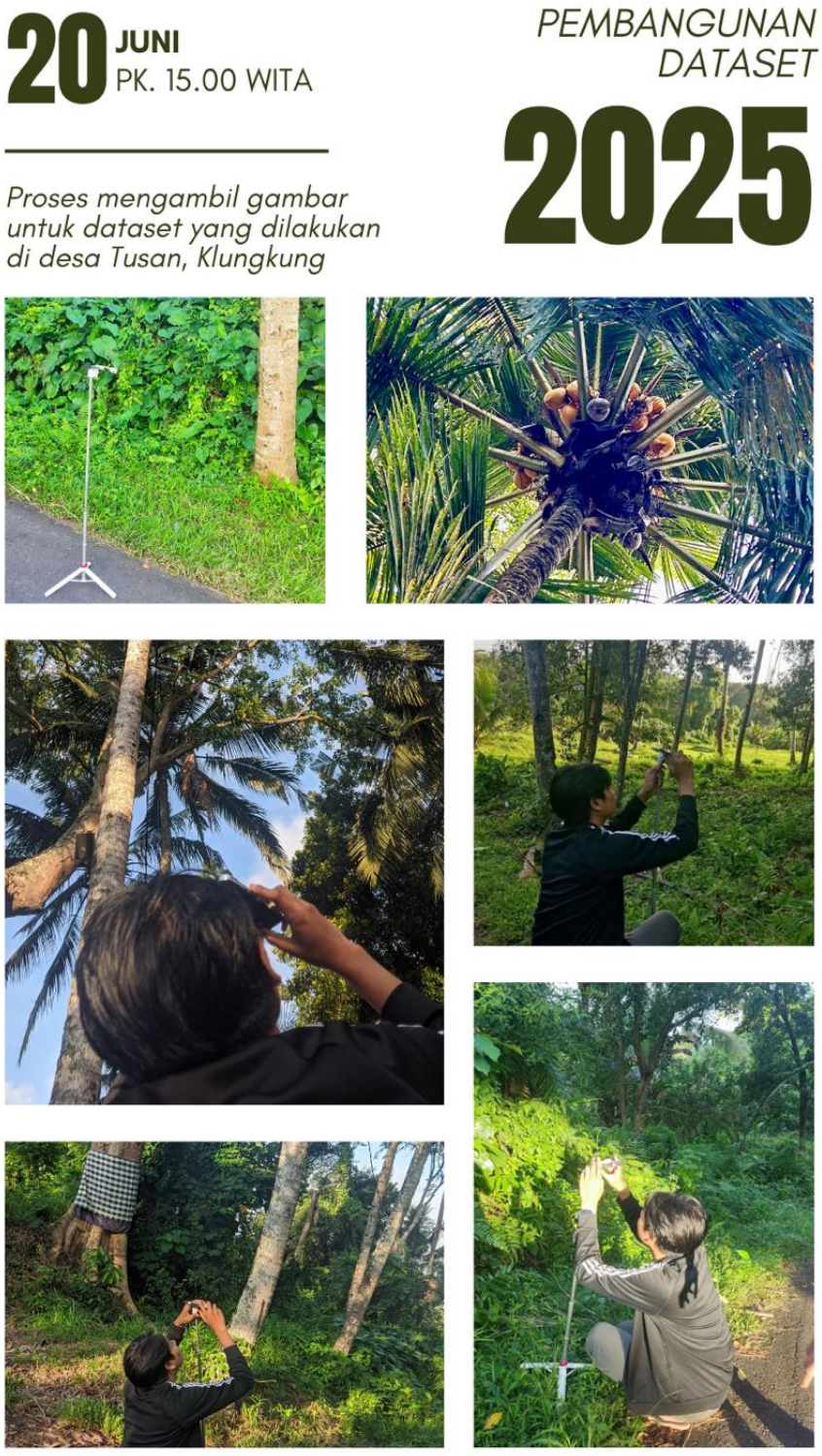
LAMPIRAN

Lampiran 1 Wawancara Terkait Majeg di Klungkung

Lampiran 2 Proses mengambil gambar dataset



Lampiran 3 Pembangunan Dataset

Lampiran 4 Stratafikasi *Code*

```python
import os
import shutil
import numpy as np
import pandas as pd
from tqdm import tqdm
from iterstrat.ml_stratifiers import MultilabelStratifiedShuffleSplit

# ===== KONFIGURASI =====
dataset_path = "image"   # folder dataset asal
images_dir  = os.path.join(dataset_path, "images")
labels_dir  = os.path.join(dataset_path, "labels")
classes_txt = os.path.join(dataset_path, "classes.txt")
output_dir  = "dataset_split_2"  # folder hasil split
train_ratio, val_ratio, test_ratio = 0.8, 0.1, 0.1  # 80/10/10

# Baca nama class
with open(classes_txt) as f:
    class_names = [c.strip() for c in f.readlines()]
num_classes = len(class_names)

# Baca semua file label
label_files = [f for f in os.listdir(labels_dir) if f.endswith(".txt")]

# Hitung jumlah box per image per class
records = []
for lf in label_files:
    counts = [0]*num_classes
    with open(os.path.join(labels_dir, lf)) as f:
        for line in f:
            cls_id = int(line.split()[0])
            counts[cls_id] += 1
    records.append([lf.replace(".txt","")] + counts)

df = pd.DataFrame(records, columns=["image"] + class_names)
X = df["image"].values
y = df[class_names].values

# ===== Split Data Stratified =====
# Train/Test
msss1 = MultilabelStratifiedShuffleSplit(n_splits=1, test_size=test_ratio, random_state=42)
train_idx, test_idx = next(msss1.split(X, y))

X_train_full, X_test = X[train_idx], X[test_idx]
y_train_full, y_test = y[train_idx], y[test_idx]

# Train/Valid
msss2 = MultilabelStratifiedShuffleSplit(n_splits=1,
        test_size=val_ratio/(train_ratio+val_ratio), random_state=42)
