BAB II
TINJAUAN PUSTAKA

2.1 *Artificial Intelligence* (AI)

Artificial Intelligence (AI) atau Kecerdasan buatan hadir sebagai cabang ilmu dari *Computer Science* yang menjanjikan banyak manfaat dalam menjawab kebutuhan manusia di masa depan. Literatur mengenai kecerdasan buatan menyebutkan bahwa ide mengenai kecerdasan buatan diawali pada awal abad 17 ketika Rene Descartes mengemukakan bahwa tubuh hewan bukanlah apa-apa melainkan hanya mesin-mesin yang rumit. Kata "*intelligence*" sendiri berasal dari bahasa Latin "*intelligo*" yang berarti "saya paham". Berarti dasar dari *intelligence* ialah kemampuan untuk memahami dan melakukan aksi. *Intelligence* merupakan istilah yang komplek yang dapat didefinisikan dengan ungkapan yang berbeda seperti logika, pemahaman, *self-awareness*, pembelajaran, perencanaan, dan *problem solving*. Sedangkan "*Artificial*" adalah sesuatu yang tidak nyata, seperti tipuan karena merupakan hasil simulasi [6].

2.2 *Deep Learning*

Deep Learning merupakan salah satu bidang dari *Machine Learning* yang memanfaatkan jaringan syaraf tiruan untuk implementasi permasalahan dengan dataset yang besar. Teknik *Deep Learning* memberikan arsitektur yang sangat kuat untuk *Supervised Learning* (pembelajaran terarah/terawasi). Dengan menambahkan lebih banyak lapisan maka model pembelajaran tersebut bisa mewakili data citra berlabel dengan lebih baik. Pada *Machine Learning* terdapat teknik untuk menggunakan ekstraksi fitur dari data pelatihan dan algoritma pembelajaran khusus untuk mengklasifikasi citra maupun untuk mengenali suara. *Feature Engineering* juga merupakan teknik yang paling penting untuk mencapai hasil yang baik pada tugas prediksi. Namun, sulit untuk dipelajari dan dikuasai karena kumpulan data dan jenis data yang berbeda memerlukan pendekatan teknik yang berbeda juga [7].

2.3 *Object Recognition*

Teknologi pengenalan objek merupakan penelitian dasar di bidang *computer vision*. Fungsi utamanya adalah untuk mengenali objek yang ada di dalam gambar serta memberikan posisi dan arah objek di dalam gambar, jadi komputer dapat mensimulasikan otak manusia dan fungsi mata manusia. Teknologi ini bisa diterapkan di berbagai bidang, seperti sistem pengenalan wajah, *Augmented Reality*, *Virtual Reality*, visi mesin dan bidang lainnya.

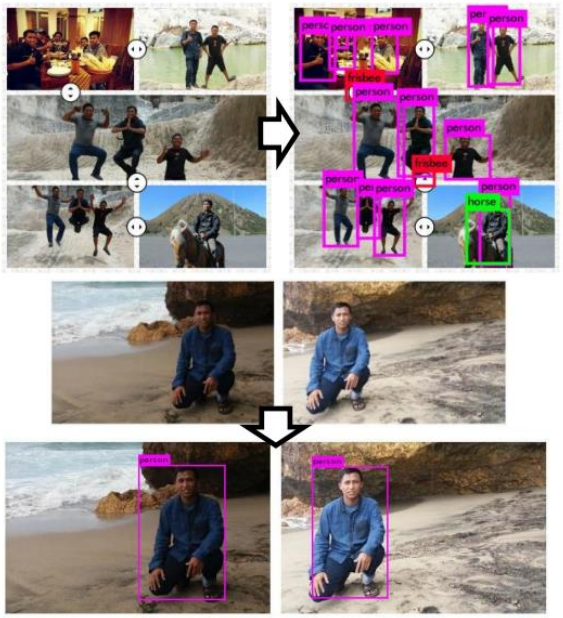
Dalam beberapa tahun terakhir, pengembangan perangkat keras dan perangkat lunak komputer, pengenalan yang efisien, pengenalan dan deteksi algoritma telah menyebabkan pengembangan teknologi pengenalan objek terhadap empat kriteria penilaian metode pengenalan objek, yaitu *robustness*, *correctness*, efisiensi dan ruang lingkup. Proses pengenalan objek dibagi menjadi proses mendapatkan satu gambar bingkai objek, *preprocessing* gambar, ekstraksi fitur, pemilihan fitur, pencocokan fitur, dan umpan balik informasi identifikasi objek dari hasil pencocokan. Kunci apakah suatu objek teknologi pengenalan yang efisien terletak pada efisiensi ekstraksi fitur objek, fitur pencocokan pemrosesan, dan metode klasifikasi dan pengenalan. Dengan terus berkembangnya *deep learning*, penerapan *deep learning* dalam bidang pengenalan objek sudah menjadi penelitian *hotspot* di berbagai perusahaan dan lembaga penelitian ilmiah.

