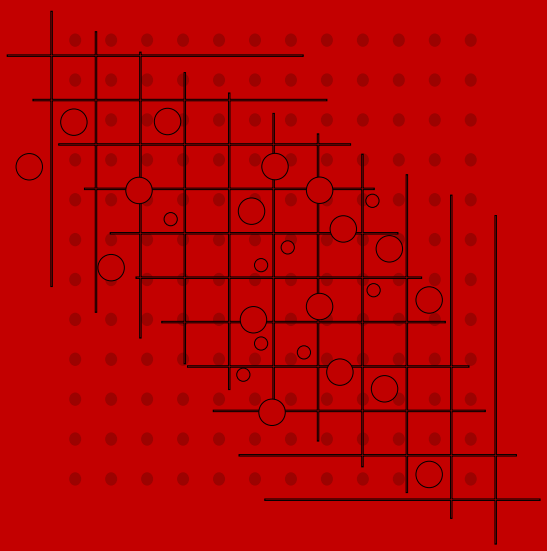
IMPLEMENTASI PENGENALAN OBJEK REAL-TIME DENGAN FRAMEWORK YOLOv5 PADA PLATFORM CLOUD COMPUTING KUBERNETES KUBEFLOW BERBASIS DEEP LEARNING



FARHAN RAMDHANI
H071171527

PROGRAM STUDI SISTEM INFORMASI
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS HASANUDDIN
MAKASSAR
2024

IMPLEMENTASI PENGENALAN OBJEK REAL-TIME DENGAN FRAMEWORK YOLOv5 PADA PLATFORM CLOUD COMPUTING KUBERNETES KUBEFLOW BERBASIS DEEP LEARNING

FARHAN RAMDHANI
H071171527

PROGRAM STUDI SISTEM INFORMASI
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS HASANUDDIN
MAKASSAR
2024

IMPLEMENTASI PENGENALAN OBJEK REAL-TIME DENGAN FRAMEWORK YOLOv5 PADA PLATFORM CLOUD COMPUTING KUBERNETES KUBEFLOW BERBASIS DEEP LEARNING

FARHAN RAMDHANI
H071171527

Skripsi

sebagai salah satu syarat untuk mencapai gelar sarjana

Program Studi Sistem Informasi

pada

PROGRAM STUDI SISTEM INFORMASI
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS HASANUDDIN
MAKASSAR
2024

SKRIPSI

IMPLEMENTASI PENGENALAN OBJEK REAL-TIME DENGAN FRAMEWORK YOLOv5 PADA PLATFORM CLOUD COMPUTING KUBERNETES KUBEFLOW BERBASIS DEEP LEARNING

FARHAN RAMDHANI
H071171527

Skripsi,

telah dipertahankan di depan Panitia Ujian Sarjana Sistem Informasi pada 31 Juli 2024 dan dinyatakan telah memenuhi syarat kelulusan

pada

Program Studi Sistem Informasi
Departemen Matematika
Fakultas Matematika dan Ilmu Pengetahuan Alam
Universitas Hasanuddin
Makassar

Mengesahkan:

Pembimbing Tugas Akhir,	Pembimbing Pendamping,
Dr.Eng. Armin Lawi, S.Si., M.Eng.	Edy Saputra Rusdi, S.Si., M.Si
NIP 197204231995121001	NIP 199104102020053001

Mengetahui:
Ketua Program Studi,

Prof. Dr. Jeffry Kusuma, Ph.D
NIP 196411121987031002

PERNYATAAN KEASLIAN SKRIPSI DAN PELIMPAHAN HAK CIPTA

Dengan ini saya menyatakan bahwa, skripsi berjudul "Implementasi Pengenalan Objek Real-Time dengan Framework YOLOv5 pada Platform Cloud Computing Kubernetes Kubeflow Berbasis Deep Learning" adalah benar karya saya dengan arahan dari pembimbing (Dr.Eng. Armin Lawi, S.Si., M.Eng. sebagai Pembimbing Utama dan Edy Saputra Rusdi, S.Si., M.Si sebagai Pembimbing Pendamping). Karya ilmiah ini belum diajukan dan tidak sedang diajukan dalam bentuk apa pun kepada perguruan tinggi mana pun. Sumber informasi yang berasal atau dikutip dari karya yang diterbitkan maupun tidak diterbitkan dari penulis lain telah disebutkan dalam teks dan dicantumkan dalam Daftar Pustaka skripsi ini. Apabila di kemudian hari terbukti atau dapat dibuktikan bahwa sebagian atau keseluruhan skripsi ini adalah karya orang lain, maka saya bersedia menerima sanksi atas perbuatan tersebut berdasarkan aturan yang berlaku.

Dengan ini saya melimpahkan hak cipta (hak ekonomis) dari karya tulis saya berupa skripsi ini kepada Universitas Hasanuddin.

Makassar, 19 Agustus 2024

Farhan Ramdhani
NIM H071171527

Ucapan Terima Kasih

Puji dan syukur penulis panjatkan ke hadirat Tuhan Yang Maha Esa atas segala limpahan rahmat dan karunia -Nya, sehingga atas izin -Nya penulis diberikan kesempatan dan kelancaran dalam menyelesaikan tugas akhir ini. Dengan berbagai kesulitan maupun rintangan yang dihadapi saat menyelesaikan tugas akhir ini, tidak lupa penulis mengucapkan terima kasih atas kontribusi dan bantuannya kepada:

1. Rektor Universitas Hasanuddin, Bapak **Prof. Dr. Ir. Jalamuddin Jompa, M.Sc.** beserta jajarannya.
2. Dekan Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Hasanuddin, **Dr.Eng. Amiruddin** beserta jajarannya.
3. Ketua Departemen Matematika Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Hasanuddin, Bapak **Prof. Dr. Nurdin, S.Si., M.Si.** atas seluruh ilmu, saran, dan masukan yang telah diberikan.
4. Ketua Program Studi Sistem Informasi, Bapak **Prof. Dr. Jeffry Kusuma, Ph.D.** atas seluruh ilmu, saran, dan masukan yang telah diberikan.
5. Pembimbing Utama sekaligus Penasihat Akademik penulis, Bapak **Dr.Eng. Armin Lawi, S.Si., M.Eng.** yang telah senantiasa memberikan bimbingan, arahan, dan bantuan ilmu yang tiada henti selama masa studi penulis.
6. Pembimbing Pendamping penulis, Bapak **Edy Saputra Rusdi, S.Si., M.Si.** yang telah senantiasa membantu, membimbing, dan memberikan arahan selama masa studi penulis.
7. Kedua Dosen Penguji, Bapak **Dr. Andi Muhammad Anwar, S.Si., M.Si.** dan Ibu **Rozalina Amran, S.T., M.Eng.**, yang telah memberikan kritik maupun masukan yang konstruktif dalam penelitian tugas akhir ini sehingga skripsi penulis dapat terselesaikan dengan baik.
8. Bapak/Ibu **Dosen Program Studi Sistem Informasi** beserta seluruh tenaga pendidik yang telah memberikan ilmu dan mendidik penulis selama masa perkuliahan. Serta kepada seluruh staf dan pegawai **Departemen Matematika** yang telah membantu penulis terutama dalam segala proses administrasi.
9. Kedua orang tua penulis, Bapak **Feby Agung** dan Ibu **Ernawati** yang selalu mendoakan, mencurahkan kasih sayang dan perhatian, memotivasi, menasihati, dan memberikan dukungan moril maupun materil. Begitu pula kedua saudara kandung penulis juga seluruh keluarga besar penulis.
10. Dengan penuh rasa terima kasih, penulis ingin menyampaikan penghargaan yang mendalam kepada **Ratu Salsabila Helmi**, yang telah menjadi sumber dukungan dan motivasi yang luar biasa. Kesabaran, perhatian, dan dorongannya menjadi penyemangat yang tidak ternilai di setiap waktu dan juga selama proses penyelesaian tugas akhir ini.
11. Seluruh teman-teman Program Studi Sistem Informasi, terutama **Sistem Informasi Angkatan 2017** dan rekan-rekan terdekat **POPAL** yang senantiasa membantu dan memberikan dukungan semasa perkuliahan penulis.

12. Seluruh pihak yang tidak dapat saya sebutkan satu per satu namun telah memberikan dukungan baik langsung maupun tidak langsung.

Semoga segala bantuan dan dukungan yang telah diberikan mendapat balasan yang setimpal dari Tuhan Yang Maha Esa. Semoga tugas akhir ini dapat memberikan manfaat kepada semua pihak yang membutuhkan dan terutama untuk penulis.

Makassar, 19 Agustus 2024

Farhan Ramdhani

ABSTRAK

FARHAN RAMDHANI. **Implementasi Pengenalan Objek Real-Time dengan Framework YOLOv5 pada Platform Cloud Computing Kubernetes Kubeflow Berbasis Deep Learning** (dibimbing oleh Dr.Eng. Armin Lawi, S.Si., M.Eng. dan Edy Saputra Rusdi, S.Si., M.Si.).

Latar belakang. Pengenalan objek secara *real-time* merupakan salah satu tantangan utama dalam bidang *computer vision* dan *deep learning*, terutama ketika diterapkan pada infrastruktur *cloud computing*. **Tujuan**. Penelitian ini berfokus dalam mengimplementasikan *framework* YOLOv5 dengan memanfaatkan layanan infrastruktur Kubernetes Kubeflow dan Amazon Web Services. **Metode**. Model dilatih menggunakan *dataset* COCO 2017 dan diintegrasikan dengan protokol RTSP untuk menerima masukan *video stream* dari perangkat *mobile*, yang kemudian diproses oleh model untuk mendeteksi objek secara *real-time* pada lingkungan *cloud*. **Hasil**. Model YOLOv5 dapat diimplementasikan pada infrastruktur *cloud computing* dan menunjukkan keberhasilan dalam mendeteksi objek dengan skor mAP sebesar 68,4% dan juga menunjukkan bahwa penggunaan GPU saat model dijalankan secara signifikan meningkatkan performa pendeteksian dibandingkan menggunakan CPU, terutama ketika model dijalankan pada resolusi *stream* yang rendah. **Kesimpulan**. Penelitian ini menunjukkan bahwa kombinasi *framework* YOLOv5, infrastruktur *cloud computing*, dan protokol RTSP dapat menghasilkan sistem deteksi objek yang efisien dan mempunyai skalabilitas sistem yang tinggi untuk berbagai aplikasi pendeteksian *real-time*.