train_idx2, val_idx = next(msss2.split(X_train_full, y_train_full))

X_train, X_val = X_train_full[train_idx2], X_train_full[val_idx]

splits = {
    "train": X_train,
    "valid": X_val,
    "test":  X_test
}

# ===== Copy File ke Folder Baru =====
for split_name, images in splits.items():
    for sub in ["images", "labels"]:
        os.makedirs(os.path.join(output_dir, split_name, sub), exist_ok=True)

    for img_id in tqdm(images, desc=f"Copying {split_name}"):
        img_file = img_id + ".jpg"  # ganti sesuai format image jika perlu
        lbl_file = img_id + ".txt"

        src_img = os.path.join(images_dir, img_file)
        src_lbl = os.path.join(labels_dir, lbl_file)
        dst_img = os.path.join(output_dir, split_name, "images", img_file)
        dst_lbl = os.path.join(output_dir, split_name, "labels", lbl_file)

        if os.path.exists(src_img):
            shutil.copy2(src_img, dst_img)
        if os.path.exists(src_lbl):
            shutil.copy2(src_lbl, dst_lbl)

print("✅ Dataset berhasil dipisah ke folder:", output_dir)

# ===== Statistik =====
print("\n=== 📊 Statistik Dataset Split ===")
for split_name, images in splits.items():
    img_count = len(images)
    class_counts = [0]*num_classes
    total_boxes = 0

    for img_id in images:
        lbl_path = os.path.join(labels_dir, img_id + ".txt")
        with open(lbl_path) as f:
            for line in f:
                cls_id = int(line.split()[0])
                class_counts[cls_id] += 1
                total_boxes += 1

    print(f"\n[{split_name.upper()}]")
    print(f"  - Jumlah gambar      : {img_count}")
    print(f"  - Total bounding box : {total_boxes}")
    for cname, ccount in zip(class_names, class_counts):
        print(f"    • {cname:<15} : {ccount}")