Teknologi pengenalan objek saat ini dapat diklasifikasikan menjadi dua kategori yaitu metode pengenalan berbasis model atau metode pengenalan berbasis konteks, pengenalan objek dua dimensi atau pengenalan objek tiga dimensi [8].

2.4 *You Only Look Once* (YOLO)

You Only Look Once (YOLO) adalah sebuah algoritma yang dikembangkan untuk mendeteksi sebuah objek secara *real-time*. Sistem pendeteksian yang dilakukan adalah dengan menggunakan *repurpose classifier* atau *localizer* untuk melakukan deteksi. Sebuah model diterapkan pada sebuah citra di beberapa lokasi dan skala. Daerah dengan citra yang diberi *score* paling tinggi akan dianggap sebagai sebuah pendeteksian [9].

YOLO menggunakan pendekatan jaringan saraf tiruan (JST) untuk jumendeteksi objek pada sebuah citra. Jaringan ini membagi citra menjadi beberapa wilayah dan memprediksi setiap kotak pembatas dan probabilitas untuk setiap wilayah. Kotak-kotak pembatas ini kemudian dibandingkan dengan setiap probabilitas yang diprediksi. YOLO memiliki beberapa kelebihan dibandingkan dengan sistem yang berorientasi pada *classifier*, terlihat dari seluruh citra pada saat dilakukan test dengan prediksi yang diinformasikan secara global pada citra. Hal tersebut juga membuat prediksi dengan sintesis jaringan saraf ini tidak seperti sistem *Region Convolutional Neural Network* (R-CNN) yang membutuhkan ribuan untuk sebuah citra sehingga membuat YOLO lebih cepat hingga beberapa kali daripada R-CNN [10].



Gambar 2.1 *You Only Look Once* (YOLO) [11]

Keluarga algoritma YOLO pada dasarnya adalah masalah regresi objek. Proses deteksi melibatkan membagi gambar masukan ke dalam grid tertentu sesuai dengan spesifikasi tertentu, melintasi seluruh gambar satu kali, dan ketika sebuah objek terdeteksi di setiap grid itu ditarik ke dalam kotak prediksi berdasarkan kotak jangkar grid, yang pada gilirannya memprediksi hasil objek secara langsung [12].

YOLO mendeteksi objek dengan menggunakan *unified model* dimana sebuah *single convolutional network* memprediksi beberapa *bounding boxes* (kotak

pembatas) serta probabilitas kelas di dalam kotak-kotak tersebut secara bersamaan. Pertama-tama, sistem YOLO membagi citra *input* ke dalam *grid* S × S. Jika pusat dari sebuah objek jatuh di dalam salah satu sel *grid*, maka sel *grid* itu bertanggung jawab untuk mendeteksi objek tersebut. Setiap sel *grid* memprediksi *bounding boxes* dan *confidence score* dari tiap *bounding box* tersebut.

Confidence score merefleksikan seberapa yakin dan akurat model bahwa terdapat sebuah objek di dalam kotak tersebut. Setiap *bounding box* terdiri dari 5 prediksi: x, y, w, h, dan *confidence*. Koordinat (x,y) mewakili pusat dari kotak relatif ke batas sel *grid*. (w,h) atau lebar dan tinggi mewakili pusat dari kotak relatif ke gambar. Dan terakhir adalah *confidence* yang mewakili *Intersection over Union* (IoU) antara kotak prediksi dan kotak *ground truth*. Setiap sel *grid* juga memprediksi probabilitas kelas. Probabilitas dikondisikan pada sel *grid* yang memuat objek dan hanya satu kelas probabilitas yang dideteksi per sel *grid* tanpa memperhitungkan jumlah *bounding boxes* [13].

Karena menggunakan *single convolutional network* dalam proses deteksi objek, YOLO memiliki beberapa kelebihan dibanding metode deteksi objek yang lain seperti berikut ini.

a. Proses deteksi yang sangat cepat

   YOLO bekerja dengan menggunakan *convolutional network* tunggal dalam mendeteksi objek, sehingga tidak membutuhkan langkah-langkah yang rumit. YOLO dapat bekerja sampai dengan 45 *frame per second* pada GPU Titan X tanpa menggunakan *batch processing*. Dengan kecepatan tersebut, YOLO dapat mendeteksi objek pada *real-time video streaming* dengan waktu kurang dari 25 *millisecond*. YOLO juga dapat mencapai nilai rata-rata presisi (*mean average precision*) dua kali lebih baik daripada sistem *real-time* lainnya.

