MIFOCAT code

1 File: load_dataset_v1.py

Bahasan:

Berikut **penjelasan isi kode secara lengkap dan dipilah berdasarkan blok fungsinya**, dari atas ke bawah. Saya jelaskan **tujuan, alur data, dan alasan teknisnya**, khusus untuk konteks **preprocessing citra medis 3D (ACDC dataset – MRI jantung)**.

(Sumber kode:)

1️⃣ Header & Import Library

```
import numpy as np
import nibabel as nib
import glob
from tensorflow.keras.utils import to_categorical
import matplotlib.pyplot as plt
from tifffile import imsave
from scipy.ndimage import zoom
from sklearn.preprocessing import MinMaxScaler
```

Fungsi blok ini

- **NumPy** → manipulasi array 3D
- **Nibabel (nib)** → membaca file medis .nii.gz (MRI)
- **glob** → membaca banyak file dengan pola tertentu
- **to_categorical** → one-hot encoding untuk mask segmentasi
- **matplotlib** → visualisasi slice 2D
- **scipy.ndimage.zoom** → resize volume 3D
- **MinMaxScaler** → normalisasi intensitas citra (0–1)

```
scaler = MinMaxScaler()
```

➡️ Digunakan global untuk normalisasi intensitas citra MRI.

2️⃣ Fungsi resize_volume()

```
def resize_volume(data, new_shape=(128, 128, 128)):
```

Tujuan

Mengubah ukuran **volume 3D MRI** menjadi ukuran standar (128,128,128).

Cara kerja

```
scale_factors = [n / float(o) for n, o in zip(new_shape, data.shape)]
resized_data = zoom(data, zoom=scale_factors, mode='nearest')
```

- Menghitung **rasio skala tiap dimensi**
- zoom() melakukan interpolasi
- mode='nearest':
  - Penting untuk **mask segmentasi**
  - Mencegah nilai label baru muncul (misalnya 1.3, 2.7)

📌 **Fungsi ini sangat penting untuk menyamakan ukuran input model deep learning**

3️⃣ Load & Normalisasi Data Contoh (Training – Patient 002)

Path dataset

```
TRAIN_DATASET_PATH = '.../trainingACDC17/'
```

Load citra End-Diastole (ED)

test_image_diastole = nib.load(...frame01.nii.gz).get_fdata()

- Membaca MRI 3D
- Bentuk data: (H, W, Slice)

Normalisasi intensitas

```
test_image_diastole = scaler.fit_transform(
   test_image_diastole.reshape(-1, test_image_diastole.shape[-1])
).reshape(test_image_diastole.shape)
```

📌 **Kenapa reshape?**

- MinMaxScaler hanya menerima data 2D
- Jadi volume 3D → diratakan → dinormalisasi → dibentuk kembali

---

4️⃣ Load Mask Segmentasi

```
test_mask_diastole = nib.load(..._gt.nii.gz).get_fdata()
test_mask_diastole = test_mask_diastole.astype(np.uint8)
```

Isi mask

```
np.unique(test_mask_diastole)
```

Biasanya:

- 0 → background
- 1–4 → struktur jantung (LV, RV, Myocardium, dll)

5️⃣ Load Data End-Systole (ES)

Blok ini **identik** dengan ED:

- Load citra frame12
- Normalisasi
- Load mask
- Cek nilai unik

📌 ED & ES diproses **terpisah** karena fase jantung berbeda.

6️⃣ Visualisasi Slice 2D (Sebelum Resize)

```
n_slice = 2
plt.imshow(test_image_diastole[:,:,n_slice])
```

Tujuan

- Debugging
- Memastikan:
  - Citra dan mask sejajar
  - Segmentasi benar
  - Tidak terbalik / rusak

7️⃣ Resize Volume ke 128^3

```
resized_image_diastole = resize_volume(test_image_diastole)
resized_mask_diastole = resize_volume(test_mask_diastole)
```

📌 **Standarisasi ukuran input untuk CNN 3D**

8️⃣ Menambahkan Channel (4D Tensor)

```
resized_diastole_4d = np.expand_dims(resized_image_diastole, axis=3)
```

Hasil:

```
(128, 128, 128, 1)
```

📌 **Format wajib untuk CNN 3D (TensorFlow / Keras)**

9️⃣ Visualisasi Setelah Resize

plt.imshow(resized_diastole_4d[:,:,n_slice,0])

Tujuan:

- Memastikan resize tidak merusak anatomi
- Mask masih sesuai citra

🔟 Preprocessing Dataset Testing (Batch Processing)

Ambil semua file ED testing

list_img_ed = sorted(glob.glob(TEST_DATASET_PATH + '*/patient*_frame01.nii.gz'))

list_mask_ed = sorted(glob.glob(TEST_DATASET_PATH + '*/patient*_frame01_gt.nii.gz'))

📌 Otomatis membaca **semua pasien testing**

1️⃣1️⃣ Loop Proses Semua Pasien ED

for img in range(len(list_img_ed)):

Isi loop:

1. Load citra
2. Normalisasi
3. Load mask
4. Resize
5. Tambah channel
6. Validasi isi mask
7. One-hot encoding
8. Simpan .npy

1️⃣2️⃣ Filtering Mask Kosong

val, counts = np.unique(resized_mask_ed, return_counts=True)

if (1 - (counts[0]/counts.sum())) > 0.01:

Artinya

- Hitung persentase **non-background**
- Jika objek jantung < 1% → **tidak disimpan**

📌 Ini mencegah:

- Data kosong
- Noise
- Slice tidak berguna

1️⃣3️⃣ One-Hot Encoding Mask

resized_mask_ed = to_categorical(resized_mask_ed, num_classes=5)

