SUMMARY HASIL DAN PEMBAHASAN

1.1. Hasil

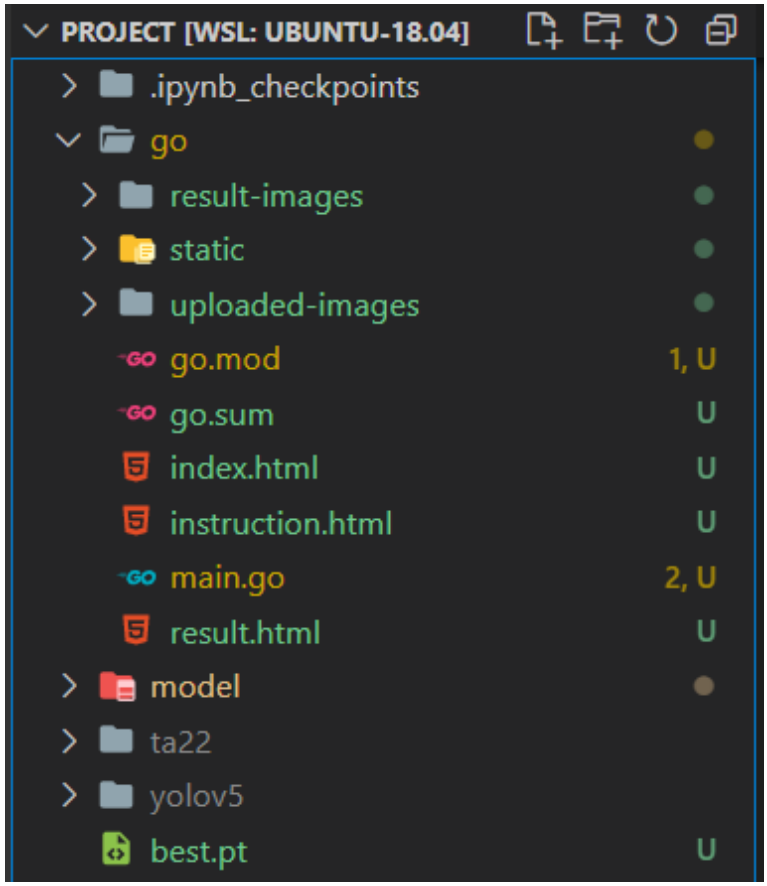
Pada bagian ini merupakan tahap pembahasan dari hasil implementasi yang penulis kumpulkan, analisa dan rancangan sehingga tercipta sistem pendeteksi dan klasifikasi tingkat kualitas tanaman padi menggunakan algoritma YOLOv5.

1.2. Pembahasan

1.2.1. Struktur Direktori

1.2.1.1. Struktur Direktori Sistem Deteksi

Struktur direktori menjelaskan urutan atau *file-file* apa saja yang terdapat pada tiap-tiap *project* dalam membangun sistem pendeteksi dan klasifikasi tingkat kualitas tanaman padi.



Gambar 4.1. Struktur direktori *root Project* Sistem Deteksi

Pada Gambar 4.1 terdapat struktur direktori *root* dari sistem yang penulis rancang. Pada *folder root* ini terdapat *folder* go yang berisi sebuah sistem *back-end* yang dibangun menggunakan bahasa pemrograman go, pada folder go terdapat *folder result-images* yang berisi semua gambar yang telah berhasil dideteksi oleh Algoritma YOLOv5s, selanjutnya ada folder *static* yang berisi *file* untuk *styling* tampilan *dashboard* seperti *css* dan *js,* lalu yang terakhir adalah *folder uploaded-images* yang berisi semua gambar yang telah diunggah oleh para petani yang nantinya akan dideteksi oleh Algoritma YOLOv5s. Selanjutnya pada *folder root* ini terdapat *folder model,* yang berisi *datasets* dan semua hasil *experiment* Algoritma *Deep Learning* seperti *Notebook*, *Classification Report*, *Confusion Matrix* dan *Model Weights*. *Folder ta22* merupakan *virtual environment* yang dibuat untuk keperluan *experiment* Algoritma *Deep Learning*. Selanjutnya *folder* yolov5 yang berisi file-file hasil *cloning* dari *repository* asli, *folder* ini berfungsi untuk pelatihan model menggunakan Algoritma YOLOv5. Selanjutnya adalah *file best.pt, file* tersebut adalah model terbaik yang telah dipilih berdasarkan *score precision, recall, f1-score*, *mAP:0.5,* waktu deteksi dan ukuran *file*.

1.2.2. Implementasi Persiapan Data

Pendeteksian dan klasifikasi tingkat kualitas tanaman padi dilakukan dengan cara menentukan terlebih dahulu kelas atau kategori tingkat kualitas yang diperoleh selama pencarian data. Tingkat Kualitas yang ditemukan selama pencarian hanya 2 tingkat kualitas. Tingkat Kualitas tersebut penulis paparkan pada Tabel 4.1 dibawah ini.