b. Dapat memahami konteks gambar secara keseluruhan

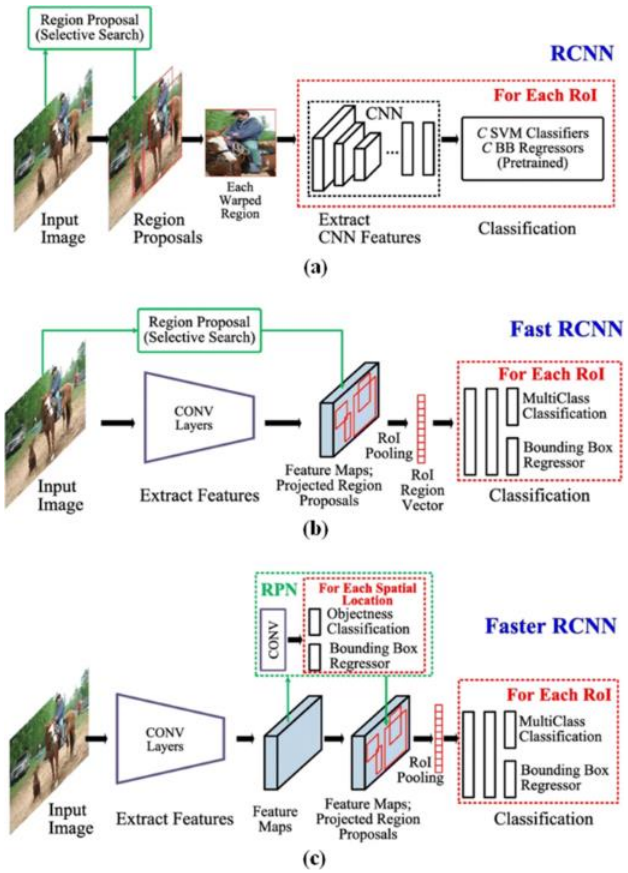
   YOLO memproses seluruh gambar secara bersamaan pada saat proses *training* maupun *testing*. Dengan demikian YOLO dapat menyimpan informasi tentang konteks objek pada suatu gambar. Fast *Region with Convolutional Neural Network* (Fast R-CNN) terkadang salah dalam

mendeteksi *background* gambar sebagai sebuah objek karena ketidakmampuan dalam melihat konteks objek pada gambar. YOLO melakukan kesalahan dalam mendeteksi *background* dua kali lebih sedikit dibandingkan dengan Fast R-CNN.

c. Dapat menggeneralisir representasi objek

Pada saat proses *training* menggunakan gambar alami dan proses *testing* menggunakan gambar hasil karya seni, YOLO menghasilkan prediksi yang lebih baik dibandingkan metode deteksi objek lain seperti *deformable parts models* (DPM) dan R-CNN.

Arsitektur dari algoritma YOLO menggunakan *Convolutional Neural Network* yang memiliki 24 *convolutional layers* dan diikuti oleh 2 *fully connected layers*. *Convolutional layer* berfungsi untuk mengekstraksi fitur dari *input* gambar, sedangkan *fully connected layer* berperan dalam memprediksi probabilitas *output* dan koordinat [14].



Gambar 2.2 Arsitektur YOLO [15]

2.5 *Convolutional Neural Network* (CNN)

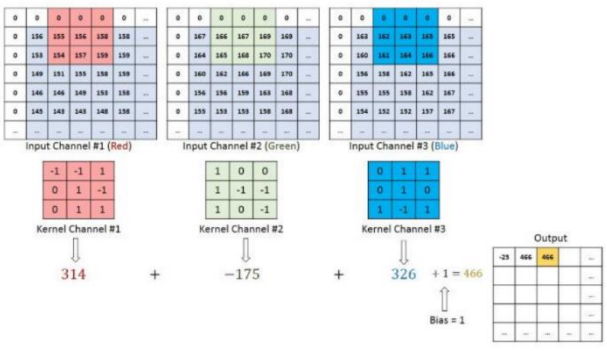
Convolutional Neural Network atau ConvNet adalah salah satu kelas *deep feed-forward artificial neural network* yang banyak digunakan pada analisis citra. CNN bisa digunakan untuk mendeteksi dan mengenali objek pada sebuah citra. Secara garis besar CNN tidak jauh berbeda dengan *neural network* biasanya. CNN terdiri dari *neuron* yang memiliki *weight*, *bias* dan *activation function*. CNN terdiri atas satu lapisan masukan (*input layer*), suatu lapisan keluaran (*output layer*) dan sejumlah lapisan tersembunyi (*hidden layer*) [5].

2.5.1 *Feature Extraction Layer*

Lapisan ekstraksi fitur (*Feature Extraction Layer*) citra terletak pada awal arsitektur yang tersusun atas beberapa lapisan dan setiap lapisan tersusun atas *neuron* yang terkoneksi pada daerah lokal (*local region*) lapisan sebelumnya. Lapisan pertama adalah lapisan konvolusi (*convolution layer*) dan lapisan kedua adalah lapisan *pooling* (*pooling layer*).

