



Pengembangan Model Identifikasi Penyakit Pada Tanaman Padi Berbasiskan Faster RCNN

Muhammad Pailus
18917121

Tesis diajukan sebagai syarat untuk meraih gelar Magister Komputer

Konsentrasi Data Sains

Program Studi Informatika Program Magister

Fakultas Teknologi Industri

Universitas Islam Indonesia

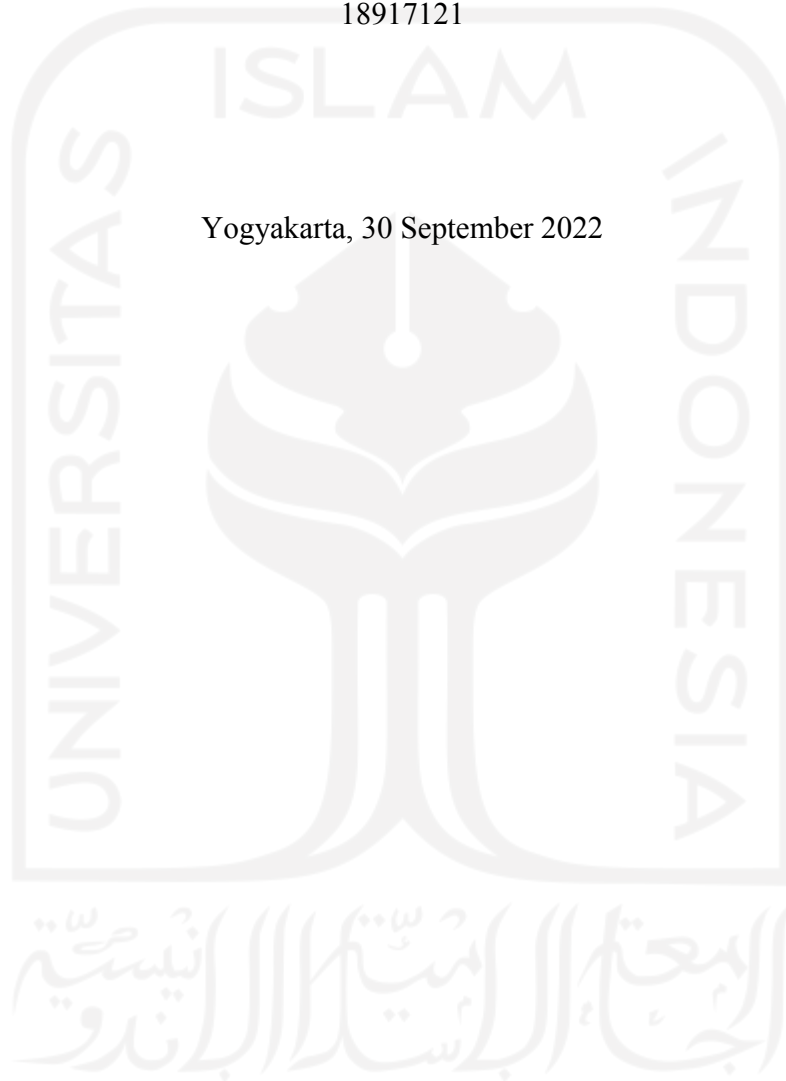
2022

Lembar Pengesahan Pembimbing
Pengembangan Model Identifikasi Penyakit Pada Tanaman Padi Berbasiskan Faster
R-CNN

Muhammad Pailus

18917121

Yogyakarta, 30 September 2022



Pembimbing

Pembimbing

A blue ink signature of Dhomas Hatta Fudholi, consisting of stylized initials 'DHF'.

Dhomas Hatta Fudholi, S. T., M.Eng, Ph.D

A blue ink signature of Dr. Syarif Hidayat, consisting of stylized initials 'SH'.

Dr. Syarif Hidayat, S.Kom., M.I.T.

Lembar Pengesahan Penguji

Pengembangan Model Identifikasi Penyakit Pada Tanaman Padi Berbasiskan Faster R-CNN

Muhammad Pailus

18917121

Yogyakarta, Oktober, 2022

Tim Penguji,

Dhomas Hatta Fudholi, S. T., M.Eng, Ph.D

Ketua

Dr. Syarif Hidayat, S.Kom., M.I.T.

Anggota I

Chandra Kusuma Dewa, S.Kom., M.Cs., Ph.D.

Anggota II

Mengetahui,

Ketua Program Studi Teknik Informatika Program Magister

Universitas Islam Indonesia

Irving Vitra Paputungan, S.T., M.Sc., Ph.D

Abstrak

Pengembangan Model Identifikasi Penyakit Pada Tanaman Padi Berbasiskan Faster RCNN

Kesalahan dalam mengidentifikasi penyakit pada tumbuhan padi bisa menyebabkan potensi kegagalan panen meningkat hingga 18%-80% menurut data kementerian pertanian Indonesia. Hal ini bisa disebabkan karena masih kurangnya ekspert di bidang pertanian. Oleh sebab itu teknologi bisa menjadi solusi dari masalah kekurangan ekspert, terutama dengan memanfaatkan deep learning. Penelitian terbaru dalam bidang deep learning menggunakan neural network telah mencapai peningkatan yang luar biasa, terutama pada bidang pertanian.

Penelitian dengan memanfaatkan *deep learning* yaitu faster rcnn pernah dilakukan untuk mendeteksi apakah padi mengidap penyakit false smut atau bukan. Namun pemanfaatan *object detection* pada penelitian sebelumnya belum optimal karena hanya untuk binary classification. Pada penelitian ini dengan menggunakan faster rcnn untuk identifikasi penyakit pada tanaman padi menggunakan 1500 data yang diambil secara mandiri. Dengan 3 kelas data penyakit dan masing-masing kelas memiliki 500 gambar sebelum di anotasi. Selanjutnya data di anotasi dan di resize dengan ukuran 512x512, baru kemudian dilakukan training. Dengan menggunakan backbone, dan pretrained model resnet50_fpn, dan Fast rcnn sebagai predictor pada layer terakhirnya. Model mampu mendeteksi penyakit spesifik di area yang diidap. Disamping itu juga model mampu mendeteksi penyakit lebih dari satu pada tanaman padi. Dengan Hasil akhir dari pengujian mAP sebesar 87.6% dan recall 85.7%. Selanjutnya inference pada dataset publik juga cukup memuaskan karena model mampu mendeteksi penyakit yang ada, dengan resolusi gambar berbeda-beda.

Kata kunci

Penyakit Daun padi , object detection, deep learning, Faster RCNN

abstract

Development of Disease Identification Model in Rice Plants Based on Faster RCNN

Errors in identifying diseases in rice plants can cause the potential for crop failure to increase by 18%-80%, according to data from the Indonesian Ministry of Agriculture. This could be due to the lack of expertise in agriculture. Therefore technology can be a solution to the problem of lack of expertise, especially by utilizing deep learning. Recent research in the field of deep learning using neural networks has achieved remarkable improvements, especially in agriculture.

Research using deep learning, namely faster RCNN, has been carried out to detect whether rice has false smut disease or not. However, the use of object detection in previous studies has not been optimal because it is only for binary classification. In this study, the faster RCNN was used to identify diseases in rice plants using 1500 data taken independently. With 3 classes of disease data and each class has 500 images before being annotated. Furthermore, the data is annotated and resized to a size of 512x512, then training is carried out. By using backbone, and pretrained models resnet50_fpn, and Fast rcnn as predictors in the last layer. The model is able to detect specific diseases in the affected area. Besides that, the model is also able to detect more than one disease in rice plants. With the final results of the mAP test of 87.6% and recall of 85.7%. Furthermore, the inference on public datasets is also quite satisfactory because the model is able to detect existing diseases, with different image resolutions.

Keywords

Rice leaf disease, object detection, deep learning, Faster RCNN

Pernyataan Keaslian Tulisan

Dengan ini saya menyatakan bahwa tesis ini merupakan tulisan asli dari penulis, dan tidak berisi material yang telah diterbitkan sebelumnya atau tulisan dari penulis lain terkecuali referensi atas material tersebut telah disebutkan dalam tesis. Apabila ada kontribusi dari penulis lain dalam tesis ini, maka penulis lain tersebut secara eksplisit telah disebutkan dalam tesis ini.

Dengan ini saya juga menyatakan bahwa segala kontribusi dari pihak lain terhadap tesis ini, termasuk bantuan analisis statistik, desain survei, analisis data, prosedur teknis yang bersifat signifikan, dan segala bentuk aktivitas penelitian yang dipergunakan atau dilaporkan dalam tesis ini telah secara eksplisit disebutkan dalam tesis ini.

Segala bentuk hak cipta yang terdapat dalam material dokumen tesis ini berada dalam kepemilikan pemilik hak cipta masing-masing. Apabila dibutuhkan, penulis juga telah mendapatkan izin dari pemilik hak cipta untuk menggunakan ulang materialnya dalam tesis ini.

Yogyakarta, Juni 2022

MUHAMMAD PAILUS

Daftar Publikasi

Pailus, Muhammad, Hatta, Dhomas, Hidayat, Syarif. “Model Identifikasi Penyakit Pada Tumbuhan Padi Berbasiskan DenseNet.” Jurnal Sains Komputer Informatika (J-SAKTI) Volume 6 No 2 (2022)

Sitasi publikasi 1

Kontributor	Jenis Kontribusi
Muhammad Pailus	Membuat model dan menulis paper
Dhomas Hatta Fudholi	Memberikan saran untuk <i>hyperparameter tuning</i>
Syarif Hidayat	Memberikan saran dan analisis pengembangan model

Halaman Kontribusi

Dalam penulisan tesis ini Dosen Pembimbing, dan Dosen-dosen Penguji memberikan masukan-masukan sebagai perbaikan dari cara penulisan tesis serta memberikan saran tentang data yang akan diolah dan dianalisis.



Halaman Persembahan

Kupersembahkan Tesis ku ini dengan setulus hati & sepenuh jiwa untuk : Istriku Mona Ananda Dewi tercinta, anakku Haitam , dan kedua Orang tuaku yang selalu mengiringi langkah kakiku dengan do'a, yang tak pernah putus asa untuk mengurus, mendidik, menasehati, menyemangati, membimbing serta memberikan motivasi kepadaku.



Kata Pengantar

Alhamdulillahirabbil'alamin, segala puji syukur kehadiran Allah SWT senantiasa penulis panjatkan karena atas limpahan rahmat dan hidayah-Nya, tesis yang berjudul "Pengembangan Model Identifikasi Penyakit Tanaman Padi Berbasiskan Faster Rcn" dapat berjalan dengan lancar dalam penyelesaiannya. Tesis ini diajukan sebagai bagian dalam menyelesaikan studi dan sebagai salah satu syarat untuk memperoleh gelar Magister Komputer pada Program Studi Magister Teknik Informatika Fakultas Teknologi Industri Universitas Islam Indonesia. Dalam penyelesaian Tesis ini, penulis banyak mendapatkan bantuan dari berbagai pihak, untuk itu penulis menyampaikan ucapan terima kasih setulusnya kepada:

1. Bapak Prof. Fathul Wahid, S.T., M.Sc., Ph.D., selaku Rektor Universitas Islam Indonesia.
2. Bapak Irving Vitra Paputungan, S.T., M.Sc., Ph.D., selaku Ketua Program Studi Teknik Informatika Program Studi Magister Universitas Islam Indonesia.
3. Bapak Dhomas Hatta Fudholi, S.T., M.Eng., Ph.D., selaku dosen pembimbing yang telah memberikan banyak pengarahan dan masukan dalam penyusunan dan penyempurnaan Tesis ini.
4. Bapak Dr. Syarif Hidayat, selaku dosen pembimbing yang telah memberikan banyak pengarahan dan masukan dalam penyusunan dan penyempurnaan Tesis ini.
5. Dewan penguji tesis yang telah memberikan banyak pengarahan dan masukan dalam penyusunan dan penyempurnaan Tesis ini.
6. Bapak/Ibu Dosen Program Studi Magister Teknik Informatika yang telah memberikan bekal ilmu pengetahuan kepada penulis, semoga ilmunya menjadi amal jariyah di dunia maupun akhirat.
7. Bapak/Ibu Staff Akademik Program Pascasarjana Fakultas Teknologi Industri Universitas Islam Indonesia, yang telah membantu dalam segala urusan administrasi di kampus.
8. Sahabat-sahabat seperjuangan Sains Data yang telah turut serta membantu, mendukung, menyemangati serta memberikan masukan kepada penulis selama menjalani studi di Program Pascasarjana Magister Teknik Informatika Fakultas Teknologi Industri Universitas Islam Indonesia Yogyakarta.
9. Teman-teman Magister Teknik Informatika Universitas Islam Indonesia Yogyakarta seangkatan. Penulis menyadari bahwa dalam penulisan tesis ini masih banyak kelemahan dan kekurangan, untuk itu kritik dan saran yang sifatnya membangun sangat penulis harapkan agar tesis ini dapat menjadi lebih baik.

Daftar Isi

Lembar Pengesahan Pembimbing	ii
Lembar Pengesahan Penguji	iii
Abstrak	iv
Pernyataan Keaslian Tulisan	vi
Daftar Publikasi	vii
Halaman Kontribusi	viii
Halaman Persembahan	ix
Kata Pengantar	x
Daftar Isi	xi
Daftar Tabel	xiii
Daftar Gambar	xiv
BAB 1 Pendahuluan	1
1.1 Latar Belakang	1
1.2 Rumusan Masalah	3
1.3 Tujuan Penelitian	4
1.4 Batasan Masalah	4
1.5 Sistematika Penulisan	4
BAB 2 Tinjauan Pustaka	5
2.1. Penelitian Terdahulu	5
2.2 Konsep Pengetahuan	7
2.2.1 Deep Learning	7
2.2.2 Object Detection	10
2.2.3 Convolutional Neural Network	10
2.2.4 Fungsi Aktivasi	14

2.2.5 R-CNN.....	17
2.2.6 Fast R-CNN.....	19
2.2.7 Faster RCNN	20
2.2.8 Resnet	24
2.2.9 Web Service	26
2.2.10 Jenis Penyakit Padi.....	27
BAB 3 Metodologi	32
3.1 Pengumpulan Data	32
3.2 Pelabelan Data.....	33
3.3 Resize Gambar	36
3.4 Training Data.....	37
3.5 Model dan Testing.....	41
3.6 Evaluasi Model.....	42
3.7 Perancangan Web Service	43
BAB 4 Hasil dan Pembahasan.....	48
4.1 Data Yang Digunakan	48
4.2 Hasil Training Dan Evaluasi	52
4.3 Pengembangan Web service.....	58
BAB 5 Kesimpulan dan Saran.....	60
5.1 Kesimpulan.....	60
5.2 Saran	60
Daftar Pustaka	61

Daftar Tabel

Table 2. 1 Penelitian-penelitian terkait penyakit tanaman	5
Table 2. 2 Perbandingan kecepatan fast r-cnn vs rcnn	20
Tabel 4. 1 Peningkatan Hasil Training Resnet50_FPN pada setiap check point	53



Daftar Gambar

Gambar 1. 1 Padi yang mengidap lebih dari satu penyakit.	3
Gambar 2. 1 Arsitektur jaringan syaraf $a = inputan$ melambangkan inputan, dan lingkaran melambangkan cell / perceptron.....	8
Gambar 2. 2 Perbedaan cara kerja machine learning dan deep learning.....	9
Gambar 2. 3 Perbedaan object detection kiri dan image classification kanan.	10
Gambar 2. 4 Arsitektur CNN	11
Gambar 2. 5 Contoh penerapan kernel 3 X 3	12
Gambar 2. 6 Penerapan filter max pooling.....	13
Gambar 2. 7 Gambar fully connected layer	14
Gambar 2. 8 fungsi aktivasi linear.....	16
Gambar 2. 9 Rectified Linear Unit.....	17
Gambar 2. 10 R-CNN	18
Gambar 2. 11 Selective search pada saat men-generate 2000 region proposal.	19
Gambar 2. 12 Fast R-CNN	20
Gambar 2. 13 Arsitektur Faster R-CNN.....	21
Gambar 2. 14 RPN memanfaatkan feature map untuk menghasilkan <i>proposed region</i>	22
Gambar 2. 15 Anchor box yang digunakan pada layer RPN.	23
Gambar 2. 16 Arsitektur Faster R-CNN.....	23
Gambar 2. 17 Perbandingan kecepatan dari R-CNN, Fast R-CNN, Faster R-CNN.	24
Gambar 2. 18 Residual Block.....	25
Gambar 2. 19 Perbedaan VGG-19 dan Resnet source = ‘ <i>Deep Residual Learning for Image Recognition</i> ’ paper	26
Gambar 2. 20 Arsitektur Web Service.....	27
Gambar 2. 21 Penyakit hawar daun.....	28
Gambar 2. 22 Penyakit Blas	29
Gambar 2. 23 Penyakit bercak daun cercospora	31
Gambar 3. 1 Metodologi penelitian.....	32
Gambar 3. 2 Contoh Dataset Hawar daun, blast, bercak sempit	33
Gambar 3. 3 penyakit padi yang dalam satu daun lebih dari satu penyakit	33

Gambar 3. 4 Proses pelabelan dengan labelImg	34
Gambar 3. 5 Multi penyakit dalam 1 tanaman	35
Gambar 3. 6 Struktur file xml dari hasil anotasi	35
Gambar 3. 7 Struktur data dari file .xml diubah menjadi csv	36
Gambar 3. 8 Contoh Dataset yang sudah di resize, termasuk dengan bounding box-nya	36
Gambar 3. 9 Potongan kode untuk melakukan proses resize gambar	37
Gambar 3. 10 Struktur direktori untuk membangun model	37
Gambar 3. 11. Konfigurasi untuk proses training data.....	38
Gambar 3. 12 potongan kode albumentation untuk augmentasi dataset.	39
Gambar 3. 13 mengambil koordinat gambar sesuai hasil anotasi dengan format VOC pada file .xml	39
Gambar 3. 14 Load data train dan validation dari folder.	40
Gambar 3. 15 Potongan Kode untuk melakukan training data.....	40
Gambar 3. 16 Fungsi untuk melakukan pembuatan model dengan pretrained resnet50_fpn	41
Gambar 3. 17 Layer restnet50_fpn.....	41
Gambar 3. 18 Ground truth bounding box dan predicted bounding box dalam object detection	43
Gambar 3. 19 Ilustrasi Intersection Over Union	43
Gambar 3. 20 Perancangan Web service.....	44
Gambar 3. 21 Struktur folder untuk deployment model dengan <i>framework</i> Django.....	44
Gambar 3. 22 File url yang mengatur routing dari web service.....	45
Gambar 3. 23 File views.py yang bertanggung jawab menghandle request dan response	45
Gambar 3. 24 File Serializer.py bertugas untuk validasi dan formating data	45
Gambar 3. 25 File Models.py bertanggung jawab sebagai konektor dengan database.....	46
Gambar 3. 26 Potongan kode	46
Gambar 3. 27 Alur Pendeteksian jenis penyakit oleh web service	47
 Gambar 4. 1 jumlah kelas masing-masing data sebelum dilakukan anotasi pada gambar.....	 48
Gambar 4. 2 Penyakit hawar daun.....	49
Gambar 4. 3 Penyakit Bercak Sempit	49
Gambar 4. 4 Penyakit Blas	50
Gambar 4. 5 Sebaran kelas data setelah dilakukan proses anotasi.....	51
Gambar 4. 6 perbedaan anotasi dari masing-masing kelas (penyakit) yang ada pada tumbuhan padi	51

Gambar 4. 7 Satu daun lebih mengidap lebih dari satu penyakit	52
Gambar 4. 8 Epoch 1 dari 105 epoch	52
Gambar 4. 9 Epoch 105 dari 105 epoch	53
Gambar 4. 10 perbandingan train loss dengan validation loss Resnet50_Fpn.....	53
Gambar 4. 11 Hasil Evaluation menggunakan mAP	54
Gambar 4. 12 Proses Inference pada kelas / penyakit hawar daun, box yang dilingkari merah merupakan gambar yang gagal teridentifikasi dari model	55
Gambar 4. 13 Proses inference pada kelas / penyakit bercak sempit.....	56
Gambar 4. 14 Proses inference pada kelas / penyakit blas box merah kegagalan deteksi	56
Gambar 4. 15 Inference pada multi kelas/penyakit pada satu tanaman, border dengan warna merah merupakan Hawar daun dan warnah hijau bercak sempit.....	57
Gambar 4. 16 Inference dengan dataset dari ics.uci.edu penyakit hawar daun.....	57
Gambar 4. 17 untuk foto yang sebelah kiri ada kesalahan deteksi yang dianggap penyakit, padahal seharusnya background.	57
Gambar 4. 18 Proses mengupload gambar untuk deteksi penyakit dengan postman.....	58
Gambar 4. 19 Aplikasi Untuk melakukan prediksi dengan model yang telah dihasilkan pada penyakit blas daun.	58
Gambar 4. 20 Hasil prediksi lebih dari satu penyakit dalam satu daun padi	59

Glosarium

DL	- Deep Learning
CNN	- Convolutional Neural Network
ML	- Machine Learning
NN	- Neural Network
VOC	- Visual Object Classes
ANN	- Artificial Neural Network
mAP	- Mean Average Precision



BAB 1

Pendahuluan

1.1 Latar Belakang

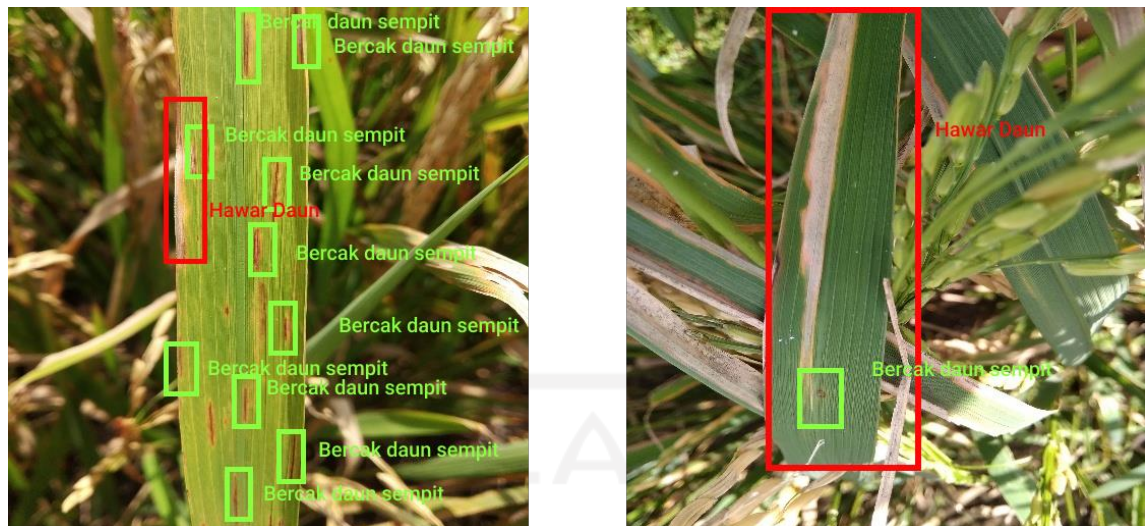
Mengetahui penyakit tanaman serta melakukan diagnosis dengan cepat dan akurat guna menentukan obat yang akan digunakan sangat penting untuk meningkatkan produksi pangan. Cara yang paling umum digunakan saat ini menggunakan tenaga ahli di bidang pertanian yang cenderung lambat dan tidak bisa mengakomodir dalam jumlah yang banyak, hal ini akan berdampak menurunkan produksi pertanian. Mendiagnosis penyakit tanaman melalui pengamatan secara optikal pada gejala yang muncul pada daun tanaman, mempunyai tingkat kesulitan yang cukup kompleks. (Ferentinos, 2018)

Penyakit tanaman merupakan ancaman utama terhadap ketahanan pangan, tetapi untuk melakukan identifikasinya dengan cepat di beberapa bagian di dunia masih sulit dilakukan, karena keterbatasan infrastruktur (Prasanna Mohanty et al., 2016). Oleh sebab itu peran teknologi yang bisa diakses dengan biaya rendah perlu disiapkan, sehingga adanya pemerataan dalam hal ketersediaan informasi penyakit tanaman bisa tercapai. Dengan banyaknya ketersediaan kamera digital dan kemajuan di bidang *computer vision*, model otomatisasi dalam mendiagnosis dengan presisi tinggi di bidang *agriculture* sangat dibutuhkan, seperti untuk fenotipe tanaman, rumah kaca pintar, dan sebagainya (Wang et al., 2017). Terlebih lagi saat ini *smartphone* yang dimiliki oleh masing-masing individu tidak terkecuali petani, sudah memiliki spesifikasi yang lebih dari cukup untuk melakukan tugas identifikasi penyakit tanaman dengan bantuan *deep learning* untuk melakukan prosesnya.

