BAB 2
DASAR TEORI

2.1 KAJIAN PUSTAKA

Studi yang dilaksanakan [6] meneliti mengenai klasifikasi jenis kendaraan roda empat dengan membagi 4 kelas yaitu sedan, *MVP*, Truk, dan Bis dengan menggunakan metode YOLO. penelitian ini melakukan Pengujian informasi penghitungan dapat mendeteksi sebanyak 93% dan memiliki tingkat kesalahan sebesar 7%.

Penelitian yang dilakukan [7] meneliti mengenai klasifikasi jenis kendaraan dengan membagi 3 bagian yaitu jenis kendaraan (motor,mobil,truk,bus), warna kendaraan, kecepatan serta arah kendaraan jika kondisi jalan memungkinkan untuk dua arah kendaraan dengan menggunakan metode *TensorFlow Python* dengan tingkat keakurasiannya mencapai 90%.

Penelitian yang dilakukan [8] meneliti mengenai klasifikasi jenis kendaraan dengan membagi Tiga jenis kendaraan (mobil, motor, sepeda) yang dianalisis dengan menggunakan pendekatan CNN dengan menggunakan 120 gambar yang meliputi gambar mobil, motor, dan sepeda didapat pada tahap pelatihan, tingkat keakurasiannya mencapai 94,4%, sedangkan pada tahap pengujian mencapai 73,3%.

Penelitian [9] menginvestigasi klasifikasi jenis kendaraan yang terdiri dari lima kelas, Kendaraan yang termasuk dalam kategori ini meliputi mobil, truk dengan empat roda, truk dengan enam roda, truk dengan delapan roda, dan truk dengan sepuluh roda, menggunakan metode CNN. *Dataset* terdiri dari total 2104 citra, dengan 631 citra digunakan untuk pengujian dan 1473 citra untuk pelatihan. *Model AlexNet* mencapai akurasi tertinggi sebesar 93,81%, sedangkan model *MobileNet* mencapai akurasi sebesar 96,19%. Eksploitasi *existing* CNN dengan mengganti *output fully connected* terakhir sesuai jumlah kelas kendaraan. Hasil pengujian menunjukkan bahwa *mobilenet* V2 lebih baik dalam klasifikasi jenis kendaraan. Visi computer dengan memanfaatkan CNN dapat menggantikan penggunaan sensor sehingga biaya implementasi lebih murah.

2.2 DASAR TEORI

2.2.1 Transjakarta (Transportasi Jakarta)

Transjakarta merupakan sistem transportasi umum yang melayani wilayah Provinsi Daerah Khusus Ibu Kota (DKI) Jakarta dan sekitarnya, beroperasi sejak tahun 2004. Sistem transportasi ini menggunakan konsep Bus Rapid Transit (BRT) dan jenis transportasi non-Bus Rapid Transit (NBRT) sebagai basisnya. TransJakarta mengusung konsep transportasi publik yang terpadu dan bebas macet untuk mengurai kemacetan dan menarik perhatian publik warga Jakarta untuk menggunakan transportasi umum [10].

Gambar 2.1 Petugas Penjaga Jalur Transjakarta [11]

Pada gambar 2.1 merupakan salah satu contoh lokasi jalur TransJakarta yang dijaga secara manual oleh petugas keamanan jalur. Jalur TransJakarta dilengkapi dengan gerbang di setiap pintu masuk jalur yang operasionalnya dibuka ketika ada *bus* TransJakarta yang akan lewat dan ditutup kembali ketika tidak ada *bus* TransJakarta yang lewat. TransJakarta dalam operasionalnya terdiri dari 14 koridor dengan total 260 halte dan jalur sepanjang 251,2 km yang beroperasi di jalur khusus (*Busway*) untuk layanan BRT dan juga menggunakan jalan biasa yang tersebar di 14 koridor dan saat ini beroperasi 24 jam. Transjakarta mengoperasikan sejumlah 1347 *unit bis*, terbagi menjadi dua jenis yakni bus tunggal dan bus gandeng, menggunakan beberapa merek bus meliputi Zhongtong untuk bus gandeng, serta Scania, Hino, dan Mercedes-Benz untuk bus tunggal [10]. TransJakarta memiliki jalur lintasan dengan beberapa rute yang dilengkapi dengan pemberhentian atau halte. Halte-halte ini strategis ditempatkan dekat dengan perkantoran, pusat perbelanjaan, tempat wisata, dan lokasi penting lainnya di DKI Jakarta. Biasanya, halte-halte ini terletak di tengah jalan dan dilengkapi

dengan jembatan penyeberangan landai sebagai akses masuk. Dalam mengelola armadanya, TransJakarta bekerja sama dengan beberapa perusahaan operator yang bertanggung jawab untuk melayani setiap koridornya:

- Unit Swakelola PT. Transportasi Jakarta (TJ)
- PT Bianglala Metropolitan (BMP)
- Perum DAMRI (DMR)
- Mayasari Bakti (MYS)
- *Steady Safe* (SAF) , dll [12]

Gambar 2.2 Operator *Steady Safe* (SAF) [13]

Pada gambar 2.2 diata merupakan salah satu bentuk *bus* TransJakarta milik operator *steady safe* (SAF) yang dimana beberapa operator lainnya juga memiliki bentuk *body bus* yang mirip dengan bentuk bis pada gambar diatas, namun ada juga operator yang memiliki bentuk *bus* yang berbeda.