Hasil:

(128, 128, 128, 5)

📌 **Format target untuk multi-class segmentation**

1️⃣4️⃣ Menyimpan Data

np.save('.../images/image_XXX_ed.npy', resized_img_ed_1)

np.save('.../masks/mask_XXX_ed.npy', resized_mask_ed)

Format akhir:

- **Image** → (128,128,128,1)

- **Mask** → (128,128,128,5)

Siap dipakai untuk:

- 3D U-Net
- V-Net
- Attention UNet 3D

1️⃣5️⃣ **Blok ES (Dikomentari)**

Bagian terakhir **sama persis** dengan ED:

- Hanya beda:
  - Frame bukan frame01
  - Disimpan sebagai ES

📌 Dikomentari kemungkinan:

- Fokus hanya testing ED
- Menghemat waktu proses

✅ **Ringkasan Fungsi Utama Kode**

Blok	Fungsi
Import	Library medis & ML
resize_volume	Standarisasi volume 3D
Normalisasi	Samakan intensitas MRI
Visualisasi	Debug & validasi
Resize + Channel	Format input CNN
Filtering	Buang data kosong
One-hot mask	Target segmentasi
Save .npy	Dataset siap training

Kalau kamu mau, aku bisa:

- 🔍 Jelaskan **alur data ke model 3D U-Net**
- 🧠 Bedah **kenapa pakai MinMaxScaler**
- 📐 Bandingkan resize vs crop
- 🧪 Optimasi pipeline biar lebih cepat

Tinggal bilang mau fokus ke bagian mana 👍

2 File: load_dataset.py

Bahasan:

Berikut penjelasan **lengkap** untuk file **load_dataset.py** (yang baru kamu upload) dan **perbedaan utamanya** dibanding file **sebelumnya load_dataset_v1.py**. Saya pilah berdasarkan blok fungsinya dan alur prosesnya.

Sumber kode: dan

A) Penjelasan load_dataset.py (file baru) per blok fungsi

1) Import & dependensi utama

```
import SimpleITK as sitk
import os
from multiprocessing import pool
import pickle5 as pickle
import numpy as np
```

from skimage.transform import resize

Peran:

- **SimpleITK**: baca citra medis .nii.gz + akses metadata penting (spacing/voxel size).
- **multiprocessing.pool**: proses banyak pasien paralel (lebih cepat).
- **pickle**: simpan & load metadata pasien (patient_info.pkl).
- **skimage.resize**: resizing volume 3D.

---

2) resize_image(image, old_spacing, new_spacing, order=3)

```
def resize_image(image, old_spacing, new_spacing, order=3):
  new_shape = (int(np.round(old_spacing[0]/new_spacing[0]*float(image.shape[0]))),
        int(np.round(old_spacing[1]/new_spacing[1]*float(image.shape[1]))),
        int(np.round(old_spacing[2]/new_spacing[2]*float(image.shape[2]))))
  return resize(image, new_shape, order=order, mode='edge')
```

Fungsi blok ini:

- Mengubah ukuran volume **berdasarkan perubahan spacing**, bukan target shape tetap.
- new_shape dihitung dari rasio **old_spacing / new_spacing** → output jadi *secara fisik lebih konsisten*.
- order = derajat interpolasi:
  - citra (continuous): biasanya 3 (cubic) atau 1 (linear)
  - mask (label): harus "nearest/orde 0/1" biar label tidak menjadi pecahan
- mode='edge': padding menggunakan nilai tepi (mengurangi artefak di batas).

---

3) Mapping label patologi

```
str_to_ind = {'DCM':0, 'HCM':1, 'MINF':2, 'NOR':3, 'RV':4}
ind_to_str = {}
for k in str_to_ind.keys():
  ind_to_str[str_to_ind[k]] = k
```

Tujuan:

- Mengubah label string patologi (mis. DCM, NOR) menjadi indeks numerik, dan sebaliknya.
- Ini untuk klasifikasi/metadata dataset, bukan segmentasi mask.

---

4) Viewer (opsional) view_patient_raw_data(patient, ...)

```
def view_patient_raw_data(patient, width=400, height=400):
  import batchviewer
  a = []
  a.append(patient['ed_data'][None])
  a.append(patient['ed_gt'][None])
  a.append(patient['es_data'][None])
  a.append(patient['es_gt'][None])
  batchviewer.view_batch(np.vstack(a), width, height)
```

Fungsi:

- Menampilkan 4 volume: **ED image, ED GT, ES image, ES GT**.
- Butuh library eksternal batchviewer (tidak diimport global).
- Ini biasanya hanya untuk inspeksi visual.

---

5) convert_to_one_hot(seg)

```
def convert_to_one_hot(seg):
  vals = np.unique(seg)
```

```
    res = np.zeros([len(vals)] + list(seg.shape), seg.dtype)
    for c in range(len(vals)):
        res[c][seg == c] = 1
    return res
```

Tujuan:

- Mengubah mask label integer → tumpukan channel one-hot berbasis nilai unik yang ada.
- Contoh: jika mask punya label {0,1,2,3}, maka output channel = 4.

Catatan: ini *one-hot internal* untuk membantu resizing mask dengan aman.

