BAB 3
METODOLOGI PENELITIAN

Penelitian ini merupakan penelitian pengembangan *(Research and Development)*. Penelitian berfokus pada pengembangan *image recognition* AI dan implementasinya dengan *input* dari kamera *drone*.

3.1 Studi Literatur

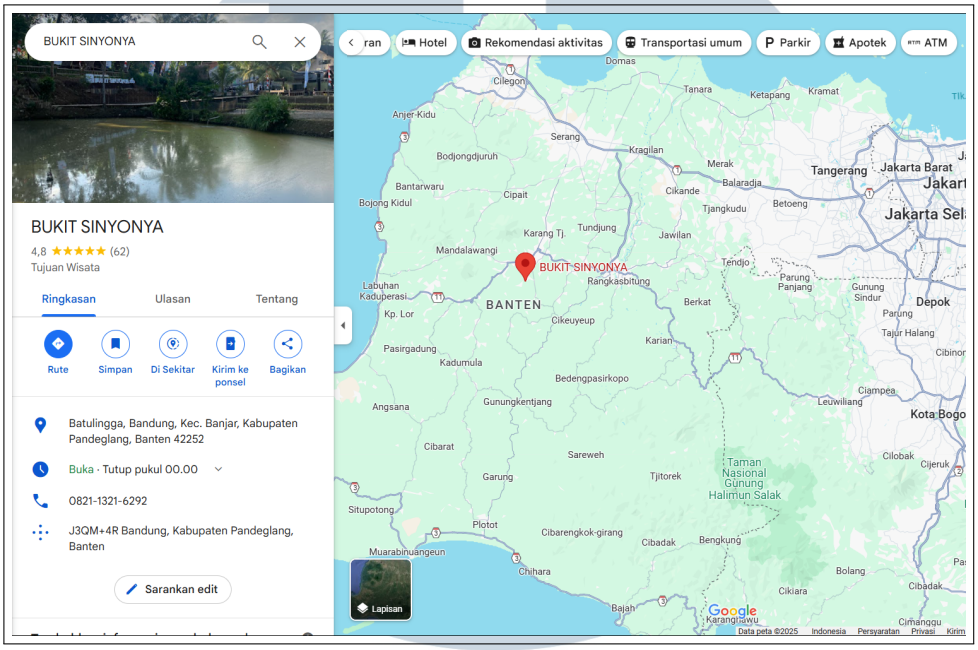
Tahap awal dalam penelitian ini adalah melakukan studi literatur untuk memahami konsep dan teknologi yang relevan dengan pengembangan sistem deteksi kesehatan tanaman padi berbasis *image recognition* AI. Studi ini mencakup eksplorasi literatur mengenai teknologi *drone*, pengolahan citra digital, *machine learning* untuk analisis tanaman, serta pendekatan yang telah digunakan dalam penelitian sebelumnya. Selain itu, analisis terhadap sistem serupa yang telah dikembangkan akan dilakukan sebagai studi banding guna mengidentifikasi kelebihan, kekurangan, serta peluang inovasi yang dapat diterapkan dalam penelitian ini. Dengan pendekatan ini, pengembangan sistem akan lebih terarah dan berbasis pada wawasan yang kuat.

3.2 Perancangan

Pada tahap perancangan ini, peneliti beserta tim mendiskusikan rancangan sistem yang sesuai dengan hasil yang diinginkan dengan sumber daya yang tersedia. Diskusi ini memperoleh hasil berupa berikut:

- Pembagian pekerjaan antara anggota tim peneliti. Penulis, Daffi Bintang Firdaus akan fokus dalam pengembangan AI *image recognition*, dan rekan peneliti Muh. Rizki Fadhil bertugas dalam otomatisasi *drone* yang akan dipakai.
- Untuk *drone* yang akan dipakai, tim peneliti setuju untuk menggunakan *drone* milik Lab FIKOM UMN.
- Untuk *hardware* yang akan digunakan untuk pengembangan dan *training* AI *image recognition*, tim peneliti setuju untuk menggunakan laptop pribadi penulis, sebuah laptop TUF Gaming A15 FA507RC.

- Untuk lokasi diambilnya data untuk penelitian, tim peneliti setuju untuk dilaksanakan di Bukit Sinyonya, Batulingga, Bandung, Kecamatan Banjar, Kabupaten Pandeglang, Banten.



Gambar 3.1. Lokasi penelitian dilihat melalui *Google Maps*.

Selain perancangan tersebut, peneliti juga melakukan pencarian program yang akan dipakai yang sesuai dengan kapabilitas yang diinginkan dan sumber daya yang dimiliki. Berikut program yang penulis gunakan dalam penelitian ini:

Tabel 3.1. Program yang digunakan selama penelitian

Nama program	Tujuan program	Alasan dipilih
Anaconda Navigator dan Anaconda Prompt	*Environment* pengembangan	Tidak berbayar dan sudah dipahami peneliti.
Jupyter Notebook	Pengembangan dan *training model*	*Web based*, dapat digunakan menggunakan Anaconda Prompt, dan sudah dipahami peneliti.
Label Studio	*Dataset labelling*	*Web based*, *open source* dan dapat digunakan menggunakan Anaconda Prompt.
Shotcut	*Editor* video	*Open source*, tidak berbayar, dan memiliki fitur ekspor video per-*frame* menjadi file gambar.

3.3 Pengumpulan Data

Pengumpulan data yang akan dilaksanakan di lokasi penelitian Bukit Sinyonya terhambat dengan tidak tersedianya *drone* Lab FIKOM UMN yang sesuai dengan spesifikasi yang dibutuhkan. Oleh karena itu, Lab FIKOM UMN setuju untuk mengadakan *drone* baru DJI Mini 4 Pro untuk keperluan penelitian sekaligus untuk peremajaan *drone* milik lab.