Gambar 2.3 Operator Damri (DMR) [14]

Pada gambar 2.3 merupakan contoh bentuk *bus* TransJakarta milik Damri yang berbeda dengan operator lain pada umumnya. Selain bus tunggal, transjakarta juga memiliki bus model gandeng untuk mengakomodasi penumpang

yang sangat ramai di beberapa koridor tertentu seperti koridor 1, 2, 3, 9, 10, 13 yang dilayani oleh operator DAMRI, Mayasari Bakti, dan Swakelola TransJakarta.

2.2.2 *Google Colaboratory*

Google Colaboratory adalah *executable document* yang biasa digunakan untuk menulis, menyimpan, dan membagikan program yang sebelumnya sudah dituliskan melalui *Google Drive*. *Tools* ini memiliki banyak manfaat diantaranya *built in machine learning*, berbasis *cloud, fleksibel*, tersedia fitur TPU dan GPU gratis [15].

Gambar 2.4 *Google Colab* [15]

Pada gambar 2.4 merupakan logo dari dari *Google Colaboratory* yang terdiri dari 2 lingkaran berwarna kuning saling bersinggungan. *Google Colab* memungkinkan siapa saja menulis dan menjalankan kode *Python* secara langsung melalui *browser*, sangat ideal untuk *machine learning*, analisis data, dan pendidikan. Secara teknis, *Colab* adalah layanan *notebook Jupyter* yang di*hosting* dan dapat digunakan tanpa persiapan, serta menyediakan akses gratis ke sumber daya komputasi termasuk GPU. Sumber daya *Colab* bersifat tidak dijamin dan terbatas, serta batas penggunaannya kadang-kadang berfluktuasi. Hal ini diperlukan agar *Colab* dapat menyediakan sumber daya secara gratis. Pengguna yang menginginkan akses lebih handal ke sumber daya yang lebih baik dapat menggunakan *Colab Pro*. Memperkenalkan *Colab Pro* merupakan langkah pertama yang diambil *Google* untuk melayani pengguna yang ingin melakukan lebih banyak hal di *Colab*. Tujuan jangka panjang *Google* adalah terus menyediakan versi gratis *Colab* sembari berkembang secara berkelanjutan untuk memenuhi kebutuhan pengguna [15].

2.2.3 Citra Digital

Citra merujuk pada gambar diam (foto) atau gambar bergerak (dari *webcam*), serta gambar digital mengacu pada pengolahan gambar menggunakan komputer. Dari segi matematis, citra dapat dijelaskan sebagai fungsi kontinu yang memperlihatkan distribusi intensitas cahaya pada bidang 2D. Untuk proses pengolahan, Citra perlu direpresentasikan secara numerik menggunakan nilai-nilai diskret. Sebuah citra digital dapat dijelaskan sebagai matriks 2D f (x, y) dengan M kolom dan N baris, di mana setiap titik pada persimpangan kolom dan barisnya dikenal sebagai piksel.

$$f\ (x,y)\begin{vmatrix} f(0,0) & f(0,1) & \dots\ f(0,M-1) \\ f(1,0) & f(1,1) & \dots\ f(1,M-1) \\ f(N-1,0) & f(N-1,1) & \dots\ f(N-1,M-1) \end{vmatrix}$$

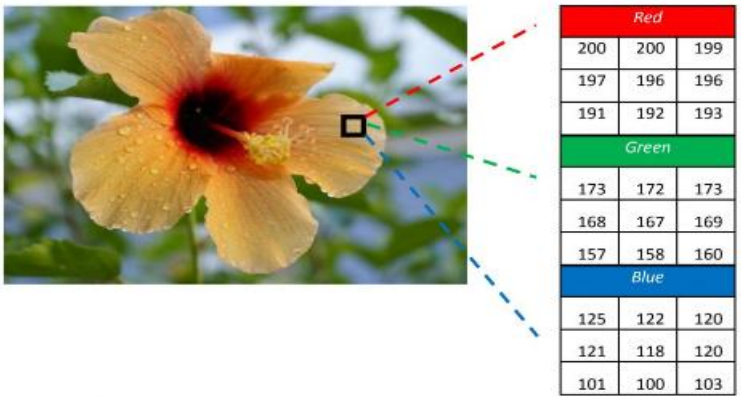
Sebuah citra f (x, y) dalam domain matematis dapat direpresentasikan sebagai berikut :

$0 \leq x \leq M\text{-}1$

$0 \leq y \leq N\text{-}1$

$0 \leq f\ (x, y) \leq G\text{-}1$

M adalah jumlah piksel dalam arah baris pada matriks citra, N adalah jumlah piksel dalam arah kolom pada matriks citra, dan G adalah nilai skala keabuan [16].



Gambar 2.5 Citra Digital Terdiri Dari Komponen *Red, Green,* dan *Blue*

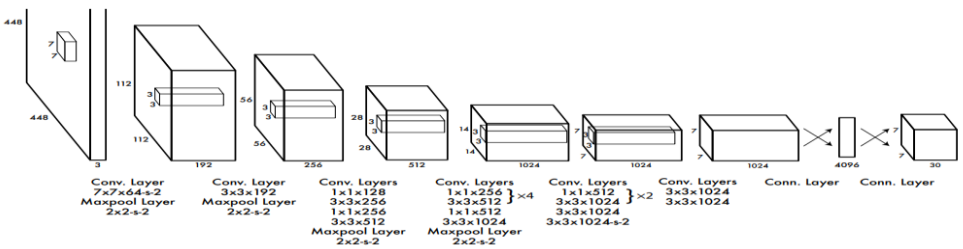
Pada gambar 2.5 menjelaskan bahwa pada setiap gambar terdapat unsur 3 warna yaitu RGB (*Red, Green, Blue*). Sebagai data, citra digital memiliki keunggulan dimana informasi dapat diambil secara *non-contact*. Hal ini membuat analisis terhadap suatu objek atau situasi dapat dilakukan tanpa kamera perlu

menyentuhnya. Analisis terhadap kondisi atau jenis objek juga dapat dilakukan tanpa merusaknya.