Tabel 4.1. Tabel Tingkat Kualitas

No	**Tingkat Kualitas**	**Label**
1	Baik / Sehat	good
2	Buruk / Tidak Sehat	bad

1.2.2.1. Pengumpulan *Dataset*

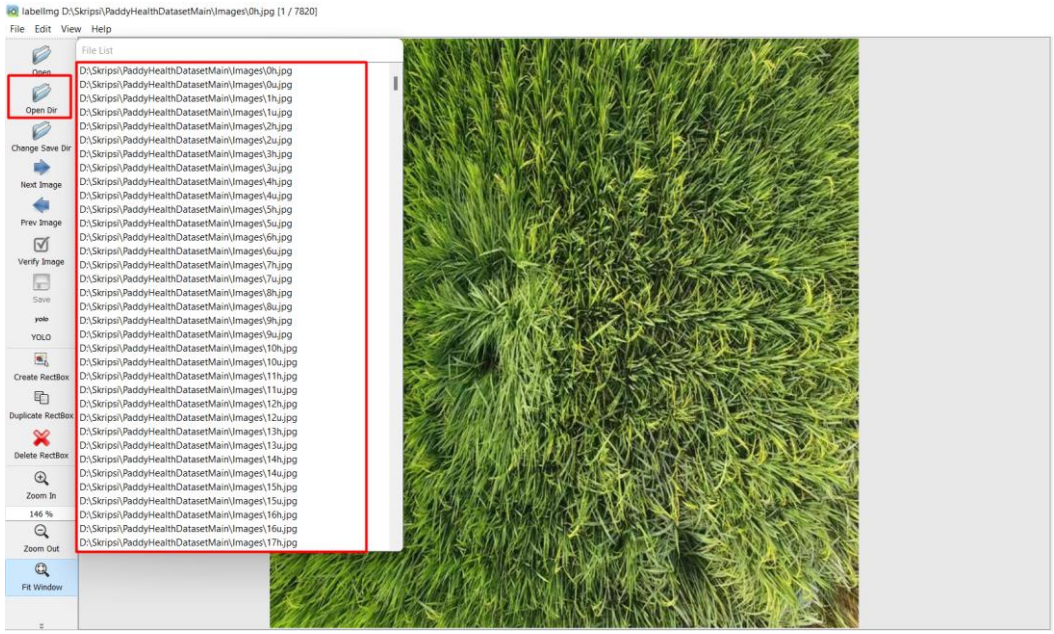
Data tanaman padi yang penulis gunakan sebagai *dataset* didapatkan dari *platform* penyedia datasets ternama yaitu Kaggle. Data tersebut berupa gambar dari tanaman padi yang diambil melalui citra udara atau *drone*.

1.2.2.2. Persiapan Data

Setelah data terkumpul, tahap selanjutnya adalah tahap persiapan data. Persiapan data yang harus dilakukan sebelum masuk ke tahap proses *training* model diantaranya yaitu *Labeling, Splitting* dan *Preprocessing.*

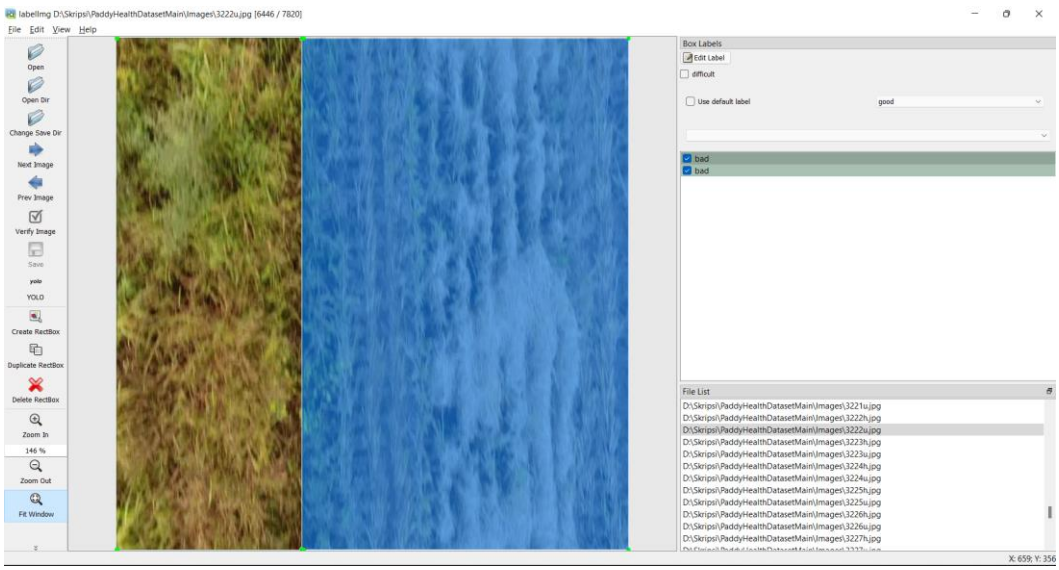
a. *Labeling*

Labeling adalah sebuah proses pemberian anotasi label terhadap objek yang menjadi target pada gambar berdasarkan *class* nya. Pada sistem yang penulis buat, pendeteksian gejala dapat dilakukan lebih dari satu objek dalam satu gambar (*Multilabel*). Proses *labeling* dilakukan menggunakan *tools*/aplikasi LabelImg. LabelImg adalah *tools* yang gratis dan *open source* berbasis *desktop,* tidk hanya membuat anotasi untuk format YOLO, *labelimg* juga bisa membuat anotasi untuk format lainya seperti *Pascal / VOC* yang akan menghasilkan anotasi dalam bentuk file XML dan format CreateML yang akan menghasilkan anotasi dalam bentuk file JSON. Sebelum melakukan proses anotasi, terlebih membuka *folder* yang berisi *datasets* yang telah dimasukan ke dalam 1 *folder*. Untuk lebih jelasnya dapat dilihat pada gambar dibawah ini.