Pengadaan *drone* baru tersebut sempat menghambat proses pengambilan data yang dibutuhkan, sekaligus menghambat proses pengembangan AI dan sistem otomatisasi *drone* yang sedang dikerjakan. Proses pengajuan pengadaan *drone* yang dilakukan pada saat registrasi penelitian ke LPPM UMN pada akhir bulan Januari, baru dapat dipenuhi pada tanggal 2 Mei 2025.

Setelah *drone* DJI Mini 4 Pro yang dibutuhkan telah tersedia, dilaksanakan pertemuan antara tim peneliti dengan perwakilan Lab FIKOM UMN atas nama Bapak Daniel Steven Nicholas pada tanggal 9 Mei 2025 di Ruang Collabo Gedung D Lantai 7 Universitas Multimedia Nusantara. Pada pertemuan tersebut, Bapak Steven Nicholas setuju untuk meminjamkan *drone* yang dibutuhkan oleh tim peneliti, serta ikut hadir dalam proses pengambilan data di lokasi penelitian untuk memantau dan mendampingi tim peneliti dalam proses operasional *drone* yang akan dilaksanakan pada Hari Minggu, 11 Mei 2025.

Pada tanggal 11 Mei 2025, di lokasi penelitian Bukit Sinyonya, tim peneliti melaksanakan proses pengumpulan data gambar dan video persawahan menggunakan *drone*. Kegiatan ini didampingi oleh perwakilan Lab FIKOM UMN, Pak Daniel Steven Nicholas, yang memiliki sertifikasi penerbang *drone*. Proses pengambilan data berlangsung selama kurang lebih 4 jam.

Gambar 3.2. Persiapan *drone* DJI Phantom 3 Pro oleh perwakilan Lab FIKOM UMN Pak Steven Nicholas.

Gambar 3.3. *Drone* diterbangkan oleh Peneliti Muh. Rizky Fadhil dibantu oleh Peneliti Daffi Bintang Firdaus dibawah supervisi perwakilan Lab FIKOM UMN Pak Steven Nicholas, beserta Dosen Peneliti Pak Winarno.

3.4 Pengembangan

Setelah proses pengumpulan data selesai, kebutuhan untuk memulai proses pengembangan telah terpenuhi. Pada 12 Mei 2025, proses pengembangan dimulai.

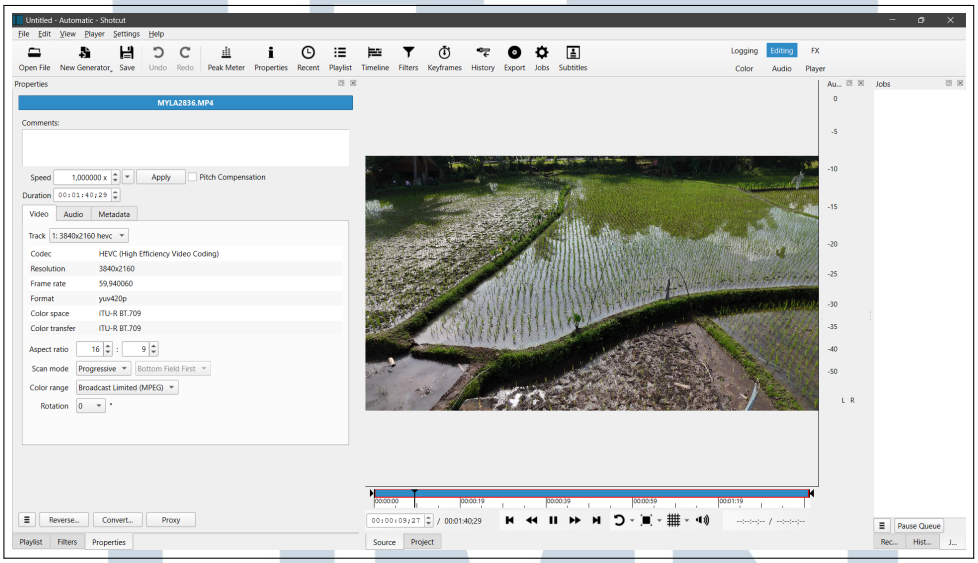
Proses pengembangan dapat dipecah menjadi beberapa langkah berikut:

- Menggunakan Label Studio dan Anaconda, mengolah gambar awal menjadi materi *training model* FR-CNN yang akan membedakan setiap individu tanaman menggunakan *bounding box*.
- Pemrograman kode *model* FR-CNN menggunakan program Jupyter Notebook.
- Proses *training model* FR-CNN menggunakan data yang sudah diolah.
- Proses *debugging* dan improvisasi akurasi *model* FR-CNN.
- Pengolahan gambar individu tanaman menjadi materi *training model* U-NET yang akan mengekstrak RGB *value* daun dari setiap tanaman dengan menggunakan *masking*.
- Pemrograman kode *model* U-NET.
- Proses *training model* U-NET.

- Proses *debugging* dan improvisasi akurasi *model* U-NET.
- Membuat *script* Python di Jupyter Notebook yang akan menangani proses pemberian data, pemanggilan masing-masing *model*, dan kompilasi hasil.

