Pengembangan Prototipe Sistem inspeksi Papan Sirkuit Cetak Berbasis Visual Menggunakan Deep Neural Network

Proposal Tugas Akhir

Oleh:

Azhani Syahputra (4212111059)

Program Studi Teknik Mekatronika
Jurusan Teknik Elektro
Politeknik Negeri Batam
2024

Pernyataan Keaslian Tugas Akhir

Saya yang bertanda tangan di bawah ini,

Nama	:	Azhani Syahputra
Nomor Induk Mahasiswa	:	4212111059
Program Studi	:	Teknik Mekatronika
Fakultas	:	Teknik Elektro
Perguruan tinggi	:	Politeknik Negeri Batam

Dengan ini menyatakan dengan sesungguhnya bahwa Tugas Akhir saya yang berjudul **'Pengembangan Prototipe Sistem inspeksi Papan Sirkuit Cetak Berbasis Visual Menggunakan Deep Neural Network'** adalah benar-benar karya saya sendiri dan bukan merupakan plagiat atau saduran dari karya orang lain yang saya akui sebagai karya saya, kecuali bagian-bagian yang sumbernya telah saya sebutkan sebagaimana mestinya sesuai dengan kaidah penulisan ilmiah yang berlaku.

Apabila di kemudian hari terbukti bahwa pernyataan ini tidak benar dan terdapat unsur plagiarisme dalam Tugas Akhir ini, maka saya bersedia menerima sanksi sesuai dengan peraturan perundang-undangan yang berlaku dan ketentuan yang ditetapkan oleh Politeknik Negeri Batam termasuk pencabutan gelar akademik yang telah saya peroleh.

Demikian surat pernyataan ini saya buat dengan sebenarnya, dalam keadaan sadar, dan tanpa ada paksaan dari pihak manapun.

Batam, 29 Mei 2025
Yang menyatakan,

Azhani Syahputra
4212111059

Lembar Pengesahan

Lembar Pengesahan

Tugas Akhir disusun untuk digunakan sebagai rencana kerja pada pelaksanaan Tugas Akhir

Disusun oleh:
Azhani Syahputra (4212111059)

Tanggal Sidang: 15 Mei 2025

Disetujui Oleh :

Rifqi Amalya Fatekha, S.Tr.T., M.Tr.T.
NIK: 113107

Penguji

Ryan Satria Wijaya, S.Tr.T., M.Tr.T.
NIK: 121249

Pembimbing

Senanjung Prayoga, S.Pd., M.T.
NIK: 198908062019031015

Penguji

Pengembangan Prototipe Sistem inspeksi Papan Sirkuit Cetak Berbasis Visual Menggunakan Deep Neural Network

Abstrak

Inspeksi kualitas pada Printed Circuit Board (PCB) krusial dalam industri elektronik, namun deteksi cacat kecil menjadi tantangan. Penelitian ini bertujuan mengembangkan sistem inspeksi visual otomatis berbasis Deep Neural Network (DNN) untuk PCB, dengan fokus pada peningkatan deteksi objek kecil dan antarmuka pengguna (GUI) interaktif. Metodologi melibatkan evaluasi model Faster R-CNN, RetinaNet, dan YOLOv11n pada dataset PCB manual. Model YOLOv11n terpilih sebagai basis sistem karena kinerja optimalnya, mencapai mAP@0.5 sebesar 0.890 dengan ukuran model hanya 5.2 MB dan kecepatan inferensi rata-rata ~4.33 FPS. Untuk meningkatkan identifikasi cacat kecil, teknik Slicing Aided Hyper Inference (SAHI) diintegrasikan dengan YOLOv11n. Analisis kualitatif visual menunjukkan bahwa SAHI meningkatkan kemampuan sistem untuk mendeteksi sejumlah cacat kecil yang sebelumnya terlewat oleh model standar. Sistem ini diimplementasikan dengan GUI berbasis Streamlit untuk mempermudah pengoperasioan dan analisis.

Kata Kunci: PCB, Industri, Teknologi, Algoritma, Deteksi Objek, Deep Learning

Development of a Visual-Based Printed Circuit Board Inspection System Prototype Using Deep Neural Network

Abstract

Quality inspection on a crucial print circuit board (PCB) in the electronic industry, but detection of small defects is a challenge. This study aims to develop a visual automatic inspection system based on Deep Neural Network (DNN) for PCB, with a focus on increasing the detection of small objects and interactive user interfaces (GUI). The methodology involves the evaluation of the R-CNN, Retinant, and Yolov11N faster models on the manual PCB dataset. The Yolov11N model was chosen as the system base because of its optimal performance, reaching map@0.5 of 0.890 with a model size of only 5.2 MB and the average inference speed ~ 4.33 fps. To increase the identification of small defects, the technique of Slicing Aided Hyper Inference (SAHI) is integrated with YOLOV11N. Visual qualitative analysis shows that Sahi increases the ability of the system to detect a number of small defects previously missed by a standard model. This system is implemented with streamlit -based GUI to facilitate operation and analysis.

Keywords: PCB, industry, technology, algorithm, object detection, deep learning.

Kata Pengantar

Alhamdulillah, segala puji bagi Allah setelah melalui perjalanan akademis yang menuntut selama empat tahun di Program Studi D4 Teknik Mekatronika, Fakultas Elektro, Politeknik Negeri Batam, akhirnya karya tulis ilmiah ini dapat terselesaikan. Tugas Akhir berjudul 'Pengembangan Prototipe Sistem inspeksi Papan Sirkuit Cetak Berbasis Visual Menggunakan Deep Neural Network' ini merupakan puncak dari serangkaian pembelajaran, tantangan, dan dedikasi waktu yang tidak sedikit.

Proses penyusunan ini tidak lepas dari berbagai rintangan dan kerja keras. Oleh karena itu, penulis ingin menyampaikan apresiasi setinggi-tingginya kepada semua pihak yang telah memberikan dukungan tak ternilai.

Ucapan terima kasih yang paling mendalam penulis sampaikan kepada kedua orang tua, khususnya kepada Ibu yang telah menjadi saksi setiap langkah, baik suka maupun duka, dalam perjalanan hidup selama berkuliah. Terima kasih juga kepada seluruh anggota keluarga yang telah sabar menjadi pendengar setia atas segala lika-liku selama empat tahun ini. Penghargaan tulus juga penulis sampaikan kepada:

- Seluruh dosen Teknik Mekatronika Politeknik Negeri Batam atas ilmu dan bimbingan yang telah diberikan.
- Kepada rekan-rekan di tempat kerja, Bosch Rexroth, terima kasih atas pengertian dan dukungannya selama masa pengerjaan Tugas Akhir ini.
- kepada para mentor dan teman-teman seperjuangan selama program MSIB Batch 5 di Myedusolve dan 6 di Stechoq atas pengalaman dan pembelajaran bersama sepanjang 2023 hingga 2024.

Secara khusus, penulis menghaturkan terima kasih sebesar-besarnya kepada dosen pembimbing, Ryan Satria Wijaya, S.Tr.T., M.Tr.T. yang telah memberikan kepercayaan dan bimbingan penuh kesabaran selama satu tahun terakhir hingga terselesaikannya karya ini.

Besar harapan penulis agar hasil penelitian dalam tugas akhir ini dapat memberikan kontribusi praktis, khususnya dalam pengembangan sistem inspeksi PCB, dan menjadi bekal berharga untuk langkah selanjutnya.

Batam, 29 Mei 2025
Azhani Syahputra

Daftar Isi

Daftar Gambar

Daftar Tabel

Bab 1.
Pendahuluan

1.1. Latar Belakang

Perkembangan teknologi di masa kini semakin pesat. Kebutuhan akan digitalisasi produk-produk konvensional hingga alat elektronik pun kian berdatangan. Produk digital membutuhkan PCB sebagai media untuk meletakkan dan menyambungkan antar komponennya sehingga dapat saling terintegrasi dan efisien dari segi ukuran. Komponen elektronika yang kecil membutuhkan ketelitian yang tinggi dalam mengamatinya, sehingga industri manufaktur modern saat ini mulai mengadaptasi teknologi machine vision untuk dapat menginspeksi hasil produksi agar lebih cepat dan akurat[1], [2].

Di antara berbagai komponen dalam perangkat elektronik, Printed Circuit Board (PCB) memiliki peran yang sangat penting. PCB atau papan sirkuit cetak bertindak sebagai fondasi bagi komponen elektronik lainnya, memungkinkan aliran daya dan sinyal antar komponen berjalan dengan stabil. Dalam industri manufaktur modern, produksi PCB umumnya dilakukan secara otomatis menggunakan mesin berpresisi tinggi untuk meminimalkan potensi cacat.

Namun demikian, proses produksi PCB secara manual masih sangat relevan, terutama dalam lingkungan akademis, komunitas penghobi elektronika, dan industri kecil menengah (IKM) dalam tahap prototyping. Produksi manual ini dinilai lebih ekonomis, fleksibel, dan mudah disesuaikan dengan kebutuhan spesifik[3]. Akan tetapi, metode ini juga membawa tantangan tersendiri, yaitu tingginya kemungkinan terjadinya cacat produksi, seperti mouse bite, spur, spurious copper, short circuit, dan open circuit. Cacat-cacat ini seringkali berukuran sangat kecil dan memiliki bentuk yang beragam, sehingga sulit dideteksi secara kasat mata.

Inspeksi visual manual terhadap PCB tidak hanya memakan waktu, tetapi juga rawan terhadap human error, baik berupa cacat yang terlewat maupun kesalahan dalam mengidentifikasi jenis cacat. Kondisi ini dapat berdampak serius terhadap kinerja rangkaian elektronik secara keseluruhan. Oleh karena itu, diperlukan solusi yang mampu membantu proses inspeksi ini secara lebih cepat, akurat, dan efisien.

Deteksi cacat pada PCB manual menuntut sistem yang tidak hanya mampu mengenali berbagai bentuk dan ukuran cacat dengan akurasi tinggi, tetapi juga mengklasifikasikannya secara spesifik (seperti *mouse bite, open circuit, short circuit*). Kemampuan untuk mengidentifikasi jenis cacat secara partikular ini menjadi krusial karena setiap jenis cacat memiliki akar penyebab dan dampak fungsional yang berbeda terhadap kinerja PCB. Dengan mengetahui jenis cacat

secara spesifik, informasi yang disampaikan oleh sistem dapat dijadikan dasar untuk analisis lebih lanjut dalam proses *quality control*, membantu dalam menentukan tindakan perbaikan yang tepat, serta memberikan umpan balik yang berharga untuk penyempurnaan proses produksi manual di masa depan. *Deep Neural Network* (DNN), khususnya melalui model deteksi objek, menawarkan solusi yang menjanjikan untuk otomatisasi proses identifikasi dan klasifikasi cacat ini secara efisien. Untuk menemukan pendekatan yang paling optimal dalam hal akurasi dan efisiensi, penelitian ini melakukan evaluasi terhadap beberapa arsitektur model DNN populer agar mengetahui keandalan masing masing model. Mengingat cacat pada PCB seringkali berukuran sangat kecil dan menjadi tantangan tersendiri bagi model deteksi objek standar, penelitian ini juga menerapkan teknik peningkatan inferensi seperti Slicing Aided Hyper Inference (SAHI) serta NMS guna memaksimalkan sensitivitas deteksi terhadap objek minor tersebut. Berdasarkan evaluasi ini, dikembangkan sebuah prototipe sistem inspeksi visual PCB berbasis DNN yang dilengkapi antarmuka pengguna (GUI) interaktif, yang tidak hanya melakukan deteksi tetapi juga menyediakan fitur pendukung untuk analisis cacat.

Melalui pengembangan sistem ini, diharapkan proses inspeksi PCB, khususnya yang diproduksi secara manual, dapat dilakukan dengan lebih optimal dan mendukung kegiatan riset, edukasi, maupun industri skala kecil secara efektif.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah diuraikan mengenai tantangan inspeksi PCB manual dan potensi teknologi Object Detection, adapun rumusan masalah dalam penelitian ini adalah :

1. Bagaimana mengembangkan sistem inspeksi visual untuk mendeteksi cacat pada PCB manual menggunakan pendekatan Deep Neural Network ?
2. Bagaimana pendekatan teknik atau metode yang dapat meningkatkan deteksi objek kecil pada PCB ?
3. Bagaimana membangun antarmuka pengguna yang interaktif untuk memfasilitasi proses pengamatan dan analisis hasil deteksi ?

1.3 Tujuan

Dengan rumusan masalah yang ada tersebut, berikut ini merupakan tujuan dari penelitian ini :

1. Mengembangkan sistem inspeksi visual berbasis DNN untuk mendeteksi cacat pada PCB manual.
2. Mengeksplorasi dan menerapkan pendekatan yang efektif untuk meningkatkan akurasi deteksi objek kecil pada PCB manual.

3. Membangun antarmuka pengguna interaktif yang memungkinkan pengguna melakukan pengamatan dan analisis secara efektif pada hasil deteksi.

1.4 Manfaat

Dengan tujuan-tujuan yang sudah disebutkan, secara luas manfaat dari penelitian ini ialah :

1. **Akademis**: Memberikan kontribusi dalam pengembangan metode deteksi cacat pada PCB menggunakan DNN dan pendekatan peningkatan deteksi objek kecil, serta menjadi referensi bagi penelitian sejenis di masa mendatang.
2. **Praktis**: Menyediakan sistem bantu yang efisien bagi akademisi, penghobi, dan industri kecil dalam melakukan inspeksi PCB manual, mengurangi ketergantungan pada inspeksi manual yang memakan waktu dan rentan terhadap kesalahan manusia.

1.5 Batasan

Terdapat batasan dari penelitian ini mencakup beberapa hal, antara lain :

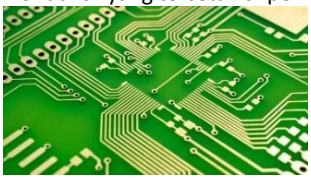
- Sebagian besar klasifikasi cacat masih dihasilkan menggunakan editan software (photoshop).
- Penelitian ini terbatas oleh perbandingan performa model pada tiga arsitektur utama, yakni Faster R-CNN, RetinaNet, dan YOLOv11n, yang dipilih untuk mewakili pendekatan deteksi objek two-stage dan one-stage yang umum digunakan.
- Metode *inference* yang terbatas pada SAHI dan NMS sebagai opsinya.

Bab 2.
Tinjauan Pustaka

2.1 PCB (Printed Circuit Board)

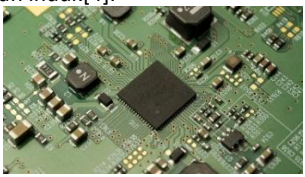
2.1.1 Pengertian PCB

Papan Sirkuit Cetak atau Printed Circuit Board (PCB) merupakan komponen fundamental dalam hampir setiap perangkat elektronik modern. PCB berfungsi sebagai wadah sekaligus penghubung antar komponen elektronik melalui jalur konduktif yang tercetak di permukaannya.



Gambar 2.1. PCB (sumber : https://www.wevolver.com/article/trace-pcb-a-comprehensive-guide, 2024)

Setelah dirakit dengan komponen elektronik, PCB sering kali disebut sebagai motherboard atau papan induk[4].



Gambar 2.2. Motherboard (sumber: https://www.wevolver.com/article/trace-pcb-a-comprehensive-guide, 2024)

Secara umum, struktur PCB terdiri dari empat lapisan utama yang saling terintegrasi, yaitu: substrat, tembaga (copper), soldermask, dan silkscreen.

Gambar 2.3. struktur PCB (sumber: https://www.pcbmay.com/blue-pcb/, 2022)

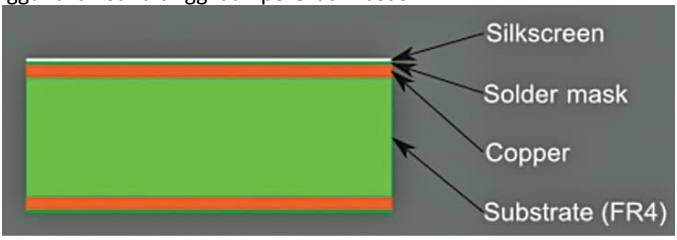
1. Substrat (Lapisan Dasar)

Substrat merupakan lapisan dasar PCB yang memberikan kekuatan mekanis dan menjadi tempat penempelan lapisan konduktor. Dua jenis bahan substrat yang umum digunakan adalah FR2 dan FR4:

- FR2 (Flame Resistant 2) adalah bahan komposit yang terbuat dari kertas yang diresapi resin plastik fenol-formaldehida. Material ini umum digunakan pada PCB murah dan memiliki ketahanan termal serta kekuatan mekanis yang relatif rendah[3].
- FR4 (Flame Resistant 4) terbuat dari anyaman fiberglass yang dilapisi resin epoksi. FR4 memiliki keunggulan berupa ketahanan terhadap panas hingga suhu sekitar 140°C, daya serap air rendah, serta sifat isolasi listrik yang sangat baik. Karena keunggulannya tersebut, FR4 banyak digunakan dalam aplikasi elektronik yang lebih kompleks dan tahan lama, meskipun biayanya lebih tinggi dibandingkan FR2[3].

2. Lapisan Tembaga (Copper Layer)

Di atas substrat, terdapat lapisan tembaga tipis yang berfungsi sebagai jalur konduktor untuk menghubungkan berbagai komponen. Lapisan ini dilaminasi menggunakan suhu tinggi dan perekat khusus.



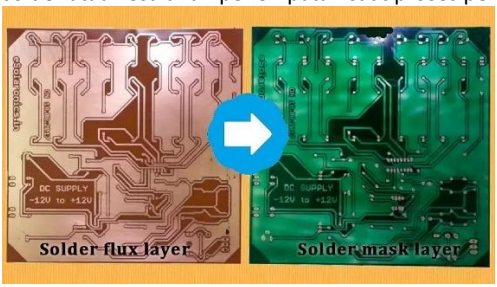
Gambar 2.4. struktur pcb double side (sumber: https://www.pcbmay.com/blue-pcb/, 2022)

Pada Single-Sided PCB, lapisan tembaga hanya terdapat di satu sisi substrat.