Kata kunci: *computer vision*; YOLOv5; *framework* deep learning; infrastruktur *cloud*; COCO 2017; pemrosesan *real-time*

ABSTRACT

FARHAN RAMDHANI. **Implementation of Real-Time Object Recognition Using YOLOv5 Framework on a Kubernetes Kubeflow Cloud Computing Platform Based on Deep Learning** (supervised by Dr.Eng. Armin Lawi, S.Si., M.Eng. and Edy Saputra Rusdi, S.Si., M.Si.).

Background. Real-time object recognition is one of the major challenges in the field of computer vision and deep learning, especially when applied to cloud computing infrastructure. **Aim**. This research focuses on implementing the YOLOv5 framework by utilizing Kubernetes Kubeflow and Amazon Web Services as cloud services infrastructure. **Method.** The model was trained using the COCO 2017 dataset and integrated with the RTSP protocol to receive video stream input from mobile devices, which was then processed by the model to detect objects in real-time in a cloud environment. **Results.** The YOLOv5 model can be implemented on cloud computing infrastructure and demonstrated success in object detection with an mAP score of 68.4%. It also showed that using a GPU during model execution significantly improved detection performance compared to using a CPU, especially when the model was run at lower stream resolutions. **Conclusion.** This study shows that the combination of the YOLOv5 framework, cloud computing infrastructure, and RTSP protocol can produce an efficient object detection system with high scalability for various real-time detection applications.

Kata kunci: computer vision; YOLOv5; deep learning framework; cloud infrastructure; COCO 2017; real-time processing

DAFTAR ISI

Halaman

DAFTAR TABEL

Nomor urut Halaman

DAFTAR GAMBAR

Nomor urut Halaman

DAFTAR LAMPIRAN

Nomor urut Halaman

BAB I
PENDAHULUAN

1.1 Latar Belakang

Beberapa tahun belakangan, *Computer Vision* (CV) dan *Deep Learning* (DL) telah diaplikasikan secara luas dan masif untuk membantu manusia dalam melakukan otomatisasi maupun prediksi pada berbagai macam keperluan seperti *Optical Character Recognition* (OCR), deteksi objek, pengenalan wajah manusia; kendaraan *autonomous,* hingga terobosan untuk dunia medis. Sebagai hasilnya, kombinasi teknologi ini memperlihatkan hasil yang efektif (Khan et al., 2018), secara tidak langsung memicu perusahaan yang bergerak di bidang teknologi; khususnya *Artificial Intelligence* (AI) untuk memperluas ke dalam lingkup CV.

Berdasarkan data analisis tahun 2018 dengan lingkup proyeksi penelitian sampai tahun 2027, nilai pasar global untuk CV mencapai 9,28 miliar USD pada tahun 2017, dan diperkirakan akan melebihi 48,32 miliar USD untuk akhir tahun 2023 mendatang (Market Research Future, 2019). Cepatnya pertumbuhan bidang teknologi, hasil yang akurat, semakin murahnya harga perangkat teknologi; juga kemudahan dalam terkoneksi ke dunia digital, merupakan faktor utama yang membuat CV diadopsi dan digunakan dengan pesat secara global (Omdia Analyst, 2020). Pemain besar teknologi maupun *startup tech company* menggunakan teknologi CV untuk keuntungan komersial mereka dengan cara memadukan model DL dan koleksi citra digital kamera maupun video dari riset mereka pada berbagai kasus permasalahan di industri yang berbeda.

Riset terkait cabang AI dilakukan oleh (Voulodimos et al., 2018) yang meninjau kembali Deep Learning pada CV, mengatakan bahwa lompatan perkembangan DL dalam beberapa tahun belakangan merupakan langkah besar dalam perkembangan CV. Dalam jurnalnya, para penulis membahas tiga kategori pengembangan, *Convolutional Neural Networks* (CNNs), “Boltzmann Families" yakni *Deep Belief Networks* (DBNs) dan *Deep Boltzmann Machines* (DBMs), juga *Stacked denoising Autoencoders* (SdAs). Ketiganya telah banyak digunakan untuk mencapai performa yang signifikan dengan tugas yang berbeda-beda dalam histori perkembangan NN, mulai dari deteksi objek, pengenalan wajah; pengenalan aktivitas, estimasi pose manusia, perolehan citra; hingga segmentasi semantik. Meski demikian, tiap kategori mempunyai kelebihan dan kekurangan masing-masing, CNN memiliki kapabilitas untuk memahami pola dengan baik (LeCun et al., 1989; Lecun et al., 1998) dan melakukan *future learning*, yang secara otomatis mempelajari fitur yang ada pada *dataset*. Dalam penerapan CV, CNN juga memiliki keunggulan invarian terhadap transformasi data, namun jika dibandingkan dengan DBMs/DBNs atau SdA; CNN sangat bergantung pada *labeled data* (Voulodimos et al., 2018).

(Papageorgiou et al., 1998) membuat sebuah kerangka model untuk deteksi objek atas dasar sulitnya mendeteksi objek *real-world*; dengan dua *domain* seperti

wajah dan manusia. Keduanya menimbulkan beberapa tantangan, sulit untuk memodelkan objek-objek tersebut karena terdapat perbedaan atau variasi yang signifikan terhadap warna dan tekstur dari sebuah citra digital, sehingga membedakan antara objek yang akan dideteksi dengan benda lain di sekitarnya harus dilakukan (Papageorgiou et al., 1998). Kerangka model tersebut mendemonstrasikan penggunaan *wavelet* Haar yang merepresentasikan skema *learning* efektif untuk deteksi objek, dengan hasil deteksi mencapai 70% sampai 75%. (Zhao et al., 2019) melakukan analisis terhadap beberapa *framework* dan arsitektur DL untuk deteksi objek. (Zhao et al., 2019) menyebutkan, bahwa masalah yang didefinisikan dari deteksi objek sendiri adalah lokalisasi objek, yakni menentukan dimana lokasi objek dalam bidang dua dimensi *x* dan y suatu citra digital; serta klasifikasi objek atau termasuk ke kategori manakah objek tersebut (Harzallah et al., 2009).

(Zhao et al., 2019) mengkategorisasikan *framework* untuk deteksi objek ke dalam dua tipe, *region proposal-based* yang menerapkan *pipeline* deteksi objek seperti umumnya, *region proposals* (lokasi-lokasi yang memungkinkan untuk tiap target) dihasilkan terlebih dahulu kemudian mengklasifikasikannya ke dalam label objek; juga terdapat *regression*/*classification-based*, tipe ini memperlakukan deteksi objek sebagai tugas regresi/klasifikasi, juga mengadopsi skema kerangka kerja *unified* (Floridi & Cowls, 2019; Zhao et al., 2019). Salah satu *framework* yang berdasar pada tipe *regression/classification* adalah YOLO (Redmon et al., 2016), sebagai *one-step framework* yang memetakan kelas probabilitas maupun lokasi koordinat *bounding box* langsung dari piksel citra digital, sehingga dapat memangkas waktu pengklasifikasian.

YOLO memprediksi *bounding box* dan *confidences rate* untuk seluruh kategori dengan *feature map* paling teratasnya. Terdiri dari 2 lapisan *Fully Connected* (FC) dan 24 lapisan *convolutional*, dimana sebagian dari lapisan ini dibentuk sebuah modul *inception* diikuti dengan lapisan *convolutional* (Redmon et al., 2016). Ketika diuji secara *real-time*, *framework* ini dapat mencapai 45 *frame per second* (fps) hingga 155 FPS (Zhao et al., 2019). Namun YOLO mempunyai ikatan atau *constraint* spasial yang kuat dengan *bounding box*-nya, hal ini dapat dianggap sebagai *drawback* yang menyebabkan kesulitan saat mendeteksi objek dengan ukuran kecil yang berada diantara objek lainnya dalam sebuah grup. Setelahnya, *framework* ini dimutakhirkan dengan adanya YOLOv2 (Redmon et al., 2016), yang mengangkat beberapa metode baru; seperti *anchor boxes*, klaster dimensi, juga *training* model secara *multi-scale*.

Seiring perkembangan CV dalam lingkup AI, teknologi *Cloud Computing* (CC) juga berkembang secara paralel. Cepatnya pertumbuhan internet dari tahun ke tahun yang linear dengan canggihnya teknologi penyimpanan/*storage* telah mengubah pandangan terhadap perangkat keras dan perangkat lunak di bidang IT (Armbrust et al., 2010). Layanan seperti *Software-as-a-Service* (SaaS), *Platform-as-a-Service* (PaaS), *Infrastructure-as-a-Service* (IaaS) dari Amazon dan perusahaan *big tech* lainnya membuktikan keefektifan CC selama beberapa tahun terakhir (Marinescu, 2013). Faktanya, 2,6 dari 7,2 miliar orang telah memakai salah satu

layanan CC pada tahun 2015, yakni *electronic mail* atau *email* (Marinescu, 2013). Dari sisi *developer* sendiri, pemilihan *resource* dan komputasi untuk sebuah *project* telah dieliminasi oleh fleksibilitas dari CC (Zhang et al., 2010). *Server*, *storage*, *database*; struktur jaringan, analisis produk, dan kebutuhan lainnya dapat diatur secara mudah CC memungkinkan pengembang/*developer* untuk *scale-up* maupun *scale-down* sehingga tidak ada sumber daya/*resource* terbengkalai (M. A. Team, 2020); hal ini berdampak positif dari segi waktu maupun biaya, yang akan dikeluarkan untuk membangun sebuah infrastruktur karena *billing* infrastruktur dihitung dengan sistem "*pay-as-you-go*".