2.5.2 *Convolution Layer*

Tahap awal dalam proses CNN yang memproses *input* akan menghasilkan banyak peta fitur *Feature map*. *Input* awal berupa citra akan dilakukan konvolusi dengan filter. Filter yang digunakan tergantung citra *input* yang dimasukan sebagai input seperti 3x3, 5x5, 7x7, dan sebagainya. Pada proses penerapan filter, digunakan parameter *stride* untuk menentukan pengulangan pada pergeseran filter dan proses *padding* untuk memanipulasi dimensi *output* pada lapisan konvolusi [5].



Gambar 2.3 Operasi Konvolusi pada Citra Matrikx MxNx3 dengan Kernel 3x3x3 [5]

2.5.3 *Stride*

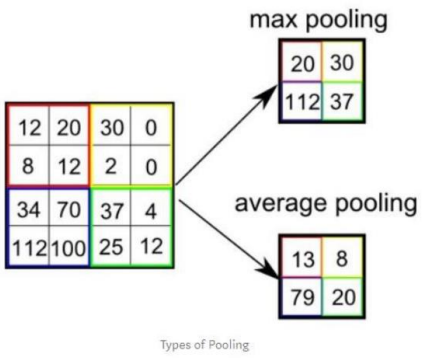
Stride adalah parameter yang menentukan berapa jumlah pergeseran filter. Jika nilai *stride* adalah 1, maka filter konvolusi akan bergeser sebanyak 1 piksel secara horizontal lalu vertikal. Semakin kecil *stride* maka akan semakin detail informasi yang diperoleh dari sebuah input, namun membutuhkan komputasi yang lebih jika dibandingkan dengan *stride* yang besar. Perlu diperhatikan bahwa dengan menggunakan *stride* yang kecil, tidak selalu didapatkan performa yang bagus [5].

2.5.4 *Padding*

Padding atau *Zero Padding* adalah parameter yang menentukan jumlah piksel berisi nilai 0 yang akan ditambahkan di setiap sisi dari *input*. Hal ini digunakan dengan tujuan untuk memanipulasi dimensi *output* dari lapisan konvolusi (*Feature map*) [5].

2.5.5 *Pooling Layer*

Pada lapisan *pooling* (*Pooling Layer*) peta-peta fitur yang telah di proses melalui ReLU akan dikecilkan dengan memilih ukuran *window*. Dengan adanya lapisan *pooling*, data akan menjadi kecil, mudah dikelola, dan mudah mengontrol *overfitting*. Cara mengecilkan peta fitur yaitu dengan menggunakan fungsi *MaxPooling*. *MaxPooling* yaitu fungsi yang memilih nilai maksimum dari daerah *window* tersebut lalu direpresentasikan sebagai piksel baru. *MaxPooling* bergerak dari pojok kiri atas ke kanan dengan ukuran *pooling* yang telah ditentukan dan pergeseran *pooling* yang telah ditentukan [5].



Gambar 2.4 *Max Pooling* dan *Average Pooling* [5]

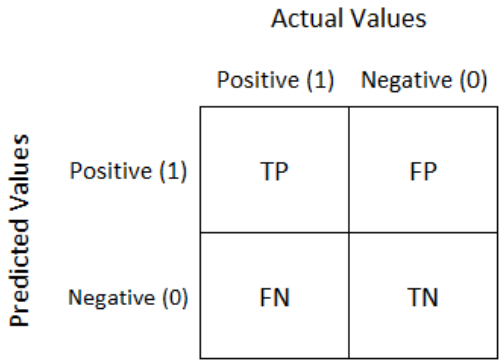
2.5.6 ReLU (*Rectified Linear Units*)

Fungsi aktivasi ReLU digunakan untuk menghilangkan linearitas yang ada pada hasil Lapisan Konvolusi (*Convolution Layer*). Fungsi aktivasi ini digunakan untuk menghasilkan keluaran dari Lapisan Konvolusi dan Lapisan *Pooling*.

2.6 *Confusion Matrix*

Confusion matrix merupakan rasio hasil prediksi dalam masalah klasifikasi objek. Jumlah penilaian benar atau salah akan dihitung kemudian dibagi dengan setiap kelas. *Confusion matrix* memberikan informasi tidak hanya tentang kesalahan pengklasifikasi, tetapi juga tentang jenis kesalahan. *Confusion matrix* menggunakan data dalam bentuk matrik [16]. Dari table matriks yang didapatkan akan dihitung akurasi, presisi, F-Score, dan *recall* berdasarkan kondisi data yang diprediksi atau diklasifikasi [17]. Berikut adalah isi dari *confusion matrix*.