2.2.4 *You Only Look Once* (YOLO)

Algoritma YOLO, yang dikembangkan oleh Joseph Redmon dengan menggunakan *framework* kustom *darknet*, adalah salah satu pendekatan untuk deteksi objek. YOLO menggunakan jaringan saraf konvolusi tunggal yang menonjolkan kecepatan deteksi yang tinggi. Metode ini memanfaatkan jaringan saraf konvolusi khusus untuk mengklasifikasikan beberapa objek dalam satu gambar secara simultan [17].

YOLO menggabungkan proses ekstraksi kotak kandidat, ekstraksi fitur, dan klasifikasi objek ke dalam satu jaringan saraf. YOLO secara langsung mengekstraksi kotak kandidat dari gambar dan mendeteksi objek menggunakan fitur keseluruhan gambar. Dalam jaringan YOLO, gambar dibagi menjadi *grid* berukuran s x s, dengan *bounding box* yang terdistribusi secara merata di sepanjang sumbu X dan sumbu Y. Dalam YOLO, apabila pusat suatu objek berada dalam sebuah sel *grid*, maka sel *grid* tersebut bertanggung jawab untuk mendeteksi objek tersebut. Setiap sel *grid* memprediksi kotak pembatas B dan nilai keyakinan (*box confidence scores*) untuk kotak tersebut, serta probabilitas kelas kondisional C. Prediksi kotak pembatas B terdiri dari lima komponen: x, y, w, h, dan nilai keyakinan kotak. Koordinat (x, y) menunjukkan posisi relatif pusat kotak dalam batas-batas sel *grid*, yang dinormalisasi menjadi rentang 0 hingga 1. Dimensi kotak (w, h) dinyatakan secara relatif terhadap ukuran gambar dan dinormalisasi juga. Jaringan ini membagi gambar menjadi beberapa wilayah (*region*) dan melakukan prediksi berdasarkan lokasi pada setiap kotak pembatas (*bounding box*), nilai keyakinan (*confidence value*), dan probabilitas kelas untuk setiap wilayah tersebut [17].



Gambar 2.6 Arsitektur *You Only Look Once* [21]

Pada gambar 2.6 menjelaskan bahwa proses deteksi objek dimulai dengan memilih area di mana objek akan dideteksi. Arsitektur algoritma YOLO menggunakan CNN yang terdiri dari 24 lapisan konvolusi, diikuti oleh 2 lapisan *fully connected*. Lapisan konvolusi bertugas untuk mengekstraksi fitur dari gambar input, sementara lapisan *fully connected* digunakan untuk memprediksi probabilitas keluaran dan koordinat objek. YOLO menetapkan nilai keyakinan *(confidence)* untuk setiap kotak B sebagai probabilitas bahwa kotak tersebut berisi objek dihitung dengan mengalikan nilai IoU *(intersection over union)* antara kotak prediksi dan ground truth yang diperoleh selama proses pelatihan. *IoU* adalah metrik evaluasi yang digunakan untuk mengukur akurasi deteksi objek dalam suatu *dataset*. Perhitungan nilai *confidence* untuk sebuah kotak ditunjukkan pada persamaan berikut.[19]

$$\textit{Confidence score} = \Pr(\textit{Class}) * \text{IoU}^{\text{truth}}_{\text{Pred}}$$

$$\text{IoU}^{\text{truth}}_{\text{Pred}} = \frac{Area\ Of\ Overlap}{Area\ Of\ Union}$$

Pr (object) : probabilitas kotak berisi objek, jika ada bernilai 1, sebaliknya bernilai 0.

IoU^{truth}_{pred} : *IoU* antara kotak prediksi dan kebenaran dasar.

Untuk mendapatkan prediksi akhir, faktor penentu adalah skor kepercayaan kelas yang dihitung berdasarkan probabilitas kondisional kelas dan nilai keyakinan kotak. Skor kepercayaan kelas mengukur tingkat keyakinan dalam klasifikasi dan lokalisasi objek. Skor kepercayaan kelas memberikan nilai keyakinan spesifik untuk setiap kotak, yang mencerminkan probabilitas kemunculan kelas dalam kotak tersebut serta kesesuaian kotak dengan objek yang diprediksi. Persamaan untuk skor kepercayaan kelas pada setiap kotak prediksi ditunjukkan pada persamaan berikut.

