program dalam Pyhton menjadi lebih ringkas dan lebih cepat dibandingkan bahasa lain.
5. Python kaya akan dukungan *library* standar. Tersedia banyak modul dan ekstensi program yang sudah siap digunakan untuk membuat program sesuai kebutuhan.
6. Pyhton dapat berinteraksi dengan bahasa pemrograman lain. Kode pyhton bisa memanggil bahasa C, C++, dan juga sebaliknya, bisa juga dipanggil dari bahasa lain.
7. Python yang mendukung ekosistem IoT (*Internet of Things*) dengan sangat baik. Banyak sistem yang mengusung IoT menggunakan Python. Terdapat berbagai macam *board* yang digunakan menjalankan sistem IoT menggunakan bahasa ini sebaagai basisnya, termasuk di dalamnya adalah Raspberry Pi.

Adapun kekurangan dari bahasa pemrograman Python sebagai berikut:

1. Beberapa penugasan terdapat di luar jangkauan kemapuan Python. Seperti bahasa pemrograman dinamis lainnya, Python tidak secepat atau efisien sebagai statis, tidak seperti bahasa pemrograman C.
2. Python cukup buruk untuk pengembangan platform mobile (Andorid/IOS).

3. Python merupakan interpreter sehingga bukan perangkat bantu terbaik untuk pengantar kinerja kritis. Jadi, Python bukan pilihan yang baik untuk tugas – tugas intensif memori.
4. Hamper mustahil untuk membuat game 3D grafis tinggi menggunakan Python.
5. Memiliki keterbatasan dengan akses basis data.
6. Python tidak dapat digunakan sebagai dasar bahasa pemrograman implementasi untuk beberapa komponen, tetapi dapat bekerja dengan baik sebagai bagian depan script interface.
7. Python memberikan tingkat efisiensi dan *flexibility trade off by* dengan tidak memberikannya secara menyeluruh.