3.4.1 Proses Pengolahan Video Menjadi Gambar

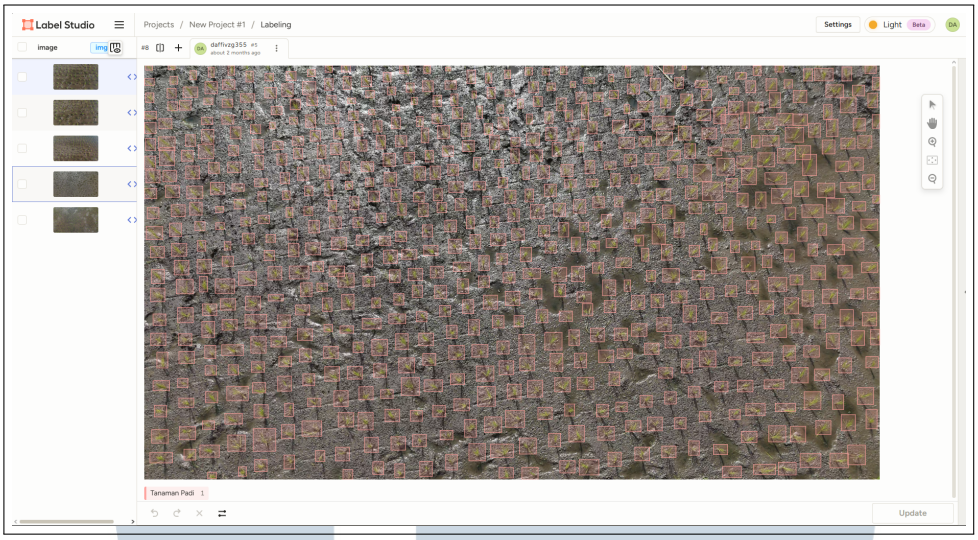
Dalam proses ini, peneliti menggunakan aplikasi Shotcut yang digunakan untuk mengekspor video rekaman dari kamera *drone* menjadi *format* gambar .JPG per-*frame* agar dapat diolah di program Label Studio.



Gambar 3.4. Proses ekspor video menjadi gambar menggunakan Shotcut.

3.4.2 Labelling Data dengan Label Studio Bounding Box

Proses *labelling* ini dilakukan untuk mengolah data awal yang berupa gambar sawah yang luas untuk isolasi objek yang diinginkan, berupa tanaman padi, menggunakan *bounding box*. Proses *labelling* tersebut menggunakan program *labelling* Label Studio. *Dataset* hasil *labelling* diekspor dengan *format* COCO.

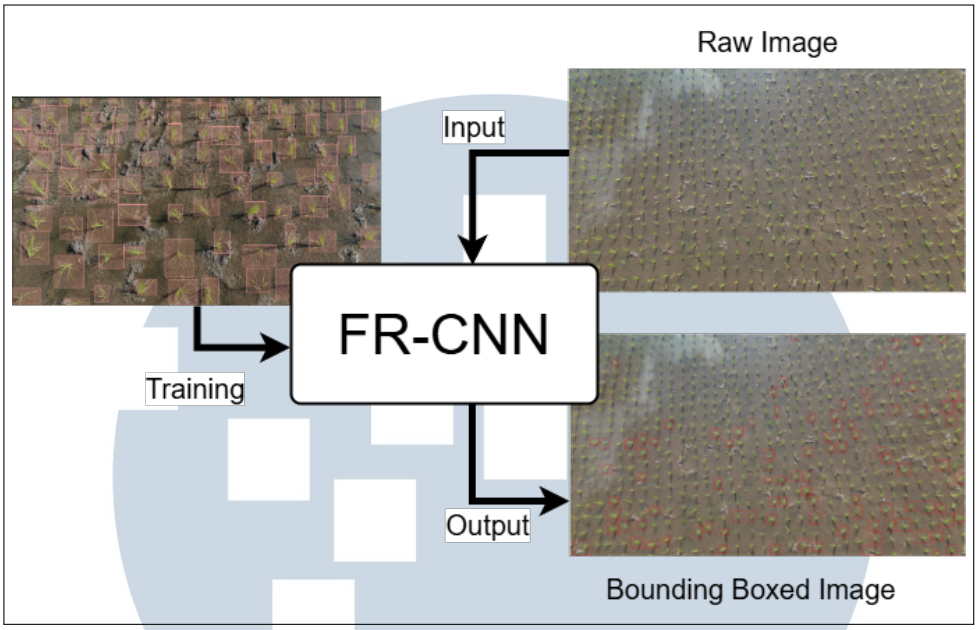


Gambar 3.5. Proses *labelling* menggunakan Label Studio.

3.4.3 Pemrograman dan Training model FR-CNN

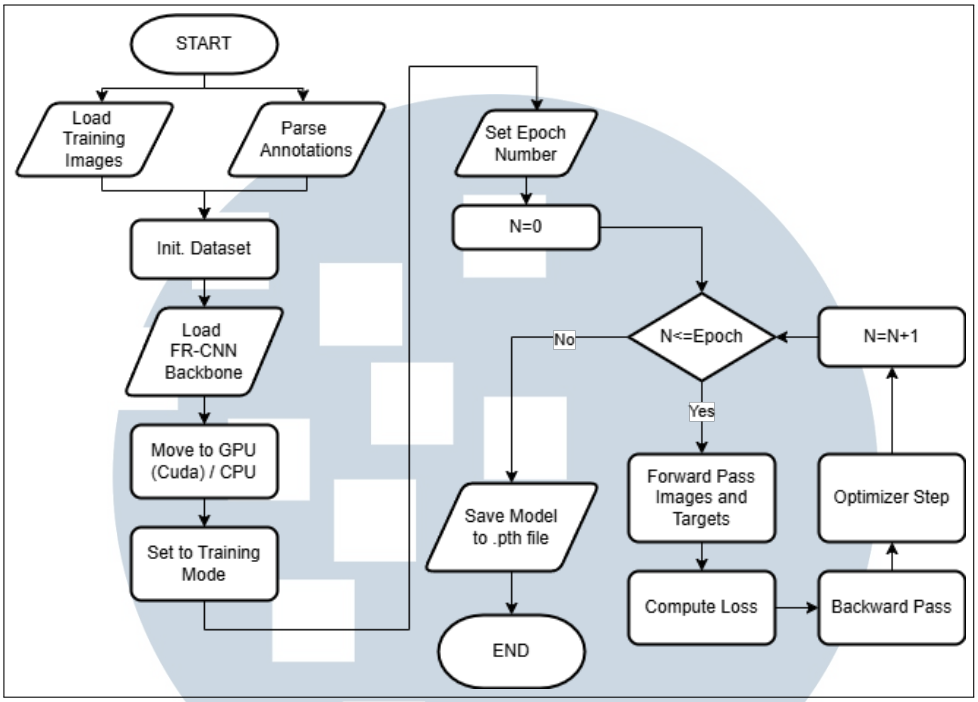
Seluruh proses pemrograman dan *training model* yang akan dipakai menggunakan bahasa *Python* dengan *environtment* program *web based* Jupyter Notebook yang mudah digunakan dan ringan pada perangkat.