Pada Double-Sided PCB, lapisan tembaga ada di kedua sisi substrat.
Untuk aplikasi yang lebih kompleks, Multilayer PCB dapat memiliki hingga 16 lapisan atau lebih, tergantung pada desain sirkuit dan kebutuhan konektivitas antar komponen.

3. Soldermask

Lapisan soldermask berada di atas lapisan tembaga dan berfungsi sebagai pelindung terhadap korosi serta mencegah hubungan pendek (short circuit) akibat percikan solder atau kesalahan penempatan saat proses penyolderan.

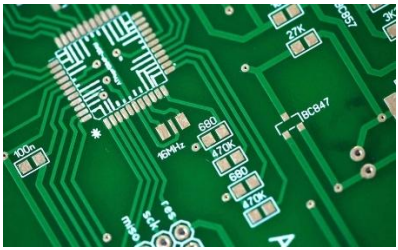


Gambar 2.5. lapisan solder flux dan solder mask (sumber: https://www.pcbaaa.com/pcb-solder-mask/)

Soldermask hanya menutupi area jalur tembaga yang tidak akan disolder, sehingga area pad atau via tetap terbuka. Warna soldermask yang paling umum adalah hijau, tetapi tersedia juga dalam warna lain seperti biru, merah, dan hitam.

4. Silkscreen

Lapisan paling atas adalah silkscreen, yaitu cetakan huruf, angka, simbol, dan kode referensi komponen.



Gambar 2.6. silkscreen pada PCB (sumber: https://www.fs-pcba.com/what-does-pcb-mean/)

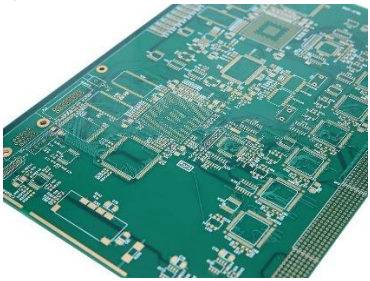
Fungsinya adalah memberikan informasi visual yang membantu proses perakitan dan identifikasi komponen di atas PCB. Warna silkscreen umumnya

putih atau hitam, tetapi dapat juga ditemui dalam warna abu-abu, merah, atau kuning keemasan, tergantung preferensi desain dan estetika.

2.1.2. Jenis-jenis PCB

Berdasarkan fleksibilitas dan aplikasinya, PCB dapat diklasifikasikan menjadi tiga jenis utama:

1. **PCB Rigid (Kaku)**



Gambar 2.7. konstruksi PCB-rigid (sumber: https://www.scscircuits.com/product-item/rigid-pcb/)

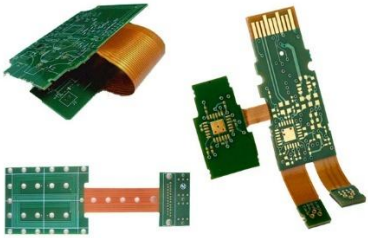
PCB jenis ini adalah yang paling umum digunakan. Memiliki bentuk yang tetap dan tidak dapat ditekuk. PCB rigid biasanya digunakan dalam perangkat elektronik konvensional seperti komputer, televisi, dan peralatan rumah tangga. Material substrat yang digunakan umumnya adalah FR4 atau FR2[3].

2. **PCB Flex (Fleksibel)**

Gambar 2.8. contoh PCB flex pada monitor (sumber: https://yic-assm.com/flexible-circuit-boards/)

PCB fleksibel dibuat dari bahan dasar seperti polyimide yang memungkinkan papan ini dibengkokkan atau dilipat tanpa merusak jalur sirkuit. PCB jenis ini cocok untuk perangkat dengan keterbatasan ruang dan kebutuhan desain melengkung, seperti kamera digital, perangkat wearable, dan smartphone.

3. **PCB Rigid-Flex**



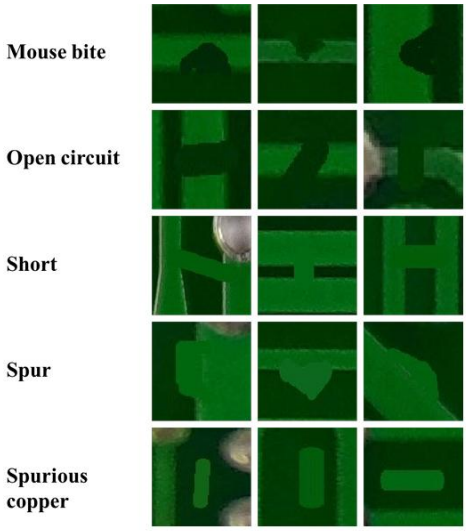
Gambar 2.9. contoh PCB rigid-flex (sumber: https://jamindopcba.com/rigid-flex-pcb/)

Merupakan kombinasi dari PCB rigid dan fleksibel, di mana sebagian area PCB bersifat kaku dan sebagian lainnya fleksibel. Jenis ini biasanya digunakan pada perangkat yang memiliki bagian bergerak atau lipatan, seperti laptop dengan layar yang dapat dilipat atau perangkat militer dan medis yang menuntut desain ringkas, ringan, dan tahan guncangan[4].

2.1.3. Potensi Cacat pada PCB hasil produksi manual

Proses pembuatan PCB secara manual, meskipun memberikan fleksibilitas dalam produksi skala kecil dan prototipe, memiliki potensi menghasilkan berbagai jenis cacat yang dapat mempengaruhi kinerja dan keandalan sirkuit elektronik. Cacat-cacat ini umumnya disebabkan oleh variabilitas dalam proses manufaktur, keterbatasan peralatan, dan faktor manusia.

Beberapa jenis cacat yang sering ditemukan meliputi :



Gambar 2.10. klasifikasi cacat yang umum pada produksi PCB (sumber:[4])

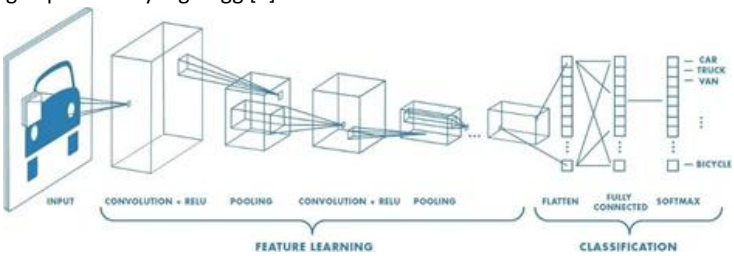
- Jalur Terputus (**Open Circuit**): Terputusnya jalur konduktif yang menyebabkan aliran listrik terganggu. Cacat ini sering disebabkan oleh tinta yang tidak atau kurang melekat pada lapisan tembaga selama proses transfer, atau gangguan fisik yang menyebabkan jalur terputus sebelum proses etsa berlangsung[4].
- Hubungan Pendek (**Short Circuit**): Terjadinya koneksi listrik yang tidak diinginkan antara dua jalur konduktif yang seharusnya terisolasi. Hal ini dapat disebabkan oleh tinta yang menyebar atau menyatu antara jalur selama proses transfer, atau residu tembaga yang tidak sepenuhnya terhapus saat etsa[4].
- Cacat Tepi (**Mouse Bite**): Cekungan atau lekukan kecil yang tidak rata pada tepi jalur konduktif atau pad. Biasanya disebabkan oleh transfer tinta yang kurang rapi, yang dapat mengurangi luas penampang jalur dan berpotensi menyebabkan koneksi listrik yang tidak stabil[4].
- Tonjolan Jalur (**Spur**): Proyeksi kecil dari jalur konduktif yang tidak sesuai dengan desain, sering kali disebabkan oleh transfer tinta yang kurang baik sehingga memudar dan membentuk jalur tembaga yang tidak diinginkan saat proses etsa[4].

- Tembaga Tak Diinginkan (**Spurious Copper**): Area tembaga sisa yang tidak diinginkan yang muncul di area isolasi pada PCB, tidak terhubung dengan desain sirkuit yang ditentukan. Cacat ini dapat disebabkan oleh tinta yang menempel secara tidak sengaja selama proses transfer, dan dapat menyebabkan hubungan pendek jika menyatu dengan dua jalur yang tidak semestinya[4].

Pemahaman terhadap jenis-jenis cacat ini penting dalam konteks pengembangan sistem inspeksi otomatis berbasis teknologi seperti Deep Neural Network (DNN). Dengan mengenali pola dan karakteristik cacat, sistem inspeksi dapat dilatih untuk mendeteksi dan mengklasifikasikan cacat secara efektif, sehingga meningkatkan kualitas dan keandalan produk akhir.

2.2 Deep Neural Network

Deep Neural Network (DNN) adalah arsitektur pembelajaran mesin yang terdiri dari banyak lapisan neuron buatan yang saling terhubung. Setiap lapisan dalam DNN memproses input dari lapisan sebelumnya dan mentransformasikannya ke dalam representasi yang lebih abstrak. Kemampuan ini memungkinkan DNN untuk menangani data kompleks seperti citra, suara, dan teks dengan performa yang tinggi[5].

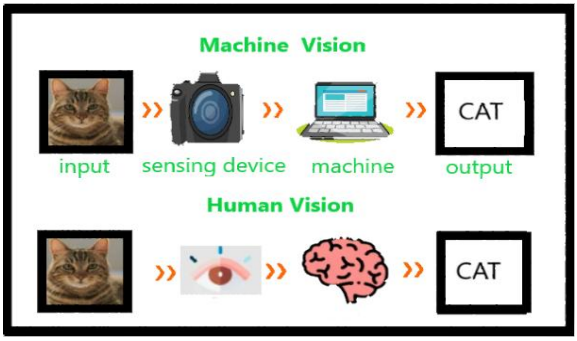


Gambar 2.11. arsitektur CNN untuk classification object (sumber: https://www.oreilly.com/library/view/strengthening-deep-neural/9781492044949/ch04.html)

Salah satu jenis DNN yang paling populer dalam pengolahan citra adalah Convolutional Neural Network (CNN). CNN dirancang khusus untuk memproses data grid-like, seperti gambar, dengan memanfaatkan operasi konvolusi untuk menangkap fitur lokal seperti tepi, tekstur, dan pola. Arsitektur CNN telah menjadi dasar bagi banyak model sukses dalam berbagai tugas visi komputer, termasuk klasifikasi gambar, segmentasi, dan deteksi objek.

2.2.1 Peran DNN dalam Computer Vision

Computer Vision adalah bidang ilmu yang memungkinkan komputer untuk "melihat" dan memahami konten visual dari dunia nyata, seperti gambar dan video. Dengan menggunakan DNN, khususnya CNN, sistem komputer dapat secara otomatis mengekstraksi fitur-fitur penting dari data visual tanpa memerlukan rekayasa fitur manual. Hal ini telah merevolusi berbagai aplikasi, mulai dari pengenalan wajah, analisis medis, hingga kendaraan otonom.

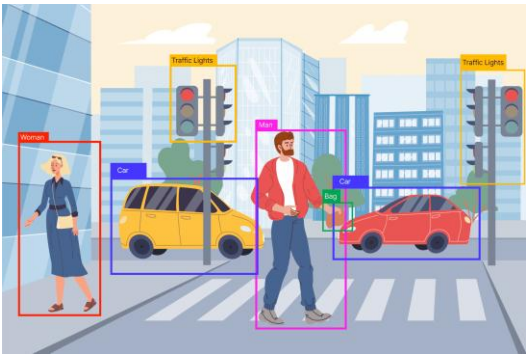


Gambar 2.12. analogi machine vision dan human vision

DNN memungkinkan komputer untuk memahami konteks visual dengan cara yang menyerupai persepsi manusia, seperti mengenali objek dalam berbagai kondisi pencahayaan, sudut pandang, dan latar belakang. Kemampuan ini sangat penting dalam pengembangan sistem yang dapat berinteraksi secara efektif dengan lingkungan sekitarnya.

2.2.2 Object Detection dalam Computer Vision

Object Detection adalah salah satu tugas utama dalam Computer Vision yang bertujuan untuk mengidentifikasi dan menentukan lokasi objek-objek dari kelas tertentu dalam sebuah citra atau video. Berbeda dengan klasifikasi gambar yang hanya memberikan label untuk seluruh gambar, Object Detection memberikan informasi tentang kelas dan posisi (bounding box) dari setiap objek yang terdeteksi[5], [6], [7].

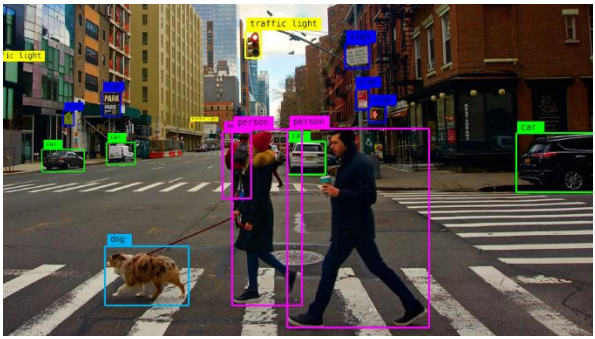


Gambar 2.13. gambaran object detection (sumber : https://deeplobe.ai/exploring-object-detection-applications-and-benefits/)

Dengan menggunakan DNN, khususnya arsitektur seperti CNN, sistem dapat dilatih untuk mendeteksi berbagai objek dalam kondisi yang kompleks dan beragam. Model-model seperti R-CNN, Fast R-CNN, Faster R-CNN, YOLO (You Only Look Once), dan SSD (Single Shot MultiBox Detector) telah menunjukkan performa yang sangat baik dalam berbagai benchmark dan telah diaplikasikan dalam berbagai bidang, termasuk pengawasan, kendaraan otonom, dan analisis medis.

2.3 Object Detection

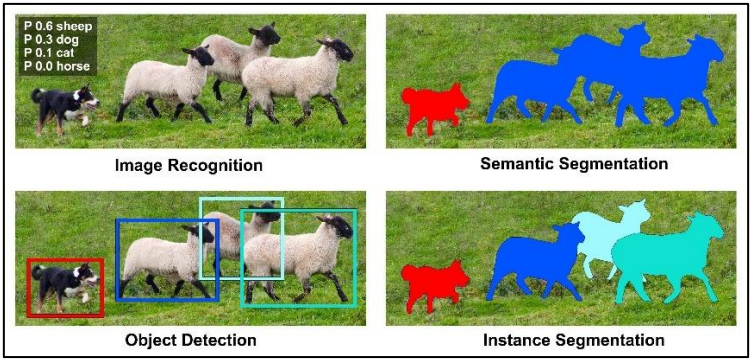
Meskipun klasifikasi melalui CNN dapat mengidentifikasi kategori utama dalam sebuah gambar, banyak implementasinya membutuhkan kemampuan untuk mendeteksi dan melokalisasi banyak objek/class berbeda dalam satu citra atau gambar secara langsung. Tugas inilah yang dikenal sebagai Object Detection[8].



Gambar 2.14. inference object detection pada dunia nyata (sumber: https://www.augmentedstartups.com/blog/how-to-implement-object-detection-using-deep-learning-a-step-by-step-guide)

Object Detection bertujuan untuk mengidentifikasi keberadaan, menentukan lokasi (dengan bounding box), dan mengklasifikasikan setiap objek menarik yang muncul dalam sebuah citra. Output dari model Object Detection beragam tergantung model yang digunakan namun biasanya berupa daftar bounding box, di mana setiap bounding box memiliki koordinat (x, y, lebar, tinggi) dari prediksi objek/class, label kelas objek yang diprediksi, dan skor kepercayaan (confidence score) terhadap prediksi tersebut[8].

Untuk memahami posisi Object Detection dalam Computer Vision, penting untuk membedakannya dari tugas-tugas lain yang terkait:



Gambar 2.15. kategori inference pada machine vision (sumber: https://www.reddit.com/r/learnmachinelearning/comments/kt0hov/difference_in_image_classification_semantic/)

- **Klasifikasi Citra (Image Classification):**

Gambar 2.16. classification v detection

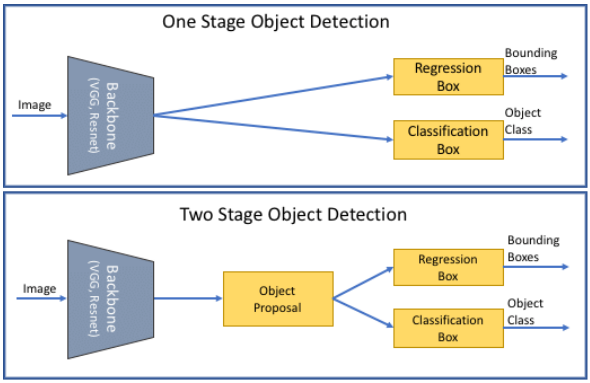
Memberikan satu label kategori untuk keseluruhan citra, mengabaikan lokasi objek spesifik atau keberadaan objek lain. Kemampuan klasifikasi juga terbatas hanya kepada keseluruhan citra yang ditampilkan, sehingga mudahnya klasifikasi hanya bisa menebak 1 klasifikasi yang menyerupai (a,b, atau c) bukan klasifikasi lebih dari 1 sekaligus.

- **Segmentasi Semantik (Semantic Segmentation):**
  Menetapkan label kelas untuk setiap piksel dalam citra, mengelompokkan piksel-piksel yang termasuk dalam kelas yang sama.
- **Segmentasi Instance (Instance Segmentation):**
  Mirip dengan segmentasi semantik, tetapi juga membedakan antar instance objek dari kelas yang sama (misal, membedakan dua mobil yang berdekatan sebagai dua instance terpisah) dan menghasilkan mask piksel untuk setiap instance.
- **Object Detection:**
  Fokus pada melokalisasi objek dengan bounding box dan mengklasifikasikan objek di dalam setiap bounding box tersebut.