Deep learning merupakan teknik modern terbaru untuk pemrosesan gambar dan analisis data, dengan menjanjikan hasil dan potensi yang besar. Karena *deep learning* telah berhasil diterapkan di berbagai domain, baru-baru ini memasuki juga domain pertanian (Kamilaris & Prenafeta-Boldú, 2018). *Deep Learning* sebenarnya subbidang dari *machine learning* (Purwins et al., 2019) yang pada awal mulanya terinspirasi oleh struktur otak manusia. Sebelum adanya *deep learning machine learning* sudah lebih dulu digunakan. *Machine learning* adalah sebuah program komputer yang mampu belajar dari pengalaman, sehingga semakin banyak pengalaman yang diberikan kepada *machine learning* akan meningkatkan performanya menjadi semakin baik. Namun dalam prakteknya algoritma

machine learning tradisional tidak akan membaik secara signifikan ketika sudah sampai titik tertentu, sebanyak apapun data yang diberikan. Untuk itulah deep learning dibuat untuk menyelesaikan masalah performa pada algoritma machine learning tradisional ketika pertumbuhan data semakin membesar (Brownlee, 2019). *Deep learning* mempunyai beberapa algoritma yang bisa digunakan dalam bidang *computer vision*.

Algoritma yang cukup terkenal dalam pengolahan citra gambar salah satunya adalah CNN, walaupun masih ada beberapa algoritma lainnya yang juga bisa digunakan untuk pengolahan gambar di penelitian-penelitian yang sudah pernah dilakukan. Penelitian yang sama-sama membahas tentang penyakit pada tanaman padi pernah dilakukan oleh (Liang et al., 2019), dimana penelitiannya berfokus pada penyakit blast dan menggunakan algoritma *Local Binary Pattern Histogram*. Dalam penelitiannya berfokus untuk membedakan tumbuhan padi yang terinfeksi penyakit blast dan tumbuhan padi yang tidak termasuk penyakit blast. Penelitian lain yang juga membahas tentang penyakit padi yaitu penyakit *false smut* dengan algoritma menggunakan *faster r-cnn* untuk membedakan apakah padi terinfeksi false mut atau bukan (Sethy et al., 2020). Hanya saja dalam penelitian yang dilakukan oleh Sethy dkk, belum bisa sepenuhnya disebut *object detection*, karena hanya membedakan apakah tanaman padi mengidap *false smut* atau bukan, yang secara teori penelitian ini lebih cenderung *binary classification* dengan penambahan *localization*. Selanjutnya penelitian yang dilakukan oleh Guoxiong Zhou dkk, menggunakan *fcm-km* dan *faster r-cnn* untuk mengidentifikasi 3 jenis penyakit pada tanaman padi yaitu *blight*, *bacterial blight*, dan *blast*, dengan hasil akurasi 96.71%, akan tetapi masih ada yang bisa di *improve* dimana pada penelitiannya dalam satu tumbuhan padi model baru bisa mengidentifikasi satu jenis penyakit saja, padahal kenyataan dilapangan ada kemungkinan berbeda. Misalnya padi mengidap lebih dari satu penyakit di waktu bersamaan, sementara kalau model hanya bisa mengidentifikasi satu penyakit saja dalam satu tanaman, informasi yang didapatkan tidak maksimal. Untuk mendukung hipotesa bahwa padi bisa mengidap lebih dari satu penyakit pada saat bersamaan, penulis mencoba menghimpun data dari ekspert PT. Mitra Sejahtera Membangun Bangsa khususnya yang menangani hama dan penyakit tanaman (**HPT**), dengan hasil kemungkinan padi mengidap lebih dari satu penyakit sering ditemukan dilapangan. Hal ini juga sejalan dengan dataset yang penulis punya. Berikut contoh temuan padi mengidap lebih dari satu penyakit perhatikan Gambar 1. 1



Gambar 1. 1 Padi yang mengidap lebih dari satu penyakit.

Berdasarkan latar belakang tersebut penulis ingin melakukan penelitian untuk mengembangkan model identifikasi penyakit pada tanaman padi dengan metode *object detection* berbasis algoritma Faster RCNN. Alasan Faster RCNN dipilih dalam penelitian ini, karena dari hasil pengujian yang dilakukan oleh Lutan dkk, pada penelitian yang berjudul *Comparison of YOLO v3, Faster R-CNN, and SSD for Real-Time Pill Identification* (Tan et al., 2021), Faster RCNN unggul dibanding dengan 2 algoritma lainnya, dengan Mean Average Precision (MAP) Faster RCNN 87%, Yolo 80% dan SSD tidak menunjukkan hasil yang memuaskan. Oleh sebab itu Faster RCNN dipilih karena MAP yang lebih tinggi berdasarkan penelitian yang dibuat oleh Tan dkk, walau Yolo lebih unggul di FPS-nya. Sedangkan untuk jenis penyakit tanaman padi pada penelitian ini ada 3 jenis penyakit yang akan menjadi pembahasan, yaitu penyakit blast, hawar daun dan bercak sempit. Ketiga penyakit tersebut sebarannya cukup banyak di Indonesia menurut data BPPADI. Hal itu lah yang melatar belakangi penulis untuk menjadikan 3 penyakit tersebut dalam penelitian ini yaitu dengan judul “Pengembangan Model Identifikasi Penyakit Pada Tanaman Padi Berbasis Faster RCNN”.

1.2 Rumusan Masalah

Berdasarkan latar belakang dan penelitian sebelumnya sudah pernah ada penelitian tentang mendeteksi penyakit pada tanaman padi menggunakan image classification. Hanya saja pada kenyataanya tanaman padi memungkinkan mengidap lebih dari satu penyakit dalam satu tanaman. Oleh sebab itu perlu membangun model yang mampu mendeteksi penyakit, walaupun lebih dari satu penyakit pada tanaman padi.

1.3 Tujuan Penelitian

Menghasilkan model yang mampu mendeteksi penyakit spesifik diarea atau lokasi penyakit pada tanaman padi.

1.4 Batasan Masalah

Pada penelitian ini terbatas pada tanaman padi saja dengan 3 jenis penyakit, penyakit terbatas hanya pada 3 jenis penyakit pada tumbuhan padi yaitu Hawar daun, bercak sempit, dan blas berdasarkan data **BBPADI** merupakan data penyakit padi yang cukup banyak di indonesia.

1. Berjalan diatas platform web service
2. Harus ada jaringan internet jika web service ingin diakses publik.
3. Algoritma yang digunakan untuk membangun model identifikasi penyakit tanaman adalah Faster R-CNN

1.5 Sistematika Penulisan

Sistematika penulisan penelitian ini disusun untuk memberikan gambaran umum tentang penelitian yang dijalankan. Sistematika penelitian ini adalah sebagai berikut:

1. BAB I PENDAHULUAN

Pada bagian ini berisi tentang latar belakang permasalahan, rumusan masalah, menentukan batasan masalah yang akan dibahas, tujuan dari penetilan, dan sistematika penulisan.

2. BAB II TINJAUAN PUSTAKA

Pada bagian ini menjelaskan teori-teori yang digunakan di dalam penelitian. Setiap teori yang digunakan di dalam penelitian akan dijabarkan di bagian ini.

3. BAB III METODOLOGI PENELITIAN

Pada bagian ini menampilkan metode yang akan digunakan pada penelitain ini mulai dari mengumpulkan data penelitian, sampai dengan penelitian dinyatakan selesai.

4. BAB IV HASIL DAN PEMBAHASAN

Pada bagian ini menjelaskan hasil dan pembahasan dalam penelitian, mulai dari data yang digunakan, hasil training dan evaluasi, sampai implementasi pada webservice.

5. BAB V KESIMPULAN DAN SARAN

Pada bagian ini akan dijabarkan kesimpulan yang didapatkan setelah menyelesaikan penelitian. Selain itu juga diberikan saran yang memberikan poin-poin yang dapat ditingkatkan.

BAB 2

Tinjauan Pustaka

2.1. Penelitian Terdahulu

Penelitian terkait penyakit tanaman terdahulu mengacu pada Table 2. 1, terdapat sub tema penelitian, algoritma yang digunakan, jenis penyakit, serta jenis tanaman yang di infeksi. Penelitian tentang deteksi penyakit dan pengenalan jenis hama pada tumbuhan tomat dengan memanfaatkan deep learning pernah dilakukan untuk 9 kelas yang terdiri dari penyakit dan hama (Fuentes, Alvaro, 2017) dengan jumlah dataset sebanyak 5000 data yang merupakan dataset public (Plant Village dataset). Dengan hasil model mampu mendeteksi 9 kelas yang berbeda walaupun resolusi gambar tidak sama. Penelitian lain menggunakan Faster R-CNN dengan kombinasi Firefly Chaos teory Maximum dan Kmeans (FCM-KM) untuk mendeteksi penyakit pada tanaman padi dengan 3 kelas data dilakukan oleh Guoxiong dkk, dengan penambahan proses masking gambar pada layer terpisah untuk mengurangi noise, kemudian baru dimasukan ke Faster R-CNN (Zhou et al., 2019). Namun disayangkan pada penelitian ini ada kegagalan deteksi terjadi ketika terdapat multi area penyakit dalam satu gambar.

Pada tahun 2020 penelitian untuk mendeteksi apakah tumbuhan padi mengidap penyakit false smut atau bukan, dengan menggunakan algoritma faster r-cnn (Sethy et al., 2020) juga pernah dilakukan, sayangnya secara teori ini masih bisa disimpulkan sebagai binary classification + localization, dengan kata lain fungsi dari *object detection* belum bisa di maksimalkan. Selanjutnya penelitian yang hampir mirip dengan penelitian sethy dkk, juga pernah dilakukan oleh Liang dkk yaitu mengenali apakah tumbuhan padi terinfeksi penyakit blast atau bukan (Liang et al., 2019). Penelitian lain ditahun yang sama pada tahun 2019 M.al-amin dkk, menggunakan CNN untuk memprediksi apakah tumbuhan padi tergolong kedalam 4 kelas penyakit Bacterial Leaf, Blight, Blast Leaf Smut, atau Brown Spot dengan akurasi mencapai 97.40% (Al-Amin et al., 2019).

Table 2. 1 Penelitian-penelitian terkait penyakit tanaman

No	Sub Tema	Algoritma	Jenis Penyakit	Jenis tanaman
1	A Robust Deep-	Faster RCNN VGG-16,	- Bacterial spot - Early Blight	Tomat

	Learning-Based Detector for Real-Time Tomato Plant Diseases and Pests Recognition	Single Shoot Detector, R-FCN	<ul style="list-style-type: none"> - Late Blight - Leaf Mold - Septoria Leaf spot - Spider Mite - Target spot - Yellow Leaf - Mosaic Virus 	
2	Rapid Detection of rice disease based on FCM-KM and Faster R-CNN fusion	FCM-KM and Faster R-CNN Fusion	<ul style="list-style-type: none"> - Rice Blast - Bacterial Blight - Blight 	Padi
3	Rice False smut Detection based on Faster-RCNN	Faster RCNN, False Smut	False Smut	Padi
4	Rice blast disease recognition using a deep convolutional neural network	Local Binary Pattern Histogram	Blast Disease	Padi
5	Prediction of rice disease from leaves using deep convolution neural network towards a	CNN	<ul style="list-style-type: none"> - Bacterial Leaf Blight - Blast - Leaf Smut - Brown Spot 	Padi

	digital agricultural system			
--	-----------------------------------	--	--	--

2.2 Konsep Pengetahuan

Saat ini, pengaplikasian deep learning (DL) tersebar luas di seluruh dunia. Pengaplikasiannya termasuk pada bidang kesehatan (Esteva et al., 2019), pemrosesan data visual (J. Yu et al., 2018), analisis sosial media (Hayat et al., 2019), serta pemrosesan audio dan ucapan (Purwins et al., 2019). Sedangkan untuk bidang pertanian, deep learning sudah sampai tahap implementasi. Dengan kata lain sudah masuk ke ranah bisnis. Salah satunya perusahaan asal Jerman yang mengembangkan aplikasi **plantix** dengan menggunakan deep learning untuk mengidentifikasi penyakit pada tanaman (Goncharov et al., 2018; Jez et al., 2021).

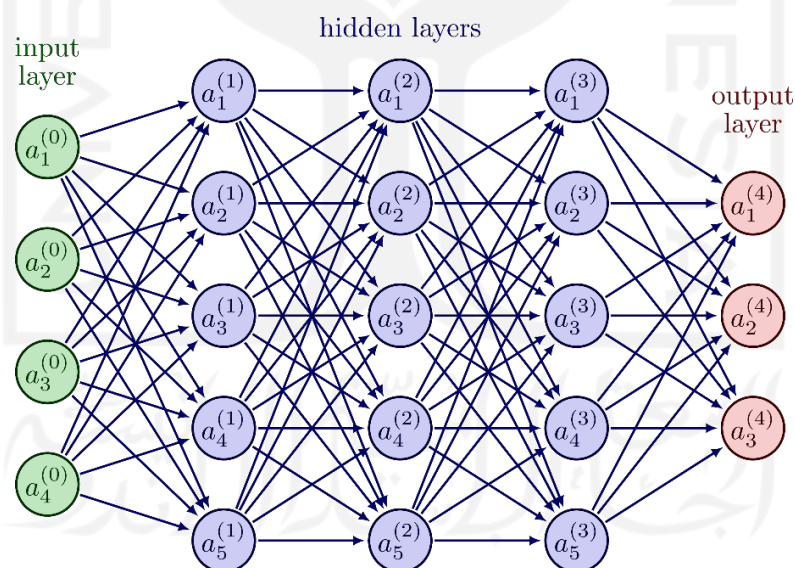
2.2.1 Deep Learning

Deep Learning menjadi pembicaraan tidak hanya dikalangan peneliti tetapi juga di masyarakat umum. Hal ini disebabkan oleh kemunculan deep fake (Agarwal et al., 2020) yang membuat heboh masyarakat dunia. Dengan beberapa foto yang ada di internet ataupun social media, tokoh-tokoh dunia bisa dibuat seakan-akan berbicara dan bersuara seperti melakukan percakapan yang sebenarnya (N. Yu et al., 2020). Di bidang yang lain seperti mobil yang mampu mengendarai secara otomatis, juga menjadi titik balik masyarakat dunia lebih sadar akan adanya teknologi seperti *deep learning* (Ajitha & Nagra, 2021).

Deep Learning sebenarnya subbidang dari *machine learning* (Purwins et al., 2019) yang pada awal mulanya terinspirasi oleh struktur otak manusia. Struktur otak tersebut dinamakan Artificial Neural Networks atau disingkat ANN. Pada dasarnya, merupakan jaringan saraf yang memiliki tiga atau lebih ANN. *Deep learning* mampu belajar dan beradaptasi untuk sejumlah data yang tak terstruktur seperti gambar dan audio, serta mampu menyelesaikan permasalahan yang sulit bisa dibandingkan algoritma *machine learning* lainnya. Sementara itu deep learning pertama kali diteorikan pada 1980-an, namun baru sekarang mulai populer, karena ada dua alasan utama mengapa hal itu bisa terjadi:

- deep learning membutuhkan sejumlah besar data berlabel. Misalnya, pengembangan mobil tanpa pengemudi membutuhkan jutaan gambar dan ribuan jam video.
- Deep learning membutuhkan daya komputasi yang besar . GPU performa tinggi memiliki arsitektur paralel yang efisien untuk deep learning. Ketika dikombinasikan dengan cluster atau komputasi awan, ini memungkinkan tim pengembangan untuk mengurangi waktu pelatihan untuk jaringan deep learning dari minggu ke jam atau kurang.

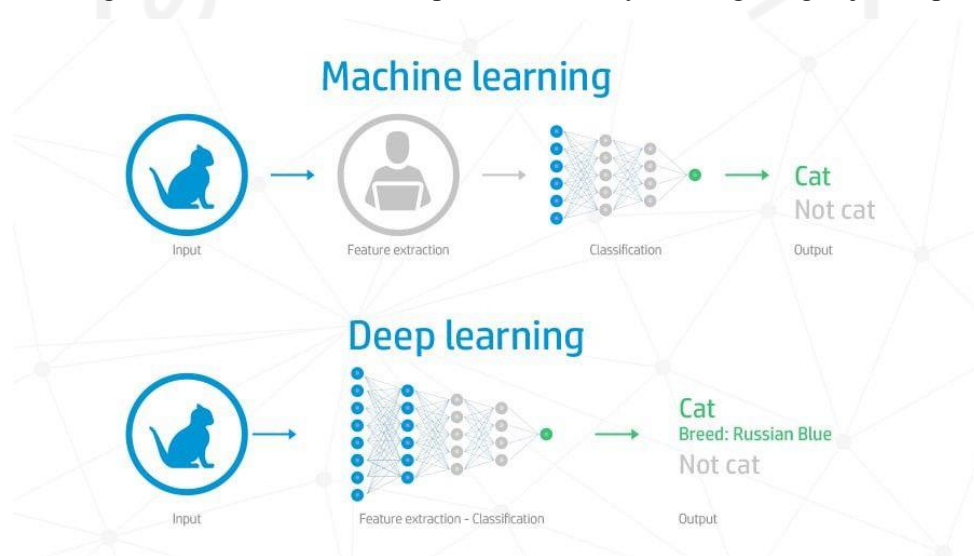
Sebagian besar metode deep learning menggunakan arsitektur jaringan saraf. itulah Karena itu model deep learning sering disebut sebagai deep neural network . Seperti yang ditunjukkan pada Gambar 2. 1 *node/cell* terlihat dibedakan menjadi 3 bagian. Pertama input layer, hidden layer, dan output layer. Hidden layer pada gambar mempunyai kedalaman yang lebih dalam dibanding 2 layer lainnya. Istilah "dalam" biasanya mengacu pada jumlah lapisan tersembunyi di jaringan saraf. Jaringan saraf tradisional hanya berisi 2-3 lapisan tersembunyi, sedangkan *deep neural network* dapat memiliki sebanyak 150 lapisan. Dikatakan deep learning apabila suatu *neural network* ini memiliki lebih dari satu hidden layer.



Gambar 2. 1 Arsitektur jaringan syaraf $a = \text{inputan}$ melambangkan inputan, dan lingkaran melambangkan cell / perceptron.

Deep learning berbeda dengan *machine learning* tradisional. Dalam *machine learning* tradisional, pemrogram harus sangat spesifik memberi tahu komputer jenis hal apa yang harus dicari untuk mengambil suatu keputusan dalam *ouput*-nya (Jordan & Mitchell, 2015;

J. Yu et al., 2018). Sedangkan *deep learning* membangun set fitur (*extraction feature*) secara otomatis tanpa perlu dilakukan secara manual oleh manusia untuk digunakan dalam mengambil suatu keputusan atau prediksi dalam *output*-nya (LeCun et al., 2015). Untuk memahami cara kerja dari masing masing mengacu pada Gambar 2. 2. Inputan *machine learning* berupa gambar kucing terlebih dahulu di ekstrak fiturnya secara manual oleh manusia. Setelahnya baru dimasukan kedalam algoritma untuk di klasifikasi apakah gambar yang dimasukan tersebut kucing atau bukan. Sedangkan untuk *deep learning*, gambar kucing di inputkan kedalam satu jaringan. Kemudian algoritma secara otomatis mengekstrak fitur yang ada pada gambar, baik itu dengan backpropagation ataupun feed forward. Dengan hasil akhir bisa lebih spesifik, misalnya kucing dengan jenis apa.



Gambar 2. 2 Perbedaan cara kerja machine learning dan deep learning¹

Hal yang paling umum dalam machine learning baik itu secara mendalam atau tidak, merupakan *supervised learning* (Vargas et al., 2017). Misalnya ketika ingin membuat sebuah sistem yang mampu membedakan apakah sebuah gambar berisikan rumah, mobil, kucing ataupun manusia. Hal yang pertama kali dilakukan adalah mengumpulkan dataset yang besar yang berisikan gambar rumah, mobil, kucing ataupun manusia. Selanjutnya data harus dilabeli sesuai dengan masing-masing kelasnya. Pada saat melakukan training algoritma akan menampilkan output yang merupakan gambaran *score* masing-masing kelas sesuai dengan kategorinya. Hasil akhir yang ingin didapatkan adalah mencari score tertinggi dari masing-masing kelas tergolong ke kategori yang mana.

¹ <https://towardsdatascience.com/understand-these-4-advanced-concepts-to-sound-like-a-machine-learning-master-d32843840b52>

2.2.2 Object Detection

Object detection adalah sebuah teknik dari *computer vision* yang bekerja untuk mengidentifikasi dan melokalisasi objek dalam sebuah gambar ataupun video (Pathak et al., 2018). *Object detection* secara umum juga akan memberikan output berupa bounding box. Kemudian bounding box tersebut juga akan menyesuaikan dengan lokasi objek yang diprediksi oleh model. *Object detection* berbeda dengan *image classification*. Dimana *image classification* akan mengklasifikasi sebuah gambar berdasarkan kelas yang paling dominan. Baru kemudian akan memberikan label terhadap gambar dengan score yang paling tinggi. Sedangkan *object detection* memberikan lokasi objek berada berupa koordinat yang hasil akhirnya bisa lebih dari satu kelas/objek. Kemudian akan diberikan koordinat atau *bounding box* pada masing-masing kelas sesuai hasil prediksinya. Pada Gambar 2. 3 terdapat perbedaan antara *image classification* dan *object detection*. Dimana dengan penyakit yang sama *object detection* memberikan persis area lokasi penyakit dengan bounding box. Sedangkan *image classification* hanya memberikan prediksi kelas dengan skor paling tinggi.