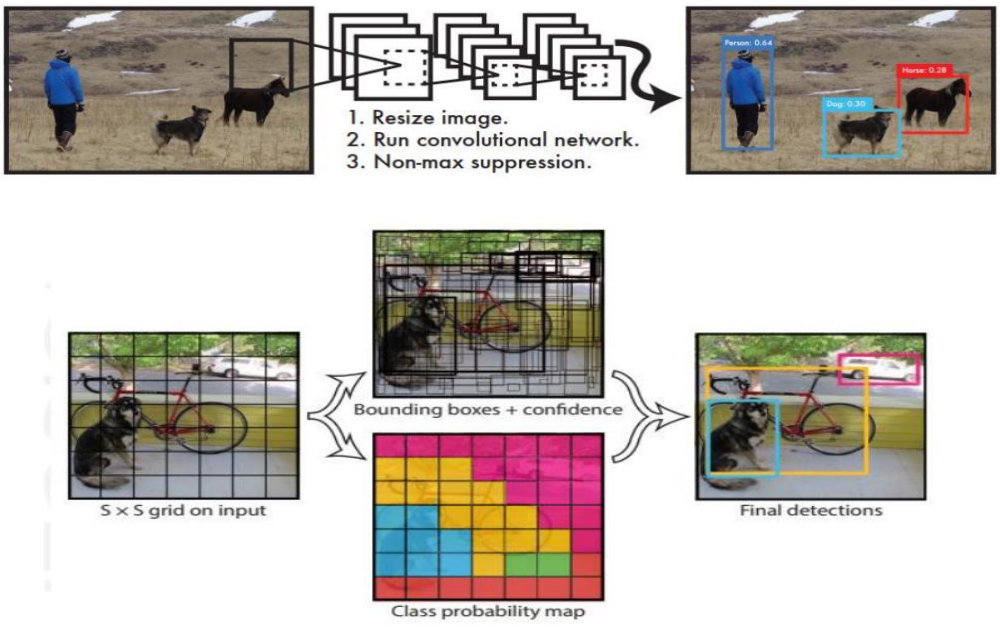
$$\textit{Pr (Class}_i|\textit{object) . box confidence score} = \textit{Pr (Class}_i) \text{ . IoU}^{\text{truth}}_{\text{Pred}}$$

Pr (Class, $_i$ *object)* : probabilitas kondisional kelas *i*.

Pr (Class$_i$ *)* : probabilitas kelas *i*.

Apabila tidak ada objek yang terdeteksi dalam kotak pembatas, maka nilai skor kepercayaan (*confidence score*) akan menjadi 0. Skor kepercayaan kelas

mengukur tingkat keyakinan pada setiap kotak pembatas yang mencerminkan kemungkinan kelas yang mungkin muncul dalam kotak tersebut dan nilai prediksiannya.[20]

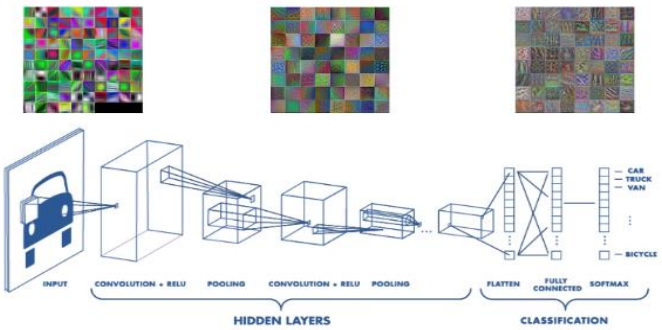


Gambar 2.7 Deteksi objek pada sistem menggunakan YOLO [20]

Pada gambar 2.7 merupakan ilustrasi nyata dalam kehidupan sehari – hari dimana YOLO menggunakan *bounding box* sebagai tanda dalam menampilkan hasil deteksi objek.

2.2.5 *Convolutional Neural Network (CNN)*

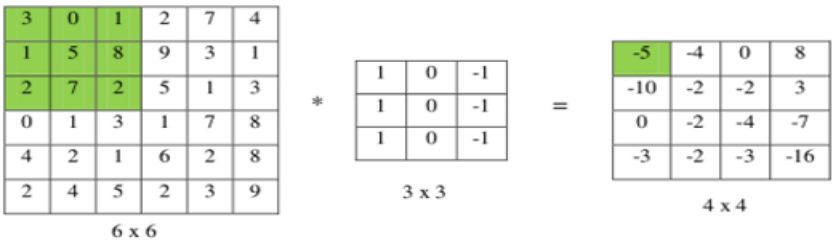
CNN merupakan salah satu pendekatan dalam *deep learning* yang menggunakan jaringan saraf tiruan untuk menyelesaikan masalah dengan memproses data yang ada. Metode CNN memiliki keunggulan yang signifikan dalam pengenalan citra digital karena didasarkan pada konsep pengenalan citra dalam korteks visual manusia. [21].



Gambar 2.8 *Convolutional Neural Networks* (CNN) [21]

Berdasarkan gambar 2.8, layer konvolusi melakukan operasi konvolusi pada keluaran dari lapisan sebelumnya, sementara layer terhubung penuh *(fully connected layer)* digunakan dalam jaringan saraf tiruan maju (MLP) untuk melakukan transformasi dimensi data agar dapat diklasifikasikan secara linear. *Convolutional Neural Network* (CNN) memiliki struktur yang terdiri dari *layer input*, *layer output*, dan beberapa *hidden layer*. Dalam umumnya, lapisan tersembunyi ini meliputi lapisan konvolusional, lapisan pooling, lapisan normalisasi, lapisan ReLU, dan lapisan *fully-connected* [21].

Convolutional layers merupakan operasi dua *function* sehingga menghasilkan *function* baru yang merupakan hasil modifikasi dari kedua *function*. Layer konvolusi ini adalah bagian utama dari struktur CNN yang melakukan sebagian besar komputasi yang memerlukan sumber daya yang signifikan. Layer ini berfungsi untuk menerima input. *Convolutional layer* pada CNN ini enggunakan lebih dari satu filter. Secara prinsip, konvolusi merupakan hasil perkalian titik antara filter dengan wilayah reseptif kecil pada citra input yang memiliki ukuran yang sama dengan filter tersebut. [21].