Gambar 4.2. *Directory File List* LabelImg

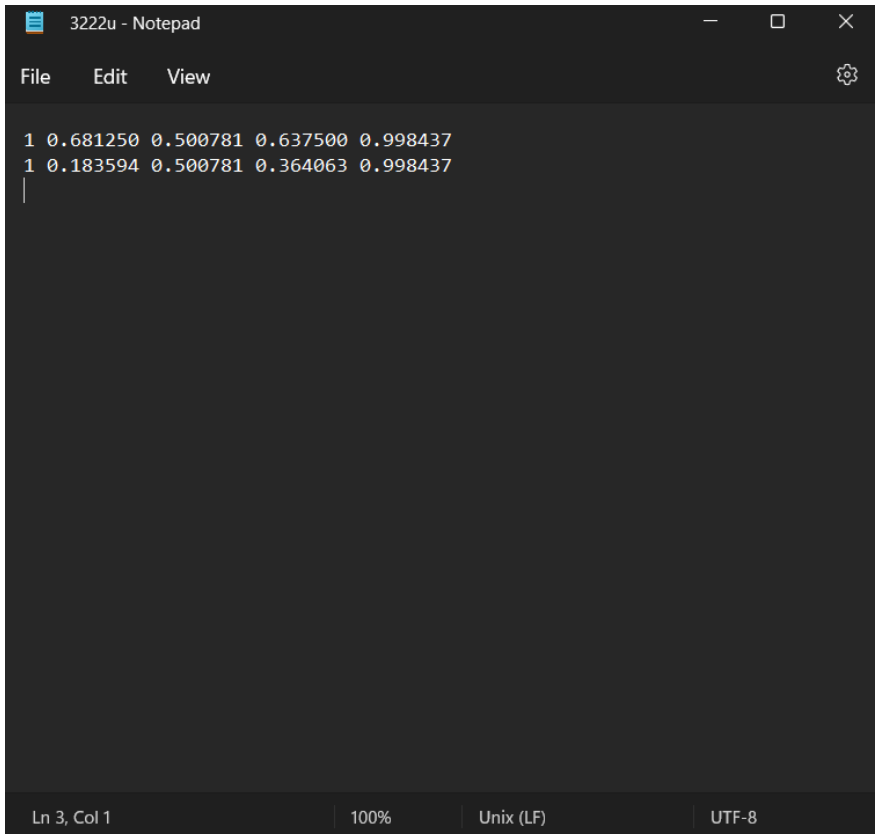
Setelah *task* dibuat, proses selanjutnya adalah pembuatan anotasi pada data yang sudah penulis masukan ke dalam *Directory File List*. Untuk lebih jelasnya dapat dilihat pada gambar dibawah ini.



Gambar 4.3. Proses *Labelling* Menggunakan LabelImg

Pada gambar diatas bisa dilihat bahwa terdapat 1 tingkat kualitas yaitu buruk / tidak sehat (bad). Maka pada objek tersebut penulis tandai menggunakan *rectangle* atau *bounding box*, lalu pilih sesuai *class* nya. Setelah membuat *bounding box* beserta pilihan *class* nya LabelImg akan otomatis membuat hasil anotasi nya

dengan format YOLO. Hasil anotasi tersebut akan menghasilkan sebuah file dengan ekstensi txt sesuai dengan nama *file* gambar yang dibuat anotasinya, untuk lebih jelasnya dapat dilihat pada gambar dibawah ini.



Gambar 4.4. *File* Hasil Anotasi

b. *Splitting*

Setelah data selesai melalui proses *labelling*, tahap selanjutnya sebelum masuk ke proses *training* dataset tersebut harus dibagi menjadi *training set* dan *validation set*. Pembagian data ini bertujuan agar model yang dibuat tidak *overfitting*. *Overfitting* terjadi saat model mendapatkan akurasi atau performa yang baik pada data *training*, namun mendapatkan akurasi atau performa yang buruk ketika menggunakan data yang belum pernah dilihat sebelumnya atau data *testing*.

Untuk pembagian dataset dalam pembuatan sistem deteksi dan klasifikasi tingkat kualitas tanaman padi ini penulis membagi dataset menjadi dua bagian. Pembagian terbanyak terdapat pada data *training* karena proses *training* sangat

penting untuk mendapatkan performa terbaik. Untuk pembagian *dataset* akan penulis jelaskan pada tabel dibawah ini.