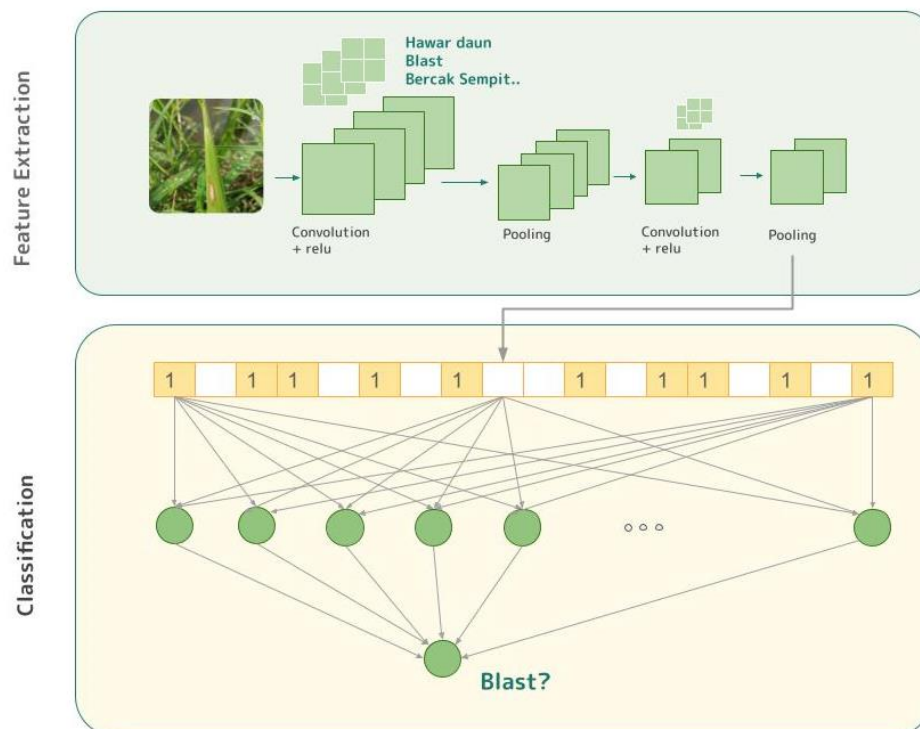
Gambar 2. 3 Perbedaan *object detection* kiri dan *image classification* kanan.

2.2.3 Convolutional Neural Network

Convolutional Neural Network (CNN) adalah salah satu bagian dari neural network yang biasa digunakan pada data gambar (O'Shea & Nash, 2015). CNN selain untuk klasifikasi juga bisa digunakan untuk mendeteksi dan mengenali object pada sebuah gambar. Convolutional Neural Network telah digunakan sebagai *framework deep learning* yang terinspirasi oleh *visual cortex hewan*. Awalnya telah banyak digunakan untuk tugas

klasifikasi pada sebuah gambar. Tetapi sekarang penelitiannya telah mulai digunakan di domain yang berbeda seperti, *object detection*, *object tracking*, *pose estimation*, *text detection and recognition*, *action recognition*, dan banyak lagi bidang-bidang lainnya (Oberweiger et al., 2015).

Pada Gambar 2. 4 dijelaskan bagaimana cara kerja dari Convolutional Neural Network. Convolutional Neural Network (CNN) yang pada dasarnya terdiri dari 3 lapisan yaitu *convolutional layer*, *pooling layer*, dan *fully connected layer*. Dimana setiap layer mempunyai fungsi yang berbeda dalam memproses data. Secara garis besar bisa dikelompokkan menjadi 2 bagian yaitu *feature extraction* dan *classification*. Pada bagian *feature extraction*, *convolution layer* yang merupakan *layer* bertanggung jawab untuk melakukan ekstraksi fitur yang kemudian hasil *output*-nya dikontrol oleh fungsi aktivasi. Dimana hasil dari outputnya disebut dengan *feature map*. Kemudian *pooling layer* yang bertugas untuk mereduksi ukuran *fitur map* yang telah dihasilkan oleh *convolution layer*. Tujuannya untuk mengambil fitur yang signifikan biasanya dengan menggunakan *max pooling*, atau *average pooling*. Sedangkan pada *classification* yang bertanggung jawab adalah *fully connected layer*. Agar data bisa diproses pada bagian ini data terlebih dahulu harus diubah menjadi data satu dimensi (*flatten*). Kemudian baru akan di masukan ke *output activation* untuk menentukan probabilitas yang paling tinggi .



Gambar 2. 4 Arsitektur CNN

Untuk memahami bagaimana CNN bekerja secara lebih detail berikut penjelasan setiap layer yang ada pada CNN.

1. Convolutional Layer

CNN bekerja dengan membandingkan gambar sepotong demi sepotong. Caranya dengan menggerakkan kernel konvolusi (filter) berukuran tertentu kepada sebuah inputan misalnya gambar. Dari hasil itu komputer akan mendapatkan informasi refresentatif baru dari hasil perkalian bagian gambar dengan kernel yang digunakan. Misalkan pada Gambar 2. 5 mempunyai matriks dengan ukuran $N * N$ sebagai inputan, dan menggunakan $m * m$ untuk filter ω . Maka *convolutional layer output*-nya didapatkan dengan ukuran $(N - m + 1) * (N - m + 1)$.

The diagram illustrates a 2D convolution operation. On the left is a 5x5 input matrix (blue and green cells):

7	2	3	3	8
4	5	3	8	4
3	3	2	8	4
2	8	7	2	7
5	4	4	5	4

This is multiplied (indicated by an asterisk) by a 3x3 kernel matrix (red and blue cells):

1	0	-1
1	0	-1
1	0	-1

The result is an equals sign followed by a 3x3 output matrix (orange cells). The top-left cell of the output matrix contains the value 6. Below the kernel matrix, the calculation for this value is shown:

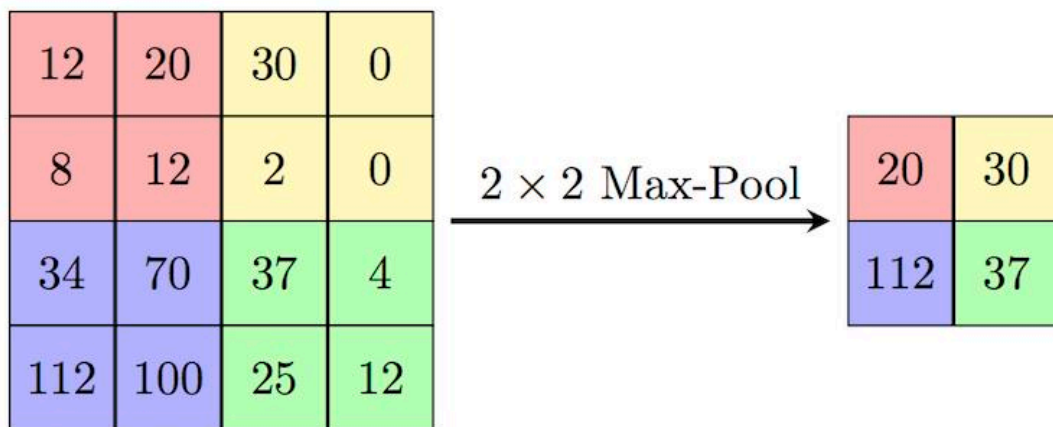
$$7 \times 1 + 4 \times 1 + 3 \times 1 + 2 \times 0 + 5 \times 0 + 3 \times 0 + 3 \times -1 + 3 \times -1 + 2 \times -1 = 6$$

Gambar 2. 5 Contoh penerapan kernel 3 X 3

Jadi berdasarkan Gambar 2. 5 sebelumnya output yang dihasilkan dari operasi konvolusi adalah $(5 - 3 + 1) \times (5 - 3 + 1) = 3 \times 3$ sedangkan hasil akhirnya setiap kolom dalam matrix dikalikan dengan kernel (filter) kemudian dijumlahkan untuk mendapatkan *output*-nya. Sementara itu jumlah langkah pergeseran kernel pada proses konvolusi disebut dengan stride. Hasil output yang dihasilkan dari proses konvolusi perlu diubah menjadi *non-linear* yang pada prosesnya akan menggunakan fungsi aktivasi.

2. Pooling Layer

Pooling layer pada CNN berada pada posisi setelah *convolutional layer*. Pada prinsipnya pooling layer terdiri dari sebuah kernel dengan ukuran dan stride tertentu yang bergeser pada area *feature map* (Arandjelovic et al., 2016). Biasanya pooling yang sering digunakan adalah *max pooling* dan *average pooling*. Pada Gambar 2. 6 dijlaskan bagaimana *pooling layer* dengan max pooling melakukan operasi untuk mengurangi dimensi dari matrix (*feature map*) atau dikenal juga dengan istilah *down sampling*. Dengan menggunakan ukuran filter **2x2** dan **stride 2** ukuran matrik inputan (**4X4**) . Setiap pergerakan kernel akan diambil nilai tertinggi. Sehingga outpunya matrix berukuran (**2X2**) dengan nilai tertinggi disetiap pergeseran kernelnya.



Gambar 2. 6 Penerapan filter max pooling²

Untuk memahami Gambar 2. 6 yang merupakan penerapan contoh dari max pooling, bisa dengan mengacu pada warna dan lihat output yang dihasilkan. Contoh untuk matrix pada operasi pertama dengan anggota 12, 20, 8, 12 maka output yang dihasilkan adalah nilai tertinggi yaitu 20. Secara matematis lebih sederhana bila dibandingkan dengan *convolutional layer*.

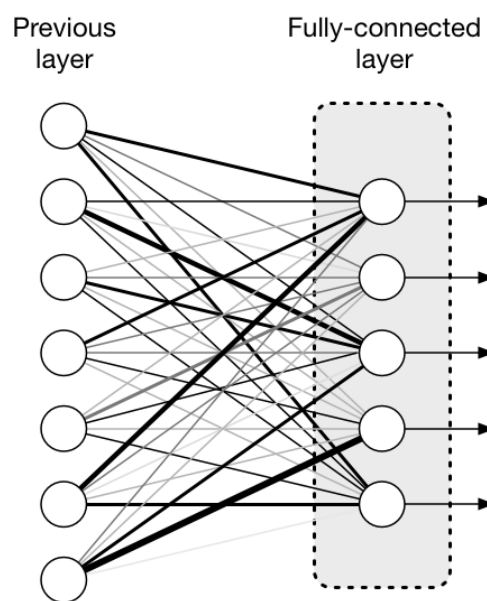
3. Fully Connected Layer

Feature map yang dihasilkan dari proses *feature extraction* masih dalam bentuk multidimensional array³. Sehingga untuk membentuknya menjadi sebuah vector diperlukan proses “**flatten**” atau “**reshape**”. Hasilnya nanti *feature map* akan menjadi inputan dari *fully connected layer*.

² <https://paperswithcode.com/method/max-pooling>

³ <https://towardsdatascience.com/confusion-matrix-and-object-detection-f0cbb634157>

Lapisan *fully connected layer* seperti Gambar 2. 7 adalah lapisan dimana semua neuron aktivitas sebelumnya (*previous layer*) terhubung kesemua neuron dilapisan *fully connected layer* seperti halnya jaringan saraf tiruan biasa⁴. Dengan demikian semua data yang akan masuk kelapisan ini sudah menjadi keharusan untuk mengubah datanya kedalam satu dimensi. Fully connected layer biasanya juga digunakan pada *multi layer perceptron* dengan tujuan untuk melakukan pengolahan data. Sehingga bisa diklasifikasikan. Perbedaan antar lapisan Fully-Connected dan lapisan konvolusi biasa adalah neuron di lapisan konvolusi terhubung hanya ke daerah tertentu pada input, Sementara fully connected layer memiliki neuron yang secara keseluruhan terhubung.



Gambar 2. 7 Gambar fully connected layer ⁵

2.2.4 Fungsi Aktivasi

Fungsi aktivasi sendiri berada pada tahap sebelum pooling layer dan setelah melakukan proses konvolusi. Fungsi aktivasi akan menentukan bagaimana perilaku dari sebuah perceptron. Lebih tepatnya output yang dihasilkan akan ditentukan setelah pemilihan fungsi aktivasi yang akan digunakan. Fungsi aktivasi diberikan untuk membatasi output yang akan dihasilkan oleh proses sebelumnya. Dimana jika tanpa fungsi aktivasi maka kita bisa melihat output atau keluarannya seperti formula 1 dengan nilai tak hingga, dimana y mewakili output.

⁴ <https://andrew.gibiansky.com/blog/machine-learning/convolutional-neural-networks/> accessed 10 juni 2022

⁵ <https://www.superdatascience.com/blogs/convolutional-neural-networks-cnn-step-4-full-connection>

$$-\infty \leq y \leq \infty \quad (2.1)$$

Jika tanpa fungsi aktivasi hasil output akan berpotensi menjadi tak hingga. Oleh sebab itu hasil dari output harus dimasukkan kedalam sebuah fungsi yang dikenal dengan fungsi aktivasi. Perhatikan persamaan matematika pada persamaan (2.2), (2.3) untuk mengontrol hasil outputnya, dimana output diwakili oleh variable y dan input ditunjukkan oleh x bobot dengan w dan bias b . Sehingga sebelum menghasilkan output berupa y hasil dari perkalian $x.w$ yang merupakan operasi perkalian matrix ditambahkan dengan bias b dimasukkan kedalam fungsi aktivasi terlebih dahulu.

$$y = f(x.w + b) \quad (2.2)$$

$$y = f\left(\sum_{i=0}^n x_i.w_i + b\right) \quad (2.3)$$

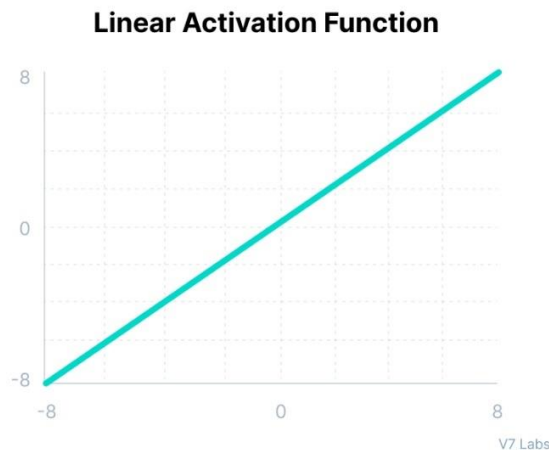
Untuk mendapatkan output yang optimal maka fungsi aktivasi menjadi sebuah keharusan untuk di implementasikan. berikut beberapa fungsi aktivasi yang digunakan pada penelitian ini.

1. Fungsi linear

Pada saat mendapatkan hasil keluaran salah satu fungsi yang umum digunakan adalah fungsi linear. Fungsi linear merupakan kembalian dari nilai output yang didapatkan. berikut persamaan matematis dari fungsi linear dengan x hasil keluaran sebelum dimasukkan kedalam fungsi linear $f(x)$.

$$f(x) = x \quad (2.4)$$

Untuk memahami gambaran fungsi linear pada grafik bisa dilihat pada Gambar 2.8, dimana hasil input yang diwakili pada sumbu x-axis berbading lurus dengan output yang ada pada sumbu y-axis.



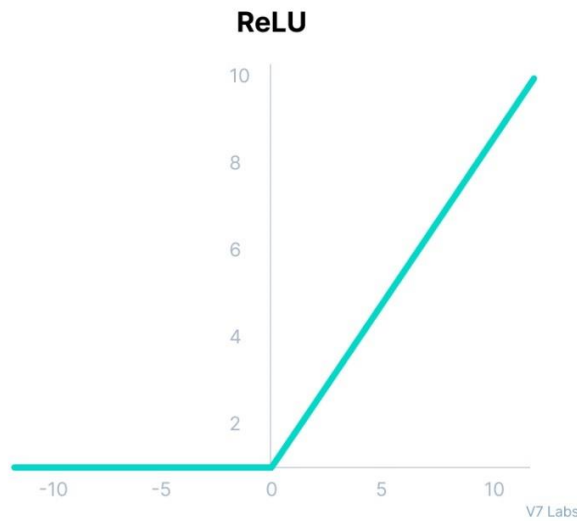
Gambar 2. 8 fungsi aktivasi linear⁶

2. Relu

Meskipun memberikan kesan fungsi linier, ReLU memiliki fungsi turunan dan memungkinkan backpropagation sekaligus membuatnya efisien secara komputasi. Dimana neuron hanya akan dinonaktifkan ketika output yang dihasilkan lebih kecil dari nol. Misalkan hasil inputan berupa x fungsi relu akan mengembalikan nilai x jika lebih besar dari **nol**. Sedangkan untuk x yang lebih kecil dari **nol** akan dikembalikan menjadi nilai **nol**. Untuk lebih jelasnya perhatikan persamaan matematis (2. 5). Jika Digambarkan dalam bentuk grafik seperti pada Gambar 2. 9, dimana dengan nilai inputan antara **-10** sampai dengan **10** akan mengembalikan nilai itu sendiri (x) jika nilainya lebih besar dari 0 dan akan bernilai 0 jika lebih kecil dari **0**.

$$f(x) = \begin{cases} 0 & \text{if } x \leq 0 \\ x & \text{if } x > 0 \end{cases} \quad (2. 5)$$

⁶ <https://www.v7labs.com/blog/neural-networks-activation-functions>



Gambar 2. 9 Rectified Linear Unit

3. Softmax

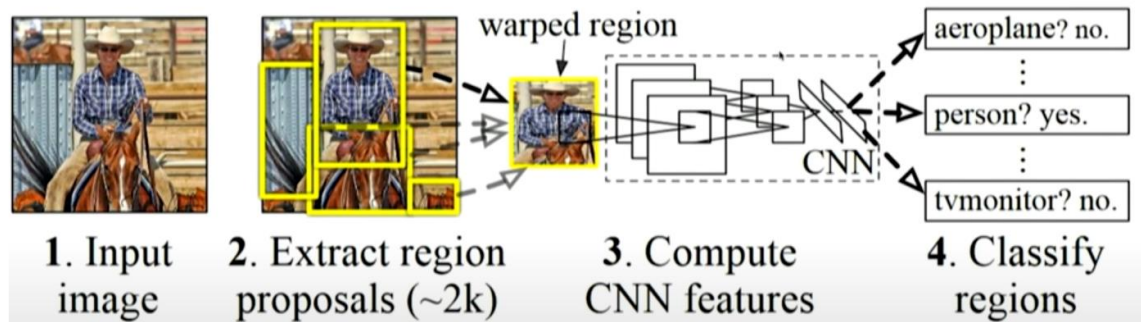
Jika dilihat secara sekilas fungsi aktivasi softmax hampir mirip dengan fungsi aktivasi sigmoid. Dimana kedua fungsi aktivasi sebenarnya walaupun terlihat sama akan tetapi mempunyai fungsi penggunaan yang berbeda. Hal yang paling muda di pahami adalah fungsi softmax digunakan untuk menghitung peluang pada *multi class classification*. Jika mengacu pada persamaan (2. 6 dimana inputan nilai dilambangkan dengan (Z_i) bisa dalam bentuk *single value* atau berupa *vector*, maka jika menggunakan fungsi softmax hasilnya adalah eksponensial *matrix* atau *vector* dibagi dengan penjumlahan keseluruhan elemen dari hasil eksponensial (Z_i) .

$$\text{Softmax}(Z_i) = \frac{\exp(Z_i)}{\sum_j \exp(Z_j)} \quad (2. 6)$$

2.2.5 R-CNN

Pada tahun 2014 grup peneliti dari UC Berkeley mengembangkan sebuah *Deep Convolutional network* yang dinamakan R-CNN yang mampu mendeteksi 80 object yang berbeda (Zhang et al., 2014). Ketika Girshik dkk menggabungkan *region proposal* dan CNNs, yang kemudian dinamakan dengan R-CNN. Jika dibandingkan dengan teknik *object detection* biasa perbedaannya hanya pada ekstraksi feature yang menggunakan CNN. Mengacu pada Gambar 2. 10 ketika proses gambar dimasukan sebagai inputan, *selective search algorithm* men-generate area sebanyak 2000, kemudian gambar akan di *crop* untuk

dimasukan kedalam CNN dan SVM (*support vector machine*) pada Gambar 2. 11 . Dalam prosesnya akan dilakukan *feature extraction* sampai dengan classification dan pada bagian akhir penentuan klasifikasi setiap objek yang telah melewati *features extraction*.

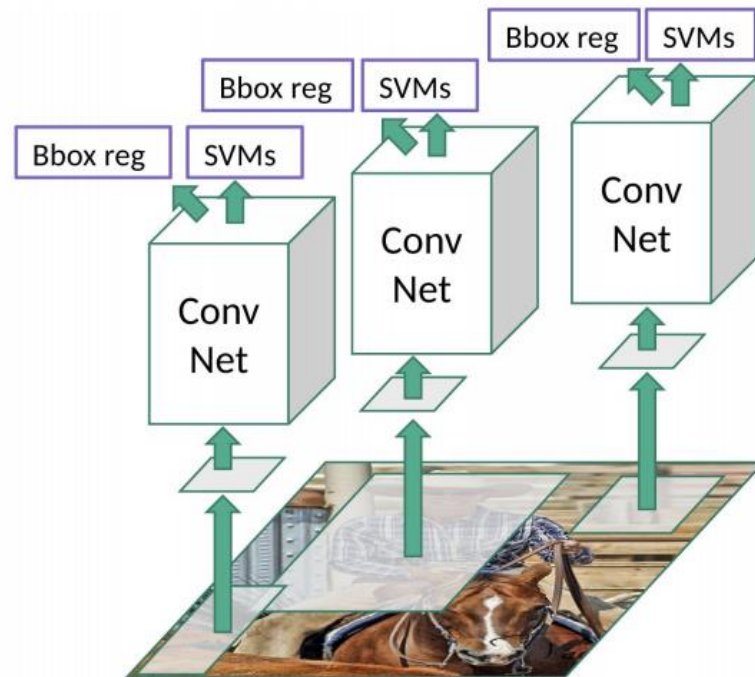


Gambar 2. 10 R-CNN ⁷

Untuk lebih jelasnya bagaimana cara kerja R-CNN sebagai berikut.

1. Pada saat gambar dimasukan algoritma akan *men-generate 2000 region proposal* dengan menggunakan *selective search algorithm*.
2. Setelah itu ukurannya diubah menjadi ukuran tetap, kemudian modul yang kedua akan melakukan ekstraksi fitur vektor dengan panjang 4096 untuk setiap region proposalnya.
3. Pada bagian akhir menggunakan *pre-trained SVM algorithm* untuk mengklasifikasikan *region proposal* ke setiap background atau ke sebuah *object class*-nya.
4. Model pada R-CNN dikenal *dengan multi stage independent component* dengan kata lain proses training tidak bisa dilakukan secara *end to end*.
5. Fitur yang di ekstraksi oleh cnn kemudian disimpan di disk yang kemudian baru di berikan ke SVM sebagai inputan, hal ini memakan ratusan gigabytes penyimpanan.