```

Lampiran 5 Convert One Class

```python
import os
import shutil
import random

# Path asal
dataset_path = "image"   # folder dataset asal
images_dir  = os.path.join(dataset_path, "images")
labels_dir  = os.path.join(dataset_path, "labels")

# Path hasil (format sesuai YOLO)
output_dir  = "Dataset_3"   # ganti nama jadi "Data"
train_ratio, val_ratio, test_ratio = 0.8, 0.1, 0.1

# Buat folder output dengan struktur YOLO
for split in ["train", "valid", "test"]:
    os.makedirs(os.path.join(output_dir, split, "images"), exist_ok=True)
    os.makedirs(os.path.join(output_dir, split, "labels"), exist_ok=True)

# Baca semua file gambar (asumsi jpg/png/jpeg)
all_images = [f for f in os.listdir(images_dir) if f.endswith(('.jpg', '.png', '.jpeg'))]
random.shuffle(all_images)

# Hitung jumlah untuk tiap split
n_total = len(all_images)
n_train = int(n_total * train_ratio)
n_val   = int(n_total * val_ratio)
n_test  = n_total - n_train - n_val

splits = {
    "train": all_images[:n_train],
    "valid": all_images[n_train:n_train+n_val],
    "test": all_images[n_train+n_val:]
}

# Fungsi untuk ubah label: semua class jadi 0 (kelapa)
def convert_label(label_path, out_path):
    new_lines = []
    with open(label_path, "r") as f:
        for line in f:
            parts = line.strip().split()
            if len(parts) >= 5:
                parts[0] = "0"  # ubah class id ke 0
                new_lines.append(" ".join(parts) + "\n")
    with open(out_path, "w") as f:
        f.writelines(new_lines)

# Copy file sesuai split
for split, images in splits.items():
    for img_file in images:
        # copy image
        src_img = os.path.join(images_dir, img_file)
        dst_img = os.path.join(output_dir, split, "images", img_file)
        shutil.copy(src_img, dst_img)

        # copy & convert label
        label_file = os.path.splitext(img_file)[0] + ".txt"
        src_label = os.path.join(labels_dir, label_file)
        dst_label = os.path.join(output_dir, split, "labels", label_file)

        if os.path.exists(src_label):
            convert_label(src_label, dst_label)

# Buat classes.txt baru (hanya 1 class: kelapa)
with open(os.path.join(output_dir, "classes.txt"), "w") as f:
    f.write("kelapa\n")

print(f"Dataset berhasil dibuat di folder: {output_dir}")
print(f"Total gambar: {n_total} (train={n_train}, valid={n_val}, test={n_test})")

```

Lampiran 6 Crop Image Base On Class

```python
import os
import cv2

# === KONFIGURASI ===
dataset_path = "image"          # folder utama dataset Roboflow
images_path = os.path.join(dataset_path, "images")
labels_path = os.path.join(dataset_path, "labels")
output_path = "cropped_dataset"   # hasil crop per class
classes_file = os.path.join(dataset_path, "classes.txt")

# Baca nama kelas
with open(classes_file, "r") as f:
    class_names = [line.strip() for line in f.readlines()]

os.makedirs(output_path, exist_ok=True)

# Loop semua label
for label_file in os.listdir(labels_path):
    if not label_file.endswith(".txt"):
        continue

    image_name = os.path.splitext(label_file)[0] + ".jpg"
    image_path = os.path.join(images_path, image_name)

    if not os.path.exists(image_path):
        print(f"[WARNING] Gambar {image_name} tidak ditemukan.")
        continue

    # Load image
    img = cv2.imread(image_path)
    h, w, _ = img.shape

    # Baca setiap baris label (bounding box)
    with open(os.path.join(labels_path, label_file), "r") as f:
        for i, line in enumerate(f):
            cls_id, x_c, y_c, bw, bh = map(float, line.strip().split())

            # Konversi dari rasio YOLO ke pixel
            x_c, y_c, bw, bh = x_c * w, y_c * h, bw * w, bh * h
            x1 = int(x_c - bw / 2)
            y1 = int(y_c - bh / 2)
            x2 = int(x_c + bw / 2)
            y2 = int(y_c + bh / 2)

            # Pastikan tidak keluar batas
            x1, y1 = max(0, x1), max(0, y1)
            x2, y2 = min(w, x2), min(h, y2)

            # Crop
            crop_img = img[y1:y2, x1:x2]

            # Buat folder sesuai class
            class_name = class_names[int(cls_id)]
            save_dir = os.path.join(output_path, class_name)
            os.makedirs(save_dir, exist_ok=True)

            # Simpan crop
            save_path = os.path.join(save_dir, f"{os.path.splitext(image_name)[0]}_{i}.jpg")
            cv2.imwrite(save_path, crop_img)

            print(f"[SAVED] {save_path}")