Training model FR-CNN (*Faster* R-CNN) yang digunakan dalam penelitian ini dilakukan dengan pendekatan *supervised learning*, yaitu *model* dilatih menggunakan data gambar yang telah diberi *label bounding box* dan *class label* ”Tanaman Padi”. *Model* FR-CNN terdiri dari dua komponen utama: *region Proposal* dan *object classification*. *Region proposal* bertugas menghasilkan area-area proposal yang berpotensi mengandung objek, kemudian proposal tersebut diklasifikasikan dan disempurnakan oleh bagian *object classification*. Selama *training*, *model* belajar secara bertahap untuk menghasilkan proposal yang lebih akurat dan klasifikasi yang lebih tepat melalui fungsi *multi-task loss* yang menggunakan Cross Entropy Loss dan *bounding box regression loss* yang menggunakan Smooth L1 Loss.



Gambar 3.6. Gambaran *training model* FR-CNN.

Dalam implementasinya, *model* FR-CNN dilatih menggunakan *dataset* hasil *labelling* manual melalui Label Studio, di mana setiap tanaman padi dalam gambar diberikan *bounding box* sesuai format COCO. Proses *training* dilakukan dengan menggunakan *library* PyTorch, dan optimalisasi parameter dilakukan menggunakan algoritma *Stochastic Gradient Descent* (SGD) dengan *learning rate* sebesar 0.005, momentum 0.9, dan *weight decay* 0.0005 untuk menjaga dari irregularitas yang dapat terjadi. Selama proses *training*, *model* melakukan evaluasi prediksi terhadap *ground truth* dan menyesuaikan *weight* jaringan melalui *backpropagation*. Setelah *training* selesai, *model* mampu mendeteksi posisi tanaman padi dalam gambar secara otomatis dengan tingkat akurasi tertentu berdasarkan *confidence threshold* yang telah ditentukan berupa 0,4.



Gambar 3.7. *Flowchart training model* FR-CNN.

Setelah FR-CNN berhasil memberikan *bounding box* pada gambar baru yang diberikan, program akan melakukan *crop* pada *bounding box* untuk mengisolasi setiap individu tanaman. Tanaman individu tersebut akan disimpan pada *folder* ”Cropped” dan akan dipakai sebagai materi *training model* U-NET.

```
model.to(device)

params = [p for p in model.parameters() if p.requires_grad]
optimizer = optim.SGD(params, lr=0.005, momentum=0.9, weight_decay=0.0005)

num_epochs = 4

for epoch in range(num_epochs):
    model.train()
    epoch_loss = 0
    for imgs, targets in tqdm(train_loader):
        imgs = list(img.to(device) for img in imgs)
        targets = [{k: v.to(device) for k, v in t.items()} for t in targets]

        loss_dict = model(imgs, targets)
        losses = sum(loss for loss in loss_dict.values())

        optimizer.zero_grad()
        losses.backward()
        optimizer.step()

        epoch_loss += losses.item()
    print(f"Epoch {epoch+1}, Loss: {epoch_loss:.4f}")
```

```
100%|██████████| 2/2 [00:33<00:00, 16.69s/it]
Epoch 1, Loss: 8.1614
100%|██████████| 2/2 [00:27<00:00, 13.75s/it]
Epoch 2, Loss: 3.6291
100%|██████████| 2/2 [00:26<00:00, 13.37s/it]
Epoch 3, Loss: 3.5054
100%|██████████| 2/2 [00:26<00:00, 13.27s/it]
Epoch 4, Loss: 2.9956
```

Gambar 3.8. Proses *training model* FR-CNN di Jupyter Notebook.