⁷ <https://blog.paperspace.com/faster-r-cnn-explained-object-detection/>



Gambar 2. 11 Selective search pada saat men-generate 2000 region proposal⁸.

2.2.6 Fast R-CNN

Fast R-CNN adalah object detector yang di kembangkan oleh **Ross Girshick**, seorang facebook ai Researcher dan *former* Microsoft Researcher. Fast R-CNN datang untuk memperbaiki beberapa issue pada RCNN. Dengan nama Fast R-CNN karena lebih cepat bila dibandingkan dengan R-CNN (Girshick, 2015). Beberapa perbaikan dengan yang di hadirkan Fast R-CNN sebagai berikut.

1. Menghadirkan layer baru dengan nama **ROI Pooling** dengan panjang fitur ekstraksi yang sama pada setiap vektor proposal digambar yang sama.
2. Dibandingkan dengan R-CNN, yang mempunyai *multiple stages (region proposal generation, feature extraction, dan classification* dengan menggunakan SVM). Fast R-CNN membuat model yang hanya mempunyai satu stage.
3. Fast R-CNN berbagi komputasi (*convolutional layer calculation*) dengan semua proposal (ROIs) dibanding dengan melakukan kalkulasi di setiap proposal secara independen. Hal ini bisa diselesaikan dengan menggunakan **ROI POOLING layer**, yang membuat **Fast R-CNN** bisa lebih cepat bila dibandingkan dengan **R-CNN**.

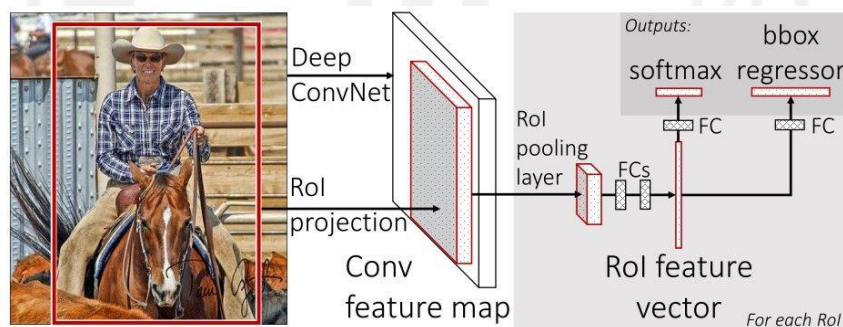
⁸ <https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e>

4. Fast R-CNN lebih cepat dibandingkan R-CNN, pada Table 2. 2 menjelaskan perbandingan yang dilakukan meliputi *training time*, *test time per image*, serta peningkatan kecepatan dari Fast R-CNN dibandingkan dengan R-CNN.

Table 2. 2 Perbandingan kecepatan fast r-cnn vs rcnn⁹

	R-CNN	Fast R-CNN
Training Time:	84 hours	9.5 hours
(Speedup)	1x	8.8x
Test time per image	47 seconds	0.32 seconds
(Speedup)	1x	146x

Mengacu pada Gambar 2. 12 model fast r-cnn terdiri dari satu *stages*, dibanding RCNN yang mempunyai 3 stages. Fast R-CNN hanya menerima sebuah inputan gambar tanpa melakukan crop / wrap dengan menggunakan SVM. Pada fast r-cnn justru gambar dimasukan langsung kedalam CNN kemudian hasil ekstraksi fiturruya (*feature map*) yang didapat baru kemudian akan diproses oleh ROI (Region Of Interest). ROI Pooling layer yang bertugas untuk meprediksi objek atau bukan. Tidak hanya itu outputnya juga berupa regresi dan class agnostic.



Gambar 2. 12 Fast R-CNN

2.2.7 Faster RCNN

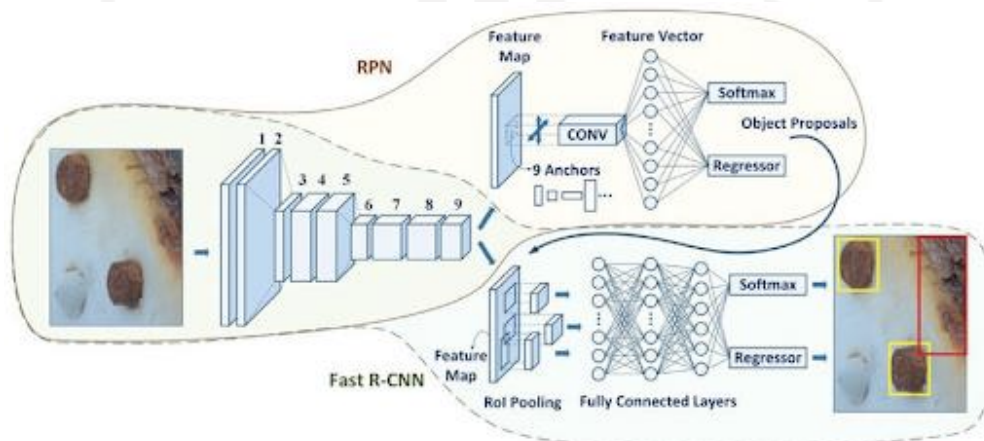
Berawal dari masih kurangnya kecepatan dalam deteksi objek yang dilakukan oleh algoritma yang berbasiskan region proposal, menjadi motivasi awal Shaoqing ren dkk, untuk mengusulkan metode baru . Dimana pada algoritma fast r-cnn waktu yang

⁹ <https://blog.paperspace.com/faster-r-cnn-explained-object-detection/>

dibutuhkan masih sekitar ~ 2 detik per gambar. Hal ini belum memungkinkan untuk mendeteksi objek secara realtime. Karenanya dibutuhkan algoritma yang bisa melakukan deteksi yang kurang dari 1 detik sehingga bisa diterapkan untuk frame persecond-nya. Sehingga pada saat itulah dikenalkan Region Proposal Network (RPN). Dimana tidak memerlukan selective search lagi dalam proses training (Ren et al., 2015). Hal ini bisa dilakukan dengan cara mengganti selective search menggunakan RPN. RPN bekerja dengan *sharing layer* secara parallel dengan *feature extractor* yang ada, sehingga bisa diklaim *cost free*. Kemudian sebagai detector tetap menggunakan seperti fast rcnn detector. Arsitektur dari Faster R-CNN terdiri dari 3 bagian berikut.

1. Feature extractor, berupa VGG-16, bisa diganti dengan arsitektur cnn lainnya. Pada penelitian ini menggunakan Resnet_50_fpn
2. RPN untuk melakukan generate proposal
3. Fast R-CNN untuk mendeteksi object yang berada di *proposed region*.

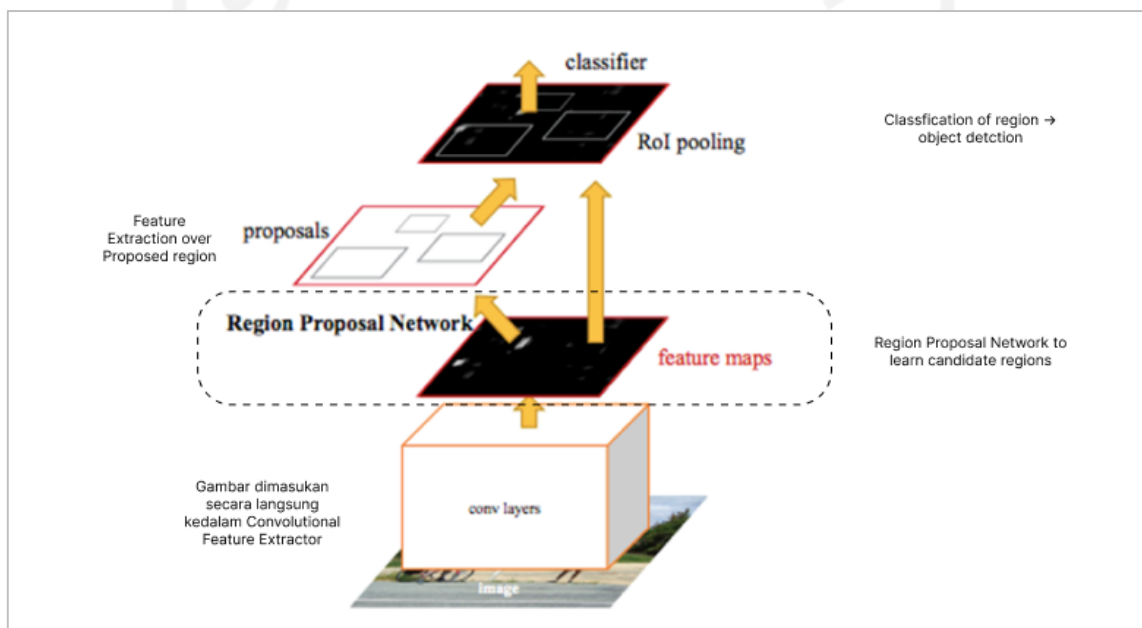
Untuk memahami lebih detailnya berikut arsitektur faster rcnn disetiap bagian, bisa mengacu pada Gambar 2. 13 yang dibagi menjadi 3 bagian. Pertama ketika gambar dimasukkan kedalam CNN yang mempunyai tugas untuk melakukan *feature extraction* hasilnya yang berupa *feature map*. Feature map ini yang akan menjadi inputan dari Region Proposal Network (RPN). Kemudian akan di proses oleh RPN untuk menghasilkan region proposal. Yaitu berupa background atau foreground dan bounding box (4 koordinat). Kemudian feature map dari proses yang dihasilkan oleh RPN akan diklasifikasikan oleh bagian fast-rcnn dengan melewati proses ROI Pooling terlebih dahulu.



Gambar 2. 13 Arsitektur Faster R-CNN¹⁰

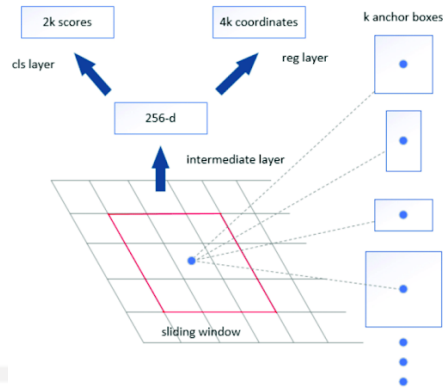
¹⁰ <https://towardsdatascience.com/faster-rcnn-object-detection-f865e5ed7fc4>

Jika pada algoritma fast r-cnn ataupun r-cnn bagian yang bertanggung jawab terhadap *proposed region* menggunakan algoritma yang terpisah dengan layer *detector*-nya. Sedangkan jika mengacu Pada Gambar 2. 14 faster r-cnn justru menggunakan CNN untu kemudian hasilnya akan berupa *feature map*. Inilah yang akan menjadi inputan untuk *region proposal network* (RPN). Setelah dilakukan operasi pada RPN barulah hasilnya akan dikirimkan ke detector. Dalam hal ini menggunakan fast r-cnn sebagai detectornya. Bagian *proposed region* dan *detector* sudah tidak perlu menggunakan algoritma dari external lagi. Karena sama-sama memanfaatkan *feature map* dari hasil *feature extraction* yang ada. Sehingga dengan metode ini peningkatan kecepatan menjadi 2-3 *ms* untuk mendeteksi objek dalam satu gambar .



Gambar 2. 14 RPN memanfaatkan feature map untuk menghasilkan *proposed region*.

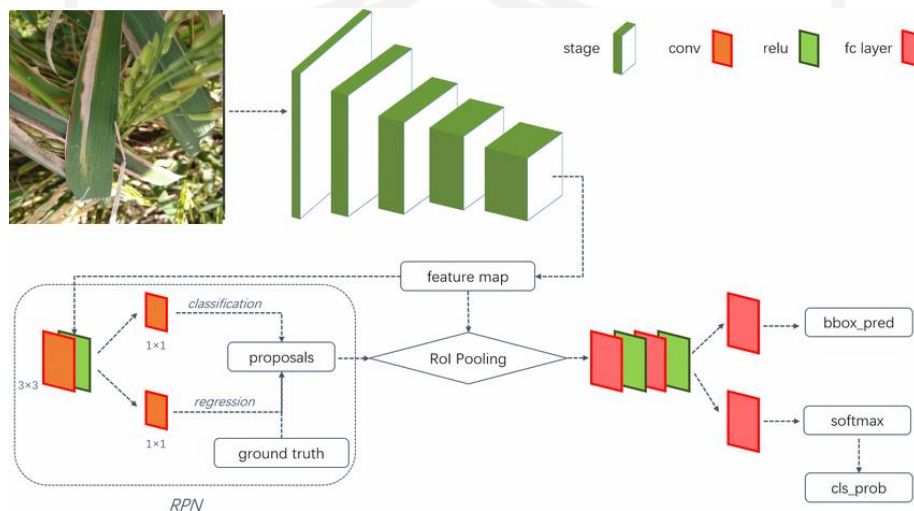
Untuk menghasilkan bounding box yang digunakan untuk object detection seperti pada Gambar 2. 15 terlihat supaya bisa mengakomodir rasio objek yang berbeda, dalam proses sliding window yang terjadi di layer RPN juga menggunakan istilah yang baru yaitu menggunakan anchor box. Dimana anchor box yang digunakan oleh RPN mempunyai 3 rasio dan 3 skala yang berbeda-beda sehingga total anchor box dalam satu kali sliding window sebanyak 9 anchor box. Sehingga nantinya model akan mampu mendeteksi objek walau dengan rasio dan ukuran yang berbeda-beda.



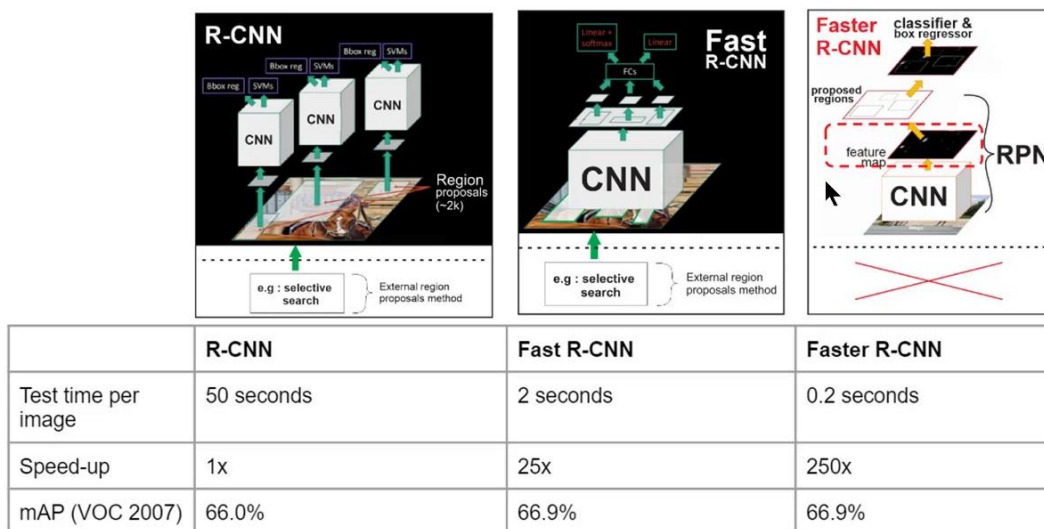
Gambar 2. 15 Anchor box yang digunakan pada layer RPN.

Pada penelitian ini tahapan-tahapan yang dilalui ketika menggunakan model faster r-cnn bisa mengacu pada Gambar 2. 16. Pada gambar terlihat mulai dari proses foto dimasukan. Kemudian akan di ekstrak featurenya oleh pretrained model cnn, disini menggunakan restnet50_fpn. Selanjutnya setelah feature

map dihasilkan barulah RPN bekerja untuk memprediksi bounding box dan menentukan foreground atau backgroundnya. Hasil dari bounding box akan dibandingkan dengan ground truth untuk mencari nilai intersection over union dari prediksi yang dbuat. Kemudian barulah feature map dari RPN dan dari CNN kemudian dimasukan kedalam ROI Pooling. Tujuannya untuk menyamakan ukuran dari feature map yang ada. Setelah ukurannya sama baru bagian akhir diklasifikasikan kelasnya dan bounding boxnya. Dengan hasil kecepatan peningkat disbanding algoritma sebelumnya lihat pada Gambar 2. 17.



Gambar 2. 16 Arsitektur Faster R-CNN



Gambar 2. 17 Perbandingan kecepatan dari R-CNN, Fast R-CNN, Faster R-CNN¹¹.

2.2.8 Resnet

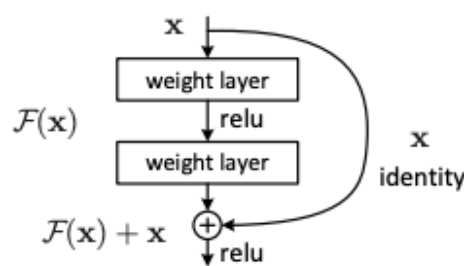
Dalam beberapa tahun terakhir terobosan dalam bidang *computer vision* terutama dalam pemanfaatan *Convolutional Neural Network* telah mendapatkan hasil yang baik terutama dalam bidang *image classification* dan *image recognition* (He et al., 2015). Untuk mendapatkan hasil yang baik para peneliti cenderung membuat *deep neural network* lebih dalam dengan cara menambahkan lebih banyak layer untuk menyelesaikan masalah yang lebih kompleks dan meningkatkan kinerja di bidang *image recognition/classification* (Zhong et al., 2017). Akan tetapi dengan semakin banyaknya layer pada *deep neural network* menyebabkan semakin sulit untuk di training dan mulai mengalami saturasi dan penurunan pada akurasi. Untuk menyelesaikan masalah tersebut resnet.

Resnet merupakan singkatan dari *Residual Network* yang mengacu kepada *neural network* yang dikenalkan oleh Kaiming he pada tahun 2015 pada *paper* yang berjudul "Deep Residual Learning for Image Recognition" (He et al., 2015). Dalam beberapa kasus penambahan layer pada *deep neural network* meningkatkan akurasi dan performa. Anggapan dengan mempunyai layer dengan kedalaman yang lebih banyak juga akan secara *progresif* akan mampu mempelajari fitur yang lebih kompleks. sebagai contoh dalam bidang *image recognition* dengan layer pertama digunakan untuk mendeteksi sudut dan layer kedua mengidentifikasi tekstur, sedangkan pada layer ketiga digunakan untuk

¹¹ <https://analyticsindiamag.com/r-cnn-vs-fast-r-cnn-vs-faster-r-cnn-a-comparative-guide/>

mendeteksi object dan lainnya. Akan tetapi ditemukan *threshold maximum* pada kedalaman *layer*. Khususnya model *Convolutional Neural Network* tradisional. Masalah akibat kedalaman *layer* pada *neural network* biasanya dikenal dengan *Vanishing Gradient / Exploding gradient* (Zhong et al., 2017).

Permasalahan *Vanishing Gradient* dan *Exploding Gradient* yang menghambat konvergen dari awal bisa diatasi dengan proses normalisasi dan inisialisasi. Sedangkan penambahan normalisasi pada bagian tengah mampu membuat convergence layer dengan puluhan layer untuk Stochastic Gradient Descent (SGD) dengan back propagation. Masalah degradasi inilah yang coba diselesaikan oleh framework *Deep Residual Network*. Dengan cara menggunakan *residual block* dengan menambahkan koneksi secara langsung dan melewati beberapa layer dalam sebuah model seperti pada Gambar 2. 18. Dibanding dengan menumpuk setiap layer secara langsung dan diproses secara berurutan. Dimana jika layer yang dimiliki terlalu dalam akan mengakibatkan terjadi *vanishing gradient* ataupun *exploding gradient*. Oleh sebab itu dibuatlah sebuah block yang mampu melakukan shortcut koneksi yang disebut *skip connection*. Metode inilah yang menjadi jantung dari *Residual Network*.

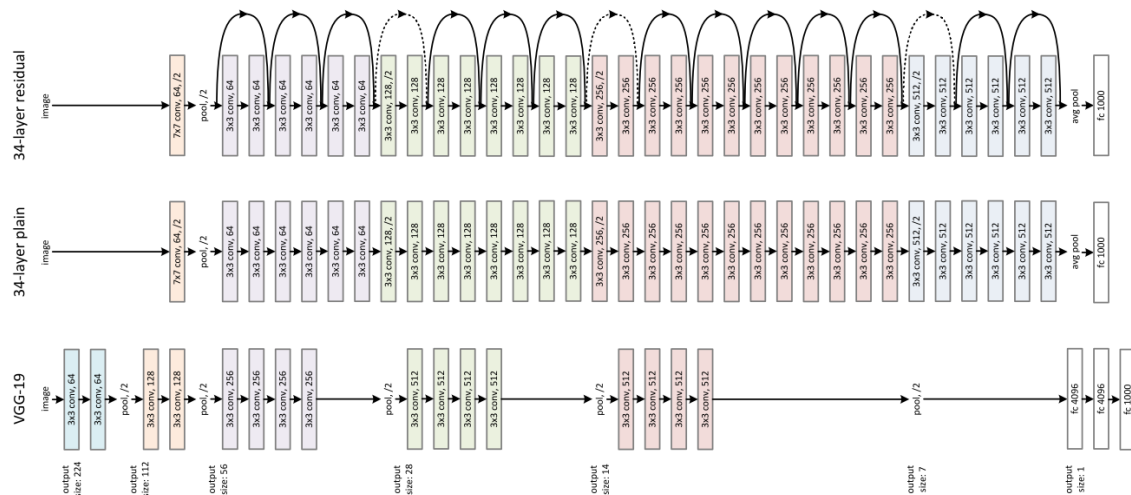


Gambar 2. 18 Residual Block¹²

Dengan menggunakan teknik *skip connection* Resnet mampu menyelesaikan masalah *Vanishing Gradient* dan *Exploding Gradient* pada CNN. Untuk mengetahui bagaimana cara kerja dari Resnet perlu mengetahui arsitektur yang digunakannya. Pada Gambar 2. 19 arsitektur Resnet sendiri terdiri dari 34 *layer plain network*. Terinspirasi dari VGG-19 dengan penambahan *skip connection*. Dimana VGG-19 adalah salah satu arsitektur CNN yang lebih awal dari Resnet mampu mengidentifikasi 1000 gambar dengan 1000 kategori yang berbeda (He et al., 2015). VGG-19 merupakan salah satu algoritma untuk image classification yang populer dan mudah digunakan untuk transfer learning.

¹² <https://www.geeksforgeeks.org/residual-networks-resnet-deep-learning/>

Dengan demikian arsitektur yang dimiliki oleh Resnet juga mirip dengan VGG-19 yang berbeda hanya penambahan *skip connection*-nya saja.