1. *True Positive* (TP) adalah kondisi dimana model memprediksi data sebagai ya (*TRUE*) dan jawaban aktualnya adalah ya (*TRUE*).
2. *True Negative* (TN) adalah kondisi dimana model memprediksi data sebagai tidak (*FALSE*) dan jawaban aktualnya adalah tidak (*FALSE*).
3. *False Positive* (FP) adalah kondisi dimana model memprediksi data sebagai ya (*TRUE*) dan jawaban aktualnya adalah tidak (*FALSE*).
4. *False Negative* (FN), kondisi dimana model memprediksi data sebagai tidak (*FALSE*) dan jawaban aktualnya adalah ya (*TRUE*).



Gambar 2.5 Tabel *Confusion Matrix* [16]

Evaluasi kinerja melibatkan perhitungan metrik seperti akurasi, presisi, sensitivitas, dan f1-*score*, yang dihitung berdasarkan nilai TP, TN, FP, dan FN. Metrik-metrik ini memberikan informasi mengenai kinerja model yang diuji.

Confusion Matrix berfungsi sebagai alat penting untuk menghitung akurasi, presisi, dan *recall*. Dibawah ini merupakan cara melakukan pengukuran *perfomance*.

1. *Accuracy*

   Accuracy didefinisikan sebagai jumlah rasio prediksi yang benar positif dan negatif lalu dibagi dengan keseluruhan data. Nilai akurasi dapat diperoleh dengan persamaan [18].

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

2. *Precision*

   Precision didefinisikan sebagai rasio prediksi benar positif dibandingkan dengan jumlah data yang negatif yang dianggap positif. Nilai *precision* diperoleh dengan persamaan [18].

   $$\textit{Precision} = \frac{TP}{TP+FP} \text{ X } 100\%$$

3. *Recall*

   Recall didefinisikan sebagai rasio jumlah prediksi positif dibandingkan dengan jumlah data benar positif. Nilai *Recall* diperoleh dengan persamaan [18].

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

4. F1-*Score*

   F1-*Score* adalah *harmonic mean* dari *precision dan recall*. Nilai F1-*Score* diperoleh dengan persamaan [18].

   $$\text{F1-}\textit{Score} = \frac{1}{2}\left(\frac{1}{Precision} + \frac{1}{Recall}\right)$$

2.7 Labelimg

LabelImg adalah alat anotasi gambar grafis sumber terbuka yang awalnya dikembangkan oleh TzuTa Lin dan dikelola oleh komunitas pengembang di Label Studio. Saat ini dihosting di organisasi GitHub bernama *heartexlabs*, LabelImg ditulis dengan Python dan menggunakan Qt untuk antarmuka grafisnya.

Saat ini, LabelImg menawarkan anotasi hanya dalam bentuk kotak pembatas yang dapat diekspor ke format PASCAL VOC, YOLO, dan CreateML dalam bentuk file XML [19].

2.8 *Mean Average Precision* (mAP)

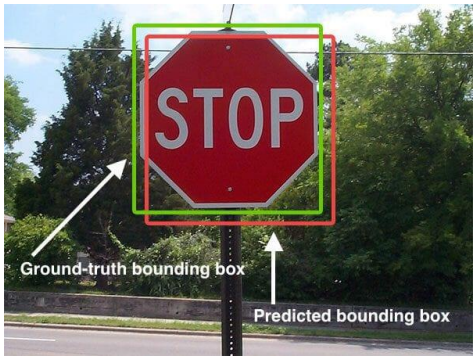
Mean Average Precision (mAP) adalah metrik yang digunakan untuk mengevaluasi model deteksi objek seperti *Fast* R-CNN, YOLO, *Mask* R-CNN, dan sebagainya. Rata-rata nilai *Average Precision* (AP) dihitung berdasarkan nilai perolehan dari 0 hingga 1 [20]. Rumus mAP didasarkan pada sub metrik: *Confusion Matrix*, *Intersection over Union* (IoU), *Recall*, dan *Precision*.

2.9 *Intersection over Union* (IoU)

Intersection over Union (IoU) merupakan metrik evaluasi untuk mengukur keakuratan detektor objek pada himpunan data tertentu. IoU dapat digunakan dengan ketentuan sebagai berikut [21].

1. Memiliki ground-truth bounding box pada himpunan data objek
2. Prediksi bounding box pada himpunan data objek

Ilustrasi perbandingan *ground-truth bounding box* dan *predicted bounding box* dari model seperti pada Gambar 2.5. *Intersection over Union* (IoU) merupakan perbandingan antara *ground-truth bounding box* dengan *predicted bounding box* pada model.