---

6) Fungsi inti preprocessing: preprocess_image(itk_image, is_seg=False, spacing_target=(1,0.5,0.5), keep_z_spacing=False)

Bagian ini yang "paling penting" di file ini.

a) Ambil spacing & ubah citra ke numpy

```
spacing = np.array(itk_image.GetSpacing())[[2, 1, 0]]
image = sitk.GetArrayFromImage(itk_image).astype(float)
```

- GetSpacing() dari SimpleITK biasanya urutan (x,y,z), lalu dibalik jadi (z,y,x) agar selaras dengan array hasil GetArrayFromImage().

b) Opsi mempertahankan spacing Z

```
if keep_z_spacing:
    spacing_target = list(spacing_target)
    spacing_target[0] = spacing[0]
```

- Jika keep_z_spacing=True, maka target spacing pada dimensi Z **tidak diubah** (sering dipakai karena slice thickness Z bisa sangat berbeda).

c) Preprocess citra (bukan mask)

```
if not is_seg:
    order_img = 3
    if not keep_z_spacing:
        order_img = 1
    image = resize_image(image, spacing, spacing_target, order=order_img).astype(np.float32)
    image -= image.mean()
    image /= image.std()
```

- Resize berdasarkan spacing (fisik).
- Normalisasi: **zero-mean, unit-std** (standardization).

d) Preprocess mask (segmentation)

```
else:
    tmp = convert_to_one_hot(image)
    vals = np.unique(image)
    results = []
    for i in range(len(tmp)):
        results.append(resize_image(tmp[i].astype(float), spacing, spacing_target, 1)[None])
    image = vals[np.vstack(results).argmax(0)]
```

Strateginya:

1. Mask → one-hot per kelas (tmp)
2. Resize tiap channel dengan interpolasi orde 1
3. Ambil argmax untuk mengembalikan label integer yang dominan per voxel

📌 Ini cara umum untuk **menghindari label jadi pecahan** saat resizing.

---

7) load_dataset(ids=range(101), root_dir=...)

```
def load_dataset(ids=range(101), root_dir=".../preprocessed dataset 3d/"):
  with open(os.path.join(root_dir, "patient_info.pkl"), 'r') as f:
    patient_info = cPickle.load(f)
  ...
```

Tujuan:

- Load dataset yang sudah dipreproses (file .npy per pasien: pat_XXX.npy)
- Isi data[i] memuat:
  - metadata: height, weight, pathology
  - volume: ed_data, ed_gt, es_data, es_gt

⚠️ Catatan penting:

- Di kode ini ada inkonsistensi: pickle5 as pickle diimport, tapi di sini memakai cPickle (tidak diimport). Secara praktik, baris ini bisa error kecuali environment kamu punya cPickle alias tertentu. (Ini bug kecil di bagian loader.)

---

8) process_patient(args) — proses satu pasien (dipanggil paralel)

Tugasnya:

1. Bentuk path pasien patient%03d
2. Baca file ED & ES sesuai frame dari patient_info
3. Preprocess semua (citra+mask)
4. Stack menjadi array:
   - index 0: ED image
   - index 1: ED seg
   - index 2: ES image
   - index 3: ES seg
5. Simpan pat_XXX.npy

Bagian kunci:

```
images[k] = preprocess_image(images[k], is_seg=(k == "ed_seg" or k == "es_seg"),
                spacing_target=(10, 1.25, 1.25), keep_z_spacing=keep_z_spc)
```

- Target spacing yang dipakai saat run: **(10, 1.25, 1.25)** (Z jauh lebih besar).

---

9) generate_patient_info(folder)

```
nfo = np.loadtxt(os.path.join(fldr, "Info.cfg"), dtype=str, delimiter=': ')
patient_info[id]['ed'] = int(nfo[0, 1])
patient_info[id]['es'] = int(nfo[1, 1])
...
```

Tujuan:

- Baca Info.cfg tiap pasien, ambil:
  - frame ED & ES
  - height, weight
  - pathology

Output: dictionary patient_info.

---

10) run_preprocessing(folder, folder_out, keep_z_spacing=True)

- Generate patient_info
- Simpan patient_info.pkl
- Multiprocessing 8 worker:

```
p = pool.Pool(8)
```

p.map(process_patient, zip(ids, [patient_info]*101, [folder]*101, [folder_out]*101, [keep_z_spacing]*101))
Tujuan: preprocessing 101 pasien training lebih cepat.

---

11) main() + entry point

```
if __name__ == "__main__":
   main()
```

Menjalankan preprocessing dari folder training ke folder output.

B) Perbedaan utama dengan load_dataset_v1.py (file sebelumnya)

Berikut perbedaannya yang paling "besar" dan berdampak ke hasil dataset:
Sumber pembanding:

1) Library pembaca NIfTI: SimpleITK vs NiBabel

- **File baru (load_dataset.py)**: pakai **SimpleITK**
  - kuat di metadata (spacing, orientasi, resampling berbasis fisik)
- **File lama (load_dataset_v1.py)**: pakai **nibabel**
  - baca data mudah, tapi pipeline kamu lebih "array-based" (lebih manual)

2) Cara resize: berbasis spacing vs berbasis shape tetap

- **File baru**: resize ditentukan oleh **target spacing** (10,1.25,1.25) → output shape bisa beda-beda tergantung data asli.
- **File lama**: resize selalu ke **new_shape = (128,128,128)** → shape output selalu sama.

Dampak:

- File baru lebih "medically consistent" (fisik voxel seragam).
- File lama lebih "DL friendly" karena semua input fixed-size.