Gambar 2. 19 Perbedaan VGG-19 dan Resnet source = 'Deep Residual Learning for Image Recognition' paper

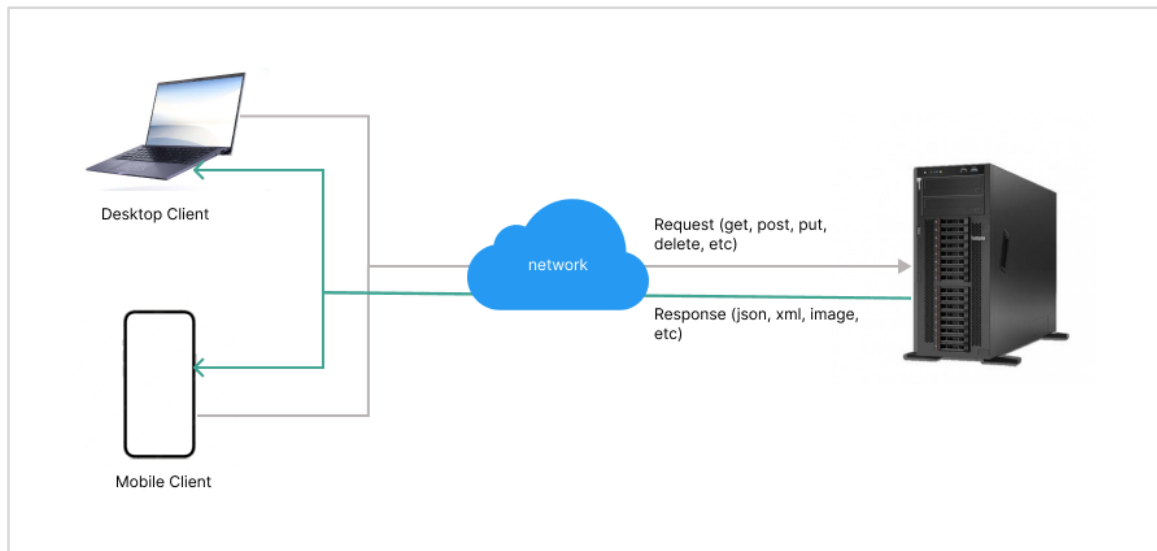
Framework yang digunakan dalam penelitian ini adalah pytorch, yang sudah umum digunakan untuk research di bidang machine learning dan deep learning. Pytorch merupakan framework yang berbasis bahasa pemrograman python dan torch library¹³. pytorch sendiri sudah mempunyai modul khusus untuk faster rcnn, sehingga lebih mudah dalam proses eksplorasi. Ketika menggunakan pytorch sebagai framework dengan memanfaatkan algoritma faster rcnn, maka sudah tersedia beberapa backbone yang bisa digunakan. Salah satunya adalah *pretrained* Restnet_50_fpn yang merupakan backbone dengan mAP yang cukup tinggi, namun kelemahannya lebih lambat bila dibandingkan dengan backbone yang lain, seperti mobilenet.

2.2.9 Web Service

Web service merupakan aplikasi yang berisi sekumpulan basis data (database) dan perangkat lunak (software) atau bagian dari program perangkat lunak yang diakses secara remote oleh piranti dengan perantara tertentu (Al-Masri & Mahmoud, 2007). Biasanya akan menerima *request* berupa *web document* (HTML, JSON, XML, Images) dan dibuat spesifik untuk menyelesaikan masalah tertentu. Pada Gambar 2. 20 terlihat ada 2 client

¹³ <https://towardsdatascience.com/what-is-pytorch-a84e4559f0e3>

(*mobile, desktop*) yang mencoba melakukan request data ke computer server dengan metode `get`, `post`, `put`, `delete` maupun yang lainnya. Jika client yang melakukan request dianggap berhak menerima informasi atas data yang ada pada server, maka server akan mengirimkan response sesuai dengan permintaan masing-masing dari client yang ada.



Gambar 2. 20 Arsitektur Web Service¹⁴

Evolusi layanan web terjadi karena semua platform utama dapat mengakses internet, tetapi platform yang berbeda tidak dapat berinteraksi satu sama lain. Oleh sebab itu salah satu cara untuk bisa berkomunikasi antar aplikasi, web service menjadi salah satu pilihan yang bisa digunakan.

2.2.10 Jenis Penyakit Padi

Pada penelitian ini penulis hanya mengambil 3 contoh dari penyakit tumbuhan padi, yang merupakan 3 penyakit yang banyak di temui di Indonesia (Nisa, 2021) . Namun kedepannya tidak tertutup kemungkinan penelitian ini bisa dikembangkan lebih lanjut, baik itu penyakit yang berbeda pada tumbuhan padi ataupun penyakit lainnya pada tumbuhan yang berbeda. Berikut penyakit yang akan dibahas pada penelitian ini.

1. Penyakit Hawar Daun

Penyakit hawar daun bakteri (HDB) merupakan salah satu penyakit penting tanaman padi di negara-negara penghasil padi, termasuk di Indonesia. Penyakit ini disebabkan oleh bakteri *Xanthomonas oryzae* pv. *oryzae* (Xoo) (Sudir et al., 2012). Patogen ini

¹⁴ <https://breefstudio.com/apa-bedanya-web-service-dan-api/>

menginfeksi daun padi pada semua fase pertumbuhan tanaman, mulai dari persemaian sampai menjelang panen, perhatikan *Gambar 2. 21*



Gambar 2. 21 Penyakit hawar daun

Gejala yang timbul pada tanaman fase vegetatif disebut kresak dan pada fase generatif disebut hawar. Apabila infeksi terjadi pada fase generatif mengakibatkan proses pengisian gabah menjadi kurang sempurna. Kehilangan hasil karena penyakit HDB bervariasi antara 15–80%, bergantung pada stadia tanaman saat penyakit timbul. Perkembangan penyakit HDB dipengaruhi oleh lingkungan terutama kelembaban, suhu, cara budidaya, varietas, dan pemupukan nitrogen. Oleh karena itu, pengendalian yang dianjurkan adalah secara terpadu dengan berbagai cara yang dapat menekan perkembangan penyakit. Varietas tahan merupakan komponen utama dalam pengendalian penyakit HDB secara terpadu (Masnilah et al., 2013). Namun aplikasi teknologi ini terkendala oleh kemampuan patogen beradaptasi membentuk patotipe (strain) baru yang lebih virulen sehingga sifat ketahanan varietas mudah patah. Pengendalian Penyakit HDB dengan penanaman varietas tahan harus disesuaikan dengan patotipe yang ada. Monitoring dan pemetaan komposisi dan penyebaran patotipe bakteri *Xoo* diperlukan sebagai dasar rekomendasi pengendalian penyakit HDB dengan varietas tahan sesuai dengan keberadaan patotipe di suatu tempat.

2. Blas

Penyakit blas merupakan salah satu penyakit utama pada tanaman padi di seluruh dunia (Sudir et al., 2015). Berbagai negara sudah memiliki data tentang intensitas dan kehilangan hasil padi oleh penyakit blas, sementara di Indonesia belum ada hasil pengujian yang khusus dirancang untuk melihat tingkat intensitas penyakit blas di daerah endemik Asia. Jamur *C. oryzae* merupakan penyebab penyakit Bercak coklat sempit yang mempunyai sebaran yang luas di daerah tropis dan sub-tropis. Penyakit ini biasanya terjadi pada tanah yang kekurangan kalium, dan di daerah dengan suhu berkisar antara 25 - 28 °C (GmbH, n.d.-a). Penyakit ini muncul selama tahap akhir pertumbuhan tanaman padi, dimulai pada fase keluar malai. Inang alternatif memungkinkan jamur bertahan hidup dan menginfeksi tanaman padi baru. Tanaman paling rentan selama inisiasi malai dan seterusnya, dan kerusakan menjadi lebih parah saat tanaman mendekati kematangan. dan potensi kehilangan hasil yang diakibatkannya, perhatikan *Gambar 2. 22*.



Gambar 2. 22 Penyakit Blas

Data sejenis ini penting untuk berbagai keperluan antara lain untuk kebijakan pengendalian dan dasar pentingnya penelitian (Suganda et al., 2016). Suatu percobaan telah dilakukan di Desa Bojong, Kecamatan Cikembar, Kabupaten Sukabumi yang merupakan daerah endemik penyakit blas selama bertahun-tahun, menggunakan varietas Ciherang yang merupakan varietas rentan namun populer ditanam petani. Delapan bahan aktif fungisida berbahan aktif tunggal dan majemuk digunakan untuk melihat pengaruh penekanannya dibandingkan kontrol. Percobaan menggunakan Rancangan

Acak Kelompok dengan lima ulangan. Parameter yang diamati adalah intensitas penyakit blas daun dan blas leher malai serta hasil padi. Hasil percobaan menunjukkan bahwa intensitas penyakit blas pada tanaman padi var. Ciherang di daerah endemik adalah 55,60% untuk blas daun, dan 37,75% untuk blas leher malai. Potensi kehilangan hasil oleh gabungan penyakit blas pada varian Ciherang adalah 3,65 ton/ha atau setara dengan 61% kehilangan hasil jika dibandingkan terhadap rata-rata produksi varian Ciherang menurut spesifikasi varietas. Data ini menjustifikasi perlunya pengambilan tindakan pengendalian penyakit blas pada tanaman padi.

3. Penyakit Bercak Sempit

Penyakit bercak daun cercospora atau bercak coklat sempit (narrow brown leaf spot) tersebar luas di negara-negara penanam padi. Di Indonesia penyakit berkembang dengan baik terutama pada daerah-daerah lahan yang miskin unsur nitrogen dan kalium. Menjadi penyakit utama pada pertanaman padi lahan sawah tadah hujan dan gogo (PRASETYO, 2017). Penyakit menyerang tanaman padi terutama pada bagian daun menyebabkan fungsi fotosintesis terganggu. Apabila serangan terjadi pada fase generatif menyebabkan pengisian gabah menjadi kurang sempurna atau hampa sehingga bobot gabah dan kualitas gabah menjadi rendah.

Gejala Penyakit Pada daun terdapat bercak-bercak sempit memanjang, berwarna coklat kemerahan, sejajar dengan ibu tulang daun. Banyaknya bercak makin meningkat pada waktu tanaman membentuk anakan. Pada serangan yang berat bercak-bercak terdapat pada upih daun, batang, dan bunga. Pada saat tanaman mulai masak gejala yang berat mulai terlihat pada daun bendera. Gejala mulai tampak 2-4 minggu setelah padi di pindah, dan gejala paling berat tampak lebih kurang satu bulan sebelum panen. Penyebab Penyakit Penyakit disebabkan oleh jamur *Cercospora janseane* (Racib) O. Const. Semula jamur disebut *Napicladium janseanum* Racib. Di Jepang disebut *Cercospora oryzae* Miyake. Jamur membentuk konidiofor berwarna coklat, keluar melalui mulut kulit, sendiri-sendiri atau berkumpul sampai 3, dengan ukuran 88-140 x 4-5 μm . Konidium berbentuk gada terbalik, bersekat 3-10, dengan ukuran 20-60 x 5 μm (Ou, 1985). Perhatikan gambar *Gambar 2. 23*



Gambar 2. 23 Penyakit bercak daun cercospora

Daur Penyakit Konidium jamur disebarkan oleh angin dan infeksi terjadi melalui mulut kulit. Gejala baru tampak 30 hari atau lebih setelah infeksi. Ini menyebabkan lambatnya gejala di lapangan, meskipun infeksi dapat terjadi pada daun muda maupun daun tua. *C. janseana* mempertahankan diri dari musim ke musim pada biji-biji dan jerami. Diduga jamur dapat bertahan pada rumput-rumput liar; antara lain di India jamur dapat menginfeksi lempuyangan (*Panicum repens*).

Faktor-faktor yang Mempengaruhi Penyakit umumnya penyakit bercak daun cercospora berkembang lebih baik pada musim kemarau. Meskipun belum diketahui secara pasti varietas-varietas yang tahan dan rentan terhadap penyakit ini (Atmaja et al., 2020), tetapi kenyataan di lapangan sering menunjukkan reaksi yang sangat beragam. Penyakit sangat dipengaruhi oleh jenis padi. Varietas Ciherang, IR 64 dan turunannya dilapangan menunjukkan reaksi sangat rentan terhadap bercak cercospora. Kandungan unsur hara terutama nitrogen dan kalium sangat berpengaruh terhadap perkembangan penyakit ini. Tanaman padi yang kekurangan unsur nitrogen dan kalium lebih rentan terhadap penyakit bercak daun cercospora.

BAB 3

Metodologi

Pada penelitian ini proses perencanaan sampai dengan penelitian ini selesai memiliki beberapa tahapan sesuai dengan Gambar 3. 1.

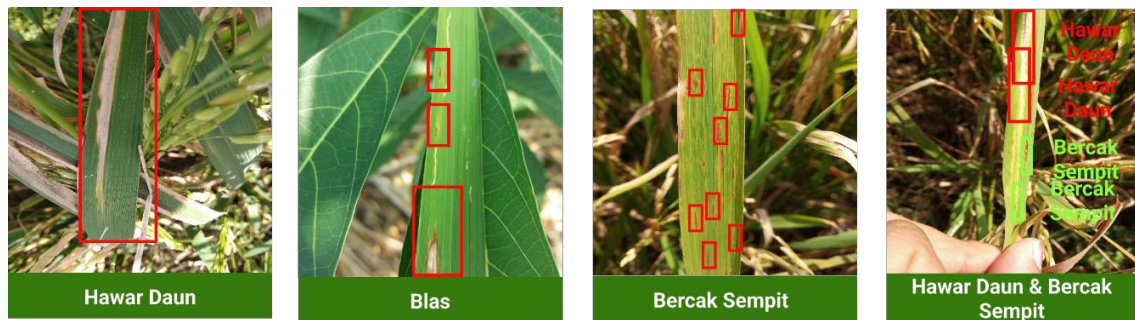


Gambar 3. 1 Metodologi penelitian

3.1 Pengumpulan Data

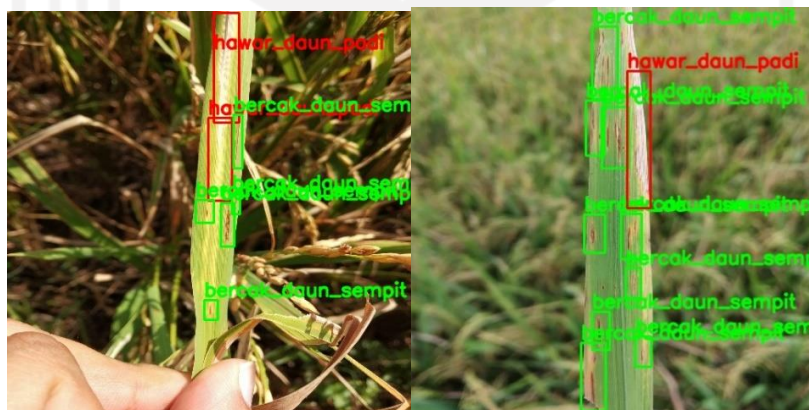
Penelitian ini menggunakan data yang diambil secara langsung menggunakan kamera hp dan kamera digital. Jumlah data yang tersedia 500 untuk setiap kelasnya. Penyakit yang ada pada penelitian ini adalah 3 penyakit yang paling mengancam di Indonesia khususnya pada tumbuhan padi¹⁵. Berikut contoh dari gambar penyakit yang ada pada Gambar 3. 2 yang merupakan 3 jenis penyakit yang paling banyak di Indonesia.

¹⁵ <https://bbpadi.litbang.pertanian.go.id/index.php/en/publikasi/booklet-leaflet-poster/tag/HawarDaunBakteri>



Gambar 3. 2 Contoh Dataset Hawar daun, blast, bercak sempit

Data diambil dari lapangan menggunakan kamera digital canon EOS 60D, dan kamera hp. Dari hasil pengamatan gambar didapatkan ternyata ditemukan ada padi yang megidap beberapa penyakit secara bersamaan, hal ini sesuai dengan hasil data yang di himpun dari expert HPT dari PT.Msmb. dengan adanya data yang multiclass dalam satu gambar akan memperkaya penelitian ini, dan menjadi tantangan tersendiri untuk menghasilkan model yang mampu mendeteksinya nanti. Tanaman yang mengidap lebih dari satu penyakit juga ditampilkan pada Gambar 3. 3 yang mana tumbuhan padi terinfeksi penyakit hawar daun dan bercak sempit sekaligus.



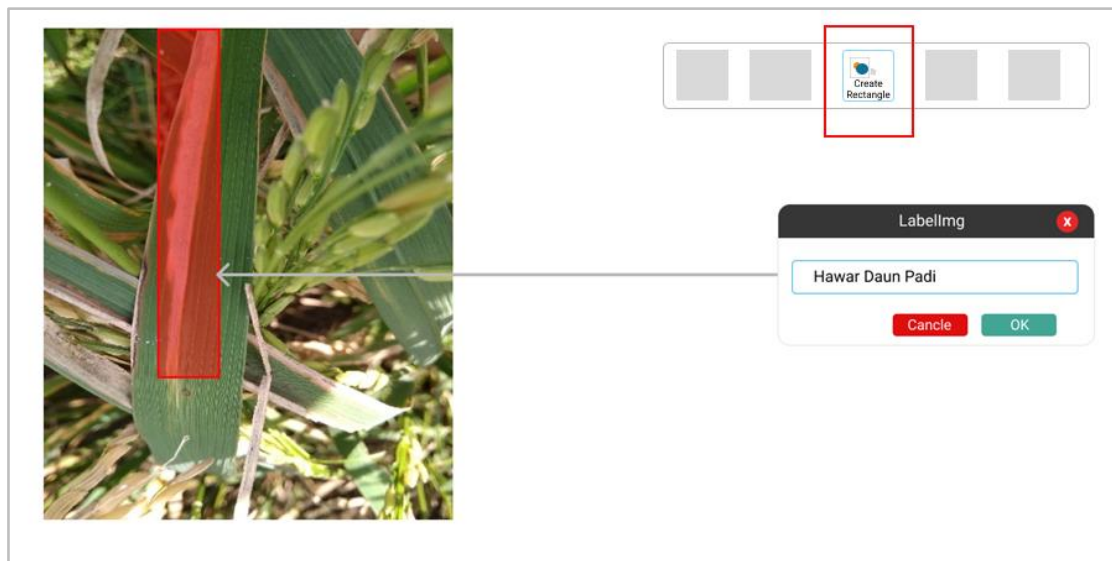
Gambar 3. 3 penyakit padi yang dalam satu daun lebih dari satu penyakit

3.2 Pelabelan Data

Dalam proses pelabelan data jika menggunakan deep learning untuk image classification, dan deep learning untuk object detection adalah dua hal yang berbeda. Pada image classification cukup dengan mengelompokan gambar berdasarkan kelasnya, Misal dengan folder yang dipisah atau cukup dengan penamaan yang menggunakan prefix. Biasanya 1 gambar hanya dianggap satu kelas yang paling dominan. Sedangkan untuk deep learning

dengan tujuan object detection data harus dilabeli dengan cara anotasi pada gambar.

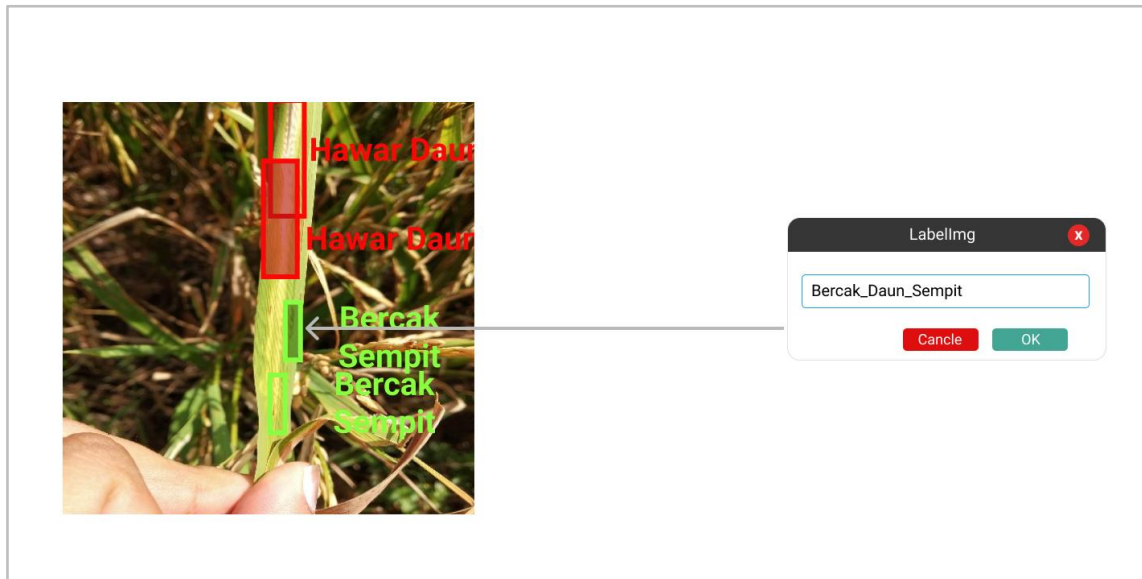
Proses anotasi diperlukan khususnya pada dataset dengan tujuan objek detection dan juga object segmentation. Akibatnya jumlah data gambar yang didapat belum tentu sama dengan hasil pelabelan dari anotasi. Pada penelitian ini untuk memudahkan proses anotasi menggunakan aplikasi khusus anotasi yang bernama labelImg¹⁶. Berdasarkan keterangan resmi dari website penyedia labelImg, aplikasi ini merupakan *graphical image annotation tools*, yang dibuat dengan menggunakan Bahasa pemrograman python dan Qt untuk sebagai interface (antarmuka) yang memudahkan dalam proses anotasi. Aplikasi ini support beberapa format anotasi seperti Pascal VOC yang digunakan oleh ImageNet, dan juga support format Yolo serta CreateML. Selain itu aplikasi ini juga *multiplatform* dengan kata lain bisa digunakan di sistem operasi *mainstream* seperti linux, windows, dan juga Mac OS. Untuk melakukan anotasi pertama lokasi folder dibuka melalui aplikasi labelImg terlebih dahulu, kemudian dipilih rectangle tool seperti pada Gambar 3. 4 untuk 1 kelas dalam satu gambar, sedangkan untuk multiclass ada pada Gambar 3. 5, kemudian gambar pada area yang terdapat penyakit serta isikan kelasnya. Proses inilah yang akan mempengaruhi kuantitas data setiap kelasnya.



Gambar 3. 4 Proses pelabelan dengan labelImg

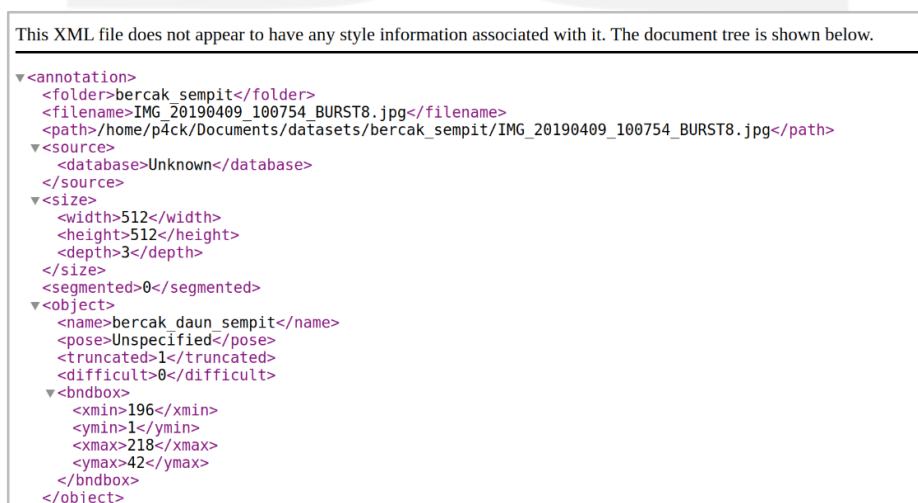
Sementara itu file anotasi akan mengandung informasi yang diperlukan untuk proses training dan validation nanti, yang berupa kordinat bounding box, yang terdiri dari 4 kordinat. Sedangkan informasi yang terkandung dalam file .xml sebagai berikut.