Dalam beberapa tahun terakhir, model Object Detection berbasis Deep Learning telah mencapai akurasi dan kecepatan yang luar biasa. Secara garis besar, arsitektur model Object Detection modern dapat dikategorikan ke dalam dua pendekatan utama:

- **Pendekatan Two-stage (Dua Tahap):** Model dalam kategori ini memecah tugas deteksi menjadi dua tahap sequential. Tahap pertama adalah menghasilkan serangkaian proposal wilayah (region proposals) yang berpotensi mengandung objek. Tahap kedua adalah mengambil setiap proposal wilayah ini, melakukan klasifikasi objek di dalamnya, dan memperbaiki (regresi) koordinat bounding box-nya. Contoh arsitektur terkenal dari pendekatan ini adalah keluarga model R-CNN, termasuk Faster

R-CNN. Model two-stage cenderung menawarkan akurasi deteksi yang sangat tinggi, namun seringkali lebih lambat karena proses dua tahap tersebut[9].



Gambar 2.17. alur one-stage v two-stage detection (sumber: https://www.researchgate.net/figure/Two-stage-vs-one-stage-object-detection-models_fig3_353284602)

- **Pendekatan One-stage (Satu Tahap):** Model dalam kategori ini melakukan deteksi objek dalam satu kali proses melalui jaringan saraf. Mereka langsung memprediksi bounding box dan probabilitas kelas untuk objek di berbagai lokasi dalam citra tanpa melalui tahap proposal wilayah yang eksplisit. Contoh arsitektur populer dari pendekatan ini adalah keluarga model YOLO (You Only Look Once), SSD (Single Shot Detector), dan RetinaNet. Model one-stage umumnya lebih cepat dan efisien secara komputasi, menjadikannya pilihan menarik untuk aplikasi real-time, meskipun secara historis akurasinya mungkin sedikit di bawah model two-stage (meskipun model one-stage terbaru semakin memperkecil gap ini)[9].

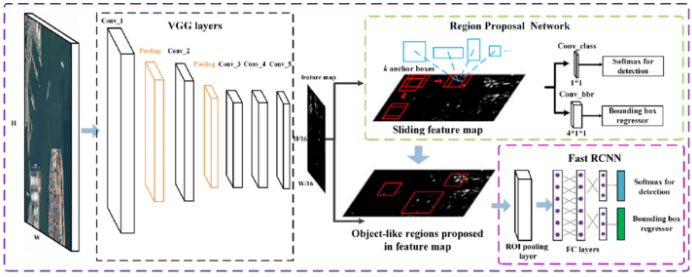
2.4 Arsitektur model Object Detection

Berdasarkan konsep Object Detection dan pendekatan umum yang telah dijelaskan, penelitian ini membandingkan dan memanfaatkan beberapa model arsitektur Deep Learning Object Detection yang merupakan representasi dari pendekatan two-stage maupun one-stage object detection, serta sebuah teknik pendukung untuk inference. Model-model ini dipilih karena representatif, performanya yang baik, dan kemudahan dalam dukungan pustaka/dependencies

dalam melakukan fine-tune (membangun model dengan weight yang sudah dilatih atau pretrained weight).

2.4.1 Faster R-CNN

Faster R-CNN merupakan salah satu arsitektur pelopor dan representasi kuat dari pendekatan two-stage object detection [RUJUKAN Paper Faster R-CNN]. Model ini meningkatkan pendahulunya (R-CNN, Fast R-CNN) dengan mengintegrasikan tahap proposal wilayah (region proposal) ke dalam jaringan saraf, menjadikannya lebih efisien. Arsitektur utamanya terdiri dari beberapa komponen kunci:



Gambar 2.18. arsitektur faster R-CNN (sumber: https://www.researchgate.net/figure/The-architecture-of-Faster-R-CNN_fig2_324903264)

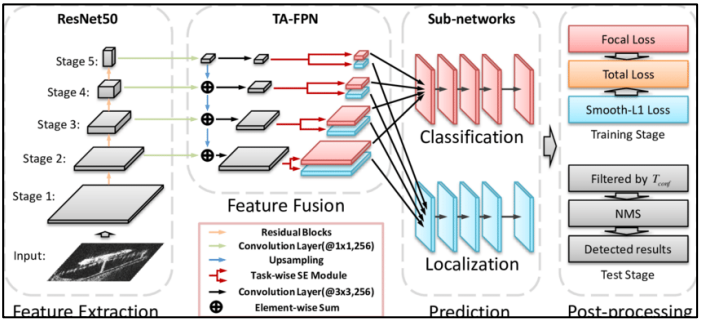
- **Backbone CNN**: Bagian awal jaringan (misal: VGG, ResNet) yang berfungsi mengekstraksi fitur hierarkis dari citra input[10].
- **Region Proposal Network (RPN):** Sebuah jaringan kecil yang berjalan di atas peta fitur dari backbone, bertugas memprediksi bounding box proposal wilayah (area yang berpotensi mengandung objek) dan skor objectness untuk setiap proposal[10].
- **RoI Pooling atau RoI Align:** Mekanisme untuk mengekstraksi fitur dengan ukuran tetap dari peta fitur backbone untuk setiap proposal wilayah yang dihasilkan oleh RPN[10].
- **Detection Head:** Jaringan yang menerima fitur dari RoI Pooling/Align dan melakukan dua tugas: klasifikasi objek di dalam proposal ke salah satu kelas yang relevan (termasuk kelas background) dan regresi untuk memperbaiki koordinat bounding box proposal agar lebih akurat[10].

Pada saat inference, citra diproses oleh backbone, RPN menghasilkan proposal, dan detection head memproses proposal ini untuk output akhir

berupa bounding box terdeteksi beserta kelas dan skornya. Model ini cenderung unggul dalam akurasi, terutama untuk lokalisasi yang tepat, namun umumnya lebih lambat dibandingkan model one-stage.

2.4.2 RetinaNet

RetinaNet adalah model Object Detection yang merupakan representasi kuat dari pendekatan one-stage object detection[11]. Arsitektur ini dirancang untuk menandingi akurasi model two-stage sambil tetap mempertahankan kecepatan model one-stage. Arsitektur intinya meliputi:



Gambar 159. arsitektur RetinaNet (sumber: https://www.superannotate.com/blog/object-detection-with-deep-learning)

- **Backbone CNN**: Sama seperti Faster R-CNN, digunakan untuk ekstraksi fitur multi-skala.
- **Feature Pyramid Network (FPN)**: Membangun piramida fitur multi-skala di atas backbone, memungkinkan deteksi objek pada resolusi yang berbeda, sangat membantu untuk mendeteksi objek berukuran kecil.
- **Classification Head**: Jaringan kecil terpisah yang diterapkan pada setiap level piramida fitur FPN untuk memprediksi probabilitas kelas untuk serangkaian anchor box yang telah ditentukan sebelumnya.
- **Regression Head**: Jaringan kecil terpisah lainnya yang juga diterapkan pada setiap level FPN untuk memprediksi offset bounding box dari anchor box yang sesuai.

Fitur paling inovatif dari RetinaNet adalah pengenalan *Focal Loss*, sebuah modifikasi dari standard *cross-entropy loss*. *Focal Loss* dirancang khusus untuk

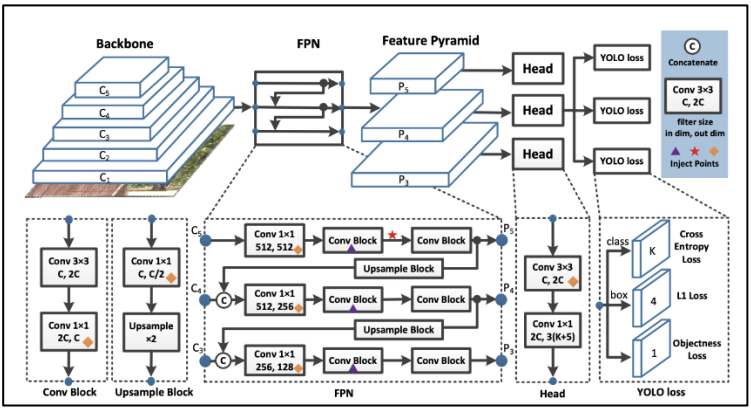
menangani masalah *class imbalance* ekstrem yang sering terjadi pada model *one-stage* (jumlah lokasi background potensial jauh lebih banyak daripada lokasi foreground/objek). Focal Loss memberikan bobot yang lebih rendah pada easy examples (contoh background yang mudah diklasifikasikan dengan benar) dan bobot lebih tinggi pada hard examples (contoh foreground atau background yang sulit), sehingga training lebih fokus pada kasus-kasus sulit. Ini sangat relevan untuk deteksi cacat di mana area "bukan cacat" jauh mendominasi citra.

2.4.3 YOLO (You Only Look Once)

YOLO adalah keluarga model object detection berbasis pendekatan one-stage, di mana proses pendeteksian objek dilakukan dalam satu tahap inference tunggal. Tidak seperti metode two-stage seperti R-CNN, YOLO secara langsung memprediksi bounding box dan kelas objek dalam satu langkah, menjadikannya sangat cepat dan efisien untuk aplikasi real-time.

Arsitektur Dasar YOLO

Arsitektur YOLO secara umum terdiri dari tiga komponen utama:



Gambar 2.20. arsitektur YOLO secara umum (sumber: https://blog.roboflow.com/scaled-yolov4-tops-efficientdet/)

- **Backbone**
  Merupakan jaringan konvolusional (CNN) yang digunakan untuk mengekstrak fitur dari citra input. YOLO awal menggunakan Darknet, sedangkan versi-versi terbaru mengadopsi arsitektur yang lebih canggih seperti CSPDarknet dan varian evolusioner lainnya untuk meningkatkan efisiensi dan akurasi.

- **Neck**
  Komponen ini bertugas menggabungkan fitur dari berbagai tingkat kedalaman (multi-scale feature fusion). Versi modern YOLO mengintegrasikan struktur seperti FPN (Feature Pyramid Network) dan PANet untuk memperkuat representasi fitur, terutama pada objek berukuran kecil.
- **Head Deteksi**
  Di bagian akhir, YOLO membagi citra menjadi grid dan memprediksi bounding box (koordinat, dimensi, skor kepercayaan) serta probabilitas kelas untuk setiap sel grid.

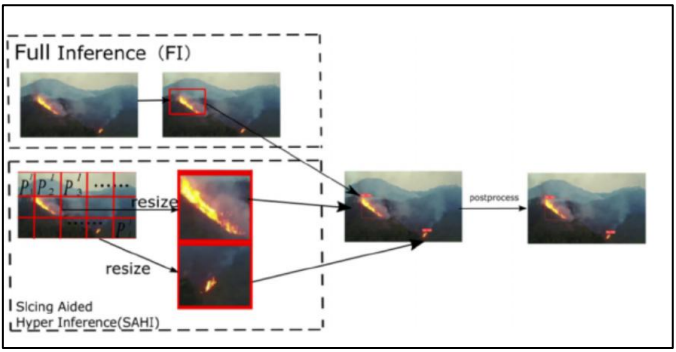
Evolusi YOLO

Sejak dirilisnya YOLOv1, model ini telah mengalami banyak peningkatan melalui versi-versi berikutnya (v2 hingga v8, dan kini bahkan sampai v11). Perubahan mencakup desain backbone yang lebih kuat, neck dan head yang lebih efisien, fungsi loss yang lebih akurat, serta teknik pelatihan modern untuk meningkatkan performa dalam hal akurasi dan kecepatan inference.

2.3 SAHI (Slicing Aided Hyper Inference)

2.3.1 Konsep SAHI dalam Inference Model

SAHI (Slicing Aided Hyper Inference) adalah teknik inference yang dirancang untuk meningkatkan deteksi objek kecil dalam citra beresolusi tinggi. Metode ini bekerja dengan membagi citra besar menjadi potongan-potongan kecil (slices), melakukan inference pada setiap potongan, dan kemudian menggabungkan hasilnya untuk mendapatkan deteksi menyeluruh pada citra asli.



Gambar 2.21. analogi slicing pada SAHI (sumber: https://www.researchgate.net/figure/The-reasoning-process-diagram-of-the-Slicing-Aided-Hyper-Inference-framework-for-forest_fig7_364257753, 2023)

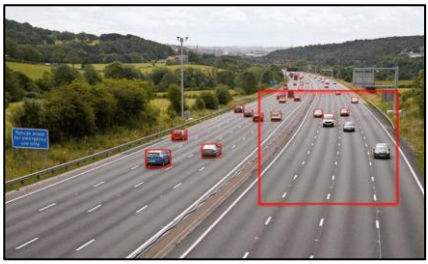
Pendekatan ini sangat efektif dalam mengatasi tantangan deteksi objek kecil yang sering terlewatkan dalam inference standar. Dengan memproses potongan citra yang lebih kecil, model dapat lebih fokus pada detail lokal, meningkatkan akurasi deteksi objek kecil tanpa memerlukan pelatihan ulang model.

2.3.2 Integrasi SAHI dengan Model Deteksi Objek

SAHI bersifat agnostik terhadap framework dan dapat diintegrasikan dengan berbagai model deteksi objek populer seperti YOLOv5, YOLOv8, dan MMDetection. Integrasi ini memungkinkan pengguna untuk memanfaatkan keunggulan SAHI tanpa perlu melakukan modifikasi signifikan pada arsitektur model yang ada.

Proses integrasi SAHI melibatkan beberapa langkah utama:

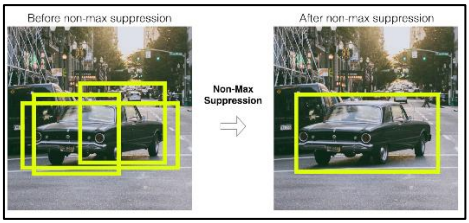
- Pemotongan Citra: Citra input dibagi menjadi beberapa potongan dengan ukuran dan overlap tertentu.
- Inference pada Potongan:



Gambar 2.22. proses slicing metode SAHI (sumber: Ultralytics Docs: Using YOLO11 with SAHI for Sliced Inference, 2025)

Setiap potongan diproses secara independen oleh model deteksi objek untuk mengidentifikasi objek yang ada.

- Penggabungan Hasil:



Gambar 2.23. penerapan NMS mengeliminasi bbox overlapse (sumber: https://www.linkedin.com/pulse/non-max-suppression-object-detection-nabeelah-maryam-2gr7f, 2024)

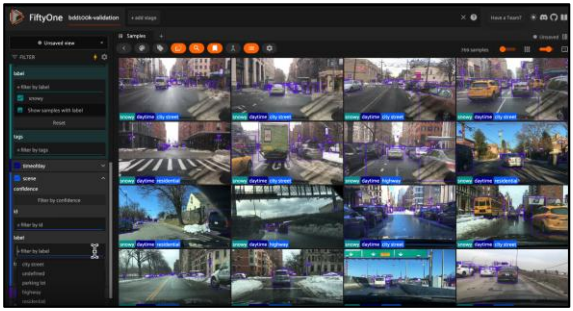
Hasil deteksi dari semua potongan digabungkan kembali ke dalam koordinat citra asli, dengan penerapan Non-Maximum Suppression (NMS) untuk mengeliminasi deteksi duplikat.

Pendekatan ini tidak hanya meningkatkan akurasi deteksi objek kecil tetapi juga memungkinkan pemrosesan citra beresolusi tinggi pada perangkat dengan sumber daya terbatas[12].

2.3.3 Antarmuka Pengguna Grafis (GUI) untuk SAHI

Untuk memudahkan penggunaan dan visualisasi hasil inference, SAHI menyediakan integrasi dengan antarmuka pengguna grafis (GUI) melalui berbagai alat bantu, seperti:

- **FiftyOne**:



Gambar 2.24. GUI fiftyone

Platform eksplorasi dan evaluasi dataset yang memungkinkan pengguna untuk memvisualisasikan hasil deteksi, melakukan analisis kesalahan, dan membandingkan performa model.

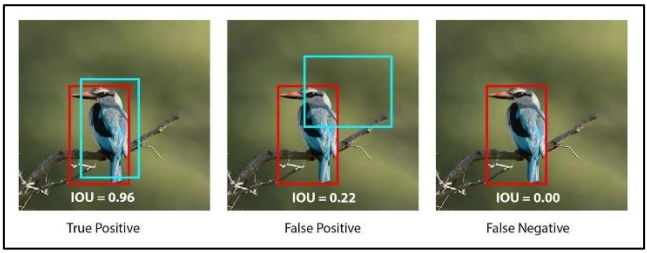
- **Hugging Face Spaces**: Integrasi dengan platform ini memungkinkan pengguna untuk menjalankan demo SAHI secara interaktif di lingkungan web, tanpa perlu instalasi lokal.

Dengan adanya GUI, pengguna dapat dengan mudah mengatur parameter inference, memantau proses deteksi, dan menganalisis hasil secara intuitif, yang sangat bermanfaat dalam pengembangan dan evaluasi model deteksi objek.

2.6 Metrik Evaluasi Kinerja dari Model Object Detection terhadap Dataset

Untuk mengukur dan membandingkan performa dari model-model Object Detection, diperlukan metrik evaluasi kuantitatif yang standar dan dapat diterima secara umum. Metrik ini penting untuk menilai seberapa baik model dalam mengidentifikasi objek (akurasi klasifikasi), menentukan lokasinya dengan tepat (akurasi lokalisasi), dan seberapa efisien model tersebut saat digunakan. Beberapa metrik kunci yang umum digunakan dalam evaluasi Object Detection meliputi :

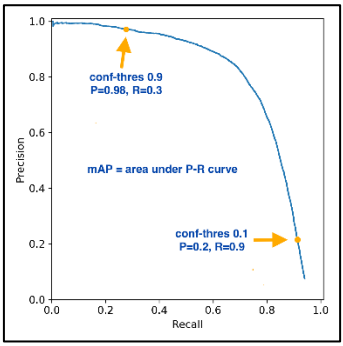
- **Intersection over Union (IoU):** IoU adalah metrik dasar yang digunakan untuk mengukur tumpang tindih (overlap) antara bounding box hasil prediksi model (predicted box) dengan bounding box objek sebenarnya (ground truth box). Didefinisikan sebagai rasio luas area irisan (intersection) antara kedua box dibagi dengan luas area gabungannya (union)[13].