Tabel 4.2. Tabel Data *Splitting*

Data Training	90%	7038
Data Validation	10%	782
Total Data	100%	7820

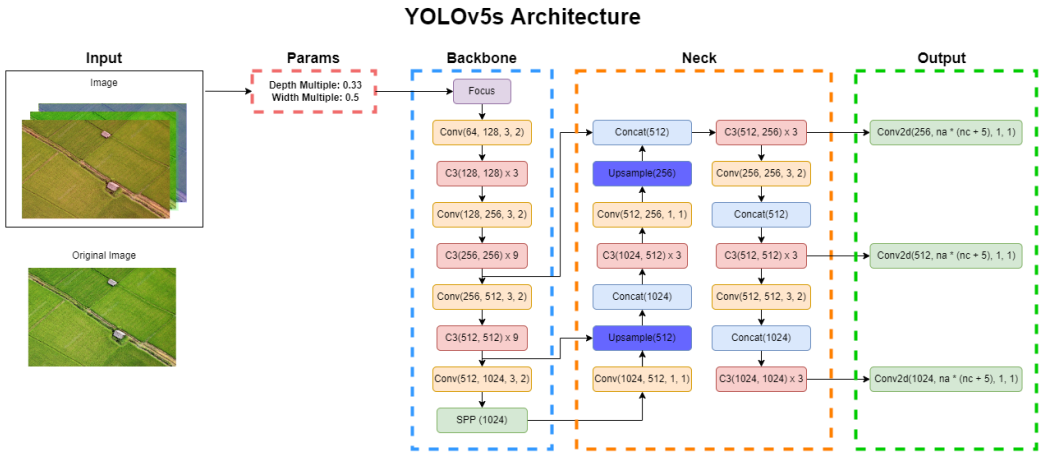
c. *Preprocessing*

Pada tahap *preprocessing* ini hanya mengubah ukuran gambar yang sebelumnya berukuran 640x640 *pixel* menjadi 512x512 *pixel*. Nilai tersebut adalah ukuran yang digunakan penulis untuk konfigurasi Model YOLOv5 dan SSDMobileNetV2. Hal ini bertujuan agar mempercepat proses *training*.

1.2.3. Training Model

1.2.3.1. Hasil Training YOLOv5s

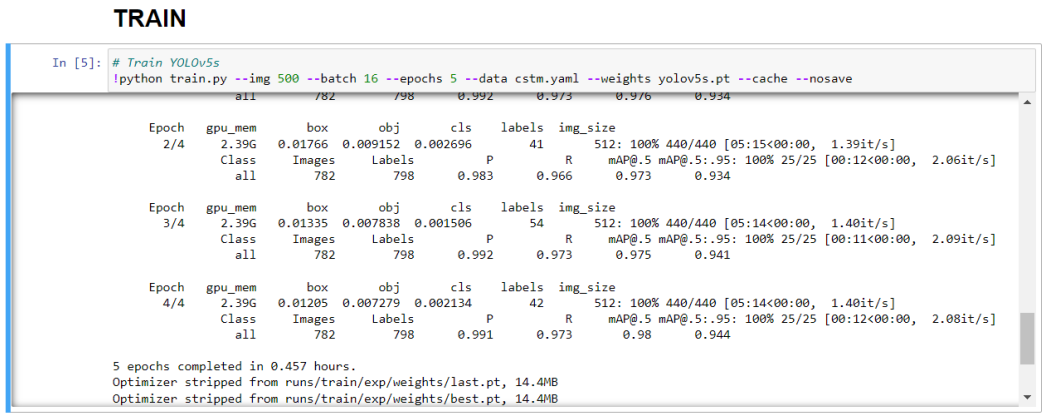
Proses *training*, merupakan sebuah proses untuk melatih model menggunakan *neural network* / jaringan saraf tiruan. Sebelum melatih model kita perlu membuat arsitektur model terlebih dahulu, karena YOLO merupakan *pre-trained model* / model yang sudah dilatih pada dataset besar seperti COCO atau ImageNet maka kita hanya akan menggunakan arsitektur nya saja yang berisi *layer neural network.* Berikut adalah gambar dari arsitektur YOLOv5s.



Gambar… Arsitektur YOLOv5s

Arsitektur YOLOv5s terbagi menjadi 3 bagian utama yaitu *Backbone*, *Neck* dan *Output Layer*. *Backbone Layer* berfungsi mengekstrak informasi penting dari gambar sekaligus mengurangi model yang kompleks. Selanjutnya adalah *Neck Layer*, *layer* ini berfungsi membuat piramida fitur, piramida fitur membantu model untuk mengidentifikasi objek yang sama dengan ukuran dan skala yang berbeda dengan cara digeneralisasi dengan baik pada penskalaan objeknya. Informasi yang dihasilkan oleh backbone layer, informasi tersebut akan di proses oleh neck layer sehingga menghasilkan informasi yang berguna di setiap level fitur yang akan menyebar langsung ke sub-jaringan. *Output Layer* berfungsi untuk melakukan pendeteksian pada bagian akhir, *layer* ini menghasilkan *box*, *confidence* / akurasi dan kelas pada objek yang terdeteksi pada gambar. Pada Bagian ini model akan menghasilkan 3 ukuran *layer* yang berbeda untuk mencapai deteksi *multi-scale*, deteksi *multi-scale* akan memastikan bahwa model dapat mengikuti perubahan ukuran.

Untuk menjalankan proses *training* dengan cara menuliskan perintah "!<python> <train.py> <img> <batch> <epoch> <data> <weights> <cache> <nosave>". Berikut ini adalah gambar proses *training* model YOLOv5s.