¹⁶ <https://github.com/tzutalin/labelImg>



Gambar 3. 5 Multi penyakit dalam 1 tanaman

Sementara itu file anotasi akan mengandung informasi yang diperlukan untuk proses training dan validation khususnya object detection, yang berupa koordinat bounding box terdiri dari 4 titik. Format yang digunakan pada penelitian ini yaitu “Pascal VOC” dengan ekstensi xml. Sedangkan informasi yang terkandung dalam file .xml pada Gambar 3. 6 ada nama folder pada bagian “<annotation><folder>nama_folder</folder>”, ukuran gambar tinggi dan lebar “<size> <width> ukuran_lebar </width> <height> ukuran_lebar </height>” kemudian untuk nama kelasnya ada pada “<object> <name>nama_kelas</name>” yang terakhir yang perlu diketahui yaitu koordinat dari bounding boxnya untuk koordinat x “<bndbox><xmin>kordinat_1</xmin><xmax>kordinat_2</xmax>” dan untuk bagian kordinat y “<bndbox><ymin>kordinat_1</ymin> <ymax>kordinat_2</ymax>”.



Gambar 3. 6 Struktur file xml dari hasil anotasi

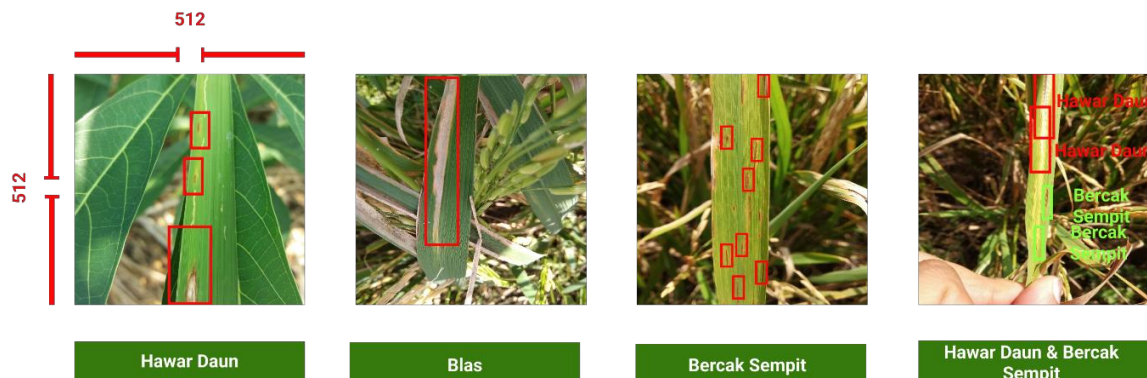
Tahap awal dalam proses pemisahan antar kelas data di pisahkan berdasarkan folder, walaupun tidak selalu dalam satu gambar memiliki satu penyakit, secara umum ada penyakit yang dominan kemudian file anotasi berekstensi .xml. Untuk melihat sebaran data maka file .xml diubah kedalam csv pada Gambar 3. 7 supaya bisa memudahkan melihat distribusi dataset setelah dianotasi. Pada file csv akan ada 8 kolom yang dihasilkan yaitu “filename” yang mengacu kepada nama gambar, “width dan height” mengacu pada lebar dan tinggi gambar, kemudian “class” mengacu kepada nama penyakit serta “xmin, ymin, xmax, ymax” mengacu kepada koordinat dari anotasi kelas.

	filename	width	height	class	xmin	ymin	xmax	ymax
0	20190430_083713_004.jpg	512	512	blas_daun_padi	232	166	257	206
1	20190430_083713_004.jpg	512	512	blas_daun_padi	233	227	274	357
2	20190430_083713_004.jpg	512	512	blas_daun_padi	246	42	263	86
3	20190430_083713_004.jpg	512	512	blas_daun_padi	275	14	295	41
4	20190430_083713_004.jpg	512	512	blas_daun_padi	287	494	310	512

Gambar 3. 7 Struktur data dari file .xml diubah menjadi csv

3.3 Resize Gambar

Proses training data adalah salah satu tahapan yang memerlukan komputasi yang tidak murah, terlebih data yang digunakan berupa data gambar, audio, dan video dengan kualitas tinggi. Untuk mengurangi beban pada saat training data, maka pada penelitian ini akan dilakukan proses resize gambar. Dengan melakukan proses resize gambar, diharapkan proses training data bisa lebih efisien dan lebih cepat. Pada Gambar 3. 8 ukuran gambar disamakan menjadi 512X512 yang awalnya gambar mempunyai ukuran yang berbeda beda.



Gambar 3. 8 Contoh Dataset yang sudah di resize, termasuk dengan bounding box-nya

Untuk melakukan proses resize gambar menjadi gambar dengan ukuran yang sama dibuat fungsi dengan menggunakan opencv seperti pada Gambar 3. 9. Sehingga gambar yang awalnya mempunyai resolusi dan ukuran yang berbeda-beda menjadi gambar dengan rasio dan resolusi yang sama.

```

26     def __getitem__(self, idx):
27         # capture the image name and the full image path
28         image_name = self.all_images[idx]
29         image_path = os.path.join(self.dir_path, image_name)
30
31         # read the image
32         image = cv2.imread(image_path)
33         # convert BGR to RGB color format
34         image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB).astype(np.float32)
35         image_resized = cv2.resize(image, (self.width, self.height))
36         image_resized /= 255.0
37

```

Gambar 3. 9 Potongan kode untuk melakukan proses resize gambar

- Pada baris 26 merupakan method dengan parameter idx yang mengacu pada index dari data
- Pada baris ke 29 merupakan potongan kode untuk membaca path gambar dan melakukan proses load ke memory, dan
- Proses resize dilakukan pada baris ke 35.

3.4 Training Data

Model pada *machine learning* bergantung pada data (Brodley & Friedl, 1999). Tanpa dasar data pelatihan berkualitas tinggi, bahkan algoritma yang paling berkinerja pun dapat dianggap tidak berguna. Pada penelitian kali ini, penulis melakukan proses pelabelan data dengan bantuan ahli dibidang pertanian yang bekerja di perusahaan berskala nasional, dengan catatan penulis tetap melakukan verifikasi ulang terhadap data, untuk mencegah human error. berikut struktur direktori untuk membangun model pada penelitian ini, perhatikan Gambar 3. 10.

```

├── engine
│   ├── test
│   ├── train
│   ├── test_labels.csv
│   └── train_labels.csv
├── outputs
│   ├── model1005.pth
│   ├── ...
│   └── valid_loss_98.png
├── src
│   ├── config.py
│   ├── datasets.py
│   ├── engine.py
│   ├── inference.py
│   ├── model.py
│   ├── utils.py
│   └── test data

```

Gambar 3. 10 Struktur direktori untuk membangun model

- Folder test berisikan dataset yang sudah dipisah untuk sebagai validation set, sebanyak 30% dari data.
 - Sedangkan pada folder train berisikan file gambar.jpg dan .xml yang menjadi data training dan anotasi dari file gambar.
 - Folder output berisikan model yang di hasilkan dari proses training data dengan format .pth yang merupakan standar dari pytorch.
 - Src berisikan potongan kode yang di pecah menjadi beberapa file, bertujuan untuk mempermudah dalam proses pengembangan.
 - Test_data berisikan gambar yang akan diuji ketika pembuatan models telah selesai.
 - Test_prediction berisikan hasil dari uji yang kemudian akan ditambahkan koordinat
- Sebelum melakukan proses training, pertama kali yang harus dilakukan adalah membuat pengaturan yang akan digunakan oleh untuk proses *training* perhatikan Gambar 3. 11.

```

1  import torch
2  BATCH_SIZE = 4 # increase / decrease according to GPU memory
3  RESIZE_TO = 512 # resize the image for training and transforms
4  NUM_EPOCHS = 100 # number of epochs to train for
5  DEVICE = torch.device('cuda') if torch.cuda.is_available() else torch.device('cpu')
6  # training images and XML files directory
7  TRAIN_DIR = 'train'
8  # validation images and XML files directory
9  VALID_DIR = 'test'
10 # classes: 0 index is reserved for background
11 CLASSES = [
12     'background', 'hawar_daun_padi', 'blas_daun_padi', 'bercak_daun_sempit'
13 ]
14 NUM_CLASSES = 4
15 # whether to visualize images after creating the data loaders
16 VISUALIZE_TRANSFORMED_IMAGES = False
17 # location to save model and plots
18 OUT_DIR = '../outputs'
19 SAVE_PLOTS_EPOCH = 2 # save loss plots after these many epochs
20 SAVE_MODEL_EPOCH = 2 # save model after these many epochs
21
22

```

Gambar 3. 11. Konfigurasi untuk proses training data.

- Baris ke 1 proses untuk mengimport pytorch akan digunakan untuk mendeteksi apakah menggunakan **GPU** atau **CPU** pada proses training nantinya yang di definisikan pada baris ke 5.
- Baris ke 7 mendefinisikan letak dari direktori dataset yang akan digunakan
- Baris ke 9 berisikan baris letak dari direktori validation set (testset).
- Baris ke 11 sampai ke 13 mendefinisikan nama-nama kelas yang akan digunakan untuk proses training, dimana pada index ke 0 merupakan default background dari gambar.

untuk mendapatkan model yang *robust* melakukan augmentasi diperlukan jika dataset yang dimiliki sedikit secara kuantitas ataupun secara variasi. Pada Gambar 3. 12 pada file `utils.py` yang berisikan potongan kode augmentasi menggunakan modul `albumentation`.

```

1  # define the training transforms
2  def get_train_transform():
3      return A.Compose([
4          A.Flip(0.5),
5          A.RandomRotate90(0.5),
6          A.MotionBlur(p=0.2),
7          A.MedianBlur(blur_limit=3, p=0.1),
8          A.Blur(blur_limit=3, p=0.1),
9          ToTensorV2(p=1.0),
10     ], bbox_params={
11         'format': 'pascal_voc',
12         'label_fields': ['labels']
13     })
14 # define the validation transforms
15 def get_valid_transform():
16     return A.Compose([
17         ToTensorV2(p=1.0),
18     ], bbox_params={
19         'format': 'pascal_voc',
20         'label_fields': ['labels']
21     })

```

Gambar 3. 12 potongan kode `albumentation` untuk augmentasi dataset.

- Baris ke 2 sampai dengan 13 merupakan potongan kode untuk augmentasi pada saat proses training
- Baris ke 15 sampai ke 21 merupakan potongan kode untuk melakukan augmentasi pada saat proses validasi.

Berikutnya untuk melakukan proses pengambilan data yang sudah di anotasi kita menggunakan `xmltree` dari python kemudian melakukan perulangan untuk mengambil objek yang lebih dari satu sesuai dengan hasil anotasi. Berikut potongan kode pengambilan **bbox** pada data `.xml` mengacu pada Gambar 3. 13.

```

8  # xmin = left corner x-coordinates
9  xmin = int(member.find('bndbox').find('xmin').text)
10 # xmax = right corner x-coordinates
11 xmax = int(member.find('bndbox').find('xmax').text)
12 # ymin = left corner y-coordinates
13 ymin = int(member.find('bndbox').find('ymin').text)
14 # ymax = right corner y-coordinates
15 ymax = int(member.find('bndbox').find('ymax').text)
16
17 # resize the bounding boxes according to the...
18 # ... desired `width`, `height`
19 xmin_final = (xmin/image_width)*self.width
20 xmax_final = (xmax/image_width)*self.width
21 ymin_final = (ymin/image_height)*self.height
22 ymax_final = (ymax/image_height)*self.height
23
24 boxes.append([xmin_final, ymin_final, xmax_final, ymax_final])
25

```

Gambar 3. 13 mengambil koordinat gambar sesuai hasil anotasi dengan format VOC pada file `.xml`

- Pada baris ke 9 sampai dengan baris 15 merupakan bagian untuk mengambil setiap titik dari bounding box pada file .xml.
- Baris ke 19 sampai dengan menyesuaikan perubahan rasio dengan ukuran gambar.
- Baris ke 24 mengembalikan nilai kordinat yang sudah diubah.

Selanjutnya untuk bisa melakukan proses training, data harus di load terlebih dahulu ke format yang digunakan oleh framework pytorch. Untuk kemudian baru bisa dilakukan ke Langkah berikutnya perhatikan Gambar 3. 14. Setelahnya baru memasukan hyper parameter pada Gambar 3. 15.

```

1
2 train_dataset = Identification(TRAIN_DIR, RESIZE_T0, RESIZE_T0, CLASSES, get_train_transform())
3 valid_dataset = Identification(VALID_DIR, RESIZE_T0, RESIZE_T0, CLASSES, get_valid_transform())
4 train_loader = DataLoader(
5     train_dataset,
6     batch_size=BATCH_SIZE,
7     shuffle=True,
8     num_workers=0,
9     collate_fn=collate_fn
10 )
11 valid_loader = DataLoader(
12     valid_dataset,
13     batch_size=BATCH_SIZE,
14     shuffle=False,
15     num_workers=0,
16     collate_fn=collate_fn
17 )
18 print(f"Number of training samples: {len(train_dataset)}")
19 print(f"Number of validation samples: {len(valid_dataset)}\n")

```

Gambar 3. 14 Load data train dan validation dari folder.

- Baris 2 dan 3 memanggil kelas identifikasi berserta parameter yang dibutuhkan.
- Baris ke 4 sampai dengan 17 proses mengubah format dataset kedalam format pytorch.

```

7 prog_bar = tqdm(train_data_loader, total=len(train_data_loader))
8
9 for i, data in enumerate(prog_bar):
10     optimizer.zero_grad()
11     images, targets = data
12
13     images = list(image.to(DEVICE) for image in images)
14     targets = [{k: v.to(DEVICE) for k, v in t.items()} for t in targets]
15     loss_dict = model(images, targets)
16     losses = sum(loss for loss in loss_dict.values())
17     loss_value = losses.item()
18     train_loss_list.append(loss_value)
19     train_loss_hist.send(loss_value)
20     losses.backward()
21     optimizer.step()
22     train_itr += 1
23
24     # update the loss value beside the progress bar for each iteration
25     prog_bar.set_description(desc=f"Loss: {loss_value:.4f}")
26 return train_loss_list

```

Gambar 3. 15 Potongan Kode untuk melakukan training data

- Baris ke 7 merupakan proses untuk training
- Baris ke 9 sampai dengan baris ke 25 adalah proses perulangan untuk melakukan proses training.

3.5 Model dan Testing

Untuk menghasilkan model pada penelitian ini menggunakan backbone resnet50_fpn seperti pada Gambar 3. 16, dimana dengan menggunakan pretrained models yang sudah disediakan oleh pytorch akan membantu proses training mencapai convergen walau dengan data yang tergolong sedikit. Setelah melakukan training pada dataset maka akan dihasilkan beberapa model sesuai dengan konfigurasi yang disimpan, pada tahapan ini lah sebenarnya yang menjadi inti dari tahapan yang dilalui dalam penelitian. Ketika model terbaik sudah dihasilkan, berikutnya yang perlu dilakukan adalah menguji models.

```

1 import torchvision
2 from torchvision.models.detection.faster_rcnn import FastRCNNPredictor
3 def create_model(num_classes):
4
5     # load Faster RCNN pre-trained model
6     model = torchvision.models.detection.fasterrcnn_resnet50_fpn(pretrained=True)
7
8     # get the number of input features
9     in_features = model.roi_heads.box_predictor.cls_score.in_features
10    # define a new head for the detector with required number of classes
11    model.roi_heads.box_predictor = FastRCNNPredictor(in_features, num_classes)
12    return model

```

Gambar 3. 16 Fungsi untuk melakukan pembuatan model dengan pretrained resnet50_fpn

Setelah model dihasilkan sesuai dengan konfigurasi yang dibuat, yaitu menggunakan resnet50_fpn sebagai backbone dari faster r-cnn di penelitian ini. Pada Gambar 3. 17 menjelaskan arsitektur dari masing-masing jumlah layer resnet, mulai dari nama layer, ukuran keluarannya dan juga jumlah layer yang dimiliki.

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$
conv3_x	28×28	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 8$
conv4_x	14×14	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 23$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 36$
conv5_x	7×7	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Gambar 3. 17 Layer resnet50_fpn

3.6 Evaluasi Model

Dalam evaluasi object detection standar yang biasa digunakan adalah mAP (mean average precision) baik itu untuk Yolo, SSD, dan juga Faster R-CNN. pada penelitian ini Faster R-CNN adalah algoritma yang digunakan untuk mendeteksi penyakit pada tanaman padi. sebelum mengetahui bagaimana mAP dihasilkan, perlu mengetahui tentang precision dan recall terlebih dahulu. *Precision* mengacu kepada ketepatan model dalam melakukan prediksi yang dibuat terhadap semua objek. Sedangkan *recall* mengacu kepada kemampuan model untuk mendeteksi semua *ground truth* yang ada. untuk lebih jelasnya perhatikan formula berikut.

$$Precision = \frac{TP}{TP+FP} \quad (3.1)$$

$$recall = \frac{TP}{TP+FN} \quad (3.2)$$

- TP = *True positif* mengacu kepada jumlah angka yang berhasil dideteksi secara benar oleh model ketika melakukan prediksi terhadap ground truth.
- FP = *False positif* mengacu kepada prediksi model yang salah bukan bagian dari ground truth
- FN = *False negative* model tidak memprediksi ground truth yang ada.
-

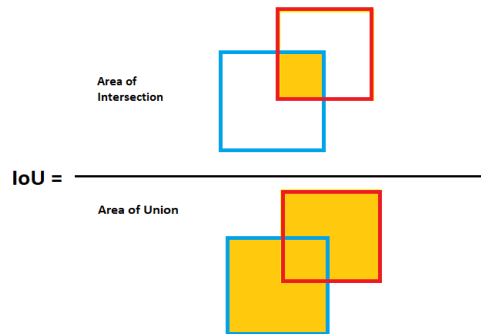
$$mAP = \frac{1}{N} \sum_{k=1}^N AP_k \quad (3.3)$$

- N = Jumlah dari kelas data
- AP_k = Average Precision dari kelas k

Pada Gambar 3. 18 merupakan ilustrasi pada gambar padi dengan penyakit sedangkan untuk melihat ilustrasi secara umum ada pada Gambar 3. 19, yang keduanya digunakan untuk melihat bagaimana ilustrasi yang terjadi pada bounding box yang dihasilkan. Pada kotak garis biru merupakan hasil anotasi yang dibuat secara manual, sedangkan pada kotak dengan garis merah merupakan hasil yang diprediksi oleh model. Pada dasarnya *Intersection Over Union* merupakan overlapping antara data yang ada, dengan data yang diprediksi oleh model. Sehingga dapat digambarkan dalam persamaan matematis seperti formula (3. 4)



Gambar 3. 18 Ground truth bounding box dan predicted bounding box dalam object detection



Gambar 3. 19 Ilustrasi Intersection Over Union

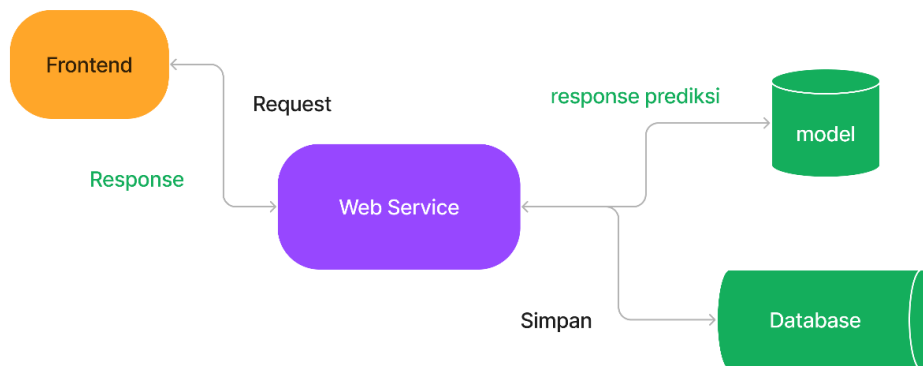
$$J(A, B) = \frac{|A \cap B|}{|A \cup B|} \quad (3.4)$$

- A = Predicted box
- B = Ground Truth box
- J = Intersection Over union

3.7 Perancangan Web Service

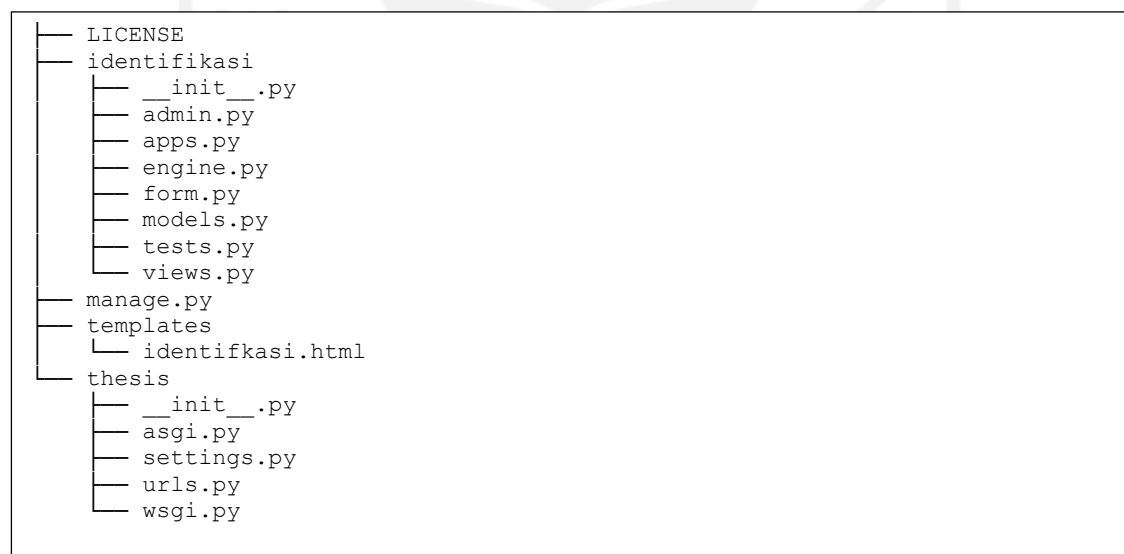
Untuk melakukan *deployment* model supaya bisa digunakan secara lebih mudah, maka diperlukan sebuah aplikasi yang bisa digunakan oleh pengguna yang mempunyai interface mengacu pada Gambar 3. 20, pada penelitian ini menggunakan web service dan front end dengan fitur untuk meng-*upload* gambar dan memberikan *response* berupa hasil prediksi

dari penyakit tanaman padi yang diidap. Berikut perancangan web service yang akan dibuat pada penelitian ini.