Gambar 2.25. analogi IOU pada object detection (sumber: https://learnopencv.com/intersection-over-union-iou-in-object-detection-and-segmentation/,2023)

Nilai IoU berkisar antara 0 (tidak ada tumpang tindih) hingga 1 (tumpang tindih sempurna). Dalam evaluasi, IoU digunakan sebagai threshold untuk menentukan apakah sebuah prediksi bounding box dianggap benar (True Positive - TP) jika IoU-nya melebihi threshold tertentu (misal: IoU > 0.5) dan label kelasnya benar. Jika IoU di bawah threshold atau label kelas salah, prediksi tersebut dianggap salah (False Positive - FP). Objek sebenarnya yang tidak terdeteksi dianggap False Negative (FN)[14].

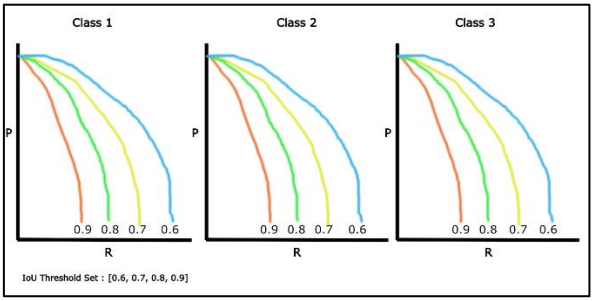
- **Precision:** Precision mengukur seberapa akurat prediksi positif model. Didefinisikan sebagai rasio True Positives (TP) terhadap jumlah total prediksi positif (TP + False Positives - FP). Precision menjawab pertanyaan: "Dari semua bounding box yang diprediksi model sebagai objek, berapa persen yang sebenarnya benar?"[14].
- **Recall:** Recall (atau Sensitivity) mengukur kemampuan model dalam menemukan semua objek yang relevan. Didefinisikan sebagai rasio True Positives (TP) terhadap jumlah total objek sebenarnya dalam dataset (TP + False Negatives - FN). Recall menjawab pertanyaan: "Dari semua objek sebenarnya dalam dataset, berapa persen yang berhasil ditemukan oleh model?" [14].
- **Precision-Recall Curve:**



Gambar 2.26. metrik precision over recall

Dengan mengubah confidence threshold (batas skor kepercayaan) pada output model, kita bisa mendapatkan pasangan nilai Precision dan Recall yang berbeda. Plot dari pasangan nilai ini membentuk kurva Precision-Recall, yang menggambarkan trade-off antara kedua metrik tersebut pada berbagai tingkat kepercayaan model [14].

- **Average Precision (AP):**



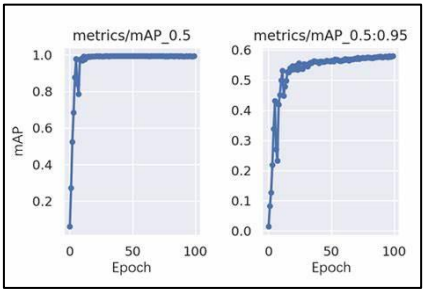
Gambar 2.27. metrik Average Precision antar kelas (sumber: https://www.v7labs.com/blog/mean-average-precision,2022)

AP adalah metrik yang merangkum informasi dari kurva Precision-Recall menjadi satu nilai tunggal. Secara teknis, AP sering dihitung sebagai area di bawah kurva Precision-Recall untuk SATU KELAS objek spesifik[14].

- **Mean Average Precision (mAP):** mAP adalah rata-rata (Mean) dari nilai AP yang dihitung untuk SEMUA KELAS objek dalam dataset. Ini adalah metrik yang paling umum digunakan untuk memberikan gambaran performa keseluruhan model Object Detection di seluruh kelas[14].

Dalam penelitian Object Detection modern (terutama yang menggunakan standar evaluasi seperti COCO), ada beberapa varian mAP:

1. **mAP@0.5 (atau mAP50):** Merupakan mAP yang dihitung dengan menggunakan IoU threshold tunggal sebesar 0.5. Metrik ini memberikan gambaran performa deteksi yang relatif kurang ketat terhadap akurasi lokalisasi bounding box[13].
2. **mAP@0.5:0.95 (atau mAP):**



Gambar 2.28. metrics mAP@0.5 dan 0.5:0.95

Merupakan mAP yang dihitung dengan mengambil rata-rata nilai AP pada berbagai IoU threshold, mulai dari 0.50 hingga 0.95 dengan langkah 0.05 (0.50, 0.55, ..., 0.95). Metrik ini jauh lebih ketat dan memberikan nilai yang lebih rendah, karena model harus memprediksi bounding box dengan akurasi lokalisasi yang tinggi agar dianggap TP pada IoU threshold yang lebih tinggi[13].

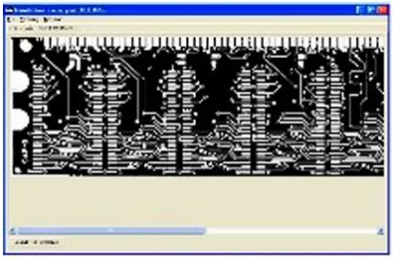
- **Kecepatan Inference (Inference Speed):** Metrik ini mengukur efisiensi komputasi model, yaitu seberapa cepat model yang sudah terlatih dapat memproses sebuah citra input dan menghasilkan prediksi deteksi. Biasanya dinyatakan dalam Frames Per Second (FPS) atau milidetik per citra (ms/citra). Kecepatan inference sangat penting untuk aplikasi yang memerlukan respon cepat, seperti inspeksi real-time dalam jalur produksi[5], [6], [7], [13].

2.7 Penelitian Terdahulu

2.7.1 PENGGUNAAN METODE TEMPLETE MATCHING UNTUK IDENTIFIKASI KECACATAN PADA PCB

Penelitian oleh Adhitya Wishnu Wardhana dan Yudi Prayudi (2008) membahas Template Matching sebagai teknik pencocokan pola dalam

pengolahan citra digital untuk mendeteksi kecacatan pada Printed Circuit Board (PCB).



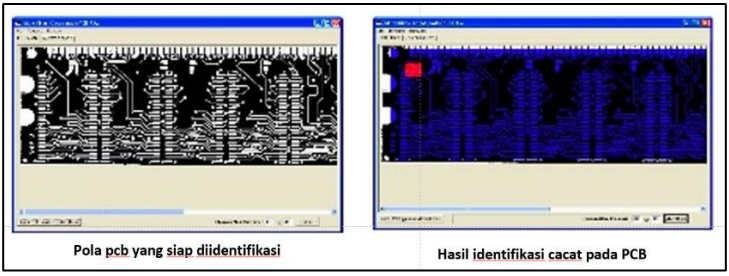
Gambar 2.29. template master sebagai acuan (sumber:[15])

Metode ini bekerja dengan membandingkan citra PCB uji dengan template acuan, mendeteksi kesalahan hingga tingkat piksel, dan memberikan indikasi visual terhadap area yang mengalami kecacatan, seperti putusnya jalur atau pelebaran/penyempitan jalur tembaga.

Sistem ini merupakan bagian dari Automated Optical Inspection (AOI) yang banyak digunakan dalam industri PCB untuk meningkatkan efisiensi dan akurasi inspeksi dibandingkan metode manual.

a. Kekurangan Metode Template Matching dibandingkan Deteksi Cacat Modern

Meskipun Template Matching efektif dalam mendeteksi pola yang sederhana dan memiliki tingkat akurasi tinggi, metode ini memiliki beberapa keterbatasan dibandingkan teknik deteksi cacat berbasis Machine Learning dan Deep Learning, yaitu :



Gambar 2.30. inference template matching pada penelitian (sumber:[15])

- **Sensitif** terhadap orientasi dan kondisi citra – Perbedaan posisi, ukuran, pencahayaan, atau kualitas gambar dapat mengurangi akurasi identifikasi cacat.

- **Tidak dapat mengenali pola cacat kompleks** – Bergantung pada template yang ada, sehingga sulit mendeteksi variasi cacat yang tidak terduga.
- **Memerlukan preprocessing citra** – Harus melalui tahap enhancement dan filtering untuk meningkatkan efektivitas pendeteksian.
- **Kurang adaptif terhadap data baru** – Berbeda dengan sistem berbasis AI, metode ini tidak bisa belajar dari dataset tambahan tanpa pemrograman ulang.

2.7.2 Pembuatan Alat Inspeksi Visual Jalur PCB menggunakan Pengolahan Citra

Penelitian oleh Rangga Ade Juliano et al. (2022) mengembangkan alat inspeksi kecacatan pada Printed Circuit Board (PCB) menggunakan metode YOLO CNN dengan Webcam Logitech C920 dan Raspberry Pi 3b+.

Gambar 2.31. sistem yang dibangun pada penelitian Rangga, dkk (sumber:[16])

Sistem ini dirancang untuk mendeteksi lima jenis cacat pada PCB, dengan hasil evaluasi yang menunjukkan mAP@0.5 tinggi:

- Short (90.67%)
- Open Circuit (97.86%)
- Mouse Bite (94.43%
- Missing Hole (96.09%)
- Spur (97.56%)

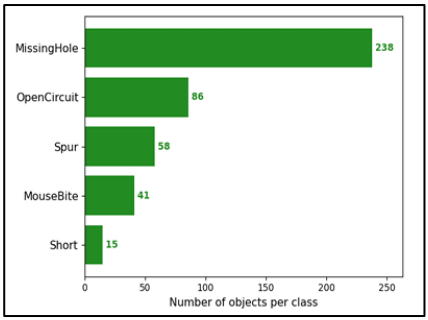
a. Keterbatasan dan Tantangan

Meskipun menunjukkan performa tinggi, penelitian ini memiliki beberapa keterbatasan yang perlu dicermati:

- **Distribusi dataset yang tidak optimal** – Semua data terlebih dahulu diaugmentasi sebelum dibagi ke dalam training dan validasi, sehingga data validasi memiliki karakteristik mirip dengan data

training. Hal ini menyebabkan model menjadi overconfident, dengan hasil evaluasi yang tampak sangat tinggi tetapi kurang mencerminkan kemampuan generalisasi di kondisi nyata.

- **Bias** terhadap kelas dominan



Gambar 2.32. klasifikasi yang dideteksi oleh model yang dibangun oleh Rangga, dkk (sumber :[16])

Terdapat ketidakseimbangan dalam jumlah sampel tiap kelas cacat PCB. Model cenderung memiliki akurasi tinggi pada kelas yang lebih dominan, tetapi kurang akurat dalam mendeteksi kelas dengan jumlah sampel lebih sedikit, tanpa adanya langkah mitigasi khusus untuk mengatasi bias ini.

- GUI hanya terbatas pada pemilihan opsi pengambilan gambar.

Bab 3.
Metodologi Penelitian

3.1 Pendekatan dan Desain Penelitian

Penelitian ini mengadopsi pendekatan kuantitatif dengan metode eksperimental rekayasa perangkat lunak untuk mengembangkan sistem inspeksi visual cacat pada *Printed Circuit Board* (PCB). Tujuan utamanya adalah membangun sistem berbasis model *deep learning* yang mampu meningkatkan akurasi deteksi, khususnya untuk cacat berukuran kecil, dan menyajikan hasilnya melalui antarmuka pengguna (GUI) yang interaktif dan intuitif untuk kemudahan analisis oleh pengguna.

Sistem yang dikembangkan melibatkan pengujian dan perbandingan beberapa model deteksi objek, yang dikelompokkan menjadi model *two-stage detector* dan *one-stage detector*. Pemilihan model dilakukan secara representatif untuk mengevaluasi kekuatan masing-masing pendekatan :

- **Faster R-CNN** dipilih sebagai perwakilan dari model ***two-stage detection***. Model ini telah terbukti memiliki akurasi tinggi dalam berbagai studi sebelumnya karena proses proposal region yang lebih selektif, menjadikannya baseline penting dalam sistem deteksi objek yang presisi[5], [6].
- **RetinaNet** dipilih sebagai model ***one-stage detection*** yang signifikan. Keunggulan RetinaNet terletak pada penggunaan focal loss untuk mengatasi ketidakseimbangan kelas (class imbalance), sehingga memperbaiki kelemahan yang terdapat pada model-model one-stage sebelumnya, dan sekaligus mendekati akurasi dari Faster R-CNN dengan efisiensi yang lebih baik[17].
- **YOLOv11** dipilih sebagai **representasi terbaru** dari model ***one-stage*** detector yang populer. Model YOLO telah lama dikenal karena kecepatannya dalam inference dan kemudahan implementasi. Versi YOLOv11 dipilih karena memiliki ukuran model yang lebih ringkas, dukungan dari komunitas, kompatibilitas dengan Pustaka-pustaka terbaru sebagai dependenciesnya[7].

Untuk kebutuhan pelatihan dan evaluasi model, dataset dirancang secara khusus dan dikembangkan secara manual. Proses pembuatan dataset melibatkan :

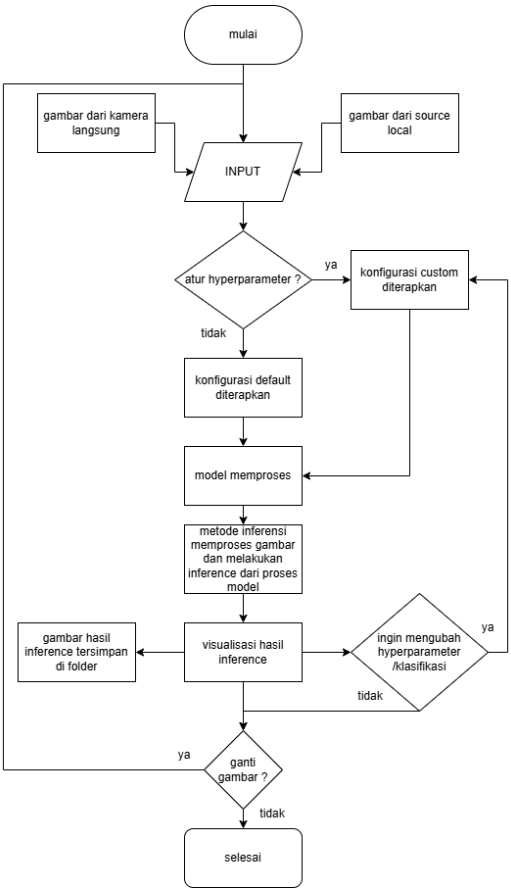
- Pembuatan dan pencetakan PCB secara manual.
- Pengambilan gambar hasil PCB menggunakan kamera.

- Penambahan cacat secara digital untuk memperluas keragaman cacat, di samping cacat yang memang terjadi secara alami dalam proses pembuatan.
- Proses anotasi dilakukan menggunakan antarmuka Label Studio.
- Dilanjutkan dengan augmentasi citra, meliputi sembilan jenis augmentasi: grayscale, blur, rotasi 15°, noise, flip, bokeh + blur, crop + pad, channel shuffle, dan HSV shift.

Seluruh model dilatih menggunakan dataset yang telah dipersiapkan dengan konfigurasi hyperparameter awal yang seragam pada beberapa aspek untuk perbandingan yang adil. Evaluasi kinerja difokuskan pada metrik validasi utama (seperti mAP, Precision, Recall dll), dengan pertimbangan tambahan terhadap kecepatan inference dan ukuran model. Model dengan performa terbaik berdasarkan kriteria tersebut kemudian dipilih sebagai fondasi sistem akhir.

Hasil deteksi dari model terpilih akan diintegrasikan ke dalam aplikasi GUI berbasis Streamlit. Antarmuka ini dirancang untuk menampilkan proses inference secara visual, memberikan output berupa anotasi pada gambar PCB, serta menyediakan fitur interaktif yang memudahkan pengguna dalam menganalisis hasil deteksi.

3.2 Alur Sistem



Gambar 3.1. proyeksi alur penggunaan protopipe sistem inspeksi PCB

Secara garis besar, sistem yang akan dibangun adalah sebuah GUI atau Graphic User Interface untuk memudahkan seorang user dalam mengamati cacat pada PCB yang diproses secara manual. Mekanisme sistem ini ketika dijalankan adalah :

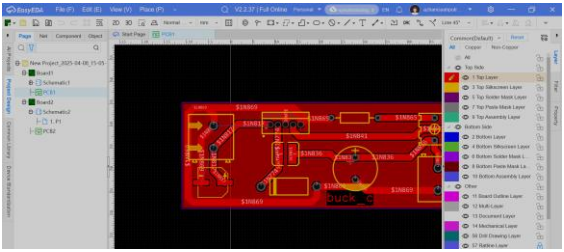
1. Ketika program dijalankan, user akan diminta untuk memberikan inputan berupa gambar dari source yang berada di device atau menggunakan kamera (menangkap gambar secara langsung)

2. Mengatur hyperparameter sebagai konfigurasi tambahan pada model dalam melakukan inference pada gambar. Contohnya klasifikasi yang mau dideteksi apa saja, dan konfigurasi lainnya.
3. Kemudian sistem akan mengingatnya sehingga proses inference akan berlangsung. Ketika proses ini berjalan... model dibantu dengan metode inference lain untuk memaksimalkan deteksi pada objek yang kecil seperti cacat pada PCB.
4. Ketika sudah selesai, sistem akan memberikan visualisasi hasil dari inference tersebut. Secara langsung pula, hasil inference ini akan tersimpan pada folder yang sudah ditentukan pada kode program.
5. Pengguna dapat mengatur ulang hyperparameter sebagai konfigurasi pada sistem. Selain itu, jika diperlukan user juga dapat melakukan inference ulang dengan mengganti inputan gambar dan mengulang prosesnya kembali tanpa mengubah konfigurasi yang sudah ditetapkan.

3.3 Dataset

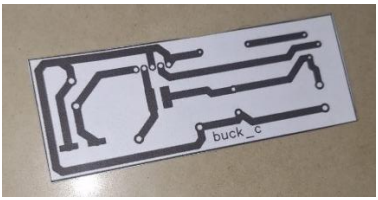
3.3.1 Pembuatan PCB Manual

Proses awal pengumpulan dataset dimulai dari menentukan design *schematic* yang akan dijadikan sebagai PCB. Terdapat 2 *schematic* yang dijadikan basis model dari dataset PCB yakni *Buck Converter* dan *Boost Converter* sebelum menuju proses produksi.