Kendati demikian, perlu diketahui bahwa infrastruktur CC tidak didirikan di atas komponen asli, melainkan dalam Virtual Machine (VM); untuk itu perhitungan dan pengukuran harus dilakukan sebelum membuat sebuah infrastruktur. Dengan banyaknya manfaat dalam penggunaan CC, teknologi ini belum terlalu "dewasa" untuk mencapai potensi maksimalnya, terdapat tantangan-tantangan untuk teknologi ini. Untuk penyedia layanan CC, keamanan pada sistem, sumber daya, dan konsumsi daya adalah hal yang harus terus ditingkatkan (Zhang et al., 2010).

(Wei & Blake, 2010) dalam penelitiannya, menjelaskan bahwa terdapat peluang yang dapat dibuka, seperti *rapid service deployment*; juga tantangan yang rumit dalam penggunaan CC seperti sistem layanan tingkat tinggi yang harus dipelihara secara berkala, menyediakan solusi keamanan secara *end-to-end*, dan menjaga *workflow* agar tetap berjalan. Seiring berjalannya dengan waktu, perkembangan CC dan AI secara konstan membuktikan bahwa keduanya merupakan kombinasi yang ideal (Ferkoun, 2014), memberikan layanan *cost effective*/ramah biaya dan *scalability* untuk pengembangan sebuah *project*, maupun infrastruktur dari suatu *stack* perangkat lunak.

Dengan paparan sebelumnya, penulis mempunyai ketertarikan terhadap *cloud computing* dan *artificial intelligence*, khususnya pada cabang *computer vision*, dan ingin melakukan penelitian untuk membuat sebuah karya mengenai hal tersebut; yakni "Implementasi Pengenalan Objek *Real-Time* dengan *Framework* YOLOv5 pada *Platform Cloud Computing* Kubernetes Kubeflow Berbasis *Deep Learning*".

1.2 Rumusan Masalah

Berdasarkan paparan pada latar belakang penelitian di atas, dapat dirumuskan masalah untuk penelitian ini:

1. Bagaimana membangun infrastruktur *cloud computing* untuk keperluan pendeteksian objek?
2. Bagaimana cara kerja komunikasi data dalam infrastruktur yang dibuat?
3. Bagaimana implementasi pendeteksian objek pada *platform cloud computing* dengan model YOLOv5?
4. Bagaimana kinerja dari pendeteksian objek yang diimplementasikan?

1.3 Batasan Masalah

Untuk mempersempit cakupan penelitian sesuai dengan maksud dan tujuan yang diinginkan, maka batasan masalah penelitian ini adalah sebagai berikut:

1. Merupakan masalah penelitian tentang penerapan *computer vision* dari *artificial intelligence*, khususnya cabang *deep learning*; yang berkaitan dengan *platform cloud computing*.
2. Pencapaian akhir penelitian merupakan klasifikasi objek atau deteksi objek secara *real-time*.
3. Model yang akan digunakan adalah YOLOv5.
4. Penelitian menggunakan *dataset* berupa kumpulan citra digital yang menggambarkan sebuah objek.
5. Proses penelitian menggunakan layanan Kubeflow Notebook dari *platform* Kubeflow, dan Amazon S3 dari *platform cloud computing* Amazon Web Service.
6. Untuk hasil yang akan dicapai, *input*-nya berupa *video stream* secara langsung dari perangkat *mobile* menggunakan protokol RTSP, dan *output*-nya adalah video dengan *bounding box* yang melekat pada objek yang dideteksi.

1.4 Tujuan Penelitian

Berdasarkan rumusan masalah penelitian, tujuan dilakukannya penelitian ini adalah:

1. Mengimplementasikan penggunaan teknologi *cloud computing* untuk *machine learning*, juga menjelaskan bagaimana infrastruktur tersebut dibuat.
2. Mendeskripsikan cara kerja komunikasi data dalam skema pendeteksian objek dengan teknologi *cloud computing*.
3. Mengimplementasikan penggunaan *computer vision*, khususnya dengan model *framework* YOLOv5 untuk melakukan deteksi objek secara *real-time*.
4. Mendeskripsikan hasil kinerja dari model yang dibangun pada infrastruktur *cloud computing*.

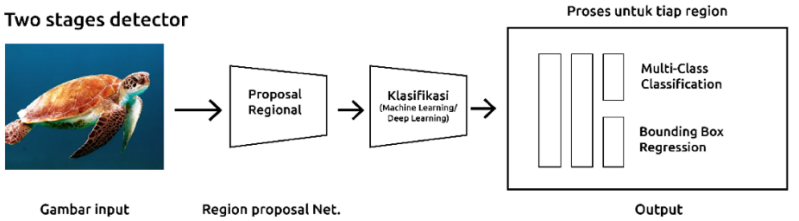
1.5 Teori

1.5.1 *Object Detection*

Deteksi objek merupakan proses mengenali objek yang berbeda-beda pada sebuah citra digital yang tergolong pada suatu label atau kategori. Prinsipnya adalah *object localization* atau mengetahui posisi/lokasi objek, dan *object classification* atau mengklasifikasikan objek tersebut ke dalam kategori label sesuai data *training*. Teknik dasar dari deteksi objek adalah membuat *sliding window* dengan berbagai ukuran $M \times N$ kemudian diiterasikan pada seluruh dimensi citra, sehingga semua lokasi *bounding boxes* yang memungkinkan dapat ditemukan. *Pipeline* dari

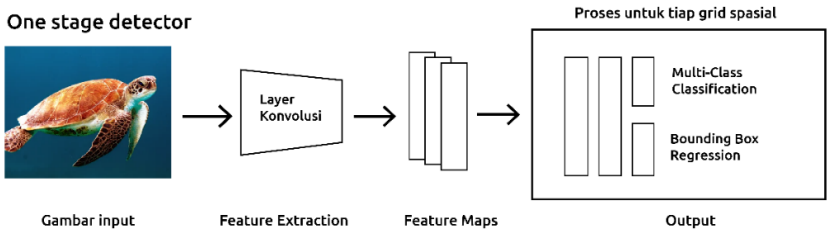
pendekteksian objek umumnya adalah: (i) pengumpulan proposal dari citra, (ii) ekstraksi fitur yang mendeskripsikan semua objek pada proposal *region*, dan (iii) klasifikasi objek dari proposal tersebut (Wu et al., 2020), namun *drawback* dari teknik tersebut adalah waktu komputasi yang tinggi karena kebutuhan iterasi pada saat pembangkitan proposal.

Hadirnya *neural networks* meningkatkan cara pendeteksian objek secara signifikan dengan membuat representasi *feature* dari sebuah citra digital dalam bentuk susunan bertingkat atau hierarki, dan hasil perkembangannya memungkinkan penggunaan koleksi data untuk menghasilkan representasi *feature* yang lebih baik (Zhao *et al.*, 2019), dibandingkan dengan cara pendeteksian objek biasa. *Output* dari pendeteksian objek berupa (a) *bounding box* persegi panjang ataupun (b) *instance segmentation* (tiap *pixel* dari suatu objek dibedakan dengan (Wu et al., 2020) objek lainnya). Tipe *framework* pendeteksian objek dengan *neural networks* dibagi ke dalam dua jenis, *one-stage detector* dan *two-stages detector*.



Gambar 1. *Workflow* dari *two-stages detector*.

Proses *two-stages detector* sebagian besar mengikuti proses pendeteksian objek secara tradisional, membuat pembangkitan proposal (*proposal generation*), mengekstraksi *feature* dari proposal, kemudian membuat prediksi kategori dari proposal tersebut dengan model *learning* yang digunakan. Namun untuk deteksi objek pada penelitian ini, yang terapkan adalah konsep *one-stage detector*.

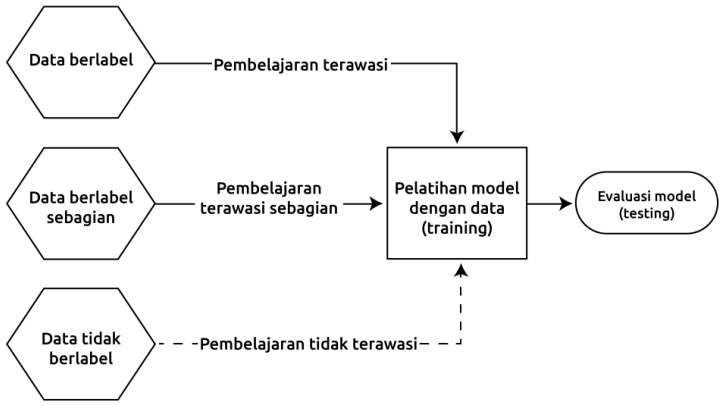


Gambar 2. *Workflow* dari *one-stage detector*.

1.5.2 *Deep Learning*

Selain menyelesaikan masalah analisis pada suatu data, algoritma mesin juga telah dikembangkan dari pendekatan maupun arsitekturnya untuk melakukan *problem solving* seperti tantangan *artificial intelligence* dengan beradaptasi dari repetisi pembelajaran secara dinamis (El Naqa dan Murphy, 2015; Jordan dan Mitchell, 2015). *Machine Learning* (ML) adalah proses *artificial* dimana suatu algoritma komputasi dikembangkan untuk dapat melakukan pembelajaran tanpa diprogram secara keseluruhan atau absolut dengan menggunakan data lampau yang telah ada, sehingga dapat melakukan analisis hasil pada data yang belum dipelajari oleh algoritma tersebut.