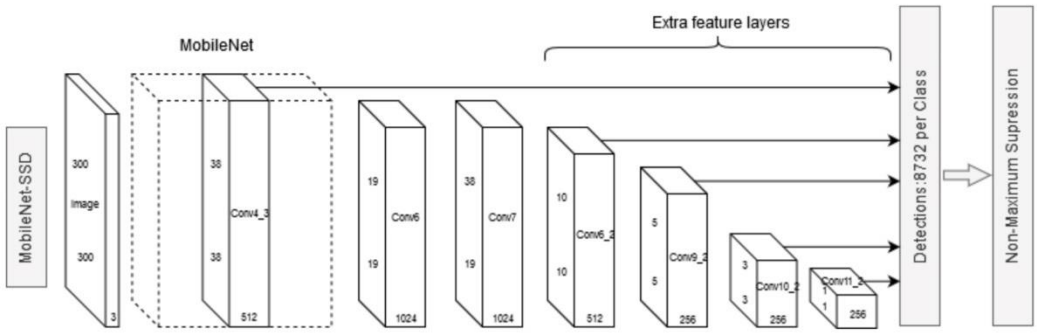
Gambar 4.5. Proses *Training* Model YOLOv5s

Proses *training* pada model pertama selesai dalam waktu 0,457 jam dengan hasil yang bisa dikatakan sudah baik. Untuk lebih jelasnya, penulis paparkan hasil *training* tersebut pada Tabel 4.4 di bawah ini.

Tabel 4.4. Kinerja Hasil *Training* Model YOLOv5s

No	Variabel Kinerja	Score
1	*Box_loss*	0,012
2	*Obj_loss*	0,007
3	*Cls_loss*	0,002
4	*Precision*	0,991
5	*Recall*	0,973
6	*mAP@0,5*	0,979
7	*mAP@0,5:0.95*	0,944

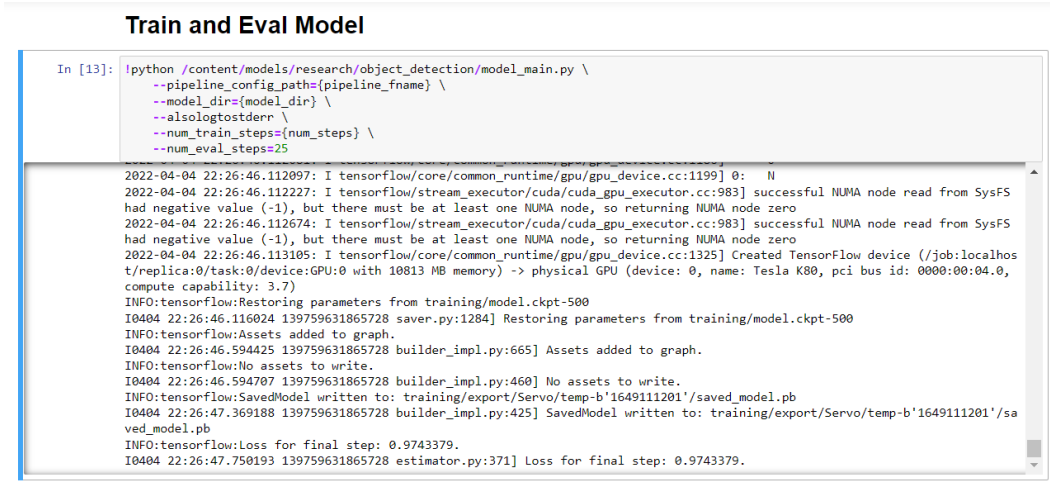
1.2.3.2. Hasil Training SSDMobileNetV2



Gambar…. Arsiteksur SSDMobileNetV2

Arsitektur SSDMobileNetV2 terdiri dari SSD yang berperan sebagai base model dan MobileNet sebagai *Network Model*. SSD akan mengatur pendeteksian objek dengan membuat *bounding box*. MobileNet akan bekerja untuk mengekstrak fitur yang akan nantinya diklasifikasi. Penggabungan SSD dan Mobilenet akan membantu dalam proses pembuatan aplikasi deteksi objek. Dalam aplikasi deteksi objek dibutuhkan SSD untuk membuat lokalisasi gambar untuk menentukan posisi objek. Sedangkan Mobilenet akan dibutuhkan untuk membantu mengklasifikasi objek yang terdapat dalam suatu gambar. Klasifikasi akan menghasilkan kategori untuk masing-masing objek yaitu Good dan Bad.

Untuk menjalankan proses *training* dengan cara menuliskan perintah "!<python> <model_main.py> <pipeline_config_path> <model_dir> <alsologtostderr> <num_train_steps> <num_eval_steps>". Berikut ini adalah gambar proses *training* model SSDMobileNetV2.



Gambar 4.2. Proses *Training* Model SSDMobileNetV2

Proses *training* pada model pertama selesai dalam waktu 0,25 jam dengan hasil yang bisa dikatakan cukup baik. Untuk lebih jelasnya, penulis paparkan hasil *training* tersebut pada Tabel 4.5 di bawah ini

Tabel 4.5. Kinerja Hasil *Training* Model SSDMobileNetV2

No	**Variabel Kinerja**	***Score***
1	*Classification_loss*	6,579
2	*Localization_loss*	2,467
3	*Regularization_loss*	0,307
4	*Precision*	0,850
5	*Recall*	0,907
6	*mAP@0,5*	0,946
7	*mAP@0,5:0.95*	0,851

Ada beberapa hal yang perlu diperhatikan setelah melakukan proses *training model*, yaitu membuat *classification report* untuk memilih model terbaik yang nantinya akan digunakan oleh sistem. Ada beberapa parameter yang menjadi tolak ukur untuk memilih model terbaik seperti *score precision, recall, f1-score, mAP:0,5,* waktu deteksi dan ukuran *file*.