Gambar 2.9 Proses Convolutional Layer

Pada Gambar 2.9, contoh proses perhitungan konvolusi matrix tersebut menggunakan matrix 6x6 pertama dan menggunakan filter 3x3. Proses perhitungan matrix ini, yaitu dengan melakukan perkalian pada setiap elemen yang dipilih kemudian menjumlahkan seluruh hasil yang telah didapatkan. Pada tahap selanjutnya, matrix digeser satu langkah ke kanan, dan begitu seterusnya sampai pada matrix akhir. Untuk menghitung ukuran matrix dari hasil konvolusi dapat dihitung dengan rumus berikut:

$$\begin{vmatrix} \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots \end{vmatrix}_{m \times m} X \begin{vmatrix} \cdots & \cdots \\ \cdots & \cdots \end{vmatrix}_{n \times n} = \begin{vmatrix} \cdots & \cdots \\ \cdots & \cdots \end{vmatrix}_{(m-n+1) \times (m-n+1)}$$

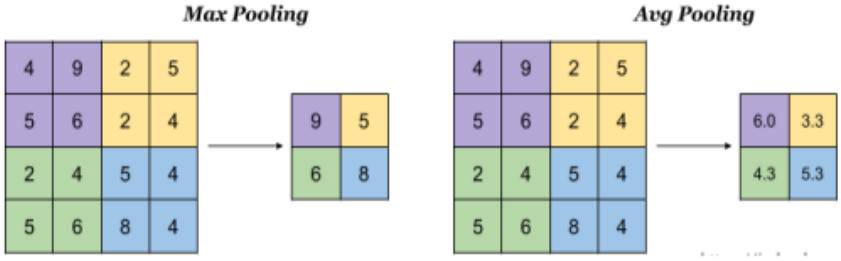
m = ukuran matrix dari input

n = ukuran matrix pada filter

Pada Gambar 2.10 langkah pengerjaanya adalah sebagai berikut, Untuk tahapan pertama adalah mengambil matrix 3x3 posisi pertama *top-left* pada gambar input yaitu $\begin{vmatrix} 3 & 0 & 1 \\ 1 & 5 & 8 \\ 2 & 7 & 2 \end{vmatrix}$. Kemudian dilakukan perhitungan konvolusi dengan cara dilakukan perkalian dengan filter 3x3 seperti berikut (3*1 + 0*0 + 1*(-1)) + (1*1 + 5*0 + 8*(-1)) + (2*1 + 7*0 + 2*(-1)) = (3 + 0 - 1) + (1 + 0 - 8) + (2 + 0 - 2) = 2 + (-7) + 0 = -5. Kemudian tahapan kedua adalah mengambil matrix 3x3 pada gambar input pada posisi *top – middle* yaitu $\begin{vmatrix} 0 & 1 & 2 \\ 5 & 8 & 9 \\ 7 & 2 & 5 \end{vmatrix}$. Kemudian dilakukan perhitungan konvolusi dengan cara dilakukan perkalian dengan filter 3x3 seperti berikut (0*1 + 1*0 + 2*(-1)) + (5*1 + 8*0 + 9*(-1)) + (7*1 + 2*0 + 5*(-1)) = (0 + 0 - 2) + (5 + 0 - 9) + (7 + 0 - 5) =-2 - 4 + 2 = -4. kemudian berlanjut hingga kolom di output 4x4 terisi seluruhnya.

*Activation Lay*er adalah setelah konvolusi, fungsi aktivasi seperti ReLU *(Rectified Linear Unit)* diterapkan untuk memperkenalkan non-linearitas ke dalam model, membantu jaringan untuk belajar dari data yang kompleks [21].

Pooling layer berfungsi untuk menjaga ukuran data dikonvolusi dengan melakukan reduksi sampel. Proses *pooling* sendiri memiliki berbagai cara, yaitu *Average pooling* melakukan pengambilan *mean pooling* menghasilkan nilai rata-rata dari suatu wilayah tertentu, sedangkan *max pooling* mengambil nilai maksimum dari wilayah yang sama, atau mempelajari kombinasi linerar dari *neuron – neuron* yang ada di suatu area [21].



Gambar 2.10 *Pooling Layer* [21]

Gambar 2.10 diatas merupakan simulasi dari *max pooling* dan *average pooling*. *Max pooling* mendapatkan nilai maksimum dari matriks *input* berdasarkan hasil operasi *'dot'* dari *convolution layer*. Sedangkan *average pooling* mengambil nilai rata – rata dari matriks hasil operasi konvolusi [21]. Berikut merupakan cara kerja dari *max pooling*:

Pembagian Input:

- Input berupa fitur peta (feature map) dibagi menjadi beberapa wilayah non-overlapping kecil yang disebut jendela (windows).
- Ukuran jendela umum adalah 2x2, 3x3, dll.

Pemilihan Nilai Maksimum:

- Untuk setiap jendela, nilai maksimum dari elemen dalam jendela tersebut dipilih.
- Nilai maksimum ini menggantikan seluruh elemen dalam jendela tersebut.

Pembentukan Output:

- Hasil dari setiap jendela yang telah dipilih nilai maksimumnya digabungkan untuk membentuk output dari lapisan pooling.

Max pooling adalah teknik penting dalam CNN yang digunakan untuk mengurangi dimensi fitur peta sambil tetap mempertahankan informasi yang penting. Dengan memilih nilai maksimum dari setiap jendela kecil dalam fitur peta, max pooling membantu mengurangi jumlah parameter, mengurangi beban komputasi, dan mengendalikan overfitting.

ReLU Layer meneraplan fungsi aktivasi dengan rumus $f(x) = \max(0, x)$ yang mana fungsi aktivasi ini akan meningkatkan sifat non-linear dari fungsi keputusan dan keseluruhan jaringan [21].