3.4.4 Labelling Materi Training U-NET

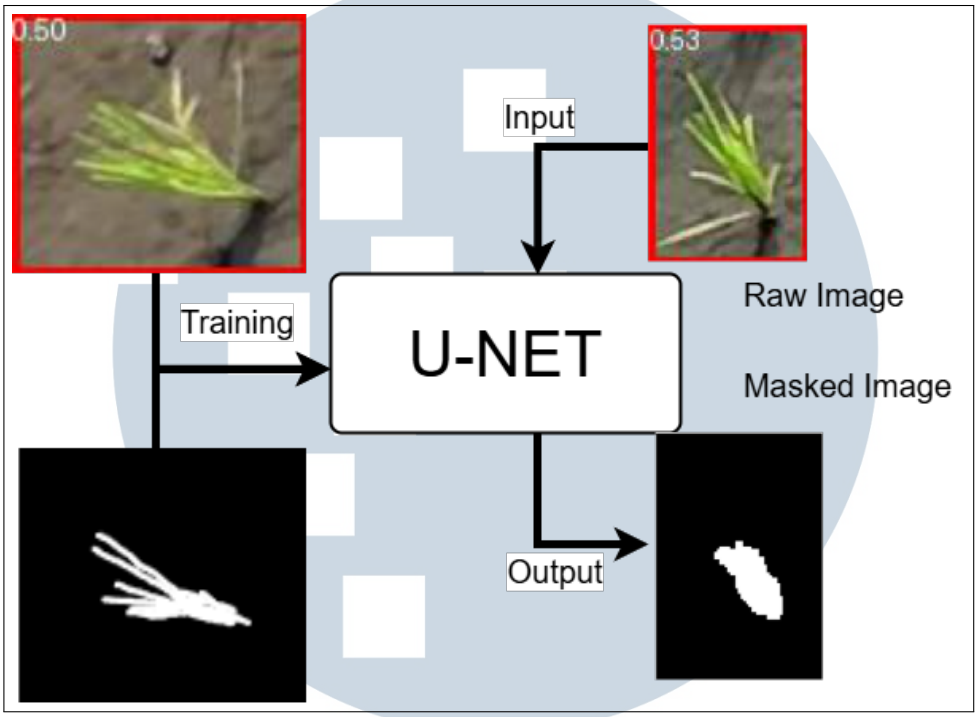
Persiapan materi *training* U-NET didapatkan dengan menggunakan hasil *crop* dari FR-CNN sebelumnya. Gambar tersebut diproses dengan program Label Studio dengan menggunakan *masking*. Target dari *masking* tersebut adalah daun dari setiap individu tanaman. Hasil *masking* tersebut diekspor dengan *format brush label to COCO* dan *YOLOv8 OBB with images*. *Format* sebenarnya yang dibutuhkan adalah *brush label to COCO*, namun *format* tersebut hanya mengekspor *mask*-nya saja, tanpa gambar awal yang dijadikan materi *training*. Oleh karena itu, peneliti juga mengekspor hasil *labelling* tersebut dengan *format YOLOv8 with images* untuk mengekstrak gambar asli yang akan digunakan sebagai materi *training*. *File* JSON yang juga diberikan oleh format *YOLOv8 with images* tidak dipakai untuk *training model* U-NET ini.

Gambar 3.9. Proses *labelling* materi *model* U-NET di Label Studio. Bagian yang berwarna merah muda adalah *masking* yang diberikan.

3.4.5 Pemrograman dan Training model U-NET

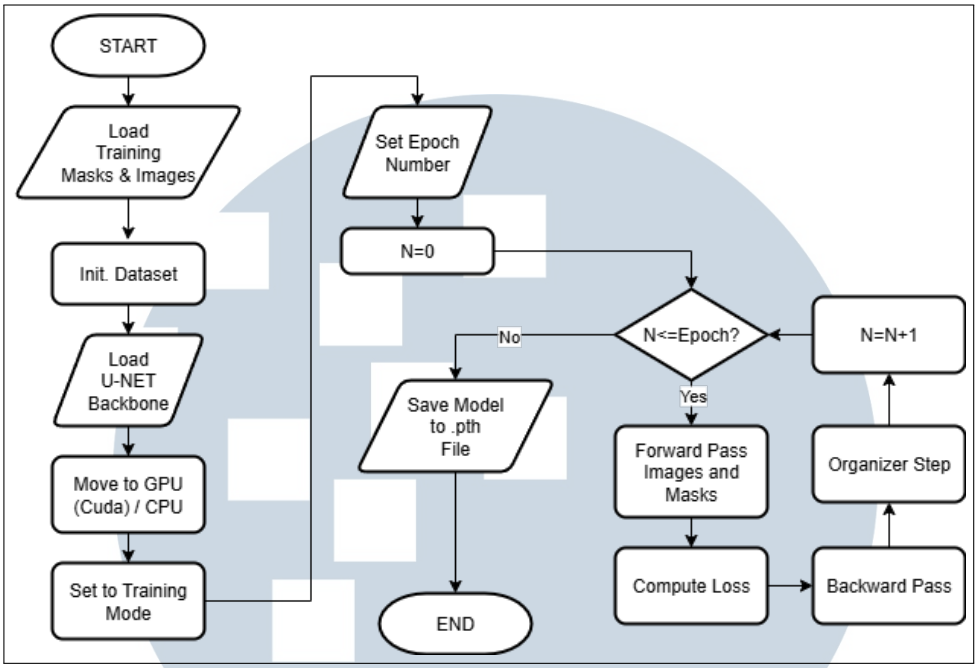
Training model U-Net dalam penelitian ini bertujuan untuk melakukan segmentasi daun tanaman padi, yaitu memisahkan area daun dari latar belakang pada citra hasil *crop* dari deteksi RCNN sebelumnya. *model* U-Net yang digunakan memiliki arsitektur *encoder-decoder* simetris, di mana bagian *encoder* bertugas mengekstraksi fitur melalui operasi *convolution* dan *pooling*, sementara *decoder* melakukan proses *upsampling* untuk merekonstruksi peta segmentasi. Setiap tahap

decoder disambungkan dengan fitur dari *encoder* yang sejajar menggunakan *skip connection*, sehingga informasi *spatial* tetap terjaga selama proses rekonstruksi.