1.2.4. Pengujian Model

Datasets untuk pengujian diambil langsung dari pertanian yang berada di daerah Kabupaten Majalengka, Cirebon dan Indramayu, ada sekitar 100 datasets dengan 111 *bounding box* pada gambar, diantaranya 40 gambar sawah kualitas bagus dengan 47 *bounding box* dan 60 gambar sawah kualitas rusak / jelek dengan 64 *bounding box* yang diambil menggunakan teknologi *drone*. Pengujian ini bertujuan untuk memilih model terbaik yang akan digunakan atau diimplementasikan, model yang digunakan untuk pengujian yaitu YOLOv5s dan SSDMobileNetV2. *Output* yang dihasilkan adalah *Precision, Recall, mAP@0.5, mAP@.5:.95*, waktu deteksi dan ukuran file dari masing-masing model.

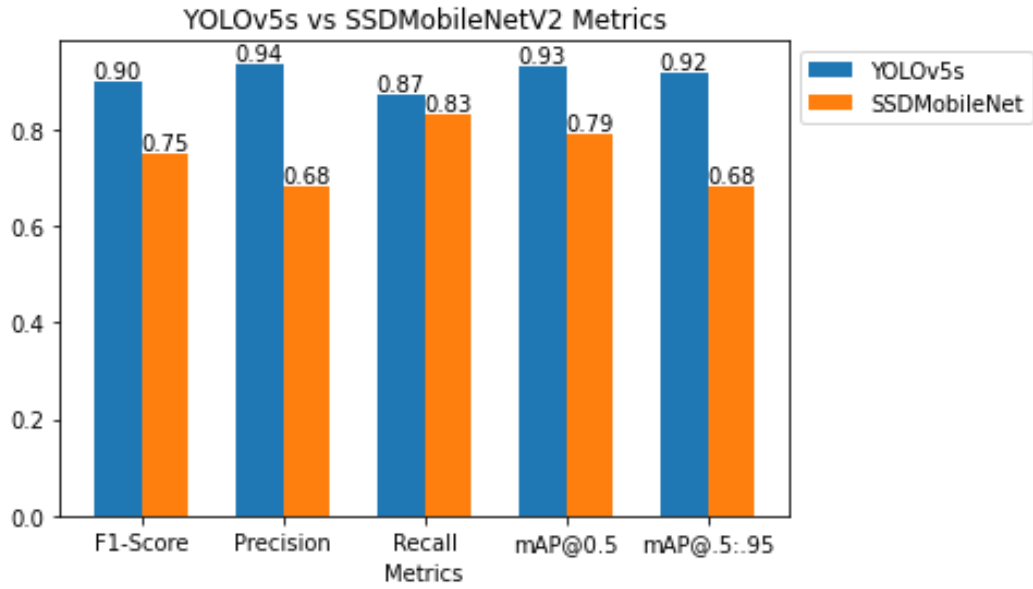
1.2.4.1. Perbandingan Hasil Pengujian

Berikut merupakan tabel hasil perbandingan antara model YOLOv5s dan SSDMobileNetV2.

Tabel 4.7 Perbandingan Hasil Uji Coba Antara Model YOLOv5s dan YOLOv5m.

No	Variabel Kinerja	YOLOv5s	SSDMobileNetV2
2	F1-Score	0,90	0,750
3	Precision	0,936	0,683
4	Recall	0,87	0,832
5	mAP@0.5	0,93	0,790
6	mAP@.5:.95	0,916	0,683
7	Waktu Deteksi	0,063s	0,144s
8	Ukuran File	14,4Mb	18,4Mb

Penulis memutuskan untuk menggunakan model YOLOv5s untuk penelitian ini, YOLOv5s dapat bekerja lebih cepat dibanding SSDMobileNetV2 serta waktu deteksi dan ukuran file dari model nya pun lebih kecil. YOLOv5s memiliki keunggulan pada nilai *F1-Score*, *Precision*, *Recall* dan mAP(*mean average precision*) dibuktikan dengan gambar grafik perbandingan berikut.