Gambar 3. 20 Perancangan Web service

Dalam pembuatan web service pada penelitian ini menggunakan *framework* Django dengan struktur direktori seperti pada Gambar 3. 21 ada 3 folder utama, yaitu identifikasi untuk business process dari penelitian ini, kemudian thesis untuk pengaturan, templates untuk interface (antarmuka).



Gambar 3. 21 Struktur folder untuk deployment model dengan *framework* Django.

Untuk membuat web service yang bisa diakses oleh aplikasi lain, baik itu secara publik maupun private, ada 4 file utama yang berfungsi untuk melakukan tugas itu. Yaitu file “identifikasi / urls.py” pada Gambar 3. 22 yang berfungsi untuk mengatur routing dari aplikasi ataupun *endpoint* seperti “{base_url}/engine/api/identifikasi/”. Sedangkan untuk *handle request* dan *response* pada *framework* django terdapat pada file “views.py”

seperti pada Gambar 3. 23. selanjutnya untuk melakukan formatting dan validation data pada saat request dan response menggunakan “serializer.py” seperti pada Gambar 3. 24 dan yang bertugas untuk menyimpan data ke database ada pada Gambar 3. 25.

```

1  from rest_framework.routers import DefaultRouter
2  from django.urls import path, include
3  router = DefaultRouter()
4  router.register(r'identifikasi')
5
6
7  urlpatterns = [
8  |   path('api/', include(router.urls)),
9
10 ]

```

Gambar 3. 22 File url yang mengatur routing dari web service

- Baris 1, 2 proses import library yang diperlukan.
- Baris ke 3 pembuatan *object router*
- Baris ke 4 alamat *routing* yang akan menjadi acuan untuk menghandle request dan response dari *client*.

```

1  from .serializer import IdentifikasiSerializer
2  from rest_framework import viewsets
3  from .models import Identifikasi
4  from rest_framework.permissions import IsAuthenticated
5
6  class IdentifikasiViewsets(viewsets.ModelViewSet):
7  |   serializer_class = IdentifikasiSerializer
8  |   queryset = Identifikasi.objects.all()
9  |   permission_classes = [IsAuthenticated]
10
11  def get_queryset(self):
12  |   return super().get_queryset()
13
14  def perform_create(self, serializer):
15  |   return super().perform_create(serializer)

```

Gambar 3. 23 File views.py yang bertanggung jawab menghandle request dan response

- Baris 1 sampai dengan baris 4 proses import library yang diperlukan
- Baris 6 sampai dengan 15 adalah bagian inti dari views yang bertugas menangani request yang dan memberikan data yang diminta. Dengan Inheritance ke parent yaitu viewset.ModelViewSet.

```

1  from dataclasses import fields
2  from rest_framework import serializers
3  from .models import *
4
5  class IdentifikasiSerializer(serializers.ModelSerializer):
6  |   class Meta:
7  |   |   model = Identifikasi
8  |   |   fields = "__all__"
9

```

Gambar 3. 24 File Serializer.py bertugas untuk validasi dan formating data

```

1 from django.db import models
2 from django.db.models.signals import post_save, pre_save
3 from django.dispatch import receiver
4 from numpy import isnat
5 from thesis.settings import BASE_DIR
6 from .engine import detection
7
8 class Identifikasi(models.Model):
9     gambar = models.ImageField(upload_to='tesis/identification')
10     created_at = models.DateTimeField(auto_now=True)
11
12

```

Gambar 3. 25 File Models.py bertanggung jawab sebagai konektor dengan database

- Baris 1 sampai dengan 6 proses mengimport library yang diperlukan
- Baris 8 sampai dengan 10 adalah inti dari model yang dibuat dimana nama kelas merepresentasikan nama table dan atributnya merepresentasikan kolom pada database.

Sementara itu untuk mengintegrasikan models yang sudah dihasilkan pada proses training di backend aplikasi dibuatkan sebuah fungsi untuk melakukan proses load model dan proses untuk deteksi gambar yang sudah di upload pada file engine.py perhatikan Gambar 3. 26.

```

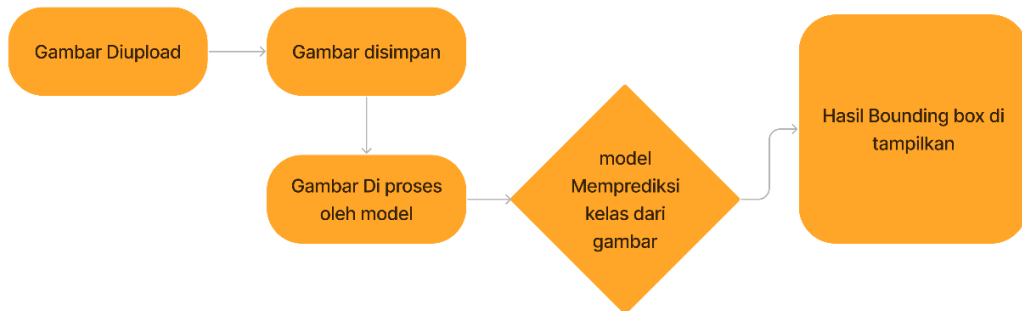
1
2 def detection(image):
3     image_name = image.split('/')[-1].split('.')[0]
4     image = cv2.imread(image)
5     orig_image = image.copy()
6     image = cv2.cvtColor(orig_image, cv2.COLOR_BGR2RGB).astype(np.float32)
7     image /= 255.0
8     image = np.transpose(image, (2, 0, 1)).astype(np.float)
9     image = torch.tensor(image, dtype=torch.float).cuda()
10    image = torch.unsqueeze(image, 0)
11    with torch.no_grad():
12        outputs = model(image)
13    outputs = [{k: v.to('cpu') for k, v in t.items()} for t in outputs]
14    if len(outputs[0]['boxes']) != 0:
15        boxes = outputs[0]['boxes'].data.numpy()
16        scores = outputs[0]['scores'].data.numpy()
17        boxes = boxes[scores >= detection_threshold].astype(np.int32)
18        draw_boxes = boxes.copy()
19        pred_classes = [CLASSES[i] for i in outputs[0]['labels'].cpu().numpy()]
20        for j, box in enumerate(draw_boxes):
21            cv2.rectangle(orig_image,
22                          (int(box[0]), int(box[1])),
23                          (int(box[2]), int(box[3])),
24                          (0, 0, 255), 2)
25            cv2.putText(orig_image, pred_classes[j],
26                       (int(box[0]), int(box[1]-5)),
27                       cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 0),
28                       2, lineType=cv2.LINE_AA)
29        cv2.imwrite(f"media/hasil/hasil_{image_name}.jpg", orig_image,)
30    return File(open(f"media/hasil/hasil_{image_name}.jpg", 'rb'))

```

Gambar 3. 26 Potongan kode

- Baris 2 pembuatan fungsi untuk mendeteksi gambar
- Baris 3 sampai dengan baris 10 proses load gambar dan konversi menjadi tensor
- Baris 11 sampai dengan 12 proses deteksi oleh model, sedangkan baris 13 sampai dengan 30, proses untuk menggambar bounding box jika threshold terpenuhi.

Pada Gambar 3. 27 merupakan alur proses deteksi penyakit oleh web service. Pada tahapan pertama gambar diupload oleh user. Setelah gambar berhasil di upload, gambar akan disimpan terlebih dahulu. Selanjutnya baru akan diproses oleh model untuk diprediksi bounding box, dan kelasnya.



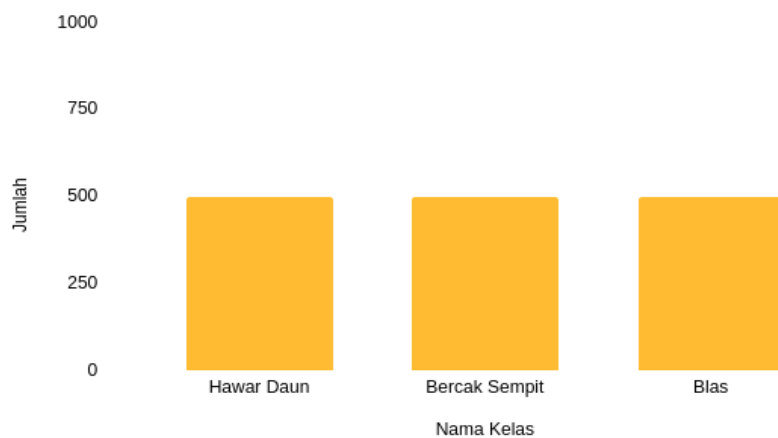
Gambar 3. 27 Alur Pendeteksian jenis penyakit oleh web service

BAB 4

Hasil dan Pembahasan

4.1 Data Yang Digunakan

Penelitian ini menggunakan dataset berupa foto yang diambil secara mandiri. Dataset yang sudah diambil terlebih dahulu diverifikasi oleh expert di bidang pertanian. Posisi dari expert saat ini adalah konsultan di salah satu startup pertanian yaitu di PT.MSMB GROUP. Expert bekerja dengan cara memisahkan foto antar penyakit yang dominan ke dalam folder masing-masing. Sehingga jika mengacu pada Gambar 4. 1 terlihat jumlah data yang dimiliki total 1500 data, dengan masing-masing kelas berjumlah 500.

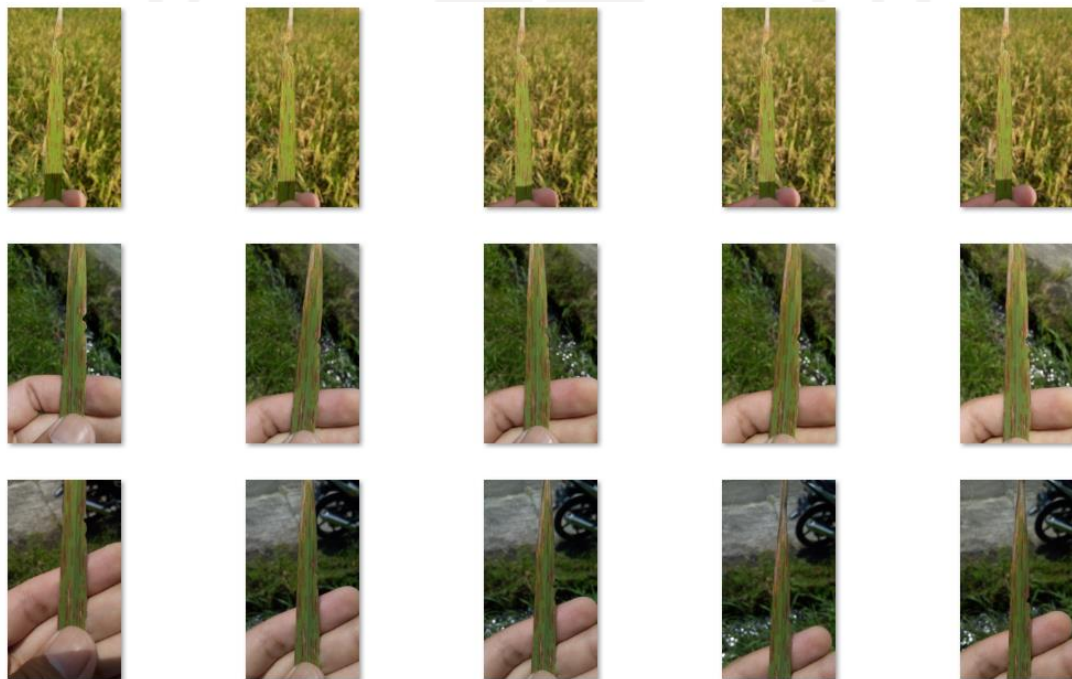


Gambar 4. 1 jumlah kelas masing-masing data sebelum dilakukan anotasi pada gambar

Data yang pada awalnya mempunyai ukuran dan kualitas yang berbeda-beda. Dengan harapan semakin bervariasi data, diharapkan model yang dihasilkan juga semakin *robust*. Pada Gambar 4. 2 merupakan contoh 15 data penyakit hawar daun dengan rasio yang berbeda-beda. Background yang terlihat pada data ada yang berupa lantai semen, ada rumput dan tumbuhan lainnya. Selanjutnya untuk penyakit bercak sempit pada Gambar 4. 3 dengan contoh 15 data. Dengan background berupa motor, parit dan tanaman padi itu sendiri. Penyakit bercak sempit mempunyai area cenderung kecil pada daun, bila dibanding dengan 2 penyakit lainnya (hawar daun, blas). Sehingga dalam satu gambar mempunyai banyak area anotasi. Data terakhir pada Gambar 4. 4 merupakan penyakit blas. Penyakit blas mempunyai warna yang mirip dengan warna hawar daun hanya saja areanya lebih kecil dari hawar daun dan lebih besar dibandingkan dengan bercak sempit.



Gambar 4. 2 Penyakit hawar daun

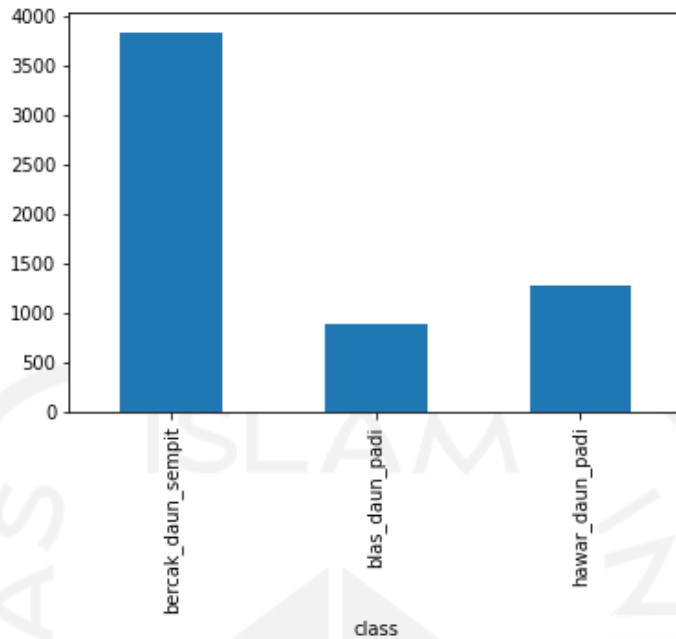


Gambar 4. 3 Penyakit Bercak Sempit



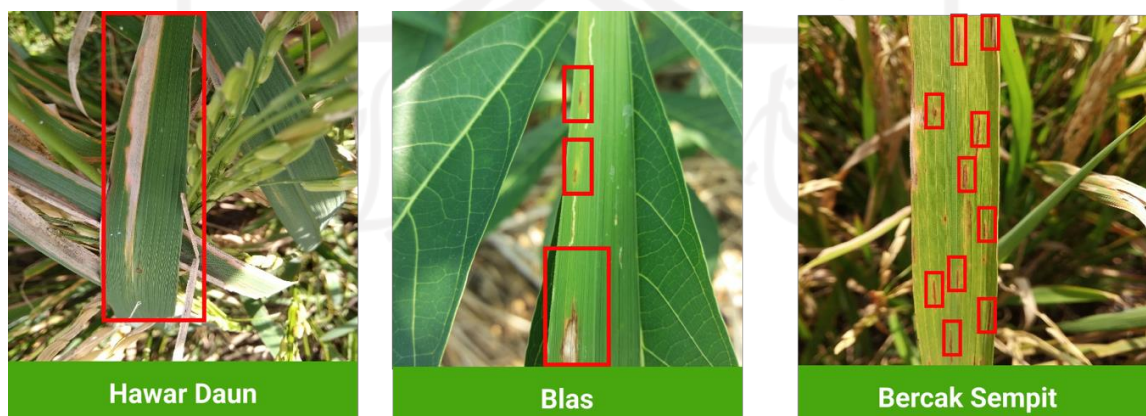
Gambar 4. 4 Penyakit Blas

Gambar yang ada pada penelitian memiliki jumlah data yang sama dimasing-masing kelasnya, namun karena pada percobaan kali ini menggunakan object detection, Perlu penambahan lokalisasi objek pada gambar. Oleh sebab itu perlu dilakukan proses anotasi disetiap gambar yang mengandung objek dari masing-masing kelas. Proses ini yang akan berpengaruh pada hasil akhir jumlah data disetiap kelasnya. Mengacu pada Gambar 4. 5 hasil anotasi dengan jumlah kelas paling banyak ada pada penyakit bercak sempit, sedangkan jumlah kelas yang paling sedikit yaitu penyakit blas. Seperti yang diketahui pada prosesnya anotasi memungkinkan objek lebih dari satu kelas pada satu gambar, baik itu kelasnya sama ataupun dengan kelas yang berbeda. Untuk lebih jelasnya perhatikan jumlah data masing-masing kelas yang dihasilkan setelah proses anotasi.



Gambar 4. 5 Sebaran kelas data setelah dilakukan proses anotasi

Mengacu pada Gambar 4. 6 penyakit bercak sempit pada tumbuhan padi memiliki ukuran yang relatif lebih kecil dan jumlah lebih banyak bila dibandingkan dengan dua penyakit lainnya pada penelitian ini. Penyakit lainnya seperti penyakit blas dan hawar daun mempunyai area yang lebih besar dan jumlah yang lebih sedikit. Sehingga setelah data dianotasi, penyakit bercak sempit mendapatkan jumlah anotasi yang paling signifikan dan relatif jauh lebih tinggi dibandingkan penyakit yang lainnya.



Gambar 4. 6 perbedaan anotasi dari masing-masing kelas (penyakit) yang ada pada tumbuhan padi

Sesuai dengan hipotesa awal dan pertanyaan yang di himpun dari expert **HTB**, kemungkinan padi terserang lebih dari satu penyakit juga terbukti ada pada dataset yang dimiliki. Dimana pada temuannya pada Gambar 4. 7 terdapat padi yang mengidap penyakit hawar daun dan penyakit bercak sempit secara bersamaan.



Gambar 4. 7 Satu daun lebih mengidap lebih dari satu penyakit

4.2 Hasil Training Dan Evaluasi

Dalam melakukan proses training pada penelitian ini menggunakan Nvidia Geforce 1050 Ti dengan 4GB memory GDDR5, sedangkan ram yang digunakan 16GB DDR4 dan system operasi menggunakan ubuntu 21 64 bit processor ryzen 7 1700 3 Ghz. Untuk konfigurasi hyperparameter dengan epoch mulai dari 50-105 di setiap percobaannya. Sebelum melakukan training model diberikan konfigurasi berupa **optimizer SGD, dan lr = 0.001, momentum=0.9, weigth_decay = 0.0005**. untuk melihat tingkat perbaikan dari proses training berdasarkan epoch bisa mengacu pada Gambar 4. 8, Gambar 4. 9.

```
EPOCH 1 of 105
Training
Loss: 0.6150: 100%|-----540/540 [08:58<00:00, 1.00it/s]
Validating
Loss: 0.6342: 100%|-----201/201 [01:21<00:00, 2.48it/s]
Epoch #1 train loss: 0.660
Epoch #1 validation loss: 0.664
Took 10.332 minutes for epoch 0
```

Gambar 4. 8 Epoch 1 dari 105 epoch

```

Epoch 105 of 105
Loss: 0.2314: 100%-----| 540/540 [00:59<00:00,
9.00it/s]
Validating
Loss: 0.764: 100%-----| 201/201 [00:11<00:00,
17.35it/s]
Epoch #0 train loss: 0.32
Epoch #0 validation loss: 0.22
Epoch 10.774 minutes for epoch 105

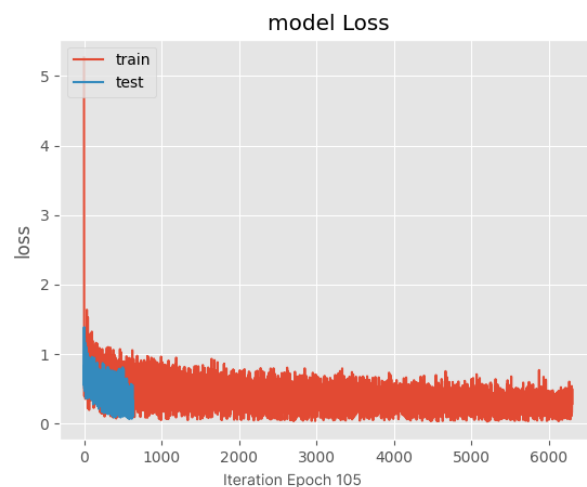
```

Gambar 4. 9 Epoch 105 dari 105 epoch

Selanjutnya melihat sebaik apa model belajar pada proses training pada penelitian ini dilakukan dokumentasi dari *localization loss*. Pada Tabel 4. 1 parameter yang direkam ada batch, epoch, trainloss, dan validation loss. Untuk melihat secepat apa model untuk konvergen proses iterasi juga direkam seperti pada Gambar 4. 10. Dengan demikian dapat dilihat perbaikan model dari setiap epoch dan iterasi yang ada dengan menggunakan *pretrained model* resnet50_fpn sebagai backbone yang digunakan untuk membangun model.

Tabel 4. 1 Peningkatan Hasil Training Resnet50_FPN pada setiap check point

BATCH	EPOCH	TRAIN LOSS	VALIDATION LOSS
2	10	0.4	0.35
2	30	0.37	0.3
2	60	0.35	0.25
2	105	0.32	0.22



Gambar 4. 10 perbandingan train loss dengan validation loss Resnet50_Fpn

Untuk mengetahui seberapa akurasi model dalam hal ini RPN dalam menentukan lokasi mana yang berisi objek yang kemudian akan diteruskan ke jaringan deteksi kelas objek untuk mengembalikan bounding box. Kesalahan dalam melakukan prediksi disebut dengan *Loss of bounding box regressor*.

Evaluasi yang digunakan pada penelitian ini adalah mean average precision (mAP), yang mengacu kepada mean, precision dan recall. Dimana precision dan recall merupakan parameter yang penting untuk image processing dan object detection. Pada Gambar 4. 11 terlihat hasil akhir dari pengujian yang pada ukuran yang digunakan (AP @[IoU=0.50:0.95]) yang merupakan standar yang digunakan oleh coco evaluator, mendapatkan hasil 87.6%. Hasil didapatkan cukup baik mengingat dataset yang dimiliki tidak terlalu besar.