Fully-connected layer adalah layer dimana setiap *neurons* yang ada pada layer sebelumnya saling terkoneksi. Model aktivasinya yaitu model perhitungan dilakukan dengan menggunakan operasi perkalian matriks yang diikuti oleh penambahan nilai bias. Tujuan dari *layer* ini adalah untuk klasifikasi citra [21].

2.2.6 *Roboflow*

Roboflow merupakan suatu *platform* yang tersedia di *website* yang berfungsi untuk membantu *engineer machine learning* untuk mengolah *dataset* untuk pengaplikasian proyek *computer vision* dengan alur kerja. *Dataset* diunggah ke

platform Roboflow untuk kemudian diorganisir sehingga memudahkan proses Anotasi atau penandaan *bounding box* dilakukan pada data tersebut. Data tersebut kemudian dapat dimasukkan sebagai input dalam proses pelatihan model. Setelah model dilatih, model tersebut dapat diimplementasikan dan performanya dievaluasi berdasarkan hasil deteksinya.

Gambar 2.11 *Roboflow* [22]

Pada gambar 2.11 merupakan logo dari *platform API Roboflow* yang memiliki warna dominasi ungu dengan gradasi warna hijau. *Roboflow* menawarkan 20 pustaka model yang dapat dipilih untuk proyek *computer vision*, termasuk YOLO, *Vision Transformer*, *OpenAI Clip*, *ResNet32*, *EfficientDet-D0-D7*, *EfficientNet*, *Detectron2*, *Faster R-CNN*, *MobileNetSSDv2*, dan *PyTorch* [22].

Roboflow memungkinkan pengembang membangun aplikasi visi komputer mereka sendiri, mulai dari persiapan data dan pelatihan model hingga penerapan dan pembelajaran aktif. *Platform* visi komputer menyederhanakan proses pengumpulan gambar, pembuatan set data, pelatihan model, dan penerapannya ke produksi. TensorFlow.js merupakan bagian inti dari beberapa penerapan *roboflow* yang kini telah mendukung lebih dari 10.000 proyek yang dibuat oleh pengembang di seluruh dunia.

Roboflow.js adalah SDK *javascript* yang dapat digunakan pengembang untuk mengintegrasikan model terlatih ke dalam aplikasi web atau aplikasi Node.js. *Roboflow inference server* adalah layanan mikro lintas *platform* yang memungkinkan pengembang menghosting sendiri dan melayani model meskipun tidak semua server inferensi *roboflow* berbasis TFjs, ini adalah salah satu sarana penerapan model yang didukung.

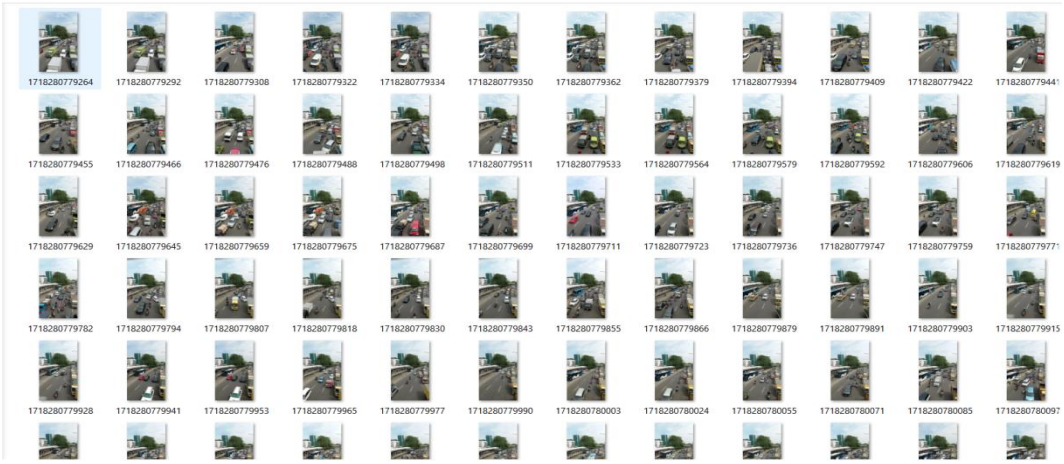
Roboflow memudahkan pengembang untuk menggunakan visi komputer dalam aplikasi. Lebih dari 100.000 pengguna telah membangun *platform end-to-*

end perusahaan untuk pengumpulan gambar dan video, pengorganisasian, anotasi, pra-pemrosesan, pelaatihan model, dan penerapan model. *Roboflow* menyediakan alat bagi Perusahaan untuk meningkatkan kumpulan data mereka dan membangun model visi komputer yang lebih akurat dengan lebih cepat sehingga dapat lebih focus pada masalah domain tanpa perlu mengubah infrastruktur visi.

2.2.7 *Dataset*

Dataset yang umumnya digunakan dalam deteksi objek meliputi gambar dan video, dan berfungsi sebagai input untuk melatih model *deep learning*. *Dataset* ini umumnya terbagi menjadi tiga bagian: *train set, validation set, dan test set.*

- *Train set* merupakan himpunan data sampel yang digunakan untuk melakukan penyesuaian pada model. Ini adalah *dataset* utama yang dimanfaatkan untuk proses pelatihan model, di mana model mengamati serta mempelajari data ini untuk menetapkan bobot dan bias dalam jaringan saraf.
- *Validation set* adalah kumpulan sampel data yang digunakan untuk mengevaluasi kinerja model secara tidak bias terhadap dataset pelatihan, terutama saat menentukan hyperparameter. *Dataset* validasi ini tidak digunakan untuk melatih model, tetapi memegang peran penting dalam pengembangan model selama proses pelatihan.
- *Test set* merupakan himpunan sampel data yang digunakan untuk menguji performa model yang telah dilatih dengan menggunakan *train set.* [23].