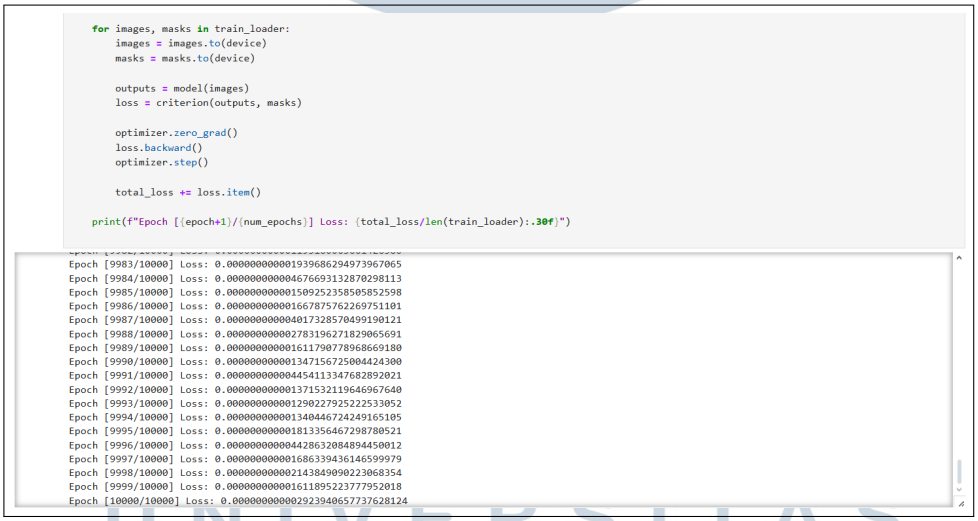


Gambar 3.10. Ilustrasi *training model* U-NET.

Model ini dilatih menggunakan *dataset* gambar daun dan *binary mask* yang terdiri dari area daun (1) dan bukan daun (0). Gambar dan *mask* terlebih dahulu diubah ukurannya menjadi resolusi tetap (256x256 *pixel*) dan dikonversi menjadi *tensor*. *Training* dilakukan menggunakan fungsi *Binary Cross Entropy Loss (BCELoss)*, yang membandingkan prediksi *model* terhadap *mask ground truth*. Optimisasi *weight model* dilakukan dengan algoritma Adam dengan *learning rate* awal sebesar 1e-4. Proses *training* dijalankan selama beberapa *epoch* dengan metode *mini-batch* untuk mempercepat *convergence* dan meningkatkan generalisasi *model*. Setelah *training* selesai, *model* digunakan untuk memprediksi *mask* daun dari gambar tanaman padi individu baru. Hasil segmentasi ini kemudian digunakan untuk menghitung nilai rata-rata warna daun dari nilai RGB sebagai dasar penilaian kesehatan tanaman.



Gambar 3.11. *Flowchart training model* U-NET.



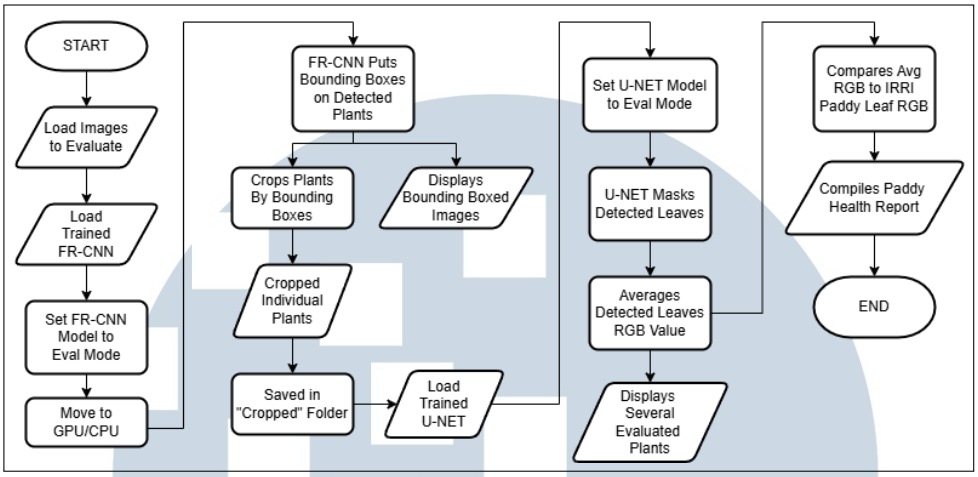
```
for images, masks in train_loader:
    images = images.to(device)
    masks = masks.to(device)

    outputs = model(images)
    loss = criterion(outputs, masks)

    optimizer.zero_grad()
    loss.backward()
    optimizer.step()

    total_loss += loss.item()

print(f"Epoch [{epoch+1}/{num_epochs}] Loss: {total_loss/len(train_loader):.30f}")
```

```
Epoch [9983/10000] Loss: 0.000000000001939686294973967065
Epoch [9984/10000] Loss: 0.000000000004676693132870298113
Epoch [9985/10000] Loss: 0.000000000001509252358505852598
Epoch [9986/10000] Loss: 0.000000000001667875762269751101
Epoch [9987/10000] Loss: 0.000000000004017328570499190121
Epoch [9988/10000] Loss: 0.000000000002783196271829065691
Epoch [9989/10000] Loss: 0.000000000001611790778968669180
Epoch [9990/10000] Loss: 0.000000000001347156725004424300
Epoch [9991/10000] Loss: 0.000000000004454113347682892021
Epoch [9992/10000] Loss: 0.000000000001371532119646967640
Epoch [9993/10000] Loss: 0.000000000001290227925222533052
Epoch [9994/10000] Loss: 0.000000000001340446724249165105
Epoch [9995/10000] Loss: 0.000000000001813356467298780521
Epoch [9996/10000] Loss: 0.000000000004428632084894450012
Epoch [9997/10000] Loss: 0.000000000001686339436146599979
Epoch [9998/10000] Loss: 0.000000000002143849090223068354
Epoch [9999/10000] Loss: 0.000000000001611895223777952018
Epoch [10000/10000] Loss: 0.000000000002923940657737628124
```

Gambar 3.12. *Training model* U-NET di Jupyter Notebook.