Gambar 3.2. EasyEda sebagai software electronic schematic design

1. Tahapan awal dimulai dengan perancangan skematik dan layout PCB menggunakan perangkat lunak easyEda. Rangkaian yang digunakan sebagai layout adalah ***buck converter dan boost converter***.



Gambar 3.3. layout trace yang diprint pada kertas HVS

2. Desain ini kemudian dicetak pada media khusus seperti kertas HVS yang tipis untuk mentransfer pola ke permukaan papan yang dilapisi tembaga menggunakan tinta bubuk yang biasa digunakan di mesin fotocopy atau inkjet printer.

Lotion anti nyamuk, papan lapis tembaga dan kertas dengan gambar schematic

kertas dan papan yang disatukan diberi lotion

Gambar 3.4. transfer toner dari kertas ke papan tembaga

3. Proses transfer-toner atau pemindahan tinta dari kertas menggunakan lotion anti-nyamuk. Tunggu hingga sekitar 5 menit.

Gambar 3.5. cairan Ferric Chloride

4. Setelah pola berhasil ditransfer ke papan lapis tembaga, tahap selanjutnya adalah penghilangan bagian tembaga yang tidak diperlukan

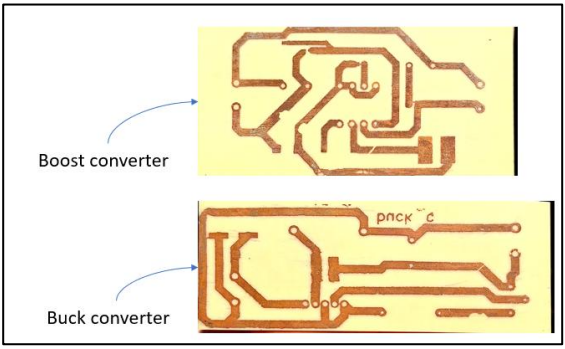
melalui proses etsa kimia. Proses ini membentuk jalur konduktif sesuai dengan desain awal menggunakan larutan air dan ferric chloride.

Gambar 3.6. PCB dibersihkan dari sisa cairan kimia pasca proses etsa(etching)

5. Tahap akhir, mengeringkan dan membersihkan hasil etsa kimia dari papan.
6. PCB dipotret untuk dijadikan basis dari dataset yang akan digunakan.

3.3.2 Pengumpulan Dataset

PCB yang sudah jadi, dilakukan pemotretan kemudain discan guna memisahkan background dengan objek pada gambar. Berikut hasil dari pemrosesan tersebut:



Gambar 3.7. 2 layout PCB yang sudah dipotret

Terdapat beberapa cacat yang secara autentik terbentuk dari pembuatan pcb manual ini. Dengan demikian, maka perlu adanya sub klasifikasi untuk

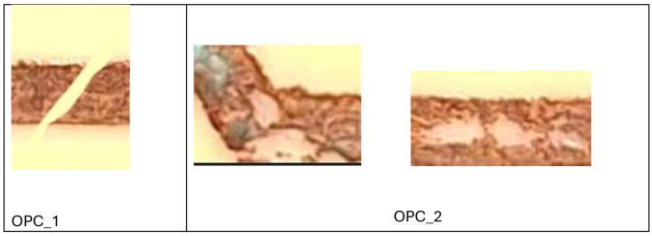
mempermudah model dan membuat deteksi bisa lebih spesifik dalam mendeteksi saat inference. Berikut ini klasifikasi yang akan diberikan dalam klasifikasi cacat pada anotasi dan deteksi.

1. **Mouse Bite**

Gambar 3.9. klasifikasi mouse bite

Hanya mencakup copper yang terkikis dari luar ke dalam (teluk) tanpa adanya kekurangan copper yang tersebar disekitarnya.

2. **Open Circuit**



Gambar 3.10. klasifikasi open circuit

Mencakup 2 kondisi, yakni yang pertama Ketika terjadi sayatan dari benda luar terhadap jalur pcb kemudian mengikis jalur tersebut dan yang ke dua yakni segala bentuk pengikisan yang terjadi pada si jalur tanpa memutus sepenuhnya.

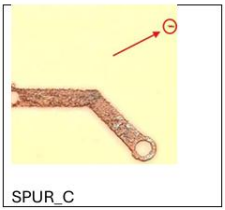
3. **Short Circuit**

Gambar 3.11. klasifikasi short circuit

Untuk memisahkan agar klasifikasi model mudah untuk mengenali, maka short dibagi menjadi 2 berdasarkan bentuk dari trace yang berada di sekitarnya. SHORT_1 akan mengenali segala bentuk

copper yang berada diantara 2 pad bulat, sedangkan SHORT_2 akan mengenali tiap copper yang menyambung antar 2 trace. Persamaan kedua klasifikasi ini didasari oleh munculnya copper tipis diantara 2 pad/trace. SHORT_3 merupakan klasifikasi terakhir yang mana copper melekat berada diantara 2 pad bulat.

4. **Spurious Copper**



Gambar 3.12. contoh spurious copper

Cacat ini diakibatkan proses pemisahan marked trace dengan lapisan copper yang tidak rapih sehingga masih menyisakan residu copper/tembaga yang tidak ikut terlarutkan ferric chloride.

5. **Spur**

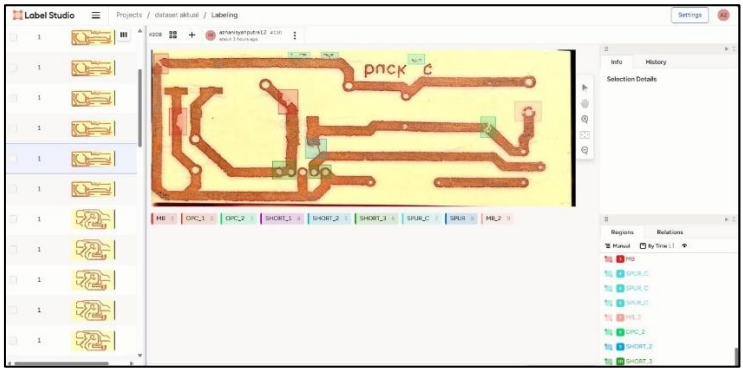
Gambar 3.13. contoh spur

Beda dengan spurious copper, spur ini terjadi akibat adanya kelebihan copper yang terjadi di jalur pcb (kebalikan daripada spurious copper). Hal ini terjadi akibat mark/penanda (tinta, dsb) jalur copper tergeser atau luntur yang mengakibatkan copper yang terkikis oleh ferric chloride tidak rapih.

3.3.3 Anotasi Data

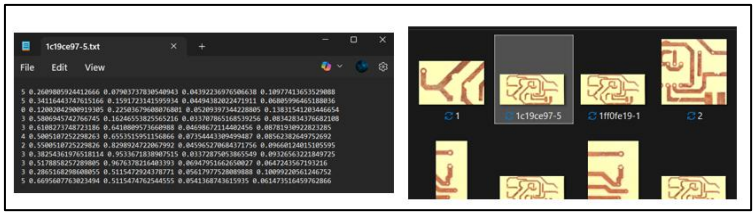
Pada proses ini, setiap cacat pada dataset akan diberikan nama kelas (classification) dan membuatnya dalam format yang bisa dibaca oleh program. Karena akan terdapat 3 model yakni Faster RCNN, Retinanet dan YOLOv11, maka format anotasi akan dibagi menjadi 2 yakni YOLO(YOLOv11) dan coco(Faster RCNN dan Retinanet).

Software yang digunakan dalam membantu proses ini Bernama Label Studio dengan tampilan sebagai berikut.



Gambar 3.14. tampilan Label-studio sebagai interface anotasi

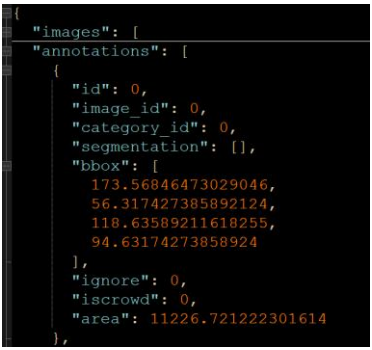
Kemudian hasil dari export anotasi ini akan terbagi menjadi berikut untuk YOLO dan coco.



Gambar 3.15. hasil anotasi format YOLO

Untuk format YOLO, dalam tiap baris anotasi menyimpan informasi berikut :

- **Angka** xxx xxx xxx
  Angka ini merepresentasikan klasifikasi cacat dalam urutan yang disusun otomatis oleh software label studio
- Angka **xxx xxx xxx**
  Xxx xxx xxx merepresentasikan bounding box dari tiap klasifikasi cacat



Gambar 3.16. hasil anotasi format coco

Sedangkan pada format coco, informasi yang termuat pada anotasinya yakni :

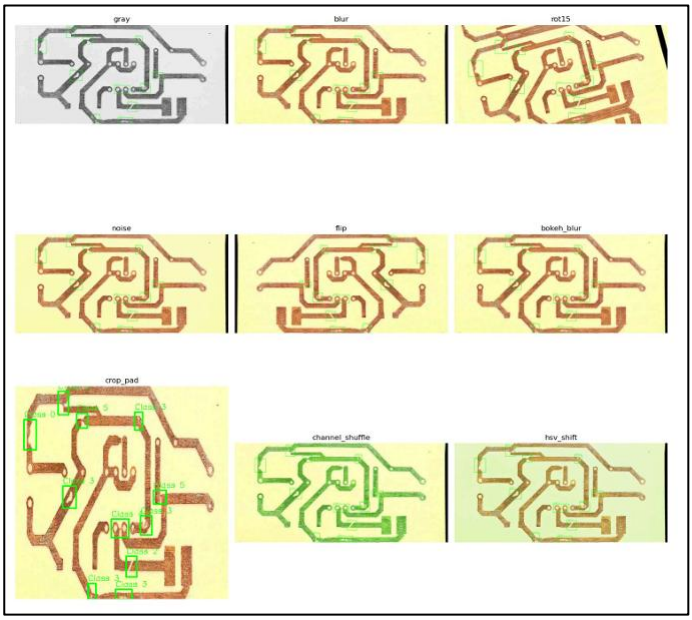
- **ID** - ID unik untuk anotasi objek ini
- **Image_id** - ID gambar tempat objek ini berada
- **Category_id** - ID klasifikasi cacat (misalnya, 0 bisa berarti "mb_1")
- **Segmentation -** Detail bentuk objek (di sini kosong, mungkin hanya info kotak)
- **Bbox –** bounding box dari klasifikasi (berisi koordinat x,y, lebar kotak dan tinggi kotak)
- **ignore –** untuk menunjukkan apakah label ini harus diabaikan atau tidak secara program
- **iscrowd** – Menunjukkan apakah anotasi ini adalah objek kelompok (crowd) atau individu.
- **area** – luas objek yang dianotasi

Anotasi dan gambar yang sudah terasosiasi akan dipisah dalam folder yang berbeda antara YOLO(images, annotations) dan coco(json file berisi annotations semua gambar, images folder). Setiap anotasi akan menyimpan identitas khusus (unique) untuk tiap gambar yang dianotasi sehingga tidak ada duplikat gambar terhdap label (penamaan klasifikasi) dan sebaliknya.

3.3.4 Augmentasi dan Oversampling data

Augmentasi merupakan salah satu metode untuk memperbanyak dataset tanpa harus melakukan pembuatan dataset tambahan ataupun edit tambahan dengan cara pengolahan citra seperti mengubah warna, rotasi, flip dll. Augmentasi yang dilakukan pada penilitian ini dikhususkan terhadap

kondisi aktual yang mungkin terjadi pada saat inference/uji coba berlangsung seperti :



Gambar 3.17. augmentasi yang diaplikasikan untuk kebutuhan dataset

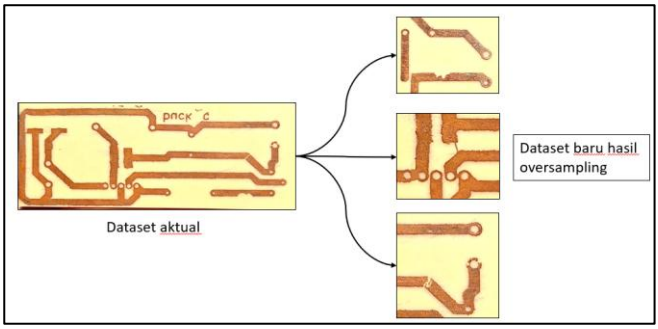
- **Grayscale**: Transformasi citra ke grayscale dilakukan untuk meningkatkan robustnes model terhadap variasi kondisi pencahayaan yang mungkin terjadi selama inspeksi, dengan mengurangi ketergantungan pada informasi warna.
- **Blur**: Penambahan efek blur bertujuan untuk meningkatkan toleransi model terhadap potensi ketidakfokusan kamera atau sedikit gerakan yang mungkin terjadi saat pengambilan gambar PCB.
- **Rotasi 15°**: Rotasi citra sebesar 15 derajat diterapkan untuk mengatasi potensi variasi kecil dalam orientasi PCB selama proses inspeksi.
- **Noise**: Penambahan noise pada citra bertujuan untuk meningkatkan ketahanan model terhadap gangguan sinyal atau noise sensor yang mungkin muncul selama akuisisi citra.

- **Flip (Horizontal dan/atau Vertikal)**: Operasi flip dilakukan untuk memungkinkan model mengenali pola cacat tanpa terpengaruh oleh orientasi spesifiknya dalam citra PCB.
- **Bokeh + Blur**: Kombinasi efek bokeh dan blur bertujuan untuk melatih model untuk fokus pada area PCB yang relevan dan mengurangi pengaruh elemen latar belakang yang tidak fokus.
- **Crop + Pad**: Operasi crop dan pad diterapkan untuk meningkatkan variasi ukuran dan posisi objek cacat dalam citra, sekaligus menjaga konsistensi ukuran input untuk model.
- **Channel Shuffle**: Pengacakan kanal warna bertujuan untuk membuat model lebih tahan terhadap distorsi warna yang tidak signifikan, sehingga lebih fokus pada fitur struktural.
- **HSV Shift (Hue, Saturation, Value Shift)**: Pergeseran pada ruang warna HSV bertujuan untuk meningkatkan adaptabilitas model terhadap perubahan kondisi pencahayaan yang mempengaruhi warna dan intensitas citra.

Pengolahan citra untuk augmentasi ini menggunakan library Albumentation dengan Bahasa pemrograman python.

Oversampling disisi lain merupakan teknik yang diterapkan untuk mengatasi masalah ketidakseimbangan kelas cacat dalam dataset. Dengan memperbanyak sampel dari kelas minoritas, model diharapkan dapat mempelajari fitur-fitur semua jenis cacat secara lebih baik dan tidak bias terhadap kelas mayoritas yang lebih sering muncul. Selain itu, augmentasi juga diterapkan pada sampel-sampel yang di-oversample untuk meningkatkan variasi data pada kelas minoritas. Oversampling dilakukan dengan cara membuat dataset baru dengan cara :

- Memotong gambar salah satu klasifikasi cacat minoritas dari dataset aktual sehingga model akan mengenali ini sebagai dataset baru karena memiliki latar citra yang berbeda.



Gambar 3.18. relasi oversampling terhadap dataset aktual

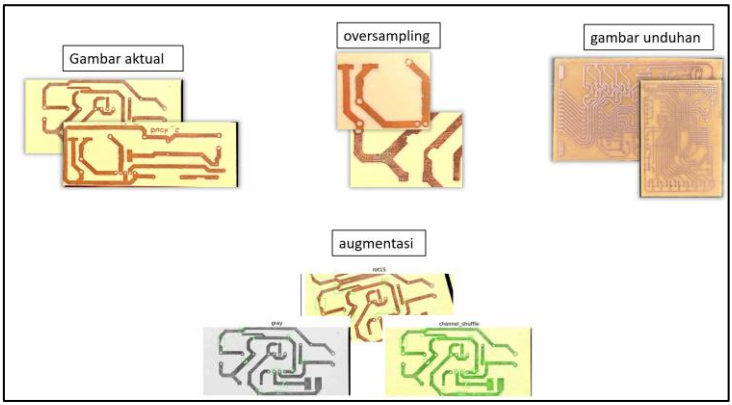
- Mencari gambar PCB dari sumber internet dengan kondisi sama (unmask dan diproduksi secara manual) dengan klasifikasi cacat khusus yang ingin diperoleh sebagai penambah variasi jika tidak ada maka dilakukan edit foto pada software photoshop untuk menambahkan. Pada penelitian ini, dataset diunduh melalui situs Roboflow.

Gambar 3.19. dataset PCB yang diunduh melalui roboflow

3.3.5 Pembagian Dataset

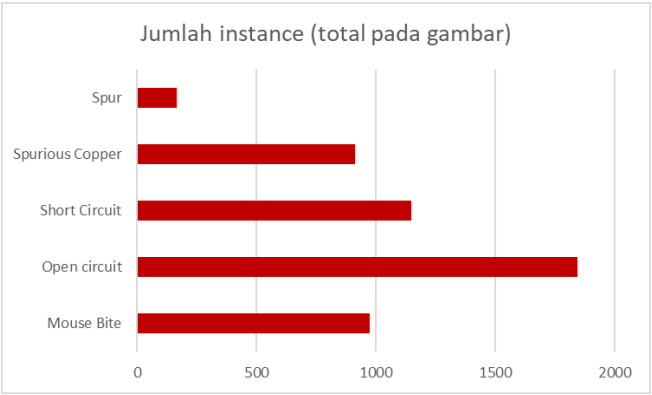
Dalam proses *fine-tuning* model, struktur direktori data yang terorganisir dengan baik sangat penting. Umumnya, diperlukan dua direktori utama: ***Training dan Validation***. Setiap direktori ini berisi dua sub-direktori yang saling berkorespondensi:

a. **annotations** (berisi file anotasi yang mendefinisikan objek dan kelasnya)
b. **images** (berisi gambar PCB yang sesuai dengan file anotasi).