Gambar 2.12 Contoh *Dataset*

Pada Gambar 2.12 merupakan contoh *dataset* yang dipergunakan dalam penelitian ini berfungsi sebagai bahan untuk melatih sistem agar dapat mempelajari gambar yang akan diuji, agar *dataset* dapat dipelajari secara optimal oleh sistem deteksi objek, ada beberapa faktor yang dapat mempengaruhi diantaranya resolusi gambar, konsistensi lokasi gambar dan hal – hal lain menyesuaikan dengan situasi dan kondisi dari penelitian yang sedang dijalankan.

Sebuah *dataset* adalah koleksi data yang disusun dalam format yang dapat digunakan untuk analisis, penelitian, atau pelatihan model *machine learning*. Dalam ilmu data dan *machine learning*, *dataset* adalah fondasi utama untuk mengidentifikasi pola, mengembangkan model, dan memperoleh pemahaman dari data. Berikut adalah struktur dasar dari sebuah dataset:

- Observasi (baris): Setiap entri dalam dataset mencerminkan satu pengamatan atau titik data.
- Fitur (kolom): Setiap atribut dalam dataset mewakili sebuah karakteristik atau atribut dari data.

Kemudian untuk klasifikasi dataset terdiri dari sebagai berikut:

- *Dataset tabular* merupakan kumpulan data yang disusun dalam format tabel, mirip dengan *spreadsheet*. Setiap baris mewakili sebuah entitas atau pengamatan, sementara setiap kolom menggambarkan atribut atau karakteristik. Contoh *dataset tabular* meliputi data yang tersedia dalam format *Excel* atau CSV.
- *Dataset gambar* adalah koleksi data yang berisi gambar, di mana setiap entitas bisa berupa gambar yang diungkapkan dalam bentuk matriks piksel.
- *Dataset teks* mencakup kumpulan data yang memuat teks atau dokumen, seringkali digunakan untuk analisis sentimen, klasifikasi teks, atau pemrosesan bahasa alami (NLP).
- *Dataset waktu seri* merupakan kumpulan data yang mencatat informasi dari waktu ke waktu, seperti data saham, cuaca, atau suhu harian.

- *Dataset* grafik adalah kumpulan data yang menggambarkan hubungan antara entitas dalam bentuk grafik atau jaringan.
- *Dataset* audio adalah kumpulan data yang berisi informasi audio, seperti rekaman suara, yang umumnya digunakan untuk pengenalan suara atau analisis audio.
- *Dataset* geospasial adalah kumpulan data yang terkait dengan informasi lokasi geografis, seperti peta atau data GPS.

Kemudian untuk asal dataset dapat diperoleh dari:

- *Dataset* publik adalah *dataset* yang tersedia untuk umum dan sering digunakan untuk keperluan latihan atau penelitian. Contohnya mencakup *dataset Iris* atau *dataset Titanic*.
- *Dataset propietary* adalah dataset yang dimiliki oleh perusahaan atau organisasi tertentu dan umumnya digunakan untuk keperluan internal.
- *Dataset* dari sumber publik diperoleh dari sumber terbuka seperti situs *web* pemerintah atau lembaga penelitian.
- Sumber buatan sendiri merujuk pada kemampuan pengguna untuk membuat dataset mereka sendiri melalui survei, eksperimen, atau metode pengumpulan data lainnya.

Dataset	Classifier	ff.ldc	c.ldc	dc.ldc	Best
Iris	SVM	65.3	6.7	2.7	dc
Iris	NNET	65.3	83.0	1.3	dc
Iris	RPART	65.3	3.3	1.7	dc
Cleveland	SVM	24.2	39.8	14.5	dc
Cleveland	NNET	24.2	16.2	10.5	dc
Cleveland	RPART	24.2	17.7	13.2	dc
Pokcr	SVM	200.4	250.9	98.2	dc
Pokcr	NNET	200.4	311.6	114.2	dc
Poker	RPART	200.4	326.5	168.4	dc
Satlog	SVM	107.0	34.1	9.6	dc
Satlog	NNET	107.0	24.4	15.9	dc
Satlog	RPART	107.0	16.0	11.4	dc
Vowels	SVM	88.6	31.3	26.3	dc
Vowels	NNET	88.6	27.3	24.3	dc
Vowels	RPART	88.6	31.6	39.9	c

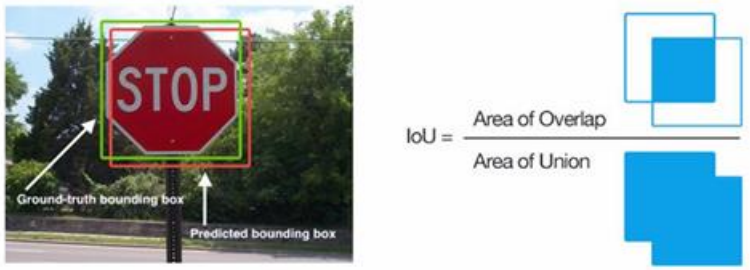
Gambar 2.13 Contoh *Dataset Tabular*

Pada gambar 2.13 merupakan salah satu contoh *dataset* tabular yang sudah dalam bentuk ekstensi csv atau excel. Tipe *dataset* yang dipilih harus disesuaikan dengan tujuan analisis atau pengembangan model pembelajaran mesin yang ingin

dicapai. Setiap tipe *dataset* memiliki karakteristik unik dan menghadirkan tantangan tersendiri dalam proses pra-pemrosesan, pembuatan model, serta evaluasinya [24].