3.4.6 Pemrograman Script Utama

Script utama ini digunakan sebagai jembatan bagi pengguna untuk menjalankan kedua *model* dan memberikan hasil. Berikut adalah cara kerja *script*:



Gambar 3.13. *Flowchart script* utama.

Pada saat dieksekusi, *script* utama akan melakukan *load* gambar-gambar awal yang dimasukkan ke dalam *folder* "input_images", beserta *model* FR-CNN yang akan dipakai, dan menyiapkan *model* FR-CNN tersebut ke dalam *mode* evaluasi.

```

model_path = "paddy_rcnn_model.pth"
input_folder = "C:/Users/Daffi Bintang/PadiVision/notebooks/
    input_images"
confidence_threshold = 0.3
device = torch.device("cuda" if torch.cuda.is_available() else "
    cpu")
model.to(device)
model.eval()
```

Kode 3.1: Bagian *load* gambar awal.

Setelah FR-CNN dan gambar awal telah selesai di *load*, *script* utama akan melakukan *image processing* berupa memuat setiap gambar ke *tensor*, menjalankan FR-CNN untuk mencari tanaman padi dalam gambar yang diberikan, dan menggambarkan *bounding box* pada tanaman padi yang ditemukan.

```

for filename in os.listdir(input_folder):
    if not filename.lower().endswith(('.jpg', '.jpeg', '.png')):
        continue

    image_path = os.path.join(input_folder, filename)
    image = Image.open(image_path).convert("RGB")
    img_tensor = F.to_tensor(image).unsqueeze(0).to(device)
```

```
    # Run detection
    with torch.no_grad():
        outputs = model(img_tensor)[0]

    boxes = outputs['boxes'].cpu()
    scores = outputs['scores'].cpu()

    # Draw boxes
    draw = ImageDraw.Draw(image)
    count = 0
    for box, score in zip(boxes, scores):
        if score < confidence_threshold:
            continue
        x1, y1, x2, y2 = box.tolist()
        draw.rectangle([x1, y1, x2, y2], outline="red", width=3)
        draw.text((x1, y1), f"{score:.2f}", fill="white")
        count += 1
```

Kode 3.2: Bagian *main script* yang menangani *image processing* diikuti dengan prediksi FR-CNN dan *bounding box* setiap tanaman padi.

Setelah proses pemberian *bounding box* telah selesai, *script* utama akan melakukan *crop* menggunakan informasi *bunding box* tersebut untuk mengisolasi setiap individu tanaman padi.

```
for i, (box, score) in enumerate(zip(boxes, scores)):
     if score < confidence_threshold:
         continue
     x1, y1, x2, y2 = map(int, box.tolist())
     cropped = image.crop((x1, y1, x2, y2))
     crop_path = f"C:/Users/Daffi Bintang/PadiVision/notebooks/
   cropped/{filename[:-4]}_crop{i}.jpg"
     cropped.save(crop_path)
```

Kode 3.3: *Crop*/potong per individu tanaman yang terdeteksi oleh FR-CNN.

Setelah proses *crop* setiap individu tanaman selesai, *script* akan mempersiapkan gambar individu tersebut untuk mempersiapkan proses U-NET. U-NET akan memberikan *masking* pada setiap *pixel* daun yang terdeteksi. Dari *masking* tersebut juga U-NET melakukan *averaging* RGB *value* dari pixel daun pada setiap individu tanaman padi.

```
# Load U-Net
```

```
unet_model = UNet()
unet_model.load_state_dict(torch.load("C:/Users/Daffi Bintang/PadiVision/notebooks/UNET/unet_paddy_leaf.pth", map_location=device, weights_only=True))
unet_model.to(device)
unet_model.eval()

# Transformation
unet_transform = transforms.Compose([
    transforms.Resize((256, 256)),
    transforms.ToTensor()
])

import cv2
all_avg_colors = []
sample_outputs = []

crop_dir = "C:/Users/Daffi Bintang/PadiVision/notebooks/cropped"
crop_files = os.listdir(crop_dir)
random.shuffle(crop_files)

for crop_file in crop_files:
    crop_path = os.path.join("C:/Users/Daffi Bintang/PadiVision/notebooks/cropped", crop_file)
    crop_img = Image.open(crop_path).convert("RGB")

    input_tensor = unet_transform(crop_img).unsqueeze(0).to(device)

    # predict mask
    with torch.no_grad():
        pred_mask = unet_model(input_tensor)[0][0].cpu().numpy()

    # treshold and resize mask back to original crop size
    binary_mask = pred_mask > 0.5
    resized_mask = Image.fromarray(binary_mask.astype(np.uint8) * 255).resize(crop_img.size)
    mask_array = np.array(resized_mask) > 127

    # apply mask and compute average RGB
    crop_np = np.array(crop_img)
    masked_pixels = crop_np[mask_array]
    if len(masked_pixels) == 0:
```

```
        continue
    avg_rgb = masked_pixels.mean(axis=0).astype(np.uint8)
    all_avg_colors.append(avg_rgb)
```

Kode 3.4: U-NET untuk mengisolasi daun tanaman padi dengan menggunakan *masking*.

Setelah U-NET memberikan rata-rata RGB *value* dari setiap tanaman, *script* akan melakukan *averaging* pada RGB *value* untuk seluruh tanaman yang diperiksa *(batch)*. Hasil *averaging* tersebut akan dibandingkan dengan RGB *value* LCC IRRI yang sudah diketahui sebelumnya, dan memberikan status kesehatan *batch* tanaman padi yang diperiksa dengan klasifikasi/*label* warna LCC IRRI.