Gambar 2.6 Ilustrasi *Predicted* dan *Ground-Truth Bounding Box* pada *Intersection over Union* [21]

2.10 Bahasa Pemrograman

Bahasa pemrograman yang dipakai pada penelitian ini adalah Python. Python merupakan salah satu bahasa pemograman tingkat tinggi yang berorientasi objek dengan semantik dinamis. Python relatif sederhana karena memerlukan sintak unik yang berfokus pada keterbacaan. Pengembang dapat membaca dan juga menerjemahkan kode python jauh lebih mudah dari bahasa lain. Selain itu Python mendukung penggunaan modul dan juga paket yang berarti bahwa program dapat dirancang dengan gaya modular dan kode dapat digunakan kembali dalam berbagai proyek. Jadi setelah modul dan juga paket dikembangkan, hal ini bisa ditingkatkan untuk digunakan dalam proyek lain dan juga bisa untuk mengimpor ataupun mengekpor modul ini.

Salah satu manfaat dari bahasa pemrograman Python adalah bahwa pustaka standar tersedia secara gratis, baik dalam bentuk biner maupun sumber. Tidak ada yang ekslusif juga karena Python dan semua alat yang diperlukan tersedia di semua platfrom utama. Jadi ini merupakan opsi yang menarik bagi pengembang yang tidak ingin khawatir membayar biaya pengembangan [22].

Gambar 2.7 Logo Python [22]

Terdapat *library* yang digunakan dalam Python yang digunakan yaitu NumPy. NumPy merupakan singkatan dari *Numerical Python*. NumPy merupakan salah satu *library* Python yang berfungsi untuk proses komputasi numerik. NumPy memiliki kemampuan untuk membuat objek N-dimensi *array*. *Array* merupakan sekumpulan variabel yang memiliki tipe data yang sama. Kelebihan dari NumPy *Array* adalah dapat memudahkan operasi komputasi pada data, cocok untuk melakukan akses secara acak, dan elemen *array* merupakan sebuah nilai yang independen sehingga penyimpanannya dianggap sangat efisien [23].

2.11 *Library*

2.11.1 OpenCV

OpenCV *(open source computer vision)* merupakan *library open source* yang tujuannya dikhususkan untuk melakukan pengolahan citra. Maksudnya adalah agar komputer mempunyai kemampuan yang mirip dengan cara pengolahan visual pada manusia. OpenCV telah menyediakan banyak algoritma visi komputer dasar. OpenCV juga menyediakan modul pendeteksian objek yang menggunakan metode computer vision [24]. Tugas OpenCV adalah sebagai berikut.

1. Membaca dan menulis gambar.
2. Mendeteksi wajah dan juga fitur-fiturnya.
3. Mendeteksi bentuk seperti lingkaran, persegi panjang dan lain-lain didalam gambar.
4. Pengenalan teks di dalam gambar seperti pelat nomor dan sebagainya.
5. Memodifikasi kualitas dan warna gambar misalnya CamScanner.
6. Mengembangkan aplikasi *augmented reality*.

Gambar 2.8 Logo OpenCV [24]

2.11.2 CUDA

CUDA adalah platform komputasi paralel dan model pemrograman yang dikembangkan oleh NVIDIA untuk komputasi umum pada unit pemrosesan grafis (GPU). Dengan CUDA, pengembang dapat secara dramatis mempercepat aplikasi komputasi dengan memanfaatkan kekuatan GPU. Saat menggunakan CUDA, pengembang memprogram dalam bahasa populer seperti C, C++, Fortran, Python, dan MATLAB dan mengekspresikan paralelisme melalui ekstensi dalam bentuk beberapa kata kunci dasar [25].

2.11.3 cuDNN

NVIDIA CUDA *Deep Neural Network library* (cuDNN) adalah pustaka primitif yang dipercepat GPU untuk *deep neural networks*. cuDNN menyediakan implementasi yang sangat disesuaikan untuk rutinitas standar seperti konvolusi maju dan mundur, penyatuan, normalisasi, dan lapisan aktivasi.

Peneliti *deep learning* dan pengembang *framework* di seluruh dunia mengandalkan cuDNN untuk akselerasi GPU performa tinggi. Ini memungkinkan mereka untuk fokus pada pelatihan *neural networks* dan mengembangkan aplikasi perangkat lunak daripada menghabiskan waktu untuk penyetelan kinerja GPU tingkat rendah. cuDNN mempercepat *framework* kerja *deep learning* yang banyak digunakan, termasuk Caffe2, Chainer, Keras, MATLAB, MxNet, PaddlePaddle, PyTorch, dan TensorFlow [26].

2.12 Dataset

2.12.1 COCO

COCO adalah singkatan dari *Common Objects in Context*, karena kumpulan data gambar dibuat dengan tujuan memajukan pengenalan gambar. Kumpulan data COCO berisi kumpulan data visual yang menantang dan berkualitas tinggi untuk visi komputer, sebagian besar *neural networks* canggih.