2.2.8 IoU (*Intersection over union*)

Intersection over union (IoU) digunakan untuk mengukur sejauh mana *bounding box* yang dihasilkan oleh model yang tumpang tindih dengan *bounding box* aslinya. Nilai IoU yang tinggi menunjukkan deteksi objek yang lebih akurat.



Gambar 2.14 IoU

Gambar 2.14 merupakan ilustrasi dari IoU pada sebuah citra atau gambar. IoU merupakan pembagian antara area yang bersinggungan dengan area secara keseluruhan. Dimana 2 *bounding box* yang ada terdapat *ground-truth* dan *predicted bounding* box. Precision, *recall*, dan *F1-Score*: Matriks ini memberikan gambaran lebih lanjut tentang kinerja model dalam mendeteksi objek. *Precision* mengukur akurasi deteksi, *recall* mengukur seberapa banyak objek yang berhasil dideteksi, dan F1-*Score* adalah perpaduan dari keduanya. Berikut merupakan rumus dari *F1-score*.

$$F1\ Score = \frac{2x(Precision\ x\ Recall)}{Precision + Recall}$$

2.2.9 *Confusion Matrix*

Confusion matrix merupakan tabel yang digunakan untuk evaluasi performa klasifikasi dalam *machine learning*, yang menampilkan jumlah prediksi yang tepat dan tidak tepat. *Confusion matrix* menyediakan informasi untuk menghitung nilai recall, presisi, dan akurasi model. Di dalamnya terdapat empat skenario: *true positive (TP), true negative (TN), false positive (FP), dan false negative (FN)* [25]. Ilustrasi *confusion matrix* dijelaskan pada tabel 2.1 dibawah berikut ini.

Tabel 2.1 *Confusion Matrix*

		ACTUAL	
		Positive	***Negative***
PREDICTION	***Positive***	***True Positive*** **(TP)**	***False Negative*** **(FN)**
	Negative	***False Negative*** **(FN)**	***True Negative*** **(TN)**

Berdasarkan Tabel 2.1 diatas dapat dijelaskan dari istilah yang terdapat pada *confusion matrix* adalah sebagai berikut:

- *True Positive* merujuk pada jumlah data positif yang diklasifikasikan dengan benar. Contohnya, ketika sistem memprediksi sebuah Bus TransJakarta yang sebenarnya normal sebagai normal.
- *True Negative* merujuk pada jumlah data negatif yang diklasifikasikan dengan benar.
- *False Positive* merujuk pada jumlah data positif yang diklasifikasikan secara salah kemudian *False Negative* merujuk pada jumlah data negatif yang diklasifikasikan secara salah.

2.2.10 Akurasi

Akurasi mengindikasikan seberapa besar proporsi prediksi yang benar dari seluruh data yang telah diproses [26]. Mengacu pada matriks konfusi, akurasi dapat diperoleh dari rumus berikut :

$$Akurasi = \frac{TP}{TP + FP + FN}$$

2.2.11 *Precision*

Presisi merupakan rasio sampel yang diprediksi dengan benar dalam suatu kelas dengan jumlah sampel yang menghasilkan prediksi untuk kelas yang diamati. Presisi ini memiliki nilai yang baik Ketika model mereduksi kesalahan klasifikasi menjadi level akurasi. Parameter ini digunakan untuk mengetahui berapa banyak prediksi afirmatif yang benar, dari himpunan data prediksi positif [26]. Persamaan presisi adalah sebagai berikut :

$$Precision = \frac{TP}{TP + FP}\ x\ 100\%$$

2.2.12 *Mean Average Precision*

Mean Average Precision (mAP) adalah nilai yang diperoleh dengan mengambil rata-rata presisi rata-rata (AP) atau ambang batas *IoU* untuk semua kelas dalam suatu *dataset*. [26]. mAP ini memiliki persamaan:

$$mAP = \frac{1}{N}\sum_{i=1}^{N} APi$$

APi = nilai AP dari kelas , N = Jumlah kelas

Untuk mencari nilai *average precision* dapat dilakukan dengan mencari nilai skor prediksi menggunakan model. Kemudian, konversikan skor prediksi menjadi label kelas. Lalu hitung *confusing matrix* – TP, FP, TN, FN. Setelah itu, hitung area di bawah kurva presisi – *recall*.

2.2.13 *Recall Rate*

Recall merupakan perbandingan antara nilai *true positive* dengan semua data yang *true positive*. Nilai *recall* ini dapat digunakan untuk melihat perkiraan kemampuan model terhadap suatu kelas [26]. Persamaan dari *recall rate* adalah sebagai berikut:

$$Recall\ Rate = \frac{TP}{TP + FN}\ x\ 100\%$$

Poin perbandingan dengan rumus *precision* Dalam perbandingan antara *precision* dan *recall*, keduanya memiliki perbedaan dalam variabel yang digunakan: precision menggunakan *False Positive (FP)* sedangkan *recall* menggunakan *False Negative (FN)*. Kedua metrik tersebut memiliki kesamaan bahwa semakin kecil dengan meningkatnya nilai *False Positive (FP), precision* akan meningkat begitu pula dengan *recall*, semakin kecil nilai *False Negative (FN),* maka nilai *recall* akan semakin tinggi [26].