```

Lampiran 7 Code Counting

```python
@app.post("/analyze")
async def analyze_image(
    file: UploadFile = File(...),
    model: str = Form(DEFAULT_MODEL),
    conf: Optional[float] = Form(0.25),
    iou: Optional[float] = Form(0.45),
    max_det: Optional[int] = Form(100),
):
    """Return coconut counts and confidence for your frontend.

    Response shape:
    {
      "young_coconuts": int,
      "mature_coconuts": int,
      "confidence": float
    }
    """
    try:
        mdl = load_model(model)
    except Exception as e:
        raise HTTPException(status_code=400, detail=str(e))

    content = await file.read()
    try:
        pil = Image.open(io.BytesIO(content)).convert("RGB")
    except Exception as e:
        raise HTTPException(status_code=400, detail=f"Invalid image: {e}")

    try:
        results = mdl.predict(source=np.array(pil), conf=conf, iou=iou, max_det=max_det, verbose=False)
    except Exception as e:
        raise HTTPException(status_code=500, detail=f"Predict error: {e}")

    res = results[0]
    names = getattr(mdl, "names", {}) or {}

    young = 0
    mature = 0
    scores: List[float] = []

    if hasattr(res, "boxes") and res.boxes is not None:
        try:
            confs = res.boxes.conf.cpu().numpy() if hasattr(res.boxes.conf, "cpu") else np.array(res.boxes.conf)
            clss = res.boxes.cls.cpu().numpy() if hasattr(res.boxes.cls, "cpu") else np.array(res.boxes.cls)
            for s, c in zip(confs, clss):
                scores.append(float(s))
                cls_name = names.get(int(c), str(int(c))).lower()
                if cls_name in ("kelapa_muda", "kelapa muda", "young_coconut", "youngcoconut"):
                    young += 1
                elif cls_name in ("kelapa_tua", "kelapa tua", "mature_coconut", "maturecoconut"):
                    mature += 1
                else:
                    # unknown class - ignore or handle as needed
                    pass
        except Exception:
            # if extraction fails, return zeros
            young = 0
            mature = 0
            scores = []

    confidence = round(max(scores) * 100, 2) if scores else 0.0

    return JSONResponse({
        "young_coconuts": young,
        "mature_coconuts": mature,
        "confidence": confidence,
    })


if __name__ == "__main__":
    import uvicorn
    uvicorn.run("main:app", host="0.0.0.0", port=8000, reload=True)

```

Lampiran 8 Tabel Detail Arsitektur YOLOv11n

Block	Layer Type	Output Size	Kernel Size	Stride	Filters	Description
Input	-	640×640×3	-	-	-	Gambar input RGB
Backbone	Conv	320×320×32	3×3	2	32	Konvolusi awal dengan stride 2
Backbone	C2f Block	320×320×64	-	-	64	Cross-stage partial fused (CSP varian)
Backbone	Conv	160×160×128	3×3	2	128	Downsampling
Backbone	C3k Block	160×160×128	-	-	128	Residual block dengan bottleneck
Backbone	Conv	80×80×256	3×3	2	256	Downsampling
Backbone	SPPF	80×80×256	-	-	256	Spatial Pyramid Pooling - Fast
Neck	C2PSA	80×80×256	-	-	256	Partial Self-Attention untuk fusi fitur
Neck	Upsample	160×160×128	-	-	128	Upsampling fitur resolusi menengah
Head	Detect(P3)	80×80×	-	-	-	Prediksi objek skala kecil
Head	Detect(P4)	40×40×	-	-	-	Prediksi objek skala sedang
Head	Detect(P5)	20×20×	-	-	-	Prediksi objek skala besar
Head	DFL	-	-	-	-	Distribution Focal Loss untuk regresi box