3) Normalisasi intensitas citra

- **File baru**: standardization → image = (image - mean) / std
- **File lama**: MinMaxScaler → skala 0–1 per volume

4) Penanganan mask saat resize

- **File baru**: one-hot → resize per channel → argmax (lebih aman untuk label)
- **File lama**: scipy.ndimage.zoom(mode='nearest') dipakai langsung untuk mask (juga aman, tapi pendekatannya berbeda)

5) Struktur output data yang disimpan

- **File baru**: simpan **1 file per pasien**: pat_XXX.npy berisi 4 volume [ed, ed_seg, es, es_seg]
- **File lama**: simpan **terpisah** per fase:
  - ED image/mask disimpan ke folder ED
  - (ES ada tapi dikomentari / belum aktif)
  - Nama file menggunakan offset img+101 untuk testing ED

6) Cakupan dataset: training vs testing

- **File baru**: fokus preprocessing **training** (ids range(101)), mengambil ED/ES berdasarkan Info.cfg
- **File lama**: contoh tampilkan patient002, lalu loop memproses **testing ED** (testingACDC17_baru) dan menyimpan jika mask tidak kosong

7) Paralelisasi

- **File baru**: multiprocessing Pool(8) → lebih cepat di banyak pasien
- **File lama**: loop biasa (single process)

8) Filtering data berdasarkan isi mask

- **File lama**: ada filter non-background > 1% sebelum save
- **File baru**: tidak ada filter semacam itu (semua pasien yang lengkap akan disimpan)

C) Ringkas super cepat (biar gampang diingat)

- load_dataset.py = **pipeline preprocessing "serius" berbasis spacing** + metadata + multiprocessing + simpan per pasien 4 volume.
- load_dataset_v1.py = **pipeline sederhana fixed-shape (128³)** + normalisasi 0–1 + fokus generate .npy khusus ED testing dengan filter isi mask.

Kalau kamu mau, aku bisa bantu juga:

- Menentukan mana yang **lebih cocok** untuk arsitektur yang kamu pakai (3D U-Net fixed input vs patch-based).
- Menyamakan output kedua pipeline supaya format dataset konsisten (mis. semua jadi (128,128,128,1) + one-hot (128,128,128,5)).

3 File : cek kelas groundtruth.py

Bahasan:

Berikut penjelasan **lengkap** isi file **cek kelas groundtruth.py** dan saya pilah berdasarkan **blok fungsinya** (urut sesuai kode).
Sumber kode:

---

1) Header file & metadata

```
# -*- coding: utf-8 -*-
"""
Created on Sun Mar 17 14:42:20 2024
@author: ramad
"""
```

Fungsi blok ini:

- Menandai encoding file UTF-8.
- Komentar metadata (tanggal dibuat & author). Tidak memengaruhi eksekusi program.

---

2) Import library

```
import cv2 as cv
import numpy as np
from matplotlib import pyplot as plt
```

Fungsi masing-masing:

- **cv2 (OpenCV)**: membaca & menulis gambar (imread, imwrite)
- **numpy**: operasi array (mencari nilai unik, masking)
- **matplotlib**: menampilkan gambar dalam plot

---

3) Menentukan path file mask (groundtruth) & membaca gambar

```
mask_path = "D:/.../Pasien101_2_ed_gt.png"
mask = cv.imread(mask_path, 0)
```

Fungsi blok ini:

- mask_path: lokasi file groundtruth (format PNG).
- cv.imread(mask_path, 0):
  - mode 0 artinya **grayscale** (hasilnya array 2D).
  - mask berisi nilai intensitas piksel 0–255.

📌 Catatan penting:

- Karena ini **mask segmentasi**, nilai pixel biasanya bukan “warna” biasa, tapi **kode kelas** (misal 0=background, 85=RV, dst tergantung format dataset PNG yang dibuat).

---

4) Mengecek nilai unik pada mask

```
unique_values = np.unique(mask)
print("Unique values in the mask:", unique_values)
```

Fungsi blok ini:

- np.unique(mask) mengambil semua nilai pixel yang muncul pada mask.
- Output ini dipakai untuk:
  - memastikan mask memang berisi beberapa kelas,
  - mengetahui kode label yang dipakai dalam file PNG tersebut.

Contoh kemungkinan output:

- [0, 85, 170, 255] atau [0,1,2,3] tergantung encoding mask.

---

5) Membuat mask biner per kelas (RV, Myo, LV)

```
kelas1 = np.where(mask == unique_values[1], 255, 0).astype(np.uint8) if len(unique_values) > 1 else np.zeros_like(mask)
kelas2 = np.where(mask == unique_values[2], 255, 0).astype(np.uint8) if len(unique_values) > 2 else np.zeros_like(mask)
kelas3 = np.where(mask == unique_values[3], 255, 0).astype(np.uint8) if len(unique_values) > 3 else np.zeros_like(mask)
```

Apa yang dilakukan?

- Membuat **tiga gambar biner**:
  - kelas1: piksel yang sama dengan unique_values[1] dibuat putih (255), lainnya hitam (0)
  - kelas2: untuk unique_values[2]
  - kelas3: untuk unique_values[3]

Kenapa pakai unique_values[1], [2], [3]?

- Karena unique_values[0] hampir pasti **background** (biasanya 0).
- Kelas objek dimulai dari indeks 1.

Kenapa ada if len(unique_values) > ...?

- Untuk menghindari error jika mask hanya punya 1–2 kelas.
- Kalau kelasnya tidak ada, dibuat gambar kosong np.zeros_like(mask).

📌 Catatan penting (potensi masalah):

- Kode ini **mengasumsikan urutan nilai unik** merepresentasikan urutan kelas (RV, Myo, LV). Kalau ternyata nilai uniknya tidak konsisten / urutannya berbeda dari definisi kelas, maka mapping RV/Myo/LV bisa tertukar.