Dalam pembelajaran atau proses *training*-nya, terdapat beberapa tipe *learning* seperti (a) *supervised,* (b) *unsupervised,* dan (c) *semi-supervised*. *Supervised learning* memanfaatkan data *input* maupun data *ouput* (*training data*) dan algoritma (regresi, *active learning*, atau klasifikasi) (Alpaydin, 2020). Model *learning* yang telah melakukan *training* dapat diuji performanya dengan pengetesan pada data uji (*data testing*), dengan tujuan akhir melihat kinerja dalam memprediksi sebuah *set* atau koleksi data baru secara akurat.



Gambar 3. Tipe pembelajaran *machine learning*.

Algoritma yang termasuk dalam tipe *unsupervised learning* cenderung berbeda dengan tipe *learning* sebelumnya. Data proses *learning* algoritma *unsupervised* hanyalah berisikan *input*, tanpa *output*/label dari variabel independennya. Dengan kata lain, algoritma ML bertipe *unsupervised* mempelajari dan/atau menghasilkan serangkaian pola sebagai *neural predilections* atau kepadatan densitas dari pembelajarannya (Celebi dan Aydin, 2016; Hinton dan Sejnowski, 1999). Terdapat pula tipe pembelajaran *semi-supervised* yang bersifat *hybrid* atau penengah dari *supervised learning* dan *unsupervised learning* (Chapelle et al., 2010; Zhu, 2008)

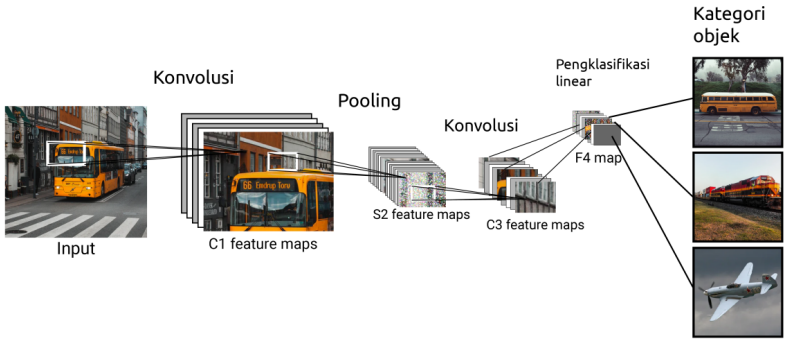
yang muncul karena sulitnya algoritma model *supervised learning* dalam menganalisis hanya dengan sedikit *labeled data*. Konsepnya sendiri, tipe *learning* ini memungkinkan untuk memproses *unlabeled data* dengan kombinasi dari *labeled data* yang lebih sedikit (van Engelen dan Hoos, 2020).

Dalam pembahasan ML, terdapat cabang metode untuk mempelajari data yang dinamakan *Deep Learning* (DL). Istilah "*deep*" mengindikasikan banyaknya penggunaan tingkatan representasi atau *layer* pada jaringan pembelajarannya, dimulai dari pembentukan komposisi modul *non-linear* dari layer terbawah (*raw input*) yang kemudian ditransformasikan pada tiap lapisan*/layer*, dan berakhir pada tingkatan *layer* teratas yang cenderung lebih abstrak (LeCun et al., 2015). Dengan kata lain, DL memanfaatkan gabungan *layer* pada jaringannya, dan secara kontinu mengekstrak fitur (reduksi dimensi) dari inputannya dengan level abstraksi yang berbeda-beda (Deng, 2014).

Dalam mengelola *raw data*, metode ML yang konvesional membutuhkan ketelitian dalam pengembangannya agar dapat membaca pola agar dapat mengekstrak fitur, yang membutuhkan keahlian dari segi *engineering*-nya sendiri (LeCun et al., 2015). *Base* dari algoritma DL pada umumnya seperti *Convolutional Neural Network* (CNN) dibuat berdasarkan *Artificial Neural Network* (ANN), terdiri dari *artificial neuron* yang mampu berkomunikasi dengan *neuron* lainnya, konsepnya sendiri ialah menirukan dan menstimulasikan cara kerja otak manusia atau otak biologis. DL memanfaatkan transformasi dimensi atau matriks dalam prosesnya, misalkan pada hal yang berkaitan dengan citra digital, masukkannya berupa koleksi *pixel* dalam bentuk matriks; kemudian *layer* selanjutnya melakukan manipulasi (*encoding* atau transformasi) pada *edges* dari sebuah citra, dan sedemikian sehingga *top level layer*-nya memproses "*cooked" data*. Menurut (Y. E. Wang et al., 2019), secara paralel kemajuan teknologi seperti unit pemrosesan grafik berpengaruh penting dalam perkembangan DL karena memungkinkan model algoritma untuk memanipulasi matriks secara efisien.

1.5.3 *Convolutional Neural Network*

Convolutional Neural Network (CNN) merupakan sebuah *neural system* yang tidak menggunakan operasi multiplikasi matriks biasa, melainkan dengan pendekatan operasi *linear* matematis yaitu konvolusi, yang menggabungkan dua informasi yang berbeda. CNN dibangun dengan ide dari sebuah model *neocognitron* oleh (Fukushima, 1980) yang menirukan struktur biologi otak untuk memproses fitur berbentuk tingkatan/*hierarchy* (Wu et al., 2020), kemudian dikembangkan ke dalam bentuk *neural network* yang secara khusus didesain agar dapat memproses data dalam bentuk koleksi *array*; seperti pada citra digital yang yang terdiri tadi tiga lapis array dua dimensi yang mewakili kanal warnanya (LeCun et al., 2015).

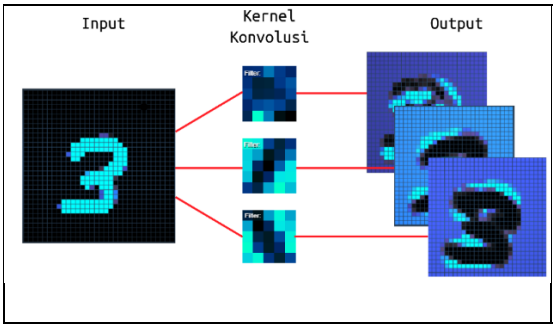


Gambar 4. Arsitektur CNN untuk pengenalan citra. Terdiri dari tiga layer utama, Convolutional (a), Pooling (b), dan diakhiri dengan top level layer yaitu Fully Connected (c).

Setiap *layer* mentransformasikan *input* awal ke dalam bentuk *output* untuk *layer* selanjutnya, sedemikian sehingga ketika telah melewati FC, *input data* dirubah ke dalam vektor *feature* berbentuk satu dimensi (Voulodimos et al., 2018). *Convolutional* layer merupakan bagian *layering* utama dari arsitektur CNN, terdapat kumpulan *filter* atau *kernel* yang dipakai untuk melakukan proses konvolusi terhadap dimensi tinggi dan lebar dari suatu data. Tiap *filter* menghasilkan matriks *feature map* atau *activation map* berdimensi dua yang merupakan hasil *dot product* dari fungsi nilai *filter* atau *kernel* dan fungsi nilai data yang di-*input*. Konvolusi dapat didefinisikan sebagai

$$y(t) = f(t) \cdot x(t) = \int_{-\infty}^{\infty} f(k) \cdot x(t-k)dk \qquad (1)$$

dimana $y(t)$ hasil konvolusi, $f(t)$ & $x(t)$ merupakan fungsi nilai input dan fungsi kernel, dan k adalah variabel integralnya.



Gambar 5. Konvolusi pada sebuah citra. Proses konvolusi bagian citra dengan tiga macam kernel berdimensi 5×5.

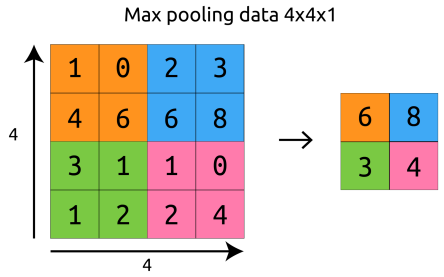
Sebelum sebuah *layer* menghasilkan *activation map* untuk *layer* selanjutnya, terdapat fungsi *non-linear activation* yang mendefinisikan sinyal *output* dari hasil

konvolusi, fungsi *non-linear activation* yang biasa digunakan adalah Linear, ReLU, Heaviside, Logistic, Tanh, PReLU, Sigmoid, dan ELU (Dhillon dan Verma, 2020). Dalam arsitektur CNN, ReLU dipakai sebagai fungsi untuk mengaktivasi nilai positif dari suatu *input*, yang dinotasikan sebagai

$$f(x) = x^{+} = max(0, x) \tag{2}$$

sinyal *output* dari fungsi aktivasi diteruskan ke layer selanjutnya seperti *pooling layer* atau *convolutional layer* lainnya.

Pooling layer berguna untuk mereduksi dimensi spasial dari *layer* sebelumnya ke *convolutional layer* selanjutnya, sehingga *layer* ini juga dapat disebut *non-linear subsampling* atau *down-sampling*. Pada *layer*, ini dilakukan pengurangan jumlah parameter, kuantitas komputasi, dan *footprint*; pengurangan yang dilakukan berguna agar model CNN tidak *overfitting*. *Layer* ini diletakkan setelah ReLU *layer* sebagai aktivator dari *convolutional layer*, dan bertujuan mengefiensiensikan waktu kalkulasi dan komputasi pada *convolutional layer* setelahnya (Géron, 2019).