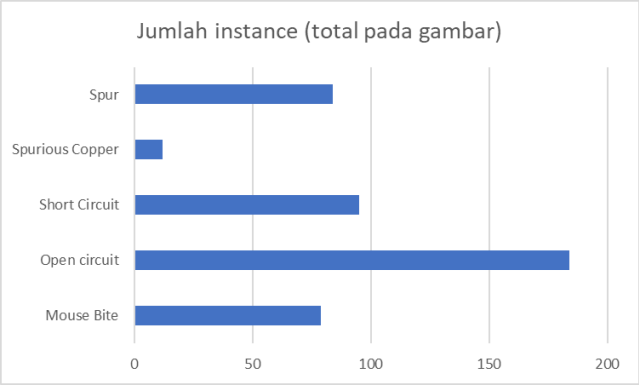
Gambar 3.20. komposisi dataset

- **Data Training:**
  Dataset *training* berjumlah **792** citra PCB, yang berasal dari 88 gambar PCB awal(gambar aktual). Pembentukan dataset ini melibatkan kombinasi gambar PCB unduhan, PCB hasil editan dan augmentasi, dan aplikasi teknik *oversampling* untuk menyeimbangkan representasi kelas cacat. Selanjutnya, setiap gambar PCB (termasuk hasil *oversampling*) diaugmentasi sebanyak 9 varian menggunakan berbagai teknik transformasi citra. Tujuan augmentasi ini adalah untuk memperkaya variasi data latih dan meningkatkan robustnes model terhadap berbagai kondisi citra.



Gambar 3.21. chart jumlah instance (klasifikasi) pada dataset train

- **Data Validasi:**
  Dataset validasi berjumlah **87** gambar PCB, yang juga terdiri dari PCB unduhan dan PCB hasil editan dari gambar aktual, serta melibatkan *oversampling* untuk memastikan representasi kelas yang baik. Berbeda dengan data *training*, data validasi tidak mengalami proses augmentasi. Hal ini bertujuan untuk menyediakan dataset yang merepresentasikan kondisi citra nyata yang akan dihadapi model saat pengujian atau implementasi, sehingga evaluasi kinerja model menjadi lebih akurat dan tidak bias oleh variasi artifisial dari augmentasi.



Gambar 3.22. chart jumlah instance (klasifikasi) pada dataset validasi

Komposisi dataset *training* yang diperkaya melalui *oversampling* dan augmentasi bertujuan untuk melatih model secara komprehensif, sementara dataset validasi yang tidak diaugmentasi berfungsi sebagai tolok ukur yang realistis untuk menguji kemampuan generalisasi model terhadap data baru.

3.4 Konfigurasi Model *Object Detection*

Penelitian ini mengimplementasikan dan membandingkan tiga arsitektur model deteksi objek utama untuk tugas inspeksi cacat PCB: Faster R-CNN, RetinaNet, dan YOLOv11. Sebagaimana telah diuraikan pada Bab 2 mengenai karakteristik dasar masing-masing arsitektur, sub-bab ini akan merinci konfigurasi spesifik dan hyperparameter yang diterapkan selama proses pelatihan. Untuk inisialisasi bobot, seluruh model memanfaatkan teknik transfer learning dengan menggunakan pre-trained weights yang telah dilatih pada dataset COCO.

3.4.1 Konfigurasi Model Faster R-CNN

Implementasi model Faster R-CNN menggunakan *library* Detectron2. Konfigurasi pelatihan diambil dari file dasar COCO-Detection/faster_rcnn_R_50_FPN_3x.yaml yang terdapat pada Model Zoo Detectron2, dengan penyesuaian parameter sebagai berikut:

- **Sumber Bobot Awal (Pre-trained Weights)**: model_zoo.get_checkpoint_url("COCO-Detection/faster_rcnn_R_50_FPN_3x.yaml")
    - **Deskripsi**: Bobot model yang telah dilatih sebelumnya pada dataset besar (COCO) yang digunakan sebagai titik awal pelatihan untuk mempercepat konvergensi dan meningkatkan performa.
- **Dataset Pelatihan (Training Dataset)**: ("train",)
    - **Deskripsi**: Nama atau identifier untuk set data yang digunakan secara eksklusif untuk melatih model.
- **Dataset Validasi (Validation Dataset)**: ("valid",)
    - **Deskripsi**: Nama atau identifier untuk set data terpisah yang digunakan untuk mengevaluasi performa model selama proses pelatihan, membantu dalam penyesuaian *hyperparameter* dan pemantauan *overfitting*.
- **Jumlah Pekerja Dataloader (Number of Dataloader Workers)**: 2
    - **Deskripsi**: Jumlah proses paralel yang bertugas memuat data dari penyimpanan ke memori, mempercepat persiapan *batch* data untuk pelatihan.
- **Ukuran Batch per Iterasi (Batch Size)**: 16
    - **Deskripsi**: Jumlah gambar yang diproses oleh model dalam satu siklus maju (*forward pass*) dan mundur (*backward pass*) sebelum bobot model diperbarui.
- **Algoritma Optimasi (Optimizer Algorithm)**: "AdamW"
    - **Deskripsi**: Algoritma yang digunakan untuk menyesuaikan bobot model berdasarkan gradien dari fungsi *loss* (misalnya, AdamW, SGD, Adam).
- **Laju Pembelajaran Awal (Initial Learning Rate)**: 0.001
    - **Deskripsi**: Ukuran langkah yang digunakan oleh algoritma optimasi untuk memperbarui bobot model.

- **Jumlah Iterasi Pelatihan Maksimum (Maximum Training Iterations)**: 3000
    - **Deskripsi**: Jumlah total pembaruan bobot (iterasi) yang dilakukan selama keseluruhan proses pelatihan model.
- **Jadwal Penurunan Laju Pembelajaran (Learning Rate Schedule Steps)**: [2500, 4000]
    - **Deskripsi**: Iterasi spesifik di mana laju pembelajaran akan diturunkan secara bertahap untuk membantu model konvergen ke solusi yang lebih baik.
- **Bobot Reduksi (Weight Decay / L2 Regularization)**: 0.0005
    - **Deskripsi**: Koefisien untuk teknik regularisasi L2 yang menambahkan penalti pada besarnya bobot model untuk mencegah *overfitting*.
- **Ukuran Batch per Gambar untuk ROI Heads (ROI Head Batch Size Per Image)**: 128
    - **Deskripsi**: (Spesifik untuk *two-stage detectors*) Jumlah *Region of Interest* (RoI) yang diproses oleh kepala klasifikasi dan regresi per gambar selama pelatihan.
- **Jumlah Kelas Output (Number of Output Classes)**: 9
    - **Deskripsi**: Jumlah total kategori atau jenis cacat PCB yang harus dideteksi oleh model.
- **Ambang Batas Skor Deteksi Saat Pengujian (Test Score Threshold)**: 0.2
    - **Deskripsi**: Nilai kepercayaan minimum yang harus dimiliki sebuah deteksi agar dianggap valid selama tahap evaluasi atau inference.

3.4.2 Konfigurasi Model RetinaNet

Model RetinaNet juga diimplementasikan menggunakan Detectron2, dengan konfigurasi dasar dari COCO-Detection/retinanet_R_101_FPN_3x.yaml dan penyesuaian berikut:

- **Sumber Bobot Awal (Pre-trained Weights)**: model_zoo.get_checkpoint_url("COCO-Detection/retinanet_R_101_FPN_3x.yaml")
- **Dataset Pelatihan (Training Dataset)**: ("train",)
- **Dataset Validasi (Validation Dataset)**: ("valid",)
- **Jumlah Pekerja Dataloader (Number of Dataloader Workers)**: 2
- **Ukuran Batch per Iterasi (Batch Size)**: 2

- **Laju Pembelajaran Awal (Initial Learning Rate)**: 0.001
- **Jumlah Iterasi Pelatihan Maksimum (Maximum Training Iterations)**: 1000
- **Jadwal Penurunan Laju Pembelajaran (Learning Rate Schedule Steps)**: []
    - **Deskripsi**: Untuk konfigurasi ini, tidak ada penurunan laju pembelajaran terjadwal pada iterasi spesifik, laju pembelajaran awal cenderung dipertahankan atau diatur oleh *scheduler default framework* jika ada.
- **Jumlah Kelas Output (Number of Output Classes)**: 9

3.4.3 Konfigurasi Model YOLOv11

Model YOLOv11 dilatih menggunakan *framework* Ultralytics. Berikut adalah parameter pelatihan yang diterapkan:

- **Sumber Bobot Awal (Pre-trained Weights)**: "YOLO11n.pt" *(Sesuaikan dengan nama file model .pt yang benar untuk versi YOLO Anda)*
- **File Konfigurasi Dataset**: "/content/training_ta (2).yaml"

```
train : "/content/must_bethefinal/train"
val : "/content/must_bethefinal/valid"

nc : 9
names : ['MB', 'MB_1', 'OPC_1', 'OPC_2', 'SHORT_1', 'SHORT_2', 'SHORT_3', 'SPUR_C', 'SPUR']
```

    - **Deskripsi**: File YAML yang mendefinisikan path ke data latih dan validasi, serta informasi jumlah dan nama kelas.
- **Jumlah Epoch Pelatihan (Number of Training Epochs)**: 200
    - **Deskripsi**: Jumlah berapa kali keseluruhan dataset pelatihan akan dilewatkan melalui model selama proses pelatihan.
- **Ukuran Gambar Input (Input Image Size)**: 640
    - **Deskripsi**: Resolusi (lebar dan tinggi dalam piksel) gambar yang diumpankan ke model.
- **Ukuran Batch (Batch Size)**: 16
- **Perangkat Pelatihan (Training Device)**: "cuda"
    - **Deskripsi**: Menentukan perangkat keras yang digunakan untuk pelatihan (misalnya, "cuda" untuk GPU NVIDIA, "cpu" untuk CPU).
- **Algoritma Optimasi (Optimizer Algorithm)**: "AdamW"
- **Laju Pembelajaran Awal (Initial Learning Rate)**: 0.001
- **Kesabaran untuk Penghentian Awal (Patience for Early Stopping)**: 20

    - **Deskripsi**: Jumlah epoch untuk menunggu tanpa adanya peningkatan signifikan pada metrik validasi sebelum pelatihan dihentikan secara otomatis.
- **Presisi Campuran Otomatis (Automatic Mixed Precision - AMP)**: True
    - **Deskripsi**: Teknik yang menggunakan kombinasi presisi *floating-point* 16-bit dan 32-bit untuk mempercepat pelatihan dan mengurangi penggunaan memori GPU.
- **Probabilitas Augmentasi Mosaik (Mosaic Data Augmentation Probability)**: 0.5
    - **Deskripsi**: Peluang diterapkannya teknik augmentasi data mosaik, di mana empat gambar pelatihan digabungkan menjadi satu.
- **Bobot Reduksi (Weight Decay / L2 Regularization)**: 0.0005
- **Faktor Laju Pembelajaran Akhir (Final Learning Rate Factor - lrf)**: 0.01
    - **Deskripsi**: Digunakan dalam *scheduler* seperti Cosine Annealing untuk menentukan laju pembelajaran akhir sebagai fraksi dari laju pembelajaran awal (lr_final = lr_initial * lrf).

3.4 Implementasi Metode Peningkatan Deteksi

Untuk meningkatkan akurasi dan kemampuan sistem dalam mendeteksi cacat pada PCB, terutama dalam mengatasi tantangan seperti tumpang tindih deteksi (overlapping boxes) dan deteksi objek berukuran kecil, penelitian ini mengimplementasikan dua metode tambahan: Non-Maximum Suppression (NMS) yang disesuaikan dan Slicing Aided Hyper Inference (SAHI).

3.4.1 Non-Maximum Suppression

Non-Maximum Suppression (NMS) adalah teknik pasca-pemrosesan esensial dalam object detection yang berfungsi untuk menghilangkan bounding box redundan yang mendeteksi objek yang sama[12]. Meskipun framework seperti Detectron2 sudah memiliki implementasi NMS bawaan, penelitian ini memastikan konfigurasi NMS dioptimalkan untuk kasus deteksi cacat PCB, dan juga menerapkan fungsi NMS kustom jika diperlukan setelah proses inference atau setelah penggabungan hasil dari metode lain seperti SAHI.

Fungsi NMS yang digunakan dalam penelitian ini didefinisikan sebagai berikut:

```python
def apply_nms(predictions, iou_threshold=0.5):
    """ Filter overlapping bounding boxes using Non-Maximum Suppression (NMS). """
    boxes = predictions.pred_boxes.tensor
    scores = predictions.scores
    keep = torch.ops.torchvision.nms(boxes, scores, iou_threshold)
    return predictions[keep]
```

Gambar 3.23. kode function sederhana NMS

Dalam fungsi **apply_nms** di atas:

- **predictions**: Merupakan objek output dari model deteksi yang berisi informasi *bounding box* dan skor kepercayaannya.
- **iou_threshold**: Parameter yang menentukan ambang batas *Intersection over Union* (IoU). Jika dua *bounding box* memiliki IoU lebih besar dari *threshold* ini, salah satunya (yang memiliki skor lebih rendah) akan dihilangkan.
- **predictions.pred_boxes.tensor**: Mengakses tensor yang berisi koordinat dari semua *bounding box* yang diprediksi.
- **predictions.scores**: Mengakses tensor yang berisi skor kepercayaan untuk setiap *bounding box*.
- **torch.ops.torchvision.nms(boxes, scores, iou_threshold)**: Ini adalah pemanggilan fungsi NMS yang disediakan oleh pustaka torchvision. Fungsi ini akan mengembalikan indeks dari *bounding box* yang dipertahankan setelah proses NMS.
- **predictions[keep]**: Mengindeks objek predictions asli menggunakan indeks keep untuk hanya mengambil *bounding box* yang tidak ditekan oleh NMS.

➢ **Parameter**: Nilai **IoU *threshold*** untuk NMS dalam fungsi kustom ini dan juga pada konfigurasi bawaan model diatur pada **0.5**.

➢ **Penerapan pada Model**: Penyesuaian dan verifikasi parameter NMS ini diterapkan setelah proses inference pada model Faster R-CNN dan RetinaNet. Untuk model YOLOv11, NMS sudah menjadi bagian integral dari proses inferencenya dan dikontrol melalui parameter bawaan, namun fungsi apply_nms dapat digunakan jika ada pemrosesan lebih lanjut atau penggabungan hasil prediksi dari sumber berbeda.

3.4.2 Penerapan *Slicing Aided Hyper Inference* (SAHI)

Untuk mengatasi tantangan deteksi cacat PCB yang seringkali berukuran sangat kecil dan mungkin terlewatkan oleh model yang dilatih pada resolusi gambar penuh, metode Slicing Aided Hyper Inference (SAHI) diimplementasikan. SAHI bekerja dengan membagi gambar input beresolusi

tinggi menjadi beberapa potongan (slices) yang lebih kecil, melakukan inference pada setiap potongan, dan kemudian menggabungkan hasilnya kembali secara cerdas. Berikut adalah langkah-langkah utama dan konfigurasi yang digunakan:

1. **Inisialisasi**

```
from sahi import AutoDetectionModel
from sahi.utils.cv import read_image
from sahi.utils.file import download_from_url
from sahi.predict import get_prediction, get_sliced_prediction, predict
from IPython.display import Image

detection_model = AutoDetectionModel.from_pretrained(
    model_type='ultralytics',
    model_path="/content/drive/MyDrive/TA/yolov11/detect/hasil training yolo v11/weights/best.pt",
    confidence_threshold=0.2,
    device="cuda" )
```

Gambar 3.24. inisialisasi SAHI

Dalam kode di atas, **model_path** merujuk pada bobot model YOLOv11 yang telah dilatih sebelumnya. **confidence_threshold** awal diatur untuk model sebelum SAHI melakukan slicing dengan bantuan model.

2. **Proses *Sliced Prediction***

```
from sahi.predict import get_sliced_prediction

result = get_sliced_prediction(
    "/content/1.jpg",
    detection_model,
    slice_height=128,
    slice_width=128,
    overlap_height_ratio=0.3,
    overlap_width_ratio=0.3,
)
result.export_visuals(export_dir="demo_data/")
Image("demo_data/prediction_visual.png")
```

Gambar 3.25. inisialisasi parameter pada slicing SAHI

Parameter **slice_height** dan **slice_width** menentukan dimensi setiap potongan, sedangkan **overlap_height_ratio** dan **overlap_width_ratio** memastikan kontinuitas deteksi pada area perbatasan antar potongan**.**

3. **Penggabungan**

Library/pustaka SAHI secara internal menangani penggabungan prediksi dari semua potongan dan menerapkan NMS pada hasil gabungan tersebut untuk menghilangkan deteksi duplikat. **result.export_visuals()** dapat digunakan untuk menyimpan visualisasi hasil deteksi.

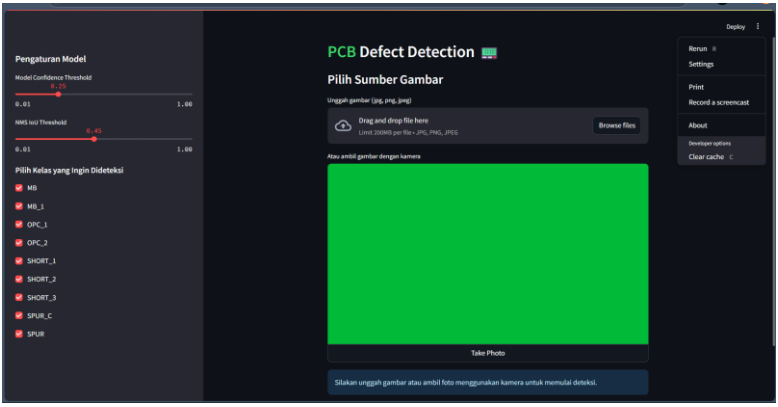
➢ **Parameter yang digunakan** :

- slice_height dan slice width = 128x128 pixel
- overlapse_ratio = 0.3 (30% panjang dan lebar)

- confidence_threshold = 0.2 (ambang batas 20% untuk prediksi)

Model SAHI hanya mendukung untuk diterapkan pada beberapa model termasuk model YOLO terbaru untuk saat ini.