---

6) Menyimpan hasil tiap kelas menjadi file PNG baru

```
cv.imwrite("D:/.../gt_rv_101_2.png", kelas1)
cv.imwrite("D:/.../gt_myo_101_2.png", kelas2)
cv.imwrite("D:/.../gt_lv_101_2.png", kelas3)
```

Fungsi blok ini:

- Menulis hasil segmentasi per kelas sebagai **mask biner**:
  - gt_rv_101_2.png → kelas1
  - gt_myo_101_2.png → kelas2
  - gt_lv_101_2.png → kelas3

Tujuannya biasanya untuk:

- inspeksi visual per kelas,
- dipakai sebagai dataset multi-output,
- atau debugging preprocessing.

---

7) Visualisasi (plot) groundtruth dan masing-masing kelas

```
plt.figure(figsize=(12, 8))

plt.subplot(231)
plt.title('Groundtruth')
plt.imshow(mask, cmap='gray')

plt.subplot(232)
plt.title('RV')
plt.imshow(kelas1, cmap='gray')

plt.subplot(233)
plt.title('Myo')
plt.imshow(kelas2, cmap='gray')

plt.subplot(234)
plt.title('LV')
plt.imshow(kelas3, cmap='gray')

plt.show()
```

Fungsi blok ini:

- Membuat figure ukuran besar agar jelas.
- Menampilkan:
    1. Mask asli (groundtruth)
    2. Mask biner RV
    3. Mask biner Myo
    4. Mask biner LV

📌 Ada subplot yang dikomentari:

```
# plt.title('Citra Asli')
# plt.imshow(mask, cmap='gray')
```

Ini awalnya mungkin ingin menampilkan citra asli (image), tapi yang dipakai tetap mask, jadi dikomentari.

Ringkasan tujuan keseluruhan program

Program ini dipakai untuk:

1. Membaca **groundtruth PNG** (mask segmentasi).
2. Mengecek **kode kelas** yang ada di mask (np.unique).
3. Memisahkan mask menjadi **3 mask biner** (per kelas).
4. Menyimpan masing-masing kelas sebagai file PNG.
5. Menampilkan hasil pemisahan untuk verifikasi visual.

Kalau kamu ingin, aku bisa bantu rapikan agar mapping kelas **tidak bergantung urutan unique_values**, misalnya kamu bisa tentukan eksplisit:

- background = 0

- RV = 1 (atau 85)
- Myo = 2 (atau 170)
- LV = 3 (atau 255)

Jadi aman walau nilai uniknya berubah-ubah antar file.

4 File : cek npy.py

Bahasan:

Berikut penjelasan **lengkap** isi file **cek npy.py** dan saya pilah **berdasarkan blok fungsinya**, dari atas ke bawah, dengan fokus pada **tujuan, alur data, dan peran tiap baris kode**.
(Sumber kode:)

---

1 Header & metadata file

```
# -*- coding: utf-8 -*-
"""
Created on Wed Feb 14 09:48:15 2024
@author: ramad
"""
```

Fungsi blok ini:

- Menentukan encoding UTF-8 (aman untuk komentar non-ASCII).
- Komentar metadata (tanggal & penulis).
- Tidak memengaruhi eksekusi program.

---

2 Import library

```
import os
import numpy as np
import keras
from matplotlib import pyplot as plt
import glob
import random
import os
from custom_datagen import imageLoader
import tensorflow as tf
```

Penjelasan per library:

- os → operasi direktori & path file
- numpy → load & manipulasi array .npy
- keras, tensorflow → konteks deep learning (tidak dipakai langsung di visualisasi)
- matplotlib.pyplot → menampilkan citra & mask
- glob → pencarian file berbasis pola (tidak dipakai di script ini)
- random → memilih data & slice secara acak
- custom_datagen.imageLoader → *data generator* (diimport tapi **tidak digunakan** di script ini)

📌 Catatan:

- Ada **import redundan** (os dua kali).
- Ada **import tidak terpakai** (glob, keras, tf, imageLoader) → aman, tapi bisa dirapikan.

---

3 Penentuan direktori dataset

```
train_img_dir = ".../train/images/"
```

train_mask_dir = ".../train/masks/"

Fungsi blok ini:

- Menentukan lokasi dataset training hasil preprocessing.
- Struktur dataset:

```
train/
 ├── images/   → file .npy citra 3D
 └── masks/    → file .npy mask 3D (one-hot)
```

4️⃣ Mengambil daftar file citra & mask

```
img_list = os.listdir(train_img_dir)
msk_list = os.listdir(train_mask_dir)
num_images = len(os.listdir(train_img_dir))
```

Fungsi blok ini:

- Mengambil seluruh nama file .npy di folder images dan masks.
- num_images = jumlah data training.

📌 Asumsi penting:

- Urutan file di images/ dan masks/ **harus sinkron** (nama dan urutan sesuai).

5️⃣ Memilih satu sampel secara acak

```
img_num = random.randint(0, num_images-1)
```

Fungsi:

- Memilih **satu volume 3D secara acak** untuk dicek secara visual.
- Berguna untuk:
  - sanity check dataset
  - debugging preprocessing

6️⃣ Load data .npy (image & mask)

```
test_img = np.load(train_img_dir + img_list[img_num])
test_mask = np.load(train_mask_dir + msk_list[img_num])
```

Bentuk data yang diharapkan

- test_img → (128, 128, 128, 1)
- test_mask → (128, 128, 128, C)
  (C = jumlah kelas, misalnya 5)

7️⃣ Mengubah mask one-hot → label integer

```
test_mask = np.argmax(test_mask, axis=3)
```

Fungsi blok ini:

- Mengubah mask dari **one-hot encoding** ke **label kelas tunggal**.
- Contoh:
  - [0,0,1,0,0] → 2
- Hasil akhir:
- test_mask.shape = (128, 128, 128)

📌 Ini langkah penting sebelum visualisasi, karena:

- imshow tidak cocok untuk tensor one-hot 4D.