Gambar 6. Proses *pooling* pada sebuah citra *grayscale*. Proses *pooling* menggunakan metode *max-pooling* dengan filter 2×2.

Metode *pooling* yang digunakan CNN adalah *max pooling*, dimana dalam suatu *pool* (kelompok matriks) direduksi dengan cara mencari nilai paling besar. Metode *max pooling* biasanya dilakukan dengan *filter* berukuran 2×2, dengan 2 langkah atau *stride*; sehingga menyisakan 25% dari hasil *activation*.

Fully Connected (FC) *layer* berfungsi untuk membuat prediksi dari proses konvolusi dan *pooling* sebelumnya, dan layer ini terhubung ke seluruh hasil dari *activation map*. Hasil *activation map* kemudian ditransformasikan secara geometris dengan metode affine melalui perkalian matriks. Hasilnya *feature map* berdimensi dua akan ditransformasikan ke dalam *feature vector* berdimensi satu, yang kemudian digunakan sebagai data input pada proses selanjutnya atau untuk dilabelkan kategori klasifikasinya (Girshick et al., 2013; Krizhevsky et al., 2017).

1.5.4 You Only Look Once (YOLO)

(Redmon et al., 2016) memperkenalkan *framework neural network* (NN) YOLO yang dapat melakukan komputasi *feature* pada sebuah citra dan melakukan prediksi pengklasifikasian objek dalam waktu yang bersamaan. Model ini membuat *feature vector* setelah citra atau *input* yang memuat objek melewati rangkaian proses konvolusi dari NN.

Konsep dari YOLO sendiri adalah mengaplikasikan *tools* berupa *grid* dengan ukuran $S \times S$ pada citra, kemudian *grid* yang memuat poin objek melakukan tugas pendeteksian objek dengan membuat *bounding box* dari parameter dan *confidence score* dari *grid* tersebut (Thatte, 2020). *Confidence score* dihitung sebagai

$$confidence\ score = p(obj) \times IoU_{pred}^{truth} \tag{3}$$

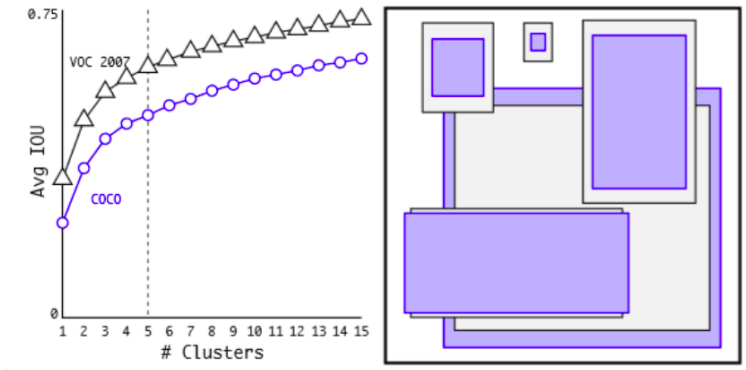
dengan *p(obj)* merupakan probabilitas bahwa terdapat objek di dalam *bounding box* (dengan range 0-1), dan *IoU* merupakan titik potong antara hasil prediksi dan nilai sebenarnya pada *bounding box*. Terdapat lima parameter *bounding box* dalam model YOLO: (a) koordinat x, (b) koordinat y, (c) lebar *b-box*, (d) tinggi *b-box*, dan (e) *confidence score;* dan output dari *grid cell* saat evaluasi model YOLO pada *dataset* adalah

$$S \times S\ * (5 \times B + C) \tag{4}$$

dimana *B* adalah jumlah *b-box* yang terdapat pada *grid cell*, dan *C* adalah jumlah label kelas yang tersedia untuk klasifikasi.

Kemudian *framework* ini menerapkan metode *batch normalization* untuk meningkatkan generalisasi dan efisiensi waktu yang digunakan saat model melakukan *training*. *Batch normalization* merupakan metode yang menormalisasikan distribusi dari data *input* (Loffe dan Szegedy, 2015) sehingga *output feature* dari *layer* aktivasi dalam keadaan *zero-mean* dan dengan standar deviasinya bernilai 1.0 (Thuan, 2021).

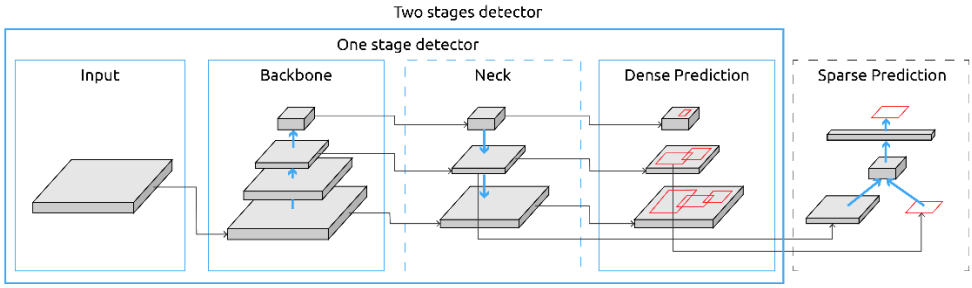
Selain *batch normalization*, terdapat *high resolution classifier* dan metode *anchor boxes*. *High-res classifier* meningkatkan *mean average precission* (mAP) dengan memanipulasi *feature extractor* dari input berupa citra dengan jumlah *epoch* lebih dari 10. Hal tersebut bertujuan agar model dapat melakukan adaptasi dengan ukuran yang lebih besar saat fase pelatihan *feature extractor* beralih ke fase pelatihan *object detector* (Liu et al., 2020; Thuan, 2021).



Gambar 7. Benchmark IoU dari penggunaan K-means Clustering. *Benchmarking* dilakukan dengan *dataset* COCO dan VOC 2007 pada YOLOv2.
Sumber: dikelola dari Redmon et al. (2016).

Pada YOLOv2 (Redmon et al., 2016) juga menerapkan arsitektur *anchor boxes* untuk mengatasi celah dimana versi sebelumnya sulit untuk mengenali beberapa objek di dalam *grid cell* yang sama. Konsep dari *anchor boxes* adalah tiap membuat *box* dari sebuah objek dengan berbagai ukuran yang telah ditetapkan. *Anchor boxes* ditetapkan dengan algoritma k-means *clustering* yang perhitungan jaraknya tidak menggunakan *euclidean distance*, melainkan dihitung dari IoU dari *centroid* dan *box* sebenarnya (*ground truth boxes*).

(Bochkovskiy et al., 2020) menekankan konsep *one-stage detector* dan *two-stages detector* yang dipakai YOLO, komponennya terbagi atas: (a) *input*, (b) *backbone*, (c) *neck*, (d) *head* (*dense prediction* & *sparse prediction*).



Gambar 8. Konsep *one-stage detector* & *two-stages detector*.

Masukan (*feature* pada citra) akan dikompres oleh *feature extractor*-nya (*backbone*), kemudian komponen *neck* berfungsi sebagai kolektor atau pengumpul *feature* (*feature aggregator*) dari proses yang ada di *backbone* untuk diteruskan ke

komponen *head* sebagai *predictor* pendeteksian objek. *Dense predictor* melakukan lokalisasi dan klasifikasi pada setiap *bounding box* objek dalam waktu yang bersamaan. *Sparse predictor* melakukan kedua proses tersebut dalam fase yang berbeda, kemudian dikombinasikan kembali setelahnya (Thuan, 2021).

1.5.5 YOLOv5

Ultralytics LLC (Jocher et al., 2021) mengembangkan dan melakukan konversi semua model YOLO yang sebelumnya ditulis dalam bahasa C; ke dalam framework PyTorch dan dilabelkan versi terbaru, yakni YOLOv5. YOLOv5 memicu banyak perbincangan karena waktu rilisnya hanya berjarak kurang dari dua bulan setelah YOLOv4. Faktor lainnya adalah kemungkinan YOLOv5 dan YOLOv4 dikembangkan dengan mengaplikasikan *state-of-the-art* teknologi yang hampir sama.

Pengembangan YOLOv5 yang dilakukan dengan bahasa Python yang terintegrasi dengan *framework* PyTorch memberikan banyak keuntungan dibanding versi sebelumnya yang memakai bahasa C dengan arsitektur DarkNet. Sifat *open source* dalam pengembangannya berdampak positif pada sektor *Internet of Things* (IoT) jika ingin melakukan implementasi yang berhubungan dengan *computer vision*.

Tabel 1. Arsitektur *one-stage detector* YOLOv5.

Komponen	Arsitektur
Backbone	Focus Structure, CSP Network
Neck	SPP Block, PANet
Head (Dense Predictor)	YOLOv3 head + GIoU-loss function

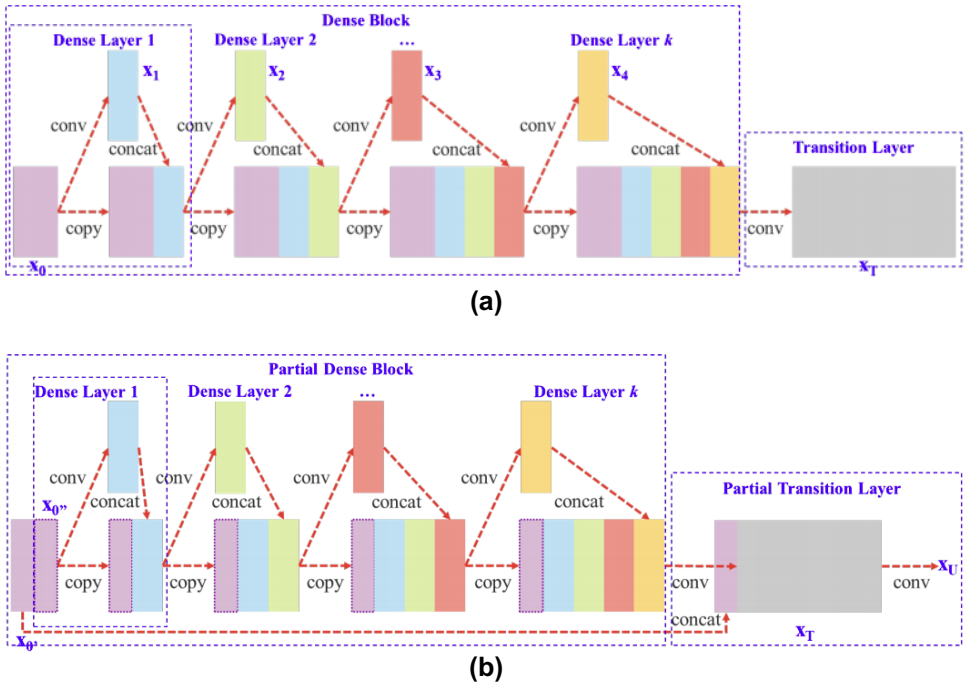
Data diolah kembali dan dimodifikasi dari Jocher (2020).