Lampiran 9 Tabel Detail Arsitektur YOLOv11s

Block	Layer Type	Output Size	Kernel Size	Stride	Filters	Description
Input	-	640×640×3	-	-	-	Gambar input 640×640 dengan 3 channel RGB
Backbone	Conv	320×320×64	3×3	2	64	Konvolusi awal dengan stride 2
Backbone	C3k2	320×320×128	-	-	128	Blok C3k2 pertama
Backbone	Conv	160×160×256	3×3	2	256	Konvolusi dengan stride 2
Backbone	C3k2	160×160×256	-	-	256	Blok C3k2 kedua
Backbone	Conv	80×80×512	3×3	2	512	Konvolusi dengan stride 2
Backbone	C3k2	80×80×512	-	-	512	Blok C3k2 ketiga
Backbone	Conv	40×40×1024	3×3	2	1024	Konvolusi dengan stride 2
Backbone	SPPF	40×40×1024	-	-	1024	Spatial Pyramid Pooling Fast
Backbone	C2PSA	40×40×512	-	-	512	Attention block (PSA)
Neck	Upsample	80×80×512	-	-	-	Upsampling
Neck	Concat	80×80×768	-	-	-	Concatenate feature maps
Neck	C3k2	80×80×256	-	-	256	Blok C3k2
Neck	Upsample	160×160×256	-	-	-	Upsampling
Neck	Concat	160×160×512	-	-	-	Concatenate feature maps
Neck	C3k2	160×160×128	-	-	128	Blok C3k2
Neck	Conv + Concat	80×80×256	3×3	2	256	Downsampling + merge
Neck	C3k2	80×80×256	-	-	256	Blok C3k2
Neck	Conv + Concat	40×40×512	3×3	2	512	Downsampling + merge
Neck	C3k2	40×40×512	-	-	512	Blok C3k2
Head	Detect	-	-	-	-	Layer deteksi (prediksi bounding box & klasifikasi)

Lampiran 10 Tabel Detail Arsitektur YOLOv11m

Block	Layer Type	Output Size	Kernel Size	Stride	Filters	Description
Input	-	640×640×3	-	-	-	Gambar input RGB
Backbone	Conv	320×320×64	3×3	2	64	Konvolusi awal dengan stride 2
Backbone	C2f Block	320×320×128	-	-	128	Cross-stage partial fused (CSP varian)
Backbone	Conv	160×160×256	3×3	2	256	Downsampling
Backbone	C3k2 Block	160×160×256	-	-	256	Residual block dengan bottleneck (2x)
Backbone	Conv	80×80×512	3×3	2	512	Downsampling
Backbone	C3k2 Block	80×80×512	-	-	512	Residual block dengan bottleneck (2x)
Backbone	Conv	40×40×512	3×3	2	512	Downsampling
Backbone	C3k2 Block	40×40×512	-	-	512	Residual block dengan bottleneck (2x)
Backbone	Conv	20×20×512	3×3	2	512	Downsampling
Backbone	C3k2 Block	20×20×512	-	-	512	Residual block dengan bottleneck (2x)
Backbone	SPPF	20×20×512	-	-	512	Spatial Pyramid Pooling - Fast
Neck	C2PSA	20×20×512	-	-	512	Position-Sensitive Attention untuk fusi fitur
Neck	Upsample	40×40×512	-	×2	-	Upsampling fitur resolusi tinggi
Neck	Concat	40×40×1024	-	-	-	Penggabungan fitur backbone dan neck
Neck	C3k2 Block	40×40×512	-	-	512	Residual block setelah concat
Neck	Upsample	80×80×512	-	×2	-	Upsampling fitur resolusi menengah
Neck	Concat	80×80×768	-	-	-	Penggabungan fitur backbone dan neck
Neck	C3k2 Block	80×80×256	-	-	256	Residual block setelah concat
Neck	Conv	40×40×512	3×3	2	512	Downsampling
Neck	Concat	40×40×1024	-	-	-	PANet fusion

Neck	C3k2 Block	40×40×512	-	-	512	Residual block setelah concat
Neck	Conv	20×20×512	3×3	2	512	Downsampling
Neck	Concat	20×20×1024	-	-	-	PANet fusion
Neck	C3k2 Block	20×20×512	-	-	512	Residual block setelah concat
Head	Detect(P3)	80×80	-	-	-	Prediksi objek skala kecil
Head	Detect(P4)	40×40	-	-	-	Prediksi objek skala sedang
Head	Detect(P5)	20×20	-	-	-	Prediksi objek skala besar
Head	DFL	-	-	-	-	Distribution Focal Loss untuk regresi box