8️⃣ Memilih slice acak pada sumbu Z

```
n_slice = random.randint(0, test_mask.shape[2])
```

Fungsi:

- Memilih satu **slice 2D** dari volume 3D.
- Sumbu ke-3 (axis=2) = arah Z (slice MRI).

9️⃣ Visualisasi citra & mask
plt.figure(figsize=(12, 8))
Membuat kanvas besar agar detail terlihat jelas.

9️⃣a Visualisasi citra
plt.subplot(221)
plt.imshow(test_img[:,:,n_slice, 0], cmap='gray')
plt.title('Image flair')
Penjelasan:

- Menampilkan citra MRI slice ke-n_slice.
- [..., 0] → channel ke-0 (karena ini citra 1-channel).
- cmap='gray' → standar citra medis.

📌 Label 'Image flair' kemungkinan:

- Copy dari kode MRI otak (FLAIR),
- Padahal dataset ini **MRI jantung ACDC** → hanya label teks, tidak memengaruhi isi.

9️⃣b Visualisasi mask
plt.subplot(222)
plt.imshow(test_mask[:,:,n_slice])
plt.title('Mask')
Fungsi:

- Menampilkan mask segmentasi hasil argmax.
- Warna default matplotlib mewakili kelas berbeda.

9️⃣c Subplot lain (dikomentari)
plt.subplot(222)
plt.imshow(test_img[:,:,n_slice, 1], cmap='gray')
plt.title('Image t1ce')
Makna:

- Script ini tampaknya diadaptasi dari:
  - MRI otak multi-modal (FLAIR, T1CE, T2)
- Namun untuk ACDC:
  - hanya **1 channel**, jadi bagian ini tidak dipakai.

🔟 **Menampilkan hasil**
plt.show()
Menampilkan semua subplot ke layar.

✅ **Ringkasan fungsi keseluruhan script**
Script **cek npy.py** digunakan untuk:

1. Membaca dataset training hasil preprocessing (.npy)
2. Memilih satu volume 3D secara acak
3. Mengonversi mask dari one-hot → label kelas
4. Mengambil satu slice 2D acak
5. Menampilkan **citra vs mask** untuk validasi visual

🎯 **Tujuan utamanya**:

- Mengecek apakah:
  - citra dan mask sejajar
  - preprocessing benar
  - channel & dimensi sesuai
  - label mask masuk akal

⚠️ **Catatan penting (debug & improvement)**

1. **Import tidak terpakai**
   - keras, tf, glob, imageLoader → bisa dihapus
2. **Label "Image flair" tidak relevan**
   - Bisa diganti → Image MRI atau Image ED
3. **Sinkronisasi image–mask**
   - Aman jika nama file konsisten
   - Lebih aman jika:
   - img_list.sort()
   - msk_list.sort()

Kalau kamu mau, aku bisa:

- Membuat versi script ini **lebih rapi & minimal**
- Menambahkan **overlay mask di atas citra**
- Menampilkan **semua kelas mask dengan colormap khusus**
- Mengecek **value range & distribusi kelas mask**

Tinggal bilang mau fokus ke bagian mana 👍

5 File : combined_images.py

Bahasan:

Berikut penjelasan **lengkap** isi file **combined_images.py** dan saya pilah berdasarkan **blok fungsinya** (urut sesuai kode).
Sumber kode:

---

1) Header & import library

```
import cv2 as cv
import numpy as np
import os
import re
```

Fungsi tiap import:

- **cv2 (OpenCV)**: baca gambar (imread), konversi warna (cvtColor), tulis gambar (imwrite)
- **numpy**: operasi array/masking boolean (dipakai untuk threshold mask)
- **os**: path & manipulasi folder/list file
- **re**: regex untuk ambil nomor pasien dari nama file

---

2) Fungsi create_folder(patient_id, saved_directory)

```
def create_folder(patient_id, saved_directory):
    loss_dir = os.path.join(saved_directory, "focalctc")
    patient_dir = os.path.join(loss_dir, f"Pasien {patient_id}")
```

```
    os.makedirs(loss_dir, exist_ok=True)
    os.makedirs(patient_dir, exist_ok=True)

    return loss_dir, patient_dir
```

Tujuan

Membuat struktur folder output untuk menyimpan hasil gabungan overlay:

- folder utama untuk metode/loss tertentu: .../focalctc/
- subfolder per pasien: .../focalctc/Pasien <id>/

Kenapa exist_ok=True?

Kalau folder sudah ada, program **tidak error**.

---

3) Fungsi natural_sort_key(s)

```
def natural_sort_key(s):
    return [int(text) if text.isdigit() else text.lower() for text in re.split('(\d+)', s)]
```

Tujuan

Membuat sorting “manusiawi” (natural sort) untuk nama file yang mengandung angka.
Contoh:

- Sort biasa: img1, img10, img2
- Natural sort: img1, img2, img10

Cara kerja singkat

- re.split('(\d+)', s) memecah string jadi bagian teks dan angka
- angka diubah ke int supaya urutan numeriknya benar

---

4) Fungsi inti combined_images(...)

```
def combined_images(sorted_imgs, sorted_myo, sorted_rv, sorted_lv, image_dir, myo_dir, rv_dir,
lv_dir, saved_directory):
    for index, (img, myo_mask, rv_mask, lv_mask) in enumerate(zip(sorted_imgs, sorted_myo,
sorted_rv, sorted_lv)):
        ...
```