Perbedaan signifikan terdapat pada penggantian komponen *backbone* yang memakai arsitektur CSP *Network*. Kemudian pada bagian *head* dikombinasikan dengan metode *loss function* yakni *General Intersect over Union* (GIoU), yang merupakan alternatif generalisasi baru pada *metric* IoU, untuk membandingkan dua *shape* deteksi objek yang saling berisisan.

Selain itu, YOLOv5 juga mengintegrasikan proses seleksi *anchor-boxes* yang dapat mempelajari *anchor-boxes* terbaik saat proses *training* tanpa memperhatikan dataset yang digunakan (Solawetz, 2020). Berbeda dengan konsep *anchor-boxes* pada YOLOv2 dengan menggunakan algoritma k-means untuk menetapkan *anchor-boxes*, yang dapat menyebabkan kelas objek selain yang ada pada dataset *default* pelatihan model tidak dapat diprediksi.

Mengikuti konsep *one-stage detector*, *backbone* YOLOv5 menggunakan arsitektur *Cross Stage Partial* (CSP), yang dikembangkan dari arsitektur DenseNet; yang

memakai metode *concatenation* (rangkaian gabungan) pada input sekarang dan input dari *layer* sebelumnya (Huang et al., 2016).

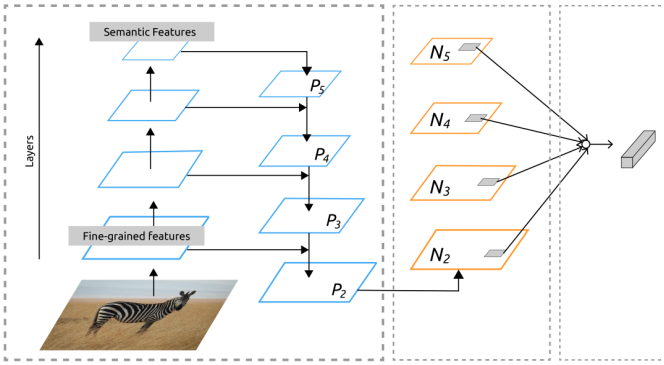


Gambar 9. *Block* dan *layering* pada arsitektur DenseNet dan CSPDenseNet. DenseNet (a) dan CSPDenseNet (b), warna merepresentasikan hasil proses tiap lapisan layer *densing*.

DenseNet terdiri dari *dense block* dengan tiap *block* berisi *k*- lapis *dense layer*, diawali dengan *input* yang dikonvolusikan dengan *kernel*; kemudian hasil konvolusi dari *input* awal akan di-*concatenate* dengan *input* mula-mula, hasil *concatenation* inilah yang menjadi *output* dari *dense layer*. *Dense layer* selanjutnya menggunakan metode yang sama, dengan *input*-nya berupa *output* dari *dense layer* sebelumnya (Gambar 9 bagian a). Namun untuk CSP-DenseNet, arsitektur memakai hasil *concatenation* dari *input* awal dengan *output* dari *dense block*. Hasil proses *concatenation* tersebut dibagi ke dalam dua bagian, bagian pertama digunakan untuk masukkan *dense block* selanjutnya, dan bagian kedua akan langsung diteruskan ke tahapan lain (Gambar 9 bagian b).

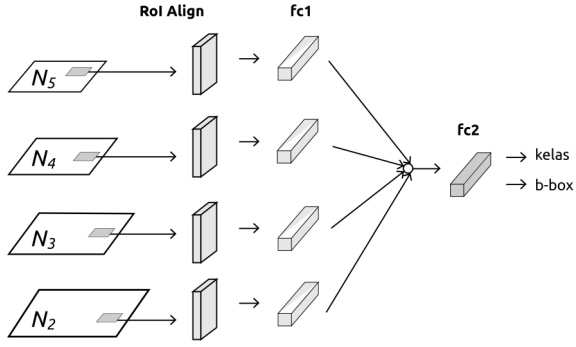
Melalui kombinasi CSP-DenseNet dan DarkNet53, YOLOv5 dapat menjaga fitur *fine-grained* untuk dilanjutkan ke dalam *layer* terdalam dari *neural network* ini. Ide awal penggabungan ini adalah menggantikan *residual block* dari DarkNet53 dengan *dense block*, namun jumlah *dense block* yang terlalu banyak dapat menyebabkan bertambah lambatnya kecepatan saat pendeteksian objek. Dari permasalahan

tersebut, penggantian *block* menjadi *dense block* hanya diterapkan pada *layer* konvolusi terakhir yang mengekstraksi *semantic feature*.



Gambar 10. Arsitektur PANet. Ilustrasi arsitektur PANet untuk segmentasi data dengan memproses informasi spasial.

Selanjutnya pada komponen *neck*, YOLOv5 menggunakan arsitektur *Spational Pyramid Pooling* (SPP) dan *Path Aggregation Network* (PANet) sebagai pemroses *additional block* maupun *path aggregation*-nya. SPP bertujuan untuk menghasilkan dimensi *output* yang konstan, tanpa memperhatikan dimensi citra yang diproses. Proses SPP sendiri bekerja dengan melakukan ekstraksi *n* macam fitur penting yang berbeda menggunakan *max-pooling* dengan *kernel pooling* yang berbeda, sehingga secara tidak langsung juga meningkatkan efisiensi YOLOv5 pada proses pelatihan atau *training* (C.-Y. Wang et al., 2019).



Gambar 11. *Region of Interest* pada augmentasi *bottom-up* PANet. Proses aligment Region of Interest (RoI) dilakukan pada tiap feature map untuk mengestrak feature pada objek.

PANet sebagai *aggregator* bertujuan untuk menjaga fitur *fine-grained* agar tetap konstan pada *hyper-parameter* awal. Hal tersebut berhubungan dengan permasalahan berkurangnya informasi spasial yang disebabkan oleh pemrosesan *semantic feature* (Thuan, 2021). PANet digunakan sebagai pengganti *Feature Pyramid Network* (FPN) yang digunakan versi YOLO sebelumnya; yang menghasilkan keluaran atau *output semantical feature* dengan alur metode *top-down* dan kemudian melakukan konkatenasi dengan *fine-grained feature* untuk kemampuan model memprediksi objek kecil.

Pada versi YOLOv4 keatas, modifikasi yang terjadi adalah penambahan *flow* augmentasi secara *bottom-up* (Gambar 10), sebagai tambahan dari metode *top-down* pada FPN dan menggunakan operasi *concatenation* pada tiap koneksinya (Thuan, 2021). Pada tiap level layer FPN, prediksi objek dilakukan secara independen dan terpisah, hal ini bisa saja menyebabkan duplikasi prediksi juga mengabaikan informasi *feature-map* lain. Sedangkan PANet menggabungkan semua *output* dari layering augmentasi *bottom-up*-nya dengan metode RoI (Gambar 11) dan operasi *element-wise* nilai terbesar.

Berbeda dengan komponen *head* pada model *two-stages detector* yang melakukan *sparse prediction*, komponen *head* model *one-stage detector* berisikan arsitektur untuk melakukan *dense prediction*. Prediksi yang dibuat pada tahap komponen ini adalah sebuah vektor yang memuat elemen *bounding box*, seperti (a) titik koordinat bounding box, (b) *confidence score*, (c) dan kelas probabilitasnya (Thuan, 2021). YOLOv5 menggunakan isi komponen *head* yang sama dengan YOLOv4 atau pun YOLOv3, yakni tahap perincian yang sama dan deteksi dengan metode *anchor-based boxes* (Solawetz, 2020).

1.5.6 Kubeflow dan Amazon Web Service

Kubernetes Kubeflow dan Amazon Web Service (AWS) adalah beberapa *platform open source* yang menyediakan layanan *Cloud Computing* (CC) yang berfokus pada *Machine Learning* (ML) *application workflows*. Kubeflow sendiri mempunyai kelebihan dalam: 1) *Deployment* model secara *portable*, mudah, dan dapat diulang-ulang pada model infrastruktur yang berbeda, 2) *Handling & deployment* layanan mikro-servis *loosely-coupling* (satu komponen tidak terasosiasi dengan kuat dengan komponen lainnya, tetapi tetap memengaruhi kinerja satu sama lain), 3) Skalabilitas model dan *resource* sesuai permintaan.