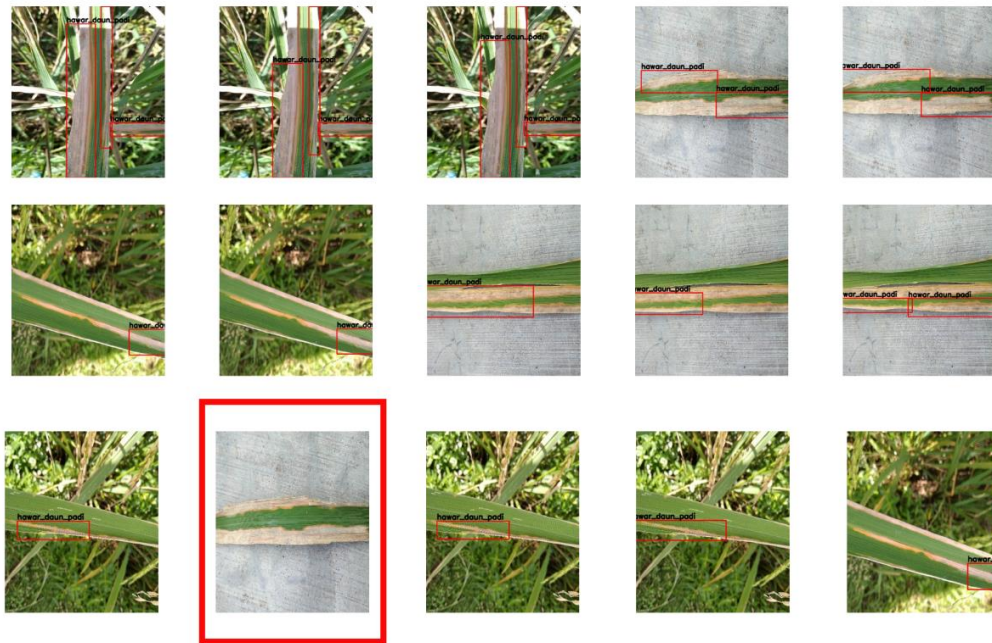
IoU metric: bbox				
Average Precision	(AP)	@[IoU=0.50:0.95	area= all maxDets=100]	= 0.876
Average Precision	(AP)	@[IoU=0.50	area= all maxDets=100]	= 0.996
Average Precision	(AP)	@[IoU=0.75	area= all maxDets=100]	= 0.996
Average Precision	(AP)	@[IoU=0.50:0.95	area= small maxDets=100]	= 0.922
Average Precision	(AP)	@[IoU=0.50:0.95	area=medium maxDets=100]	= 0.878
Average Precision	(AP)	@[IoU=0.50:0.95	area= large maxDets=100]	= 0.854
Average Recall	(AR)	@[IoU=0.50:0.95	area= all maxDets= 1]	= 0.208
Average Recall	(AR)	@[IoU=0.50:0.95	area= all maxDets= 10]	= 0.857
Average Recall	(AR)	@[IoU=0.50:0.95	area= all maxDets=100]	= 0.896
Average Recall	(AR)	@[IoU=0.50:0.95	area= small maxDets=100]	= 0.938
Average Recall	(AR)	@[IoU=0.50:0.95	area=medium maxDets=100]	= 0.887
Average Recall	(AR)	@[IoU=0.50:0.95	area= large maxDets=100]	= 0.863

Gambar 4. 11 Hasil Evaluation menggunakan mAP

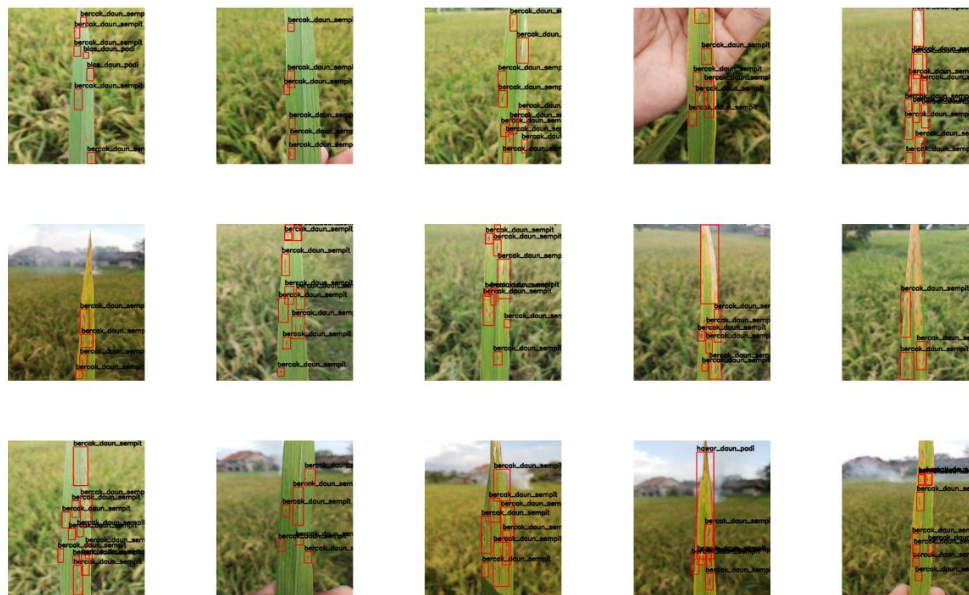
selanjutnya untuk menguji model yang dihasilkan pada penelitian ini, proses *inference* dilakukan dengan 2 cara yaitu dengan menggunakan *command* pada file *inference.py* dan *webservice* pada sub bab 4.3. Untuk skenario menggunakan *command* setiap kelas gambar yang merupakan data untuk inference dipisah menjadi 3 folder berdasarkan jenis penyakitnya, Perhatikan Gambar 4. 12 yang merupakan contoh hasil pengujian pada penyakit hawar daun. Dengan pengujian secara langsung terhadap gambar hawar daun ternyata ada 1 gambar yang tidak berhasil di deteksi oleh model dengan ditunjukkan pada box berwarna merah.

Pada Gambar 4. 13 merupakan contoh gambar penyakit bercak sempit, dari hasil pengujian dengan sample 15 gambar hamper semua bisa terdeteksi dengan baik walaupun penyakit bercak sempit tergolong kecil secara area penyakitnya. Hal ini juga sejalan

dengan banyaknya jumlah data training yang ada dalam penyakit bercak sempit setelah anotasi. Walaupun mungkin dengan scenario yang berbeda dengan variasi gambar yang lebih banyak mungkin juga akan mendapatkan hasil yang berbeda. Terakhir pada Gambar 4. 14 yang merupakan penyakit blas juga ditemukan 2 gambar dari 15 gambar yang tidak berhasil diidentifikasi oleh model. Hal yang lain yang ditemukan pada saat pengujian seperti pada Gambar 4. 15 adanya penyakit yang lebih dari satu dalam tumbuhan padi. Sehingga model yang bisa dikatan berhasil dan terbukti sesuai dengan hipotesis awal.



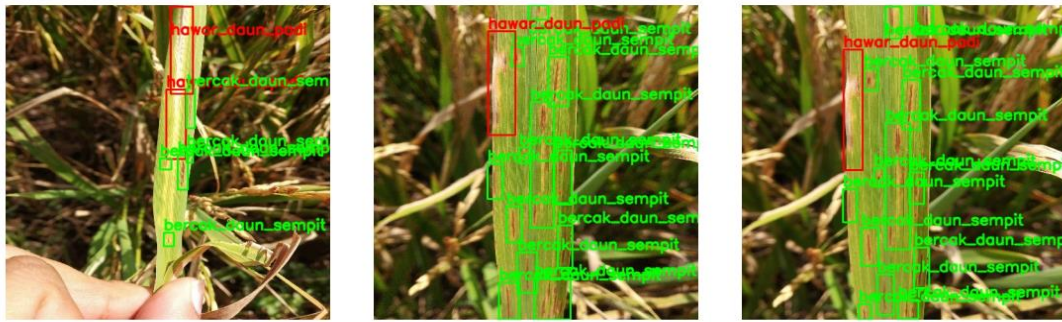
Gambar 4. 12 Proses Inference pada kelas / penyakit hewar daun, box yang dilingkari merah merupakan gambar yang gagal teridentifikasi dari model



Gambar 4. 13 Proses inference pada kelas / penyakit bercak sempit



Gambar 4. 14 Proses inference pada kelas / penyakit blas box merah kegagalan deteksi



Gambar 4. 15 Inference pada multi kelas/penyakit pada satu tanaman, border dengan warna merah merupakan Hawar daun dan warna hijau bercak sempit
Kemudian skenario selanjutnya yang dicoba pada penelitian ini yaitu menggunakan dataset yang diambil dari ics.uci.edu dengan resolusi dan rasio 3081x2897 pada Gambar 4. 16.



Gambar 4. 16 Inference dengan dataset dari ics.uci.edu penyakit hawar daun¹⁷
Selanjutnya pengujian juga dilakukan terhadap gambar dengan resolusi yang kecil, yaitu 256X256 yang diambil dari dataset publik kaggle¹⁸. untuk menguji apakah model mampu bekerja dengan baik.perhatikan



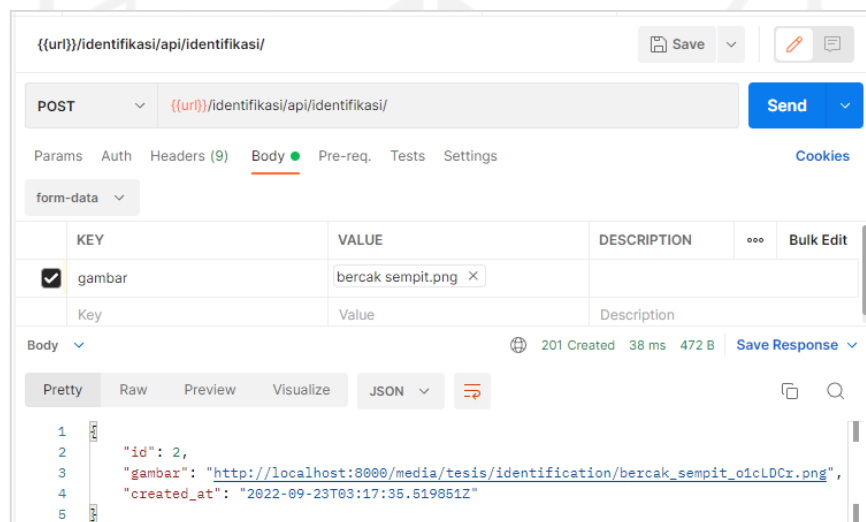
Gambar 4. 17 untuk foto yang sebelah kiri ada kesalahan deteksi yang dianggap penyakit, padahal seharusnya background.

¹⁷ <https://archive.ics.uci.edu/ml/machine-learning-databases/00486/>

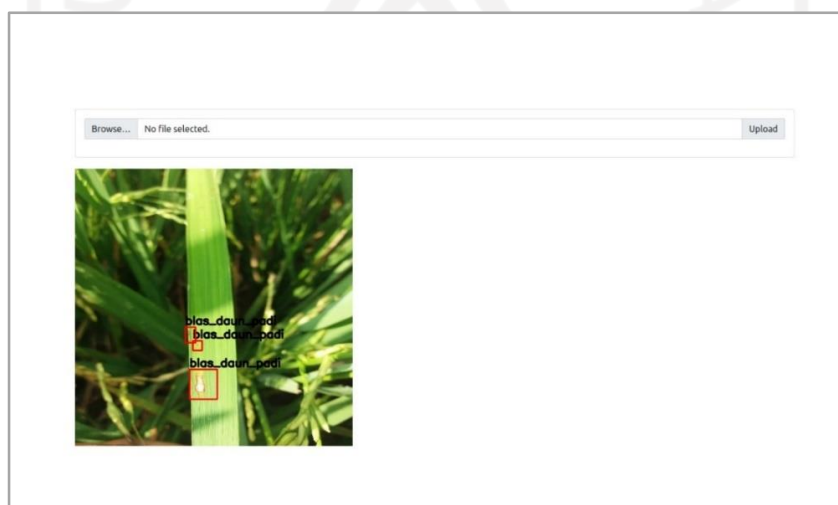
¹⁸ <https://www.kaggle.com/datasets/vbookshelf/rice-leaf-diseases>

4.3 Pengembangan Web service

Untuk mengimplementasikan hasil dari training model yang dibuat, pada penelitian ini penulis mengembangkan web service. Mengacu pada Gambar 4. 18 web service yang dibuat pada penelitian ini memiliki fitur untuk mengupload gambar, kemudian akan memberikan response berupa json. Aplikasi yang digunakan untuk melakukan percobaan menggunakan postman yang sudah lazim digunakan untuk ujicoba dari webservice. Selanjutnya untuk mencoba mengimplementasikan webservice yang dibuat dengan interface yang sederhana mengacu pada *Gambar 4. 19* yang memiliki fitur upload gambar dan mendisplay hasil dalam bentuk gambar yang sudah membunyai bounding box sesuai dengan hasil prediksi model yang ada.



Gambar 4. 18 Proses mengupload gambar untuk deteksi penyakit dengan postman



Gambar 4. 19 Aplikasi Untuk melakukan prediksi dengan model yang telah dihasilkan pada penyakit blas daun.

Dari Gambar 4. 19 dapat terlihat model mampu memberikan prediksi terhadap penyakit yang ada pada tumbuhan padi dengan tipe penyakit hawar daun. Tidak hanya itu model juga mampu memberikan lokasi perkiraan pada titik mana yang mengalami penyakit sesuai dengan hasil deteksinya. Selanjutnya percobaan dilakukan pada Gambar 4. 20 yaitu multi penyakit dalam satu tanaman padi, dengan hasil model juga mampu mendeteksinya. Dengan demikian terbukti bahwa tanaman padi bisa mengidap lebih dari satu penyakit pada saat bersamaan dan model yang dibuat pada penelitian ini mampu untuk mendeteksi itu.



Gambar 4. 20 Hasil prediksi lebih dari satu penyakit dalam satu daun padi

Berdasarkan implementasi menggunakan web service, berikut asas kemanfaatan dan improvement yang diharapkan oleh expert yang terlibat dalam penelitian ini. Sebelumnya para expert secara manual menjawab pertanyaan yang diajukan oleh petani lewat photo. Dengan adanya implementasi web service mengurangi beban expert untuk mengidentifikasi penyakit pada padi secara manual. Terutama penyakit yang ada dalam penelitian ini. Disamping itu expert dibidang HTB juga terbantu untuk mengidentifikasi padi yang terserang lebih dari satu penyakit secara bersamaan. Namun saran dari expert untuk bisa menambah jenis penyakit dan jenis tanaman yang mampu di deteksi oleh model.

BAB 5

Kesimpulan dan Saran

5.1 Kesimpulan

Proses identifikasi penyakit tanaman padi lebih awal sangat penting untuk dilakukan, karena dengan mengetahui jenis penyakit petani bisa memberikan pengendalian lebih awal yang bisa meningkatkan peluang untuk keberhasilan panen. Terbukti dengan data yang di *release* oleh kementerian pertanian jika penyakit tanaman tidak ditangani lebih awal bisa menyebabkan potensi gagal panen mencapai 18-80%.

Penelitian sebelumnya untuk mengidentifikasi dan mengenali penyakit pada tumbuhan padi pernah dilakukan dengan menggunakan faster r-cnn untuk mendeteksi penyakit false smut pada buah padi. Hasil yang didapatkan pada penelitian sebelumnya sudah berhasil mengidentifikasi penyakit false smut, namun harus disempurnakan lagi mengingat pada fase tanam padi proses pembuahan terjadi pada fase generative. Sementara itu untuk fase yang lebih awal yang dikenal dengan fase vegetatif masih banyak penyakit padi yang bisa menyebabkan gagal panen. Oleh sebab itu pada penelitian ini mencoba memperluas jenis penyakit yang bisa dideteksi, yang hampir bisa menyerang di semua fase. Pada penelitian ini akurasi menggunakan Faster R-CNN didapat mAP 87.6%, dan juga model mampu mendeteksi lebih dari 1 penyakit dalam satu gambar. Sehingga diharapkan pada saat deteksi penyakit bisa lebih akurat tidak selalu dalam satu gambar daun padi hanya memiliki satu penyakit tetapi ada kemungkinan lebih.

5.2 Saran

Penelitian ini fokus pada percobaan identifikasi penyakit pada tumbuhan padi, dengan 3 kelas penyakit yang paling banyak di temui di jogja dan jawa tengah. Untuk penelitian berikutnya diharapkan bisa menambah jenis penyakit pada tumbuhan padi dan tumbuhan-tumbuhan lain dengan dataset yang bervariasi, sehingga model yang dihasilkan akan lebih robust, dan lebih banyak penyakit tanaman yang bisa di deteksi.

Daftar Pustaka

- Agarwal, S., Farid, H., El-Gaaly, T., & Lim, S.-N. (2020). Detecting deep-fake videos from appearance and behavior. *2020 IEEE International Workshop on Information Forensics and Security (WIFS)*, 1–6.
- Ajitha, P. V., & Nagra, A. (2021). An Overview of Artificial Intelligence in Automobile Industry—A Case Study on Tesla Cars. *Solid State Technology*, 64(2), 503–512.
- Al-Amin, M., Karim, D. Z., & Bushra, T. A. (2019). Prediction of rice disease from leaves using deep convolution neural network towards a digital agricultural system. *2019 22nd International Conference on Computer and Information Technology (ICCIT)*, 1–5.
- Al-Masri, E., & Mahmoud, Q. H. (2007). Discovering the best web service. *Proceedings of the 16th International Conference on World Wide Web*, 1257–1258.
- Arandjelovic, R., Gronat, P., Torii, A., Pajdla, T., & Sivic, J. (2016). NetVLAD: CNN architecture for weakly supervised place recognition. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 5297–5307.
- Atmaja, R. A. W., Santosa, S. J., & Siswadi, S. (2020). KAJIAN PUPUK KANDANG TERHADAP INTENSITAS PENYAKIT BERCAK COKLAT SEMPIT (*Cercospora oryzae*) PADA TANAMAN PADI KETAN PUTIH (*Oryzae sativa* L. Glutinosa). *Innofarm: Jurnal Inovasi Pertanian*, 22(1), 15–22.
- Brownlee, J. (2019). *Deep learning for computer vision: Image classification, object detection, and face recognition in python*. Machine Learning Mastery.
- Esteva, A., Robicquet, A., Ramsundar, B., Kuleshov, V., DePristo, M., Chou, K., Cui, C., Corrado, G., Thrun, S., & Dean, J. (2019). A guide to deep learning in healthcare. *Nature Medicine*, 25(1), 24–29.

- Ferentinos, K. P. (2018). Deep learning models for plant disease detection and diagnosis. *Computers and Electronics in Agriculture*, 145, 311–318.
- Fuentes, Alvaro. (2017). *A Robust Deep-Learning-Based Detector for Real-Time Tomato Plant Diseases and Pests Recognition*.
- Goncharov, P., Ososkov, G., Nechaevskiy, A., Uzhinskiy, A., & Nestsiaenia, I. (2018). Disease detection on the plant leaves by deep learning. *International Conference on Neuroinformatics*, 151–159.
- Hayat, M. K., Daud, A., Alshdadi, A. A., Banjar, A., Abbasi, R. A., Bao, Y., & Dawood, H. (2019). Towards deep learning prospects: Insights for social media analytics. *IEEE Access*, 7, 36958–36979.
- He, K., Zhang, X., Ren, S., & Sun, J. (2015). *Deep Residual Learning for Image Recognition* (arXiv:1512.03385). arXiv. <https://doi.org/10.48550/arXiv.1512.03385>
- Jez, J. M., Topp, C. N., Silva, G., Tomlinson, J., Onkokesung, N., Sommer, S., Mrisho, L., Legg, J., Adams, I. P., & Gutierrez-Vazquez, Y. (2021). Plant pest surveillance: From satellites to molecules. *Emerging Topics in Life Sciences*, 5(2), 275–287.
- Jordan, M. I., & Mitchell, T. M. (2015). Machine learning: Trends, perspectives, and prospects. *Science*, 349(6245), 255–260.
- LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, 521(7553), 436–444.
- Liang, W., Zhang, H., Zhang, G., & Cao, H. (2019). Rice blast disease recognition using a deep convolutional neural network. *Scientific Reports*, 9(1), 1–10.
- Masnilah, R., Astono, T. H., & Aini, L. Q. (2013). Karakterisasi bakteri penyebab penyakit hawar daun edamame di Jember. *Berkala Ilmiah Pertanian*, 1(1), 10–14.
- Nisa, K. (2021). *Isolasi Xanthomonas oryzae pv. Oryzae dan Pengujian Ketahanan Varietas Padi terhadap Hawar Daun Bakteri di BB Padi Subang*.

- Oberweger, M., Wohlhart, P., & Lepetit, V. (2015). Hands deep in deep learning for hand pose estimation. *ArXiv Preprint ArXiv:1502.06807*.
- O'Shea, K., & Nash, R. (2015). An introduction to convolutional neural networks. *ArXiv Preprint ArXiv:1511.08458*.
- Pathak, A. R., Pandey, M., & Rautaray, S. (2018). Application of deep learning for object detection. *Procedia Computer Science*, 132, 1706–1717.
- Prasanna Mohanty, S., Hughes, D., & Salathe, M. (2016). Using Deep Learning for Image-Based Plant Disease Detection. *ArXiv E-Prints*, arXiv-1604.
- PRASETYO, M. S. H. (2017). *Kajian Intensitas Penyakit Bercak Coklat Sempit (Cercospora oryzae) dan Teknik Pengendaliannya pada Pertanaman Padi di Kecamatan Tanggul Kabupaten Jember*.
- Purwins, H., Li, B., Virtanen, T., Schlüter, J., Chang, S.-Y., & Sainath, T. (2019). Deep learning for audio signal processing. *IEEE Journal of Selected Topics in Signal Processing*, 13(2), 206–219.
- Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in Neural Information Processing Systems*, 28.
- Sethy, P. K., Barpanda, N. K., Rath, A. K., & Behera, S. K. (2020). Rice false smut detection based on faster R-CNN. *Indonesian Journal of Electrical Engineering and Computer Science*, 19(3), 1590–1595.
- Sudir, S., Nasution, A., Santoso, S., & Nuryanto, B. (2015). *Penyakit blas Pyricularia grisea pada tanaman padi dan strategi pengendaliannya*.
- Sudir, S., Nuryanto, B., & Kadir, T. S. (2012). *Epidemiologi, patotipe, dan strategi pengendalian penyakit hawar daun bakteri pada tanaman padi*.

- Suganda, T., Yulia, E., Widiyanti, F., & Hersanti, H. (2016). Intensitas penyakit blas (*Pyricularia oryzae* Cav.) pada padi varietas Ciherang di lokasi endemik dan pengaruhnya terhadap kehilangan hasil. *Agrikultura*, 27(3).
- Tan, L., Huangfu, T., Wu, L., & Chen, W. (2021). *Comparison of YOLO v3, Faster R-CNN, and SSD for Real-Time Pill Identification*.
- Vargas, R., Mosavi, A., & Ruiz, R. (2017). *Deep learning: A review*.
- Wang, G., Sun, Y., & Wang, J. (2017). Automatic image-based plant disease severity estimation using deep learning. *Computational Intelligence and Neuroscience*, 2017.
- Yu, J., Mei, X., Porikli, F., & Corso, J. (2018). *Machine learning for big visual analysis*. Springer.
- Yu, N., Skripniuk, V., Abdelnabi, S., & Fritz, M. (2020). *Artificial GAN fingerprints: Rooting deepfake attribution in training data*.
- Zhang, N., Donahue, J., Girshick, R., & Darrell, T. (2014). Part-based R-CNNs for fine-grained category detection. *European Conference on Computer Vision*, 834–849.
- Zhong, Z., Li, J., Luo, Z., & Chapman, M. (2017). Spectral–spatial residual network for hyperspectral image classification: A 3-D deep learning framework. *IEEE Transactions on Geoscience and Remote Sensing*, 56(2), 847–858.
- Zhou, G., Zhang, W., Chen, A., He, M., & Ma, X. (2019). Rapid detection of rice disease based on FCM-KM and faster R-CNN fusion. *IEEE Access*, 7, 143190–143206.