Fungsi ini melakukan loop per-slice, lalu:

1. memastikan file image + mask berasal dari pasien yang sama
2. load image & mask
3. overlay mask ke citra
4. simpan hasil overlay ke folder output

4.1) Ekstraksi nomor pasien dari nama file (regex)

```
match_img = re.search(r'Pasien(\d+)_', img)
...
if match_img ... and match_img.group(1) == match_myo.group(1) == match_rv.group(1) ==
match_lv.group(1):
    patient_id = match_img.group(1)
else:
    print("Tidak dapat mengekstrak...")
    continue
```

Tujuan:

- Mengambil angka pasien dari nama file yang polanya seperti:
  Pasien101_...png

Validasi penting:

- Program hanya lanjut kalau:
    - semua file berhasil diambil patient id-nya
    - dan semuanya sama (image, myo, rv, lv sinkron)

Kalau tidak, dilewati (continue) untuk mencegah overlay salah pasangan.

4.2) Membuat folder output pasien

loss_dir, patient_dir = create_folder(patient_id, saved_directory)

Hasilnya: output disimpan dalam folder pasien tersebut.

4.3) Menyusun path file & debugging print

```
img_path = os.path.join(image_dir, img)
...
print(f"images = {img_path}")
...
```

Tujuan:

- Membentuk path file lengkap untuk dibaca OpenCV
- Print dipakai untuk memastikan file yang diproses benar

4.4) Membaca image & mask (grayscale)

```
imgs = cv.imread(img_path, 0)
myo_mask = cv.imread(myo_mask_path, 0)
rv_mask = cv.imread(rv_mask_path, 0)
lv_mask = cv.imread(lv_mask_path, 0)
```

- Mode 0 = grayscale → output array 2D
- Mask diasumsikan berupa gambar biner / grayscale yang >0 berarti “ada prediksi”.

4.5) Konversi citra ke RGB/BGR untuk overlay warna

img_rgb = cv.cvtColor(imgs, cv.COLOR_GRAY2BGR)

OpenCV menyimpan warna dalam urutan **BGR**, bukan RGB.

Kenapa perlu diubah?

- Kalau tetap grayscale, tidak bisa mewarnai area mask dengan warna tertentu (3 channel).

4.6) Membuat mask boolean (area objek)

```
myo_mask_1 = myo_mask > 0
rv_mask_1 = rv_mask > 0
lv_mask_1 = lv_mask > 0
```

Tujuan:

- Ubah mask menjadi boolean:
    - True untuk pixel yang termasuk objek
    - False untuk background

Ini yang dipakai untuk indexing overlay:

img_rgb[myo_mask_1] = [235, 183, 0]

4.7) Overlay warna ke citra

```
# img_rgb[lv_mask_1] = [139, 51, 204]  # magenta
img_rgb[myo_mask_1] = [235, 183, 0]  # cyan
# img_rgb[rv_mask_1] = [255, 0, 17]  # pink
```

Yang aktif hanya myocardium (myo):

- myo_mask_1 diberi warna [235, 183, 0] (ingat: ini BGR di OpenCV)

RV dan LV dikomentari, jadi saat ini script:
✅ menggabungkan **citra + prediksi myocardium** saja
❌ belum menampilkan RV/LV (tapi sudah disiapkan)
📌 Catatan: komentarnya bilang "cyan/magenta/pink", tapi angka warnanya belum tentu sesuai label warna yang kamu maksud (karena BGR). Tapi fungsinya tetap: memberi warna tetap pada area mask.

4.8) Menyimpan gambar hasil overlay

```
name_file_images = os.path.join(patient_dir, f"Pasien{patient_id}_{index+1}_img.png")
cv.imwrite(name_file_images, img_rgb)
```

- Nama file output dibuat berurutan berdasarkan index+1
- Disimpan per pasien di folder output

---

5) Fungsi main() — loop pasien 101 sampai 150

```
for i in range(101, 151):
```

Tujuan

Memproses pasien test dari **101–150** (50 pasien).

5.1) Menentukan direktori input per pasien

```
image_dir = ".../Pasien {i}/images/"
myo_dir = ".../Pasien {i}/prediksi myocardium/"
rv_dir  = ".../Pasien {i}/prediksi right ventricel/"
lv_dir  = ".../Pasien {i}/prediksi left ventricel/"
saved_directory = ".../Gabung citra dan myo/transunet/"
```

Makna struktur:

- image_dir: citra 2D ED yang sudah di-resize (128)
- myo_dir, rv_dir, lv_dir: hasil prediksi model (TransUNet) per kelas
- saved_directory: root output overlay

5.2) Validasi direktori prediksi ada atau tidak

```
if not os.path.exists(myo_dir) or not os.path.exists(rv_dir) or not os.path.exists(lv_dir):
   print("Jalur direktori tidak ditemukan...")
   continue
```

Tujuan:

- Kalau salah satu folder prediksi belum ada, pasien itu dilewati agar tidak error.

5.3) Ambil list file dan urutkan natural sort

```
get_imgs = os.listdir(image_dir)
sorted_imgs = sorted(get_imgs, key=natural_sort_key)
...
sorted_myo = sorted(get_myo, key=natural_sort_key)
sorted_rv  = sorted(get_rv,  key=natural_sort_key)
sorted_lv  = sorted(get_lv,  key=natural_sort_key)
```

Tujuan:

- Menjaga urutan slice konsisten (slice1 cocok dengan slice1 mask, dst).

5.4) Panggil proses overlay

combined_images(sorted_imgs, sorted_myo, sorted_rv, sorted_lv, image_dir, myo_dir, rv_dir, lv_dir, saved_directory)

---

6) Entry point

if __name__ == "__main__":
  main()

Agar script berjalan saat file dieksekusi langsung.