Sistem CC terbagi ke dalam tiga kategori: 1) *Infrastructure as a Service* (IaaS), 2) *Platform as a Service* (PaaS), dan 3) *Software as a Service* (SaaS). IaaS menyediakan fleksibilitas tertinggi untuk layanan CC, karena mencakup layanan infrastruktur dasar seperti *virtual computer* atau *dedicated hardware* dan layanan ruang untuk penyimpanan data. PaaS mencakup layanan selain infrastruktur dasar pada tingkat perangkat lunak, layanan yang disediakan pada tipe PaaS biasanya berfokus pada *deployment project*. Sedangkan SaaS dapat diartikan sebagai *micro-services* atau *end-user applications*, dengan kata lain adalah aplikasi yang

disediakan langsung dari penyedia; pengguna hanya difokuskan pada pemakaian aplikasi yang digunakan, sedangkan hal terkait infrastruktur ditangani oleh penyedia layanan CC.

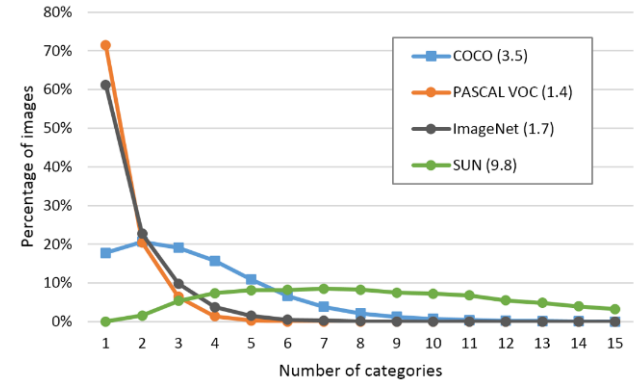
Kubeflow. Kubeflow merupakan layanan CC dibangun di atas sistem Kubernetes yang berfungsi untuk membuat sebuah model ML yang mengutamakan skalabilitas dan portabilitas dengan *platform* CC lain, dengan tujuan simplifikasi *tasks* ML yang dibangun. Beberapa *key feature* yang bisa didapatkan dalam layanan ini mencakup ML *modeling*, automatisasi *workflow* atau *pipelines*, aplikasi *monitoring* dan *logging* model seperti Tensorboard, juga fitur untuk melakukan *hyperparameter tuning* pada model (Bisong, 2019).

AWS Simple Storage Service. Amazon Simple Storage Service (S3) menyediakan layanan *storage* atau penyimpanan data berbasis *cloud* untuk kepentingan personal sampai tingkat industri. S3 didesain untuk berbagai penggunaan penyimpanan seperti *website storage*, aplikasi *mobile*, pencadangan data, pengarsipan, hingga penggunaan *storage* lainnya. Amazon S3 menerapkan pengaturan kontrol akses yang dapat dimodifikasi sesuai *system requirements* dari sebuah infrastruktur, selain itu juga dibangun untuk mencapai 99.9% *up-time* atau *availability* (Mathew, 2021).

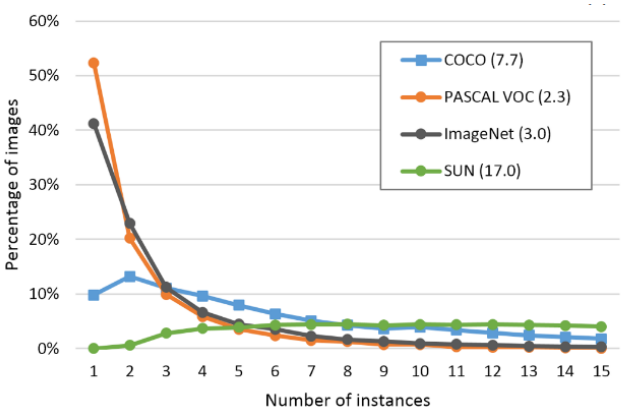
1.5.7 Common Object in Context (COCO) Dataset

COCO Dataset (Lin et al., 2014) adalah sebuah *dataset* yang berisi kumpulan data citra yang dirancang untuk berbagai task CV, terutama dalam *object detection* dan segmentasi citra. Dataset ini mencakup objek-objek yang sering terlihat dalam kehidupan sehari-hari (*common objects*) maupun yang lebih jarang ditemui (*uncommon objects*). Dengan total 91 kategori objek yang berbeda, setiap citra dalam dataset COCO disertai dengan *caption* yang mendeskripsikan objek-objek tersebut, memberikan konteks tambahan untuk proses pelatihan model. Dalam penelitian ini, digunakan versi terbaru dari dataset tersebut, yaitu COCO 2017.

COCO 2017 Dataset digunakan untuk *training*, *testing*, dan validasi model YOLOv5 yang dikembangkan dalam penelitian ini. Dataset ini dibagi ke dalam tiga bagian utama. Data *training* mencakup 72% dari total *dataset*, digunakan untuk melatih model mengenali pola-pola objek di dalam citra. Sementara itu, 25% dari dataset dialokasikan untuk data *testing*, yang akan digunakan untuk mengevaluasi kinerja model setelah pelatihan. Sisanya, 3% dari dataset digunakan sebagai data *validation*, yang bertujuan untuk memvalidasi hasil pelatihan dan membantu dalam penyetelan hyperparameter guna meningkatkan performa model.



(a)



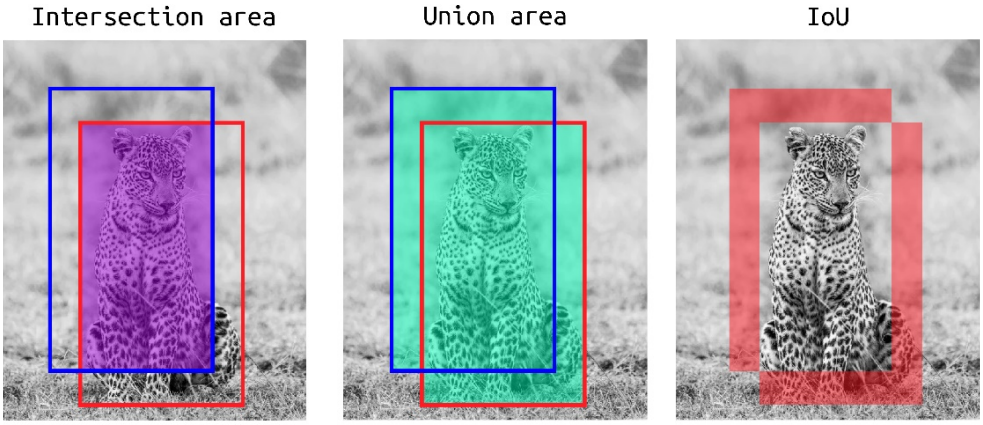
(b)

Gambar 12. Perbandingan MS-COCO dengan *dataset* lain. Persentase jumlah kategori pada setiap citra dalam dataset (a) dan persentase instansi (benda yang dapat diidentifikasi) pada setiap citra (b).
Sumber: diolah dari Lin et al. (2014).

1.5.8 Ukuran Kinerja

Dalam konteks pengukuran performa pendeteksian objek, digunakan konsep *Intersection over Union* (IoU) dan *mean Average Precision* (mAP). IoU mengukur seberapa tepat *prediction bounding box* dari hasil prediksi meng-*overlap* posisi *ground truth bounding box* dengan skala 0.0 (*completely not overlaps*) sampai 1.0 (*completely overlaps*).

$$IoU = \frac{Intersection\ area}{Union\ area} \tag{5}$$



Gambar 13. Ilustrasi *intersect* dan *union area* pada sebuah citra. Kotak ber-*outline* biru: *prediction bounding box*, (ii) kotak ber-*outline* merah: *ground truth bounding box*. Kotak transparan ungu & cyan adalah *intersect area* & *union area*, kotak transparan merah adalah area IoU.

IoU dimanfaatkan untuk mendapatkan metrik *precision* dan metrik *recall* dari dalam *confusion matrix*. *Precision* dan *recall* nantinya diolah kembali menjadi mAP.

$$Precision = \frac{TP}{TP + FP} \tag{6}$$

$$Recall = \frac{TP}{TP + FN} \tag{7}$$

$$mAP = \frac{1}{11} \sum_{Recall_i} Precision(Recall_i) = 1 \tag{8}$$

dengan mAP membagi nilai *recall* menjadi 11 poin bagian sama rata. Kemudian nilai *True Positive* (TP), *True Negative* (TN), *False Positive* (FP), dan *False Negative* (FN) diperoleh dari mengukur nilai *threshold* IoU.

Tabel 2. Pengukuran *confusion matrix* dalam konteks IoU.

Pengukuran	Kondisi
True Positive (TP)	Jika nilai IoU ≥ 0.6 (mendeteksi objek dengan benar)
False Positive (FP)	Jika nilai IoU < 0.6 (salah mendeteksi)
True Negative (TN)	Jika tidak ada objek yang dideteksi dan tidak ada *ground truth box* untuk objek tersebut
False Negative (FN)	Jika tidak ada *prediction box* yang dihasilkan namun sebenarnya terdapat *ground truth box*

Keterangan: diasumsikan *threshold* dari contoh pengukuran *confusion matrix* IoU diatas adalah 0.

1.6 Penelitian Terkait

(Loey et al., 2021) membuat sebuah model *deep learning* pendeteksi pemakaian masker medis dalam sebuah citra digital. Modelnya sendiri terdiri dari dua komponen, ResNet-50 sebagai *feature extractor*, dan YOLOv5 untuk deteksi maskernya. Estimasi *anchor boxes* dari model ini dilakukan dengan perhitungan mean IoU untuk meningkatkan proses deteksi objeknya. Performa model yang dipresentasikan mendapatkan skor presisi 81% dengan dua *dataset* yang berisi kumpulan citra wajah memakai masker (Loey et al., 2021).