Dataset MS COCO adalah deteksi objek skala besar, segmentasi gambar, dan dataset teks yang diterbitkan oleh Microsoft. Insinyur *Machine Learning* dan

Computer Vision populer menggunakan dataset COCO untuk berbagai proyek *computer vision* [27].

2.12.2 Kaggle

Kaggle adalah situs untuk berbagi ide, mendapatkan inspirasi, bersaing dengan ilmuwan data lainnya, mempelajari informasi baru dan trik pengkodean, dan melihat contoh penerapan ilmu data. Ada banyak kumpulan data seperti data polusi udara yang dapat digunakan untuk hal-hal sederhana seperti menjual video game hingga sesuatu yang lebih kompleks dan penting. Data ini sebenarnya direferensikan sehingga dapat melatih dan menguji model dalam proyek yang pada akhirnya dapat membantu orang lain [28].

2.13 *Tools*

2.13.1 *CMake*

CMake adalah sistem yang mengelola proses kompilasi pada sistem operasi yang tidak tergantung pada *compiler* tertentu. Dalam hal *CMake* membantu peneliti untuk melakukan kompilasi *framework* OpenCV menjadi *library* yang nantinya digunakan dalam aplikasi [29].

Gambar 2.9 Logo CMake [29]

2.13.2 *Powershell*

Windows Powershell merupakan suatu kerangka kerja atau *framework* yang dibangun di atas .NET *Framework Common Language Runtime* (CLR) dan *accept and return* .Net *Framework*. *Windows* menyediakan wadah untuk menulis dan menguji skrip yang sedang dikerjakan yang disebut *Powershell Integrated Scripting Environtment* (ISE) dan akan menghasilkan file skrip *Powershell*. Ekstensi dari file skrip *Powershell* tersebut adalah .ps1 [30].

2.13.3 *Anaconda*

Anaconda dikembangkan oleh Organisasi *Anaconda, Inc*. (*Continuum Analytics*). *Anaconda* merupakan sebuah paket distribusi dari bahasa pemrograman *Python* dan R serta berisi beberapa paket tambahan guna pemrograman *data science*, komputasi ilmiah (*scientific computing*) seperti *data science*, *machine learning*, *data processing* skala-luas, analisis prediksi, matematika hingga teknik dalam satu distribusi *platform* yang *user friendly*. *Anaconda* menyediakan banyak pustaka dan paket yang sudah diinstal sebelumnya. Beberapa di antaranya adalah NumPy, SciPy, Pandas, Scikit *learn*, nltk, dan Jupiter. *Anaconda* menyediakan *conda* sebagai pengelola paket sedangkan bahasa *Python* menyediakan *pip* sebagai pengelola paket. *Pip Python* memungkinkan penginstalan dependensi *Python*. Disisi lain, *Anaconda conda* memungkinkan penginstalan dependensi *library Python* dan *non-Python*. Kegunaan anaconda berkaitan dengan *Data Science* dan *Machine Learning* [31].

Gambar 2.10 Logo *Anaconda* [31]

2.13.4 Git

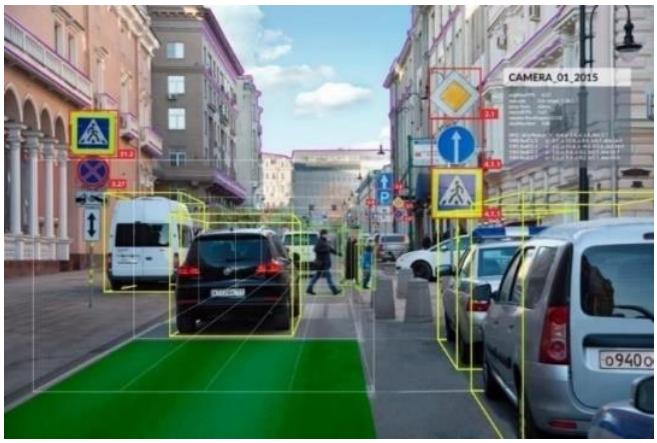
Git (*Group Inclusive Tour*) adalah salah satu tipe *Version Control System* (VCS) yang memudahkan proses pelacakan dan pencatatan perubahan dari sebuah dokumen. Hal ini memudahkan *developer* untuk melihat secara detail perubahan yang terjadi pada kode aplikasi atau *website*. Tujuan penggunaan GIT yakni untuk mengelola versi source code program dengan menentukan baris serta kode yang akan ditambahkan atau diganti [32].