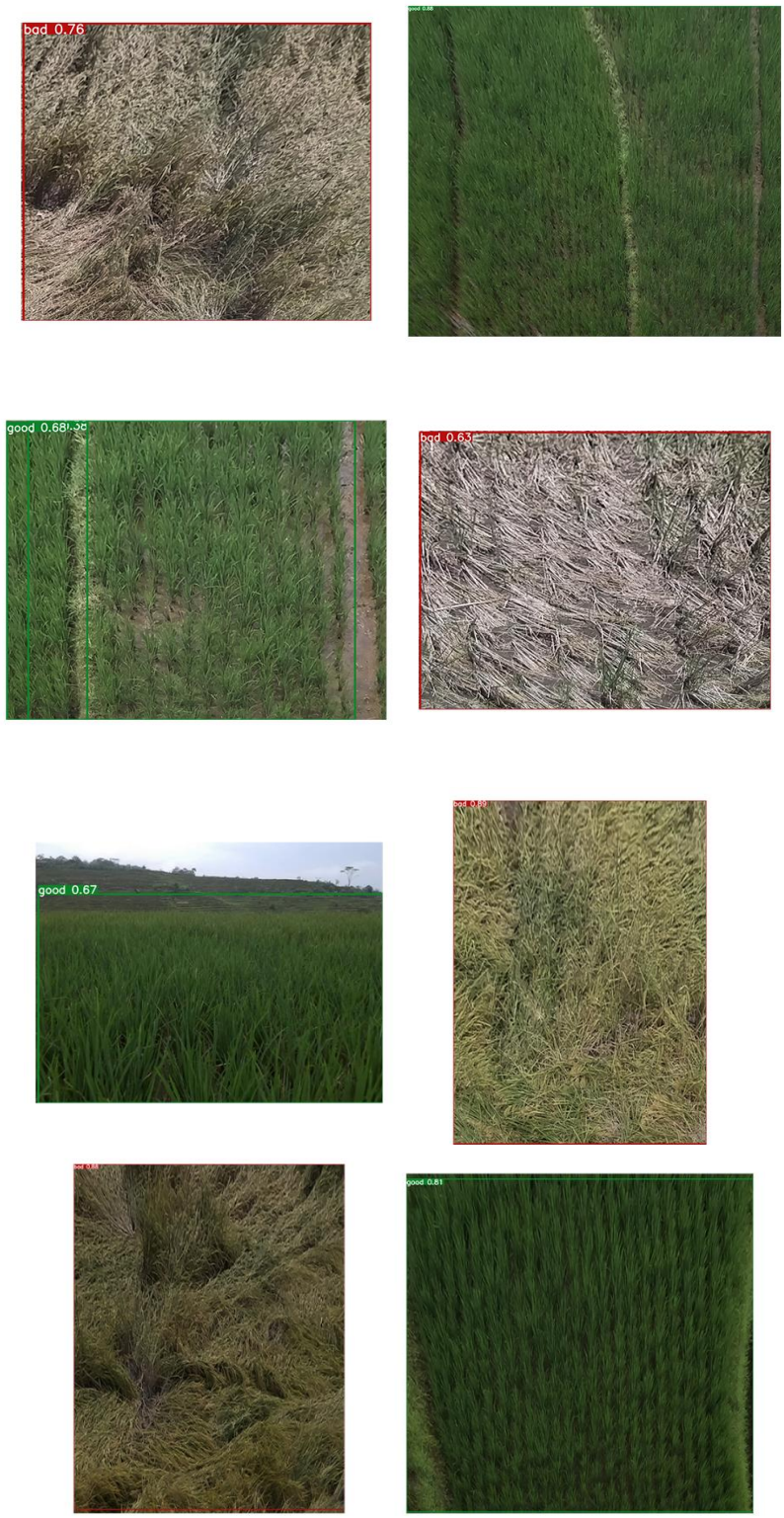
Gambar ….. YOLOv5s vs SSDMobileNetV2 Metrics

Gambar … merupakan grafik perbandingan dari masing-masing model terhadap *metrics* yang digunakan oleh kedua model yaitu COCO. Dapat dilihat pada gambar diatas, YOLOv5s selalu unggul di semua *metrics* yang digunakan, ini membuktikan bahwa YOLOv5s terbukti lebih baik dalam memprediksi datasets yang diambil langsung dari lapangan. Waktu deteksi YOLOv5s pun lebih cepat dibandingkan waktu deteksi SSDMobileNetV2, ini berarti FPS yang dihasilkan akan jauh lebih tinggi dan waktu komputasi pun tidak akan terlalu lama. Maka dari itu model YOLOv5s merupakan pilihan terbaik untuk mendeteksi tingkat kualitas tanaman padi.

1.2.4.2. Pengujian Deteksi Tingkat Kualitas Tanaman Padi

Setelah dibandingkan dan mendapat kesimpulan, bahwa model yang menggunakan arsitektur YOLOv5s adalah yang terbaik untuk mendeteksi tingkat

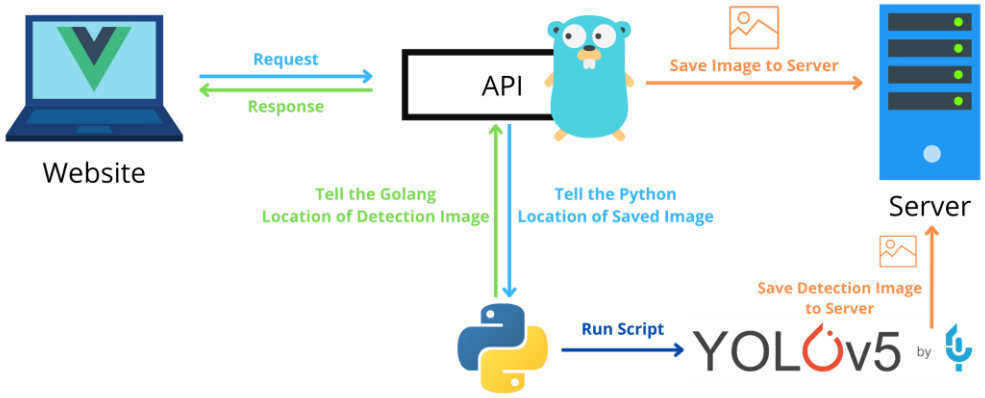
kualitas tanaman padi. Model YOLOv5s menjadi acuan untuk sistem deteksi ini, berikut beberapa hasil deteksi yang dilakukan:



Gambar Deteksi Tingkat Kualitas Tanaman Padi menggunakan YOLOv5s

1.2.5. Hasil Integrasi Sistem

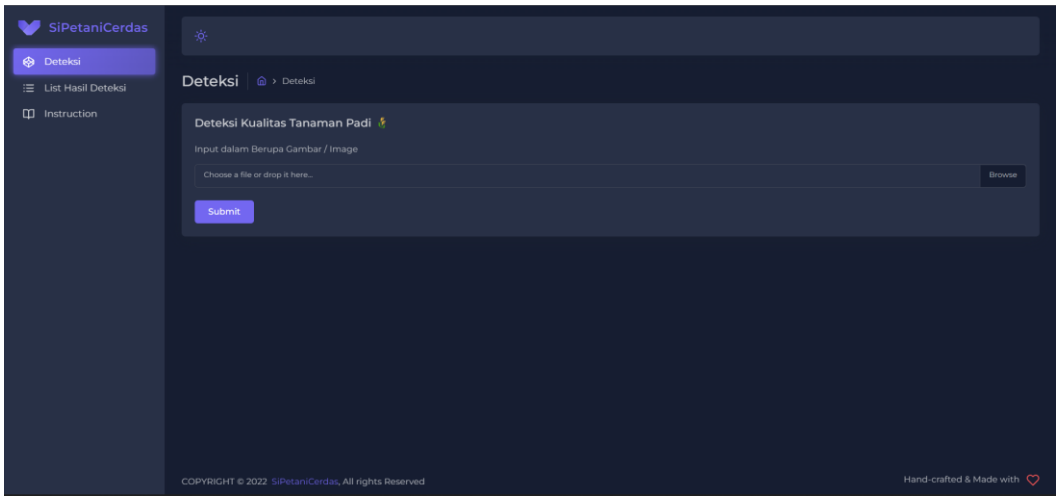
Pada bagian ini akan dibahas hasil dari integrasi antara sistem yang penulis buat dengan Aplikasi SiPetaniCerdas (Sistem Informasi Petani Cerdas). Berikut adalah Gambar Arsitektur Sistem yang digunakan.



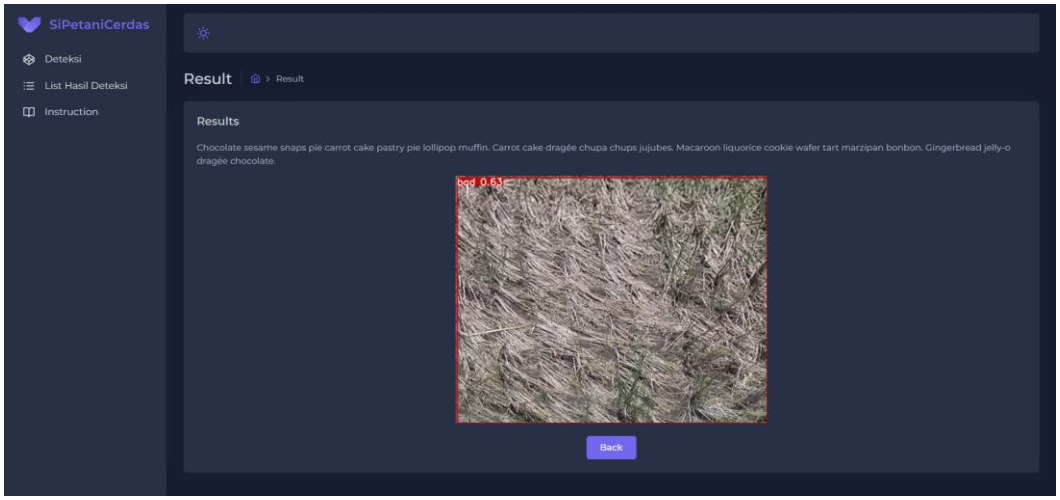
Gambar .. Arsitektur Sistem

Seperti yang digambarkan diatas bahwa untuk menjembatani hubungan antara *server* dengan Aplikasi SiPetaniCerdas yaitu dengan menggunakan API sebagai perantara. *Website* mengirimkan gambar menggunakan format *form-data* ke API yang dibuat menggunakan golang, gambar tersebut akan disimpan ke dalam *server*, dan lokasi gambar yang disimpan akan dimasukan ke dalam format eksekusi *script* python. Selanjutnya setelah format *script* python terbentuk, golang akan membuat perintah untuk mengeksekusi nya melalui *command-line*. Setelah *script* python dijalankan, *script* tersebut akan mengeksekusi perintah untuk mendeteksi tingkat kualitas tanaman padi menggunakan model YOLOv5s, selanjutnya *script* akan menyimpan hasil deteksi ke dalam bentuk gambar yang akan disimpan di *server*, *script* akan mengeluarkan *output* berupa lokasi gambar pada *server* yang telah dideteksi. *Output* berisi lokasi gambar yang telah dideteksi tersebut akan digunakan golang untuk mengembalikan *response* kepada *website*, dan nantinya akan ditampilkan oleh *website* menggunakan *framework* vue-js. Pada Aplikasi SiPetaniCerdas, pengecekan tingkat kualitas tanaman padi dilakukan secara otomatis dengan hanya mengunggah gambar. Metode tersebut sangat cocok diterapkan pada fitur yang membutuhkan input manual dari petani. Adapaun fitur dan format sebagai berikut.

Gambar Format API



Gambar Halaman *Form* Unggah Gambar



Gambar Halaman Hasil Deteksi Tingkat Kualitas Tanaman Padi