3.5 Graphic User Interface



Gambar 3.26. desain GUI untuk sistem inspeksi PCB dengan parameter NMS

Pengembangan Antarmuka Pengguna Grafis (GUI) bertujuan untuk menyediakan sarana interaksi yang intuitif dan efisien bagi pengguna untuk memanfaatkan sistem deteksi cacat PCB yang telah dibangun. GUI ini memungkinkan pengguna untuk melakukan inspeksi visual, melihat hasil deteksi secara langsung, dan menganalisis performa model.

Pustaka/library yang digunakan untuk membangun ini adalah :

- streamlit versi 1.42.0
- opencv versi 4.10.0.84
- sahi 0.11.22
- ultralytics 8.3.91

3.5.1 Tujuan dan Fungsionalitas GUI

- **Tujuan Utama**:
    - Mempermudah proses inspeksi PCB dengan menyediakan platform visual.
    - Menampilkan hasil deteksi cacat secara jelas dan informatif.
    - Memungkinkan pengguna untuk berinteraksi dengan hasil prediksi dan mengevaluasi kinerja sistem secara kualitatif.

- Menyediakan alat bantu untuk analisis lebih lanjut terhadap jenis dan lokasi cacat.

- **Fungsionalitas Utama yang Dikembangkan**:
  1. **Unggah Gambar PCB**: Pengguna dapat mengunggah gambar PCB yang akan diinspeksi melalui kamera yang terhubung atau folder pada device program berjalan. Dapat memberikan 2 inputan sekaligus walau pada akhirnya harus mematikan salah satu inputan sebelum mendeteksi.
  2. **Seleksi Klasifikasi Cacat** : pengguna dapat memilih klasifikasi cacat mana yang hendak diamati. Secara otomatis, inference juga hanya akan menampilkan klasifikasi yang dipilih.
  3. **Hyperparameter tuning** : pengaturan hyperparameter pada metode inference dapat diatur secara langsung melalui slider intuitif.
  4. **Visualisasi Hasil Deteksi**:
     - Menampilkan gambar PCB asli dengan *bounding box* yang menandai area cacat yang terdeteksi.
     - Menampilkan label kelas (jenis cacat) dan skor kepercayaan (*confidence score*) untuk setiap deteksi.
     - Tabel untuk melihat rangkuman deteksi
     - Gambar dapat diperbesar dan diperkecil baik inputan maupun hasil inference.
     - Gambar hasil inference otomatis tersimpan pada folder yang sudah diatur.

3.6 Metrik dan Prosedur Evaluasi Kinerja

Evaluasi kinerja sistem deteksi cacat PCB dilakukan secara kuantitatif dan kualitatif untuk mendapatkan pemahaman yang komprehensif mengenai efektivitas model dan metode yang diusulkan.

3.6.1 Metrik Evaluasi Kuantitatif

Untuk memastikan perbandingan yang adil dan objektif antara kinerja model *object detection* yang diuji (Faster R-CNN, RetinaNet, dan YOLOvX), evaluasi kuantitatif akan difokuskan pada metrik-metrik inti yang dapat diukur secara konsisten pada ketiga model tersebut:

1. ***Mean Average Precision (mAP)@0.5***: ...
2. ***Mean Average Precision (mAP)@0.5:0.95***: ...
3. **Ukuran Model (*Model Size*)**: ...

Metrik inti ini dipilih karena secara bersama-sama memberikan evaluasi paling komprehensif dan seimbang terhadap aspek krusial kinerja model object detection untuk penelitianmu:

a. **Mean Average Precision (mAP)**: Merupakan standar industri dan akademik untuk menilai kualitas deteksi objek secara keseluruhan.
   - mAP@0.5 digunakan karena mengukur kemampuan dasar model dalam mendeteksi dan mengklasifikasikan objek dengan tumpang tindih (Intersection over Union - IoU) yang moderat (50%) dengan ground truth, menunjukkan akurasi deteksi secara umum[13].
   - mAP@0.5:0.95 (standar COCO) dipilih karena memberikan evaluasi yang lebih ketat dan detail terhadap kemampuan model dalam melokalisasi objek dengan presisi tinggi pada berbagai tingkat IoU. Ini menguji seberapa akurat bounding box yang dihasilkan. Menggunakan kedua varian mAP ini memberikan gambaran lengkap, dari deteksi umum hingga presisi lokalisasi[13].

b. **Ukuran Model (Model Size)**: Metrik ini sangat penting untuk menilai efisiensi dan kelayakan praktis model. Ukuran model yang lebih kecil seringkali berarti kebutuhan sumber daya (memori, penyimpanan) yang lebih rendah dan potensi kecepatan inferensi yang lebih tinggi, yang krusial untuk implementasi di dunia nyata, terutama pada perangkat dengan keterbatasan[7].

Meskipun berbagai metrik dapat dihasilkan, analisis perbandingan utama antar ketiga model akan ditekankan pada nilai mAP@0.5, mAP@0.5:0.95, dan Ukuran Model untuk memastikan objektivitas evaluasi.

3.6.2 Prosedur Evaluasi

Proses evaluasi kinerja akan mengikuti langkah-langkah berikut:

1. **Penggunaan Dataset Uji**: Seluruh model dan konfigurasi metode akan dievaluasi menggunakan dataset yang terpisah dari data uji.
2. **Perbandingan Antar Model Berdasarkan Metrik Inti**: Kinerja ketiga model utama (Faster R-CNN, RetinaNet, YOLOv11) akan dibandingkan secara langsung berdasarkan nilai mAP@0.5, mAP@0.5:0.95, dan Ukuran Model.
3. **Evaluasi Kualitatif Metode Peningkatan**:
   Optimalisasi dan penerapan **NMS** (sebagaimana dijelaskan pada 3.4.1), khususnya untuk memastikan output yang bersih dari Faster R-CNN dan

RetinaNet atau sebagai langkah pasca-pemrosesan umum, akan dievaluasi dampaknya secara kualitatif melalui observasi visual terhadap hasil deteksi.

4. Jika model YOLOv11 terpilih sebagai model utama, penerapan SAHI (sebagaimana dijelaskan pada 3.4.2) akan dievaluasi kemampuannya dalam meningkatkan deteksi objek kecil. Peningkatan ini akan didemonstrasikan dan dianalisis secara kualitatif melalui studi kasus visual pada gambar-gambar yang mengandung cacat berukuran kecil, membandingkan hasil deteksi YOLOv11 dengan dan tanpa SAHI secara visual.
5. **Analisis Kualitatif Tambahan**: Selain poin di atas, analisis kualitatif umum akan dilakukan dengan mengamati visualisasi hasil deteksi melalui GUI untuk mengidentifikasi pola keberhasilan atau kegagalan spesifik dari sistem pada berbagai jenis cacat dan kondisi gambar.

Bab 4.
Hasil dan Pembahasan

Bab ini menyajikan hasil dari implementasi dan evaluasi sistem deteksi cacat PCB menggunakan tiga model *object detection*: Faster R-CNN, RetinaNet, dan YOLOv11n. Pembahasan akan difokuskan pada analisis kinerja kuantitatif berdasarkan metrik yang telah ditetapkan, yang mengarah pada pemilihan model terbaik untuk implementasi lebih lanjut, serta analisis kualitatif terhadap kemampuan deteksi dan dampak penerapan metode peningkatan.

4.1 Perbandingan Kinerja Kuantitatif Model *Object Detection*

Ketiga model *object detection* telah dilatih menggunakan dataset dan konfigurasi yang telah diuraikan pada Bab 3. Evaluasi dilakukan pada dataset uji untuk membandingkan kinerjanya secara objektif. Fokus perbandingan adalah pada metrik *Mean Average Precision* (mAP) pada ambang batas IoU 0.5 (mAP@0.5) dan mAP pada rentang IoU 0.5 hingga 0.95 (mAP@0.5:0.95), ukuran model, serta kecepatan inference.

Hasil evaluasi kuantitatif dari ketiga model disajikan pada Tabel di bawah ini. Kecepatan inference diukur berdasarkan waktu rata-rata yang dibutuhkan untuk memproses 10 gambar dari dataset validasi.

Tabel 4.1. rangkuman metrik validasi model terhadap dataset

Model	mAP@0.5	mAP@0.5:0.95	Ukuran Model (MB)	Kecepatan Inference (rata-rata per 10 gambar)
Faster R-CNN	0.916	0.491	315	12.3 detik
RetinaNet	0.924	0.588	422.7	5.4 detik
YOLOv11n	0.89	0.566	5.2	2.31 detik

Berdasarkan Tabel 1 , model RetinaNet menunjukkan performa akurasi tertinggi di antara ketiga model yang diuji, dengan nilai mAP@0.5 sebesar 0.924 dan mAP@0.5:0.95 sebesar 0.588. Model Faster R-CNN mengikuti dengan mAP@0.5 sebesar 0.916 dan mAP@0.5:0.95 sebesar 0.491.

Namun, ketika mempertimbangkan efisiensi, model YOLOv11n menunjukkan keunggulan yang sangat signifikan. Dengan ukuran model hanya 5.2 MB dan kecepatan inference tercepat (2.31 detik untuk 10 gambar), YOLOv11n jauh

mengungguli Faster R-CNN (315.0 MB; 12.3 detik) dan RetinaNet (422.7 MB; 5.4 detik). Meskipun nilai mAP YOLOv11n (mAP@0.5: 0.890; mAP@0.5:0.95: 0.566) sedikit di bawah RetinaNet, kombinasi ukuran model yang sangat kecil dan kecepatan inference yang superior menjadikannya kandidat yang sangat kuat untuk aplikasi yang membutuhkan respons cepat dan efisiensi sumber daya.

Selain itu, perlu dicatat bahwa YOLOv11n mencapai performa terbaiknya pada epoch ke-24 dari total 45 epoch yang dijalankan, mengindikasikan efisiensi dalam proses pelatihan berkat mekanisme early stopping. Sebaliknya, model Faster R-CNN dan RetinaNet berjalan hingga seluruh iterasi yang ditetapkan selesai.

Temuan ini menyoroti adanya *trade-off* yang jelas antara akurasi deteksi tertinggi dengan efisiensi operasional (ukuran model dan kecepatan inference). Pemilihan model terbaik akan sangat bergantung pada prioritas kebutuhan aplikasi akhir.

4.2 Pemilihan Model Deteksi Objek Utama

Berdasarkan hasil evaluasi kuantitatif yang disajikan pada Tabel 4.1, terlihat bahwa meskipun RetinaNet menawarkan akurasi mAP tertinggi, **YOLOv11n menunjukkan kombinasi kinerja yang paling seimbang untuk tujuan penelitian ini**, terutama ketika mempertimbangkan efisiensi. Dengan ukuran model yang hanya **5.2 MB** dan kecepatan inference rata-rata **2.31 detik untuk 10 gambar (sekitar 4.33 FPS)**, YOLOv11n menawarkan keunggulan signifikan dalam hal portabilitas, potensi implementasi pada perangkat dengan sumber daya terbatas, dan kemampuan untuk respons yang lebih cepat.

Meskipun mAP YOLOv11n sedikit di bawah RetinaNet, perbedaan tersebut dianggap dapat diterima mengingat keuntungan besar dalam efisiensi. Selain itu, arsitektur YOLOv11n yang sudah secara inheren mengimplementasikan *Non-Maximum Suppression* (NMS) secara efektif menyederhanakan alur pasca-pemrosesan. Oleh karena itu, **YOLOv11n dipilih sebagai model deteksi objek utama** untuk pengembangan sistem inspeksi visual cacat PCB lebih lanjut dan akan diuji dengan metode peningkatan inference SAHI.

4.3 Evaluasi Penerapan *Slicing Aided Hyper Inference* (SAHI) pada YOLOv11n

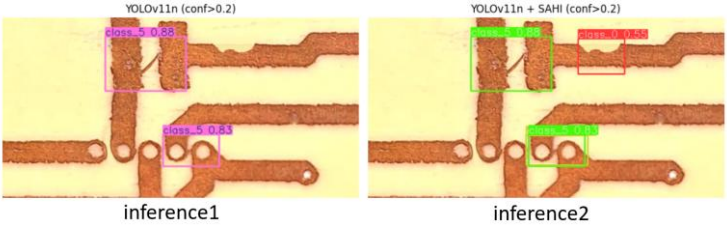
Setelah YOLOv11n terpilih sebagai model utama, metode *Slicing Aided Hyper Inference* (SAHI) diterapkan. Tujuan utama penerapan SAHI adalah untuk meningkatkan sensitivitas model terhadap cacat berukuran kecil yang mungkin terlewatkan oleh inference standar. Evaluasi dampak SAHI dilakukan melalui analisis kualitatif dan perbandingan visual hasil deteksi.

Meskipun pengamatan menunjukkan bahwa SAHI dapat meningkatkan jumlah deteksi cacat, terutama untuk objek berukuran kecil, terkadang juga teridentifikasi adanya potensi *false positive*. Namun, untuk tujuan penelitian ini yang menekankan pada kemampuan untuk mengidentifikasi sebanyak mungkin potensi cacat (dengan asumsi verifikasi lebih lanjut dapat dilakukan), peningkatan cakupan deteksi oleh SAHI dianggap sebagai keunggulan.

4.3.1 Analisis Visual Dampak SAHI pada Deteksi Objek Kecil

Untuk mempermudah pemahaman terhadap hasil deteksi visual yang akan disajikan, berikut adalah pemetaan antara nama kelas cacat yang digunakan dalam penelitian ini dengan indeks kelas numerik yang mungkin direferensikan oleh output model:

- Kelas 0: MB (Mouse Bite)
- Kelas 1: MB_1 (Mouse Bite Tipe 1)
- Kelas 2: OPC_1 (Open Circuit Tipe 1)
- Kelas 3: OPC_2 (Open Circuit Tipe 2)
- Kelas 4: SHORT_1 (Short Circuit Tipe 1)
- Kelas 5: SHORT_2 (Short Circuit Tipe 2)
- Kelas 6: SHORT_3 (Short Circuit Tipe 3)
- Kelas 7: SPUR_C (Spurious Copper)
- Kelas 8: SPUR (Spur)

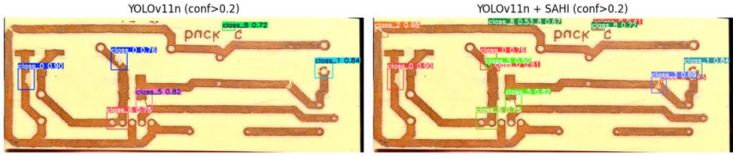


Gambar 4.1. Hasil inference YOLOv11n v YOLOv11n + SAHI

Kasus 1 : pada contoh kasus diatas, dapat diamati terdapat 3 cacat pada pcb tersebut yakni short_1, short_2 dan mb_1, berikut model mendeteksi cacat tersebut :

- YOLOv11n hanya mendeteksi **2 cacat** yakni **2 short(short_1)**
- YOLOv11n+SAHI dapat mendeteksi 4 cacat yakni 3 short (short_1 dan short_3) dan 1 mouse bite (mb_1)

Terdapat False positive dari segi klasifikasi walau mendeteksi benar adanya cacat disitu. Dapat disimpulkan SAHI membantu model untuk mendeteksi adanya cacat dengan baik walau model YOLOv11n sebagai backbonenya belum begitu optimal dalam mengklasifikasikan cacat tersebut.



inference1 inference2

Gambar 4.2. hasil inference YOLOv11n v YOLOv11n + SAHI

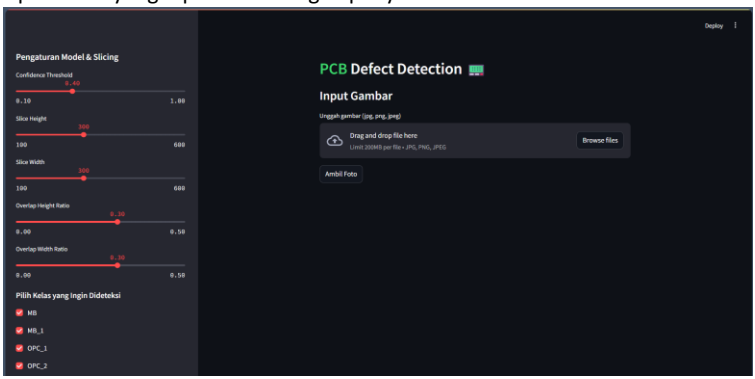
Kasus 2 : menyajikan kasus yang lebih kompleks di mana PCB memiliki total sepuluh cacat aktual, terdiri dari dua *open circuit* ('OPC_1'), tiga *mouse bite* (dua diklasifikasikan sebagai 'MB' dan satu sebagai 'MB_2'), dua *short circuit* ('SHORT_1' dan 'SHORT_2'), dan tiga *spurious copper*.

- **Inference1** hanya mampu mendeteksi enam dari sepuluh cacat tersebut.
- Sebaliknya, ketika SAHI diterapkan pada YOLOv11n (**Inference2**), jumlah deteksi meningkat drastis menjadi empat belas. Peningkatan jumlah deteksi ini berhasil mencakup lebih banyak cacat aktual yang sebelumnya terlewat, mendekati jumlah sebenarnya. Meskipun jumlah deteksi melebihi jumlah cacat aktual (mengindikasikan adanya beberapa *false positive* atau deteksi ganda yang mungkin belum sepenuhnya tersaring oleh NMS pasca-SAHI), kemampuan SAHI untuk mengungkap lebih banyak area cacat potensial sangat terlihat. Peningkatan sensitivitas ini krusial untuk memastikan tidak ada cacat signifikan yang terlewat, meskipun memerlukan tahap verifikasi lebih lanjut untuk beberapa deteksi.

Dari observasi visual ini, disimpulkan bahwa SAHI secara efektif meningkatkan sensitivitas model YOLOv11n terhadap cacat-cacat minor. Meskipun ada potensi munculnya beberapa deteksi keliru (*false positive*), kemampuan SAHI untuk mengungkap lebih banyak kemungkinan area bermasalah pada PCB dinilai lebih bermanfaat untuk tahap inspeksi awal. Oleh karena itu, implementasi YOLOv11n dengan SAHI akan digunakan dalam sistem GUI yang dikembangkan.