Gambar 2.11 Logo Git [32]

2.14 *Computer Vision*

Computer Vision merupakan suatu bidang ilmu komputer yang bekerja untuk membuat komputer yang mungkin untuk melihat, mengidentifikasi, dan memproses gambar dengan cara yang sama seperti yang dilakukan oleh manusia dan kemudian memberikan *output* yang sesuai. Hal ini seperti menanamkan naluri dan kecerdasan manusia ke dalam *computer*. *Computer Vision* berkaitan erat dengan kecerdasan buatan karena komputer harus menginterprestasikan apa yang dilihatnya dan kemudian melakukan analisis.

Tujuan *Computer Vision* tidak hanya untuk melihat, tetapi juga untuk memproses dan juga memberikan hasil yang bermanfaat berdasarkan pengamatan. Misalnya, komputer dapat membuat gambar 3D dari gambar 2D. Seperti halnya pada sebuah mobil. Saat sebuah mobil dilengkapi dengan *computer vision*, maka mobil dapat mengidentifikasi dan membedakan objek yang berada di jalan seperti lampu lalu lintas, rambu lalu lintas, pejalan kaki dan sebagainya, dan kemudian *computer vision* akan memberikan tanda kepada pengemudi atau bahkan akan membuat mobil berhenti apabila ada hambatan yang mendadak di jalan [33]



Gambar 2.12 *Computer Vision* [33]

Saat seseorang mengendarai mobil dan kemudian melihat ada objek yang tiba-tiba bergerak ke dalam jalur mobil, maka pengemudi harus bereaksi langsung seperti langsung mengerem, maka pengemudi tersebut telah melakukan tugas yang sangat kompleks, yaitu mengidentifikasi objek, memproses dan kemudian

memutuskan tindakan yang akan diambil. Tujuan Computer Vision adalah untuk memungkinkan sebuah komputer untuk melakukan tugas yang sama seperti yang dilakukan manusia dengan efisien yang sama.

2.15 *Webcam*

Webcam merupakan sebuah kamera yang terhubung dengan komputer. *Webcam* menangkap gambar diam dan juga gambar yang bergerak atau video. Dengan bantuan perangkat lunak, *webcam* dapat mengirimkan videonya ke internet secara *real-time*. Dibawah ini merupakan salah satu contoh *Webcam* [34].

Gambar 2.13 *Webcam* [34]

Webcam tidak seperti kamera digital dan juga *camrecorder* digital yang memiliki penyimpanan sendiri. Jadi *webcam* selalu terhubung ke komputer dan menggunakan penyimpanan komputer untuk menyimpan gambar ataupun video yang telah direkam.

2.16 Penelitian Terdahulu

Berikut adalah rangkuman dari beberapa penelitian yang pernah dilakukan sebelumnya.

Tabel 2.1 Penelitian Terdahulu

Penulis	Judul	Metode	Akurasi Model
Shu-Jun Ji, Qing-Hua Ling, Fei Han	*An Improved Algorithm for Small Object Detection Based on* YOLO v4 *and Multi-scale Contextual Information*	YOLO v4 *Multi-scale Contextual Information*	73.51%
Lusiana Rahma, Hadi Syaputra, A.Haidar Mirza, Susan Dian Purnamasari	Objek Deteksi Makanan Khas Palembang Menggunakan Algortima YOLO (*You Only Look Once*)	*Deep learning models You Only Look Once*	96.00%
Sisco Jupiyandi, Fadhil R. Saniputra, Yoga Pratama, M. Robby Dharmawan	Pengembangan Deteksi Citra Mobil Untuk Mengetahui Jumlah Tempat Parkir Menggunakan CUDA dan *Modified* YOLO	CUDA dan *Modified* YOLO (M-YOLO)	100%
Oktaviani E. Karlina, Dina Indarti	Pengenalan Objek Makanan Cepat Saji pada Video dan *Real*-time *Webcam* Menggunakan Metode *You Only Look Once* (YOLO)	*You Only Look Once* (YOLO)	76.30%
Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi	*You Only Look Once*: *Unified, Real-Time Object Detection*	*You Only Look Once with Deformable Parts Models and Fast* R-CNN	70.70%
Calvin Geraldy, Chairisni Lubis	Pendeteksian dan Pengenalan Jenis Mobil Menggunakan Algoritma *You Only Look Once* dan *Convolutional Neural Network*	*You Only Look Once* dan *Convolutional Neural Network*	88.10%

Berdasarkan beberapa penelitian sebelumnya yang menggunakan beberapa metode dan kumpulan data, terlihat jelas bahwa penelitian pendeteksian dengan algoritma *You Only Look Once* (YOLO) mampu memberikan tingkat ketelitian yang tinggi. Oleh sebab itu pada riset ini akan diaplikasikan simulasi sistem deteksi serta identifikasi daging sapi dan daging babi dengan menggunakan algoritma *You Only Look Once* (YOLO).