Penelitian oleh (Heo et al., 2020), membahas tentang sistem deteksi objek dengan *multi-path neural network*, dengan dasar model *worst-case execution time* (WCET) pada sebuah *Graphic Processing Unit* (GPU). *Neural network* ini didesain sehingga dapat memilih jalur eksekusi sendiri untuk menyesuaikannya terhadap *constraint waktu* yang diberikan, karena terdapat operator pembangkit proposal dinamis dan operator seperti *switch* dan *skip* (Heo et al., 2020). Dalam mendeteksi objek secara *real-time*, model WCET memperoleh hasil prediksi *error* sebesar 28% pada *worst-case execution latency* dan 81% pada layer normalisasi berkelompok atau *group normalization layers*.

Tabel 3. *State-of-the-art* deteksi objek.

Judul	Metode & dataset	Hasil
Rich feature hier-archies for accurate object detection and semantic segmentation. (Girshick et al., 2014)	Deep VGG16, R-CNN & PASCAL VOC 2012	Dalam mendeteksi objek, mendapatkan nilai *mean Average Precision* sebesar 65,7%
Recognition, object detection and segmentation of white background photos based on deep learning. (Ning et al., 2017)	Faster R-CNN, Region Proposal Network & VOC 2012 Dataset	Dalam melakukan segmentasi pada citra dengan layar putih, skor model mencapai 96%
The Object Detection Based on Deep Learning. (Tang et al., 2017)	DNN, R-CNN & ILSVRC 2012	Mendapatkan skor akurasi 96% dalam mendeteksi objek pada dataset testing
Field Effect Deep Networks for Image Recognition with Incomplete Data. z(Zhong et al., 2016)	DL Model & MNIST, CIFAR-10, BioID face dataset	Skor *max average precision* sebesar 98,92% pada proses pengenalan citra dengan data tidak komplit
Self Paced Deep Learning for Weakly Supervised Object Detection. (Sangineto et al., 2019)	Faster R-CNN & ILSVRC, Pascal VOC 2007/2010	Hasil pendeteksian objek mencapai skor akurasi sebesar 12,13%
Enhancing iris authentication on handheld devices using deep learning derived segmentation techniques. (Bazrafkan et al., 2018)	Deep Learning & CASIA Thousand, Bath 800 datasets	Hasil segmentasi citra digital sebesar 98,55 untuk autentikasi atau deteksi iris pada sebuah perangkat
Deep Regionlets for Object Detection. (Xu et al., 2018)	DCNN, Deep regionlets & PASCAL VOC dan Microsoft	Mencapai skor *mean Average Precision* sebesar 82% untuk pendeteksian objek

Keterangan: data diolah kembali dan dimodifikasi dari (Dhillon dan Verma, 2020).

BAB II
METODE PENELITIAN

2.1 Waktu dan Tempat Penelitian

Penelitian ini dilaksanakan terhitung dari perencanaan penelitian, pelaksanaan penelitian, sampai pembuatan laporan lengkap mengenai penelitian yang dirincikan dengan alur waktu sebagai berikut:

Tabel 4. Alur waktu penelitian.

Uraian Kegiatan	2023			2023		
	OKT	NOV	DES	JAN	FEB	MAR
Persiapan penelitian						
Pelaksanaan penelitian						
Pembuatan model						
Pelatihan model						
Pengujian model secara realtime						
Pengolahan hasil uji model						
Penyusunan hasil dan pembahasan						

Keterangan: waktu pengerjaan direpresentasikan dalam rentang waktu bulanan.

2.2 Tahapan Penelitian

Tahapan penelitian terbagi menjadi beberapa bagian, yakni: tahap eksplorasi & *preprocessing* data, tahap *training* & *testing* pada model, tahap *deployment* model pada infrastruktur *cloud computing*, dan analisis hasil.

2.2.1 Eksplorasi dan praproses data

Pada tahapan eksplorasi dan *preprocessing* data, dilakukan penelusuran menyeluruh terhadap dataset COCO 2017, yang akan digunakan untuk melatih model YOLOv5. Eksplorasi dilakukan untuk memahami karakteristik data dan distribusi kelas objek, dimana hal ini penting untuk mengidentifikasi kebutuhan spesifik dalam *preprocessing*. Langkah *preprocessing* selanjutnya melibatkan anotasi atau pelabelan setiap citra sesuai dengan format YOLO, yang mencakup

informasi seperti kelas objek, koordinat *bounding box*, serta dimensi objek (lebar dan tinggi sebuah objek pada citra). Proses ini dilakukan untuk memastikan bahwa model dapat mengenali dan memprediksi objek dengan akurat berdasarkan data yang disiapkan. Dataset kemudian dibagi menjadi tiga bagian, yaitu data *training*, *testing*, dan *validation*, untuk mendukung proses pelatihan dan evaluasi model. Seluruh data yang telah di-*preprocess* diunggah ke Amazon S3, sehingga dapat diakses dengan mudah selama pelatihan model di *platform* Kubeflow.

2.2.2 *Training* dan *testing*

Selanjutnya pada tahap training dan testing, model YOLOv5 dilatih menggunakan dataset COCO 2017 yang telah dianotasi sebelumnya. Proses pelatihan dilakukan di *platform* Kubeflow, dengan memanfaatkan *resource* GPU & CPU untuk mempercepat iterasi dan meningkatkan akurasi model. Selama pelatihan, berbagai *hyperparameter* seperti *learning rate* dan *batch size* diatur untuk mengoptimalkan performa model. *Testing* dilakukan secara berkala pada tiap *epoch* pelatihan untuk mengevaluasi kinerja model, dengan menggunakan metrik evaluasi seperti *mean Average Precision* (mAP). Hasil dari *testing* memberikan gambaran sejauh mana model mampu mendeteksi objek dengan tepat, serta membantu dalam mengidentifikasi area yang perlu dioptimalkan lebih lanjut sebelum tahap *deployment*.

2.2.3 Deployment

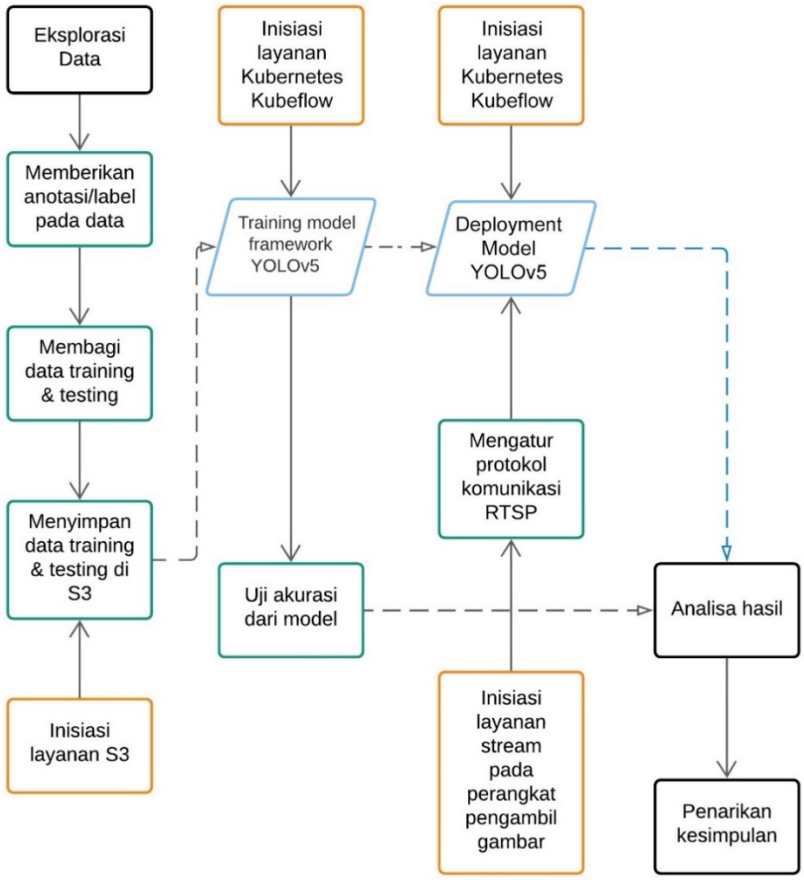
Pada tahap deployment, model yang telah dilatih diimplementasikan pada *platform cloud computing* menggunakan Kubeflow dengan kapabilitas melakukan deteksi objek secara *real-time*. Infrastruktur *cloud* disiapkan untuk mendukung model dalam memproses data secara efisien, termasuk pengaturan komunikasi data melalui protokol RTSP yang memungkinkan streaming video dari perangkat mobile ke *server*. Model yang telah di-*deploy* diintegrasikan dengan sistem RTSP untuk memungkinkan deteksi objek langsung dari *video streaming*, yang kemudian diolah oleh model YOLOv5 untuk menghasilkan *output* berupa video dengan *bounding box* yang mendeteksi objek secara *real-time*.

2.2.4 Analisis hasil

Hasil yang diperoleh dari tahapan-tahapan sebelumnya disimpulkan dalam bentuk visual dan rangkuman dari penelitian yang dilakukan. Kemudian dibentuk konklusi penelitian sebagai pendukung capaian atau tujuan penelitan.

2.3 Alur Penelitian

Flowchart yang menggambarkan alur dan urutan penelitian dapat dilihat pada Gambar 14.



Gambar 14. Alur penelitian.

2.4 Instrumen Penelitian

Perangkat keras yang digunakan pada penelitian ini adalah sebuah *notebook* berspesifikasi *processor* Apple silicon M1, RAM 8GB, dan menggunakan sistem operasi MacOS Ventura 13.0 (*localhost*). Kemudian perangkat lunak yang digunakan untuk melakukan penelitian ini adalah: 1) AWS S3, 2) Kubeflow (Cloud Computer, GPU, Notebook), 3) Chrome, 4) VSCodium, 5) draw.io, 6) RTSP *streaming service*.