4.4 Implementasi Model terhadap GUI Sebagai Sistem Utuh

Setelah penentuan model dan metode inference tambahan maka selanjutnya adalah implementasi model yang sudah dibangun. Namun, terdapat perubahan mengingat SAHI memiliki hyperparameter yang berbeda maka GUI perlu untuk disesuaikan dengan penentuan sistem backbone yang sudah dibangun. Berikut ini tampilan GUI yang diperbarui dengan penyesuaian metode inference :



Gambar 4.3. Tampilan GUI final untuk sistem inspeksi PCB

Pada gambar tersebut, terlihat bahwa hyperparameter yang digunakan SAHI dapat dikonfigurasi secara langsung.

4.4.1 Demonstrasi Inference Melalui Sistem GUI

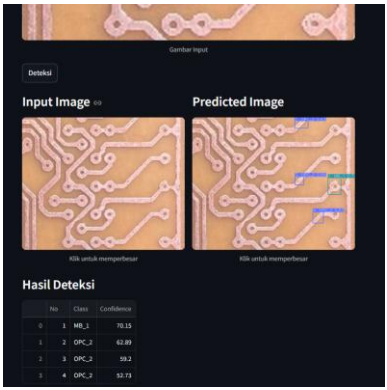
Seperti yang sudah tertuang pada bab 3.2 tentang alur penggunaan sistem, maka hal pertama yang perlu dilakukan adalah memberikan inputan gambar baik itu melalui kamera maupun file yang berada pada device (laptop atau komputer).

Gambar 4.4. opsi browse file dan ambil foto sebagai input sistem

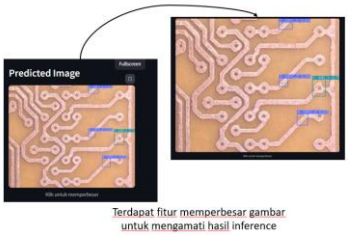
Langkah selanjutnya adalah melakukan konfigurasi khusus jika dibutuhkan untuk hyperparameter maupun seleksi terhadap klasifikasi cacat yang hendak diamati.

Gambar 4.5. opsi hyperparameter pada sistem

Kemudian melakukan menekan tombol **deteksi** untuk melakukan inference menggunakan backbone yang sudah dibangun.

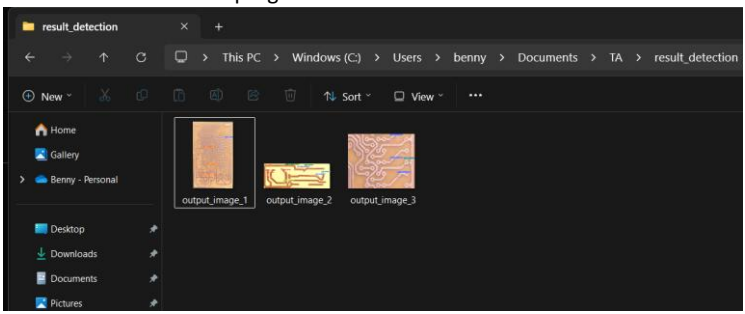


Gambar 4.6. tampilan hasil inference pada sistem



Gambar 4.7. fitur untuk zoom pada gambar

Seperti pada gambar, hasil inference akan berupa gambar prediksi dan tabel berisi urutan cacat berikut dengan nama klasifikasi serta confidence score nya. Gambar hasil prediksi/inference juga akan tersimpan pada folder yang sudah ditentukan dalam kode program.



Gambar 4.8. folder output dari hasil inference sistem

4.5 Pembahasan Hasil Penelitian

Penelitian ini secara keseluruhan telah berhasil mengembangkan dan mengevaluasi sistem deteksi cacat pada Printed Circuit Board (PCB) yang fungsional. Sistem ini mengintegrasikan model YOLOv11n yang efisien dengan metode Slicing Aided Hyper Inference (SAHI) untuk peningkatan deteksi objek kecil, serta disajikan melalui Antarmuka Pengguna Grafis (GUI) yang interaktif. Keunggulan utama dari sistem yang dihasilkan terletak pada efisiensi operasional yang tinggi. model YOLOv11n menunjukkan ukuran yang sangat ringkas (5.2 MB) dan kecepatan inference superior (~4.33 FPS) sambil tetap mempertahankan kinerja deteksi yang kompetitif (mAP@0.5: 0.890).

Lebih lanjut, penerapan SAHI secara visual terbukti signifikan dalam meningkatkan sensitivitas sistem terhadap cacat-cacat berukuran kecil yang sebelumnya berpotensi terlewat. kemampuan untuk menyesuaikan hyperparameter SAHI secara spesifik, seperti ukuran potongan dan rasio tumpang tindih, juga terbukti sangat membantu dalam mengoptimalkan konfigurasi untuk deteksi objek yang sangat kecil tersebut.

GUI yang dikembangkan juga berhasil menyediakan platform yang efektif bagi pengguna untuk melakukan visualisasi dan analisis hasil deteksi secara mudah. Selain itu, pendekatan analisis yang mendalam terhadap kemampuan model dalam mengenali klasifikasi cacat secara partikular selama proses evaluasi memberikan pemahaman berharga mengenai respons dan kekuatan model terhadap setiap jenis cacat. Wawasan ini dapat menjadi landasan penting untuk upaya perbaikan dan pengembangan model di masa depan, khususnya dalam meningkatkan akurasi dan memperluas pengenalan terhadap varietas klasifikasi cacat yang lebih beragam.

Meskipun demikian, penelitian ini memiliki beberapa aspek yang menjadi catatan dan keterbatasan penting. Keterbatasan utama terletak pada dataset yang digunakan, yang masih terbatas dalam hal kompleksitas dan variasi jenis cacat. Hal ini berpotensi membatasi kemampuan generalisasi model ketika dihadapkan pada cacat dari sumber atau jenis PCB yang sangat berbeda. Terkait metode SAHI, efektivitasnya sangat bergantung pada keandalan klasifikasi model dasar pada setiap potongan gambar dan teramati kurang optimal untuk objek berukuran besar. Selain itu, SAHI juga menambah waktu inference secara nyata lebih lama beberapa detik dan memiliki potensi untuk meningkatkan insiden *false positive*, yang memerlukan keseimbangan dalam penerapannya.

Bab 5.
Kesimpulan

Bab ini menyajikan kesimpulan dari keseluruhan hasil penelitian dan pembahasan yang telah diuraikan, serta memberikan saran untuk pengembangan penelitian selanjutnya.

5.1 Kesimpulan

Berdasarkan analisis hasil penelitian dan pembahasan yang telah dilakukan untuk mengembangkan sistem inspeksi visual cacat pada Printed Circuit Board (PCB) manual, dapat ditarik kesimpulan sebagai berikut:

a. **Pengembangan Sistem Inspeksi Visual Berbasis DNN**: Sistem inspeksi visual untuk mendeteksi cacat pada PCB manual telah berhasil dikembangkan dengan mengimplementasikan dan mengevaluasi beberapa model Deep Neural Network (DNN). Model YOLOv11n terpilih sebagai model utama karena menawarkan kombinasi terbaik antara efisiensi operasional (ukuran model 5.2 MB, kecepatan inference ~4.33 FPS) dan kinerja deteksi yang kompetitif (mAP@0.5: 0.890), sehingga berhasil menjawab rumusan masalah pertama dan mencapai tujuan pertama penelitian.

b. **Peningkatan Deteksi Objek Kecil**: Pendekatan untuk meningkatkan deteksi objek kecil pada PCB telah dieksplorasi dan diterapkan melalui implementasi metode Slicing Aided Hyper Inference (SAHI) pada model YOLOv11n. Secara kualitatif melalui analisis visual, penerapan SAHI terbukti meningkatkan sensitivitas model terhadap cacat berukuran kecil yang sebelumnya berpotensi terlewat, sejalan dengan rumusan masalah kedua dan tujuan kedua penelitian, meskipun perlu dicatat adanya potensi peningkatan false positive sebagai trade-off.

c. **Pembangunan Antarmuka Pengguna Interaktif**: Sebuah antarmuka pengguna (GUI) berbasis Streamlit telah berhasil dibangun. GUI ini memfasilitasi proses pengamatan dan analisis hasil deteksi secara interaktif dan intuitif, memungkinkan pengguna untuk mengunggah gambar PCB, menjalankan proses deteksi, dan memvisualisasikan hasilnya dengan jelas. Hal ini menjawab rumusan masalah ketiga dan mencapai tujuan ketiga penelitian.

Secara keseluruhan, penelitian ini berhasil mengembangkan prototipe sistem deteksi cacat PCB yang efisien, memiliki sensitivitas yang ditingkatkan untuk cacat

kecil melalui SAHI, dan dilengkapi dengan GUI yang fungsional. Keunggulan utama sistem terletak pada efisiensi dan kemampuan adaptasinya, namun keterbatasan pada generalisasi dataset dan nuansa operasional SAHI menjadi catatan penting.

5.2 Saran

Berdasarkan kesimpulan dan keterbatasan yang telah diidentifikasi dalam penelitian ini, berikut adalah beberapa saran yang dapat menjadi pertimbangan untuk pengembangan penelitian selanjutnya:

a. **Pengembangan dan Diversifikasi Dataset**: Disarankan untuk melakukan pengembangan dataset dengan menambah jumlah sampel, variasi jenis PCB, dan terutama kompleksitas jenis cacat. Hal ini bertujuan untuk meningkatkan kemampuan generalisasi model sehingga lebih andal ketika dihadapkan pada data baru yang beragam.

b. **Analisis Kuantitatif dan Optimasi SAHI**: Melakukan analisis kuantitatif yang lebih mendalam mengenai dampak SAHI terhadap metrik evaluasi (misalnya, mAP pada subset objek kecil) dan trade-off antara peningkatan sensitivitas dengan false positive rate. Eksplorasi lebih lanjut terhadap optimasi parameter SAHI (ukuran potongan, rasio tumpang tindih) juga dapat dilakukan[12].

c. **Eksplorasi Model dan Teknik Lanjutan**: Menguji arsitektur model object detection atau segmentasi yang lebih baru atau yang memiliki mekanisme khusus untuk menangani objek kecil. Menyelidiki teknik pasca-pemrosesan tambahan untuk mengurangi false positive yang mungkin dihasilkan oleh SAHI atau model itu sendiri.

d. **Peningkatan Fungsionalitas GUI**: Mengembangkan fitur-fitur yang lebih canggih pada GUI, seperti integrasi dengan basis data LLM, fitur anotasi manual untuk koreksi, atau analisis statistik sederhana terhadap hasil deteksi.

e. **Pengujian Keandalan dan Validasi Lapangan**: Melakukan pengujian sistem pada lingkungan yang lebih mendekati kondisi nyata industri untuk mengukur ketahanan dan keandalan sistem dalam jangka panjang, serta pada berbagai kondisi pencahayaan dan kualitas gambar PCB.

Daftar Pustaka

[1] M. Faisal Riftiarrasyid, D. Arif Setyawan, and H. Maulana, "Klasifikasi Kesegaran Daging Sapi Menggunakan Metode Gray Level Cooccurrence Matrix dan DNN," 2021. [Online]. Available: https://www.kaggle.com/crowww/meat-quality-assessment-

[2] H. Muchtar and R. Apriadi, "Implementasi Pengenalan Wajah Pada Sistem Penguncian Rumah dengan Metode Template Matching Menggunakan Open Source Computer Vision Library (Opencv)," vol. 2, no. 1, Feb. 2017.

[3] R. Balma, K. Petsch, and T. Kaya, "Development of Thin Film Photolithography Process: Patterning Printed Circuit Boards(PCBs) and Copper Electroplating," Jun. 2011.

[4] W. Huang and P. Wei, "A PCB Dataset for Defects Detection and Classification," Jan. 2019, [Online]. Available: http://arxiv.org/abs/1901.08204

[5] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," Jun. 2015, [Online]. Available: http://arxiv.org/abs/1506.01497

[6] P. Soviany and R. T. Ionescu, "Optimizing the Trade-off between Single-Stage and Two-Stage Object Detectors using Image Difficulty Prediction," Mar. 2018, [Online]. Available: http://arxiv.org/abs/1803.08707

[7] R. Khanam and M. Hussain, "YOLOV11: AN OVERVIEW OF THE KEY ARCHITECTURAL ENHANCEMENTS," 2024.

[8] "Object Detection untuk Mendeteksi Citra Buah Buahan Menggunakan Metode YOLO".

[9] P. Soviany and R. T. Ionescu, "Optimizing the Trade-off between Single-Stage and Two-Stage Object Detectors using Image Difficulty Prediction," Mar. 2018, [Online]. Available: http://arxiv.org/abs/1803.08707

[10] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," Jun. 2015, [Online]. Available: http://arxiv.org/abs/1506.01497

[11] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal Loss for Dense Object Detection," Aug. 2017, [Online]. Available: http://arxiv.org/abs/1708.02002

[12] F. C. Akyon, S. O. Altinuc, and A. Temizel, "Slicing Aided Hyper Inference and Fine-tuning for Small Object Detection," Feb. 2022, doi: 10.1109/ICIP46576.2022.9897990.

[13] rafael padilla, L. N. sergio, and A. B. da S. Eduardo, “A survey on performance metrics for object detection algorithms,” 2020, *IEEE*.

[14] *2020 International Conference on Systems, Signals and Image Processing (IWSSIP)*. IEEE, 2020.

[15] A. Wishnu Wardhana and Y. Prayudi, “PENGGUNAAN METODE TEMPLETE MATCHING UNTUK IDENTIFIKASI KECACATAN PADA PCB,” 2008.

[16] R. A. Julianto, E. Efrizon, H. Hendrick, L. Devy, S. Suryadi, and Y. Antonisfia, “Pembuatan Alat Inspeksi Visual Jalur PCB Menggunakan Pengolahan Citra Untuk Kegiatan Praktikum Pengawatan Dan Teknologi PCB,” *Elektron : Jurnal Ilmiah*, pp. 61–66, Dec. 2022, doi: 10.30630/eji.14.2.295.

[17] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, “Focal Loss for Dense Object Detection,” Feb. 2018. [Online]. Available: https://github.com/facebookresearch/Detectron.

Daftar Pustaka Gambar

1. Jamindo PCBA. *Rigid Flex PCB*. Diakses dari https://jamindopcba.com/rigid-flex-pcb/ pada 6 Juni 2025.
2. Learn OpenCV. *Intersection over Union (IoU) in Object Detection and Segmentation*. Diakses dari https://learnopencv.com/intersection-over-union-iou-in-object-detection-and-segmentation/ pada 6 Juni 2025.
3. Maryam, N. *Non-Max Suppression in Object Detection*. LinkedIn. Diakses dari https://www.linkedin.com/pulse/non-max-suppression-object-detection-nabeelah-maryam-2gr7f pada 6 Juni 2025.
4. O'Reilly. *Chapter 4. Convolutional Neural Networks* (dari *Strengthening Deep Neural Networks*). Diakses dari https://www.oreilly.com/library/view/strengthening-deep-neural/9781492044949/ch04.html pada 6 Juni 2025.
5. PCBAAA. *PCB Solder Mask*. Diakses dari https://www.pcbaaa.com/pcb-solder-mask/ pada 6 Juni 2025.
6. PCBMay. *Blue PCB*. Diakses dari https://www.pcbmay.com/blue-pcb/ pada 6 Juni 2025.
7. Reddit. *Difference in Image classification, Semantic segmentation, Object detection, Instance segmentation?*. Diakses dari https://www.reddit.com/r/learnmachinelearning/comments/kt0hov/difference_in_image_classification_semantic/ pada 6 Juni 2025.

8. ResearchGate. *Figure 2: The architecture of Faster R-CNN*. Diakses dari https://www.researchgate.net/figure/The-architecture-of-Faster-R-CNN_fig2_324903264 pada 6 Juni 2025.
9. ResearchGate. *Figure 3: Two-stage vs. one-stage object detection models*. Diakses dari https://www.researchgate.net/figure/Two-stage-vs-one-stage-object-detection-models_fig3_353284602 pada 6 Juni 2025.
10. ResearchGate. (2023). *Figure 7: The reasoning process diagram of the Slicing Aided Hyper Inference framework for forest....* . Diakses dari https://www.researchgate.net/figure/The-reasoning-process-diagram-of-the-Slicing-Aided-Hyper-Inference-framework-for-forest_fig7_364257753 pada 6 Juni 2025.
11. Roboflow. *Scaled-YOLOv4 Tops EfficientDet*. Diakses dari https://blog.roboflow.com/scaled-yolov4-tops-efficientdet/ pada 6 Juni 2025.
12. SCS Circuits. *Rigid PCB*. Diakses dari https://www.scscircuits.com/product-item/rigid-pcb/ pada 6 Juni 2025.
13. SuperAnnotate. *Object Detection with Deep Learning: A Practical Guide*. [Blog Post]. Diakses dari https://www.superannotate.com/blog/object-detection-with-deep-learning pada 6 Juni 2025.
14. V7 Labs. *Mean Average Precision (mAP) Explained: Everything You Need to Know*. [Blog Post]. Diakses dari https://www.v7labs.com/blog/mean-average-precision pada 6 Juni 2025.
15. Wevolver. Diakses dari https://www.wevolver.com/article/trace-pcb-a-comprehensive-guide pada 6 Juni 2025.
16. YIC Technologies. *Flexible Circuit Boards*. Diakses dari https://yic-assm.com/flexible-circuit-boards/ pada 6 Juni 2025.

Lampiran

Lampiran berisikan berkas berikut :

- Ipynb script kode model detectron2 faster rcnn, detectron2 retinanet, yolov11n dan yolov8n
- Dataset format coco dan yolo (must_bethefinal, coco)
- Script kode untuk GUI
- Output training dan validasi faster rcnn, retinanet, yolov8n dan yolov11n berupa metrics dan model yang sudah dilatih pada dataset pcb
- Dataset actual
- Dependencies yang dibutuhkan dalam membangun system (kecuali detectron2)

Dapat diakses melalui hyperlink berikut [LINK]