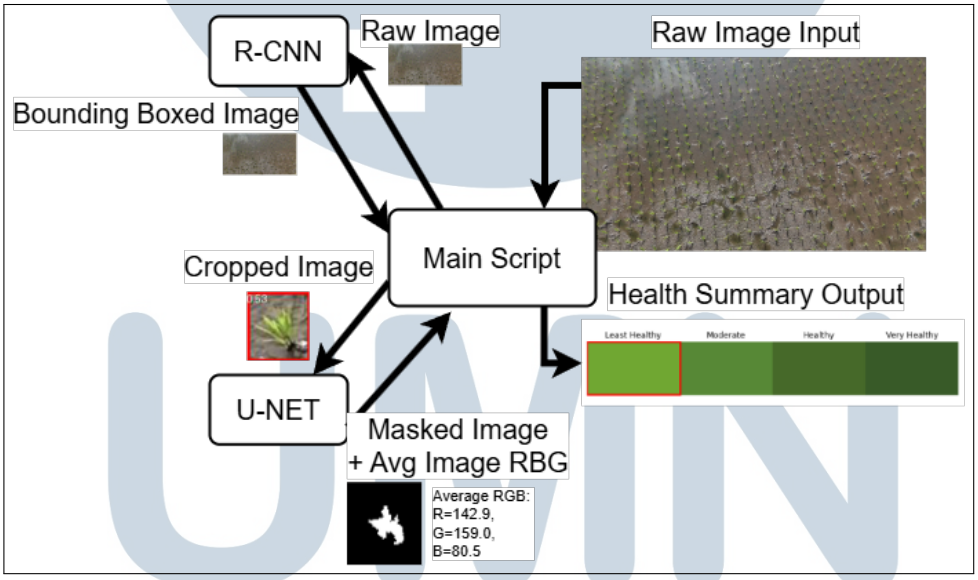
```

# IRRI LCC values
lcc_chart = [
    {"label": "Least Healthy", "rgba": (112, 167, 50, 255)},
    {"label": "Moderate",      "rgba": (87, 136, 54, 255)},
    {"label": "Healthy",       "rgba": (70, 105, 41, 255)},
    {"label": "Very Healthy",  "rgba": (56, 90, 40, 255)}
]

def get_health_label(rgb):
    def dist(color): return sum((a - b) ** 2 for a, b in zip(color, rgb))
    closest = min(lcc_chart, key=lambda item: dist(item["rgba"][:3]))
    return closest["label"]

def find_closest_index(rgb):
    def dist(color): return sum((a - b) ** 2 for a, b in zip(color, rgb))
    return min(range(len(lcc_chart)), key=lambda i: dist(lcc_chart[i]["rgba"][:3]))

if all_avg_colors:
    batch_avg_rgb = np.mean(all_avg_colors, axis=0).astype(np.uint8)
    batch_health = get_health_label(batch_avg_rgb)
    print(f"\n========================================")
    print(f"Batch Avg RGB: {batch_avg_rgb}")
    print(f"Batch Health Rating: {batch_health}")

    # plot LCC
    fig, ax = plt.subplots(figsize=(6, 2))
```

```
    bar_height = 1
    patch_width = 1

    for i, item in enumerate(lcc_chart):
        ax.add_patch(Rectangle((i * patch_width, 0), patch_width,
bar_height,
                               color=np.array(item["rgba"][:3]) /
255.0))
        ax.text(i * patch_width + 0.5, 1.05, item["label"],
                ha='center', va='bottom', fontsize=9)

    closest_idx = find_closest_index(batch_avg_rgb)
    ax.add_patch(Rectangle((closest_idx * patch_width, 0),
patch_width, bar_height,
                           edgecolor='red', facecolor='none',
linewidth=2))

    ax.text(len(lcc_chart) / 2, -0.4,
            f"Avg RGB: {batch_avg_rgb.tolist()}      {lcc_chart[
closest_idx]['label']}",
            ha='center', va='top', fontsize=11)

    ax.set_xlim(0, len(lcc_chart))
    ax.set_ylim(-0.5, 1.5)
    ax.axis('off')
    plt.tight_layout()
    plt.show()

    print(f"========================================")
    print(f"Batch Average RGB: {batch_avg_rgb.tolist()}")
    print(f"Batch Health Rating: {batch_health}")

    if batch_health in ["Least Healthy", "Moderate"]:
        print("\n    Keterangan:")
        print("Warna daun menunjukkan tanaman padi mungkin belum
optimal dalam menyerap nitrogen.")
        print("Pertimbangkan penambahan pupuk nitrogen (N) secara
bijak sesuai rekomendasi agronomi.")
    else:
        print("\n    Keterangan:")
        print("Warna daun menunjukkan tanaman padi berada dalam
kondisi sehat.")
        print("Tidak perlu penambahan pupuk nitrogen berlebih saat
```

```
        ini.")
63
64      print(f"========================================\n")
65
66  else:
67      print("No leaves detected for analysis.")
```

Kode 3.5: Klasifikasi kesehatan tanaman padi berdasarkan LCC IRRI.

Berdasarkan klasifikasi warna tanaman dari LCC IRRI, dapat ditarik kondisi kesehatan tanaman padi yang sangat prima pada warna tanaman paling hijau tua, sedangkan ketiga warna hijau lainnya yang lebih muda mengidentifikasi kurangnya asupan nitrogen pada tanaman padi tersebut. Pesan saran akan diberikan setelah proses identifikasi selesai berupa perlunya ditambahkan pupuk nitrogen atau tidak. Hal ini akan sangat membantu petani untuk menjaga kesehatan tanaman padi yang ditanam.



Gambar 3.14. Cara kerja *script* utama.