✅ **Ringkasan tujuan keseluruhan script**

Script ini dipakai untuk:

1. mengambil citra 2D ED per pasien (101–150),
2. mengambil mask prediksi per kelas (myo, rv, lv),
3. mengecek sinkronisasi ID pasien dari nama file,
4. overlay mask (saat ini **hanya myocardium**) ke citra grayscale,
5. menyimpan hasilnya per pasien ke folder output.

⚠️ **Catatan penting / potensi perbaikan**

1. **Sinkronisasi slice hanya pakai zip**
- zip(sorted_imgs, sorted_myo, sorted_rv, sorted_lv) akan berhenti di list terpendek.
- Kalau jumlah slice di salah satu folder beda, sebagian data bisa tidak diproses tanpa disadari.

2. **RV/LV sudah dibaca tapi belum ditampilkan**
- Tinggal uncomment baris overlay RV/LV.

3. **Komentar warna vs nilai BGR**
- OpenCV pakai BGR, jadi komentar “cyan/pink/magenta” bisa meleset dari warna sebenarnya.

4. **Validasi file match lebih kuat**
- Sekarang hanya cek “Pasien(\d+)_” → bagus, tapi tidak cek indeks slice. Jika urutan file kacau, bisa salah pasangan slice.

Kalau kamu mau, aku bisa bantu ubah script ini jadi:

- overlay **3 kelas sekaligus** (myo+rv+lv) dengan transparansi (alpha blending),
- memastikan pairing berdasarkan **nomor slice** (bukan hanya urutan list),
- atau menghasilkan satu gambar “panel” (original + overlay + tiap mask) untuk setiap slice.

6 File : custom_datagen.py

Bahasan:

Berikut penjelasan **lengkap** isi file **custom_datagen.py** dan saya pilah **berdasarkan blok fungsinya** (urut sesuai kode).

Sumber kode:

---

1) Header & import

```
# -*- coding: utf-8 -*-
"""
Created on Thu Feb 15 12:38:14 2024
@author: ramad
"""

import os
```

import numpy as np

Fungsi blok ini

- Header hanya metadata (tidak memengaruhi eksekusi).
- numpy dipakai untuk np.load dan menyusun batch menjadi array.
- os diimport tapi **tidak digunakan** di kode ini (bisa dihapus tanpa mengubah fungsi).

---

2) Fungsi load_img(img_dir, img_list)

```
def load_img(img_dir, img_list):
   images=[]
   for i, image_name in enumerate(img_list):
      if (image_name.split('.')[1] == 'npy'):

         image = np.load(img_dir+image_name)

         images.append(image)
   images = np.array(images)

   return(images)
```

Tujuan

Membaca sekumpulan file **.npy** dari sebuah folder dan mengembalikannya sebagai **numpy array** (batch kecil).

Alur kerja per baris

- images=[] → list penampung sementara.
- Loop setiap image_name dalam img_list.
- Cek ekstensi:
- if (image_name.split('.')[1] == 'npy'):
  Artinya hanya file yang nama formatnya nama.npy yang diproses.
- Load file:
- image = np.load(img_dir+image_name)
  Ini akan menghasilkan array numpy, misalnya:
  - citra: (128,128,128,1) atau
  - mask: (128,128,128,5) (one-hot)
- images.append(image) → kumpulkan semua sampel.
- images = np.array(images) → jadikan satu array:
  - hasil biasanya berbentuk:
  - (batch, 128, 128, 128, channels)

Catatan penting (potensi masalah)

- img_dir+image_name raw string concat. Lebih aman pakai os.path.join(img_dir, image_name) supaya tidak salah slash.
- image_name.split('.')[1] bisa error kalau nama file punya banyak titik (mis. a.b.npy) atau tidak punya titik. Cara yang lebih aman: image_name.endswith(".npy").

---

3) Fungsi generator imageLoader(...)

```
def imageLoader(img_dir, img_list, mask_dir, mask_list, batch_size):

   L = len(img_list)

   #keras needs the generator infinite, so we will use while true
```

```
  while True:

    batch_start = 0
    batch_end = batch_size

    while batch_start < L:
      limit = min(batch_end, L)

      X = load_img(img_dir, img_list[batch_start:limit])
      Y = load_img(mask_dir, mask_list[batch_start:limit])

      yield (X,Y) #a tuple with two numpy arrays with batch_size samples

      batch_start += batch_size
      batch_end += batch_size
```

Tujuan utama

Membuat **data generator** untuk Keras/TensorFlow yang:

- Mengambil data citra dan mask dari disk secara bertahap (batch),
- Menghasilkan pasangan (X, Y) terus menerus (infinite loop),
- Supaya training tidak perlu memuat semua data ke RAM.

Struktur loop-nya

a) Panjang dataset

L = len(img_list)

Jumlah sample = jumlah file image (diasumsikan sama dengan mask).

b) Infinite loop untuk Keras

while True:

Keras model.fit() membutuhkan generator yang bisa “mengalir” terus (tiap epoch Keras yang mengatur kapan berhenti).

c) Sliding window batching

batch_start = 0

batch_end = batch_size

Mulai dari indeks 0 sampai batch_size-1.

d) Loop sampai habis satu epoch (secara manual)

while batch_start < L:

  limit = min(batch_end, L)

- limit memastikan batch terakhir tidak melewati panjang data.

e) Load batch X dan Y

X = load_img(img_dir, img_list[batch_start:limit])

Y = load_img(mask_dir, mask_list[batch_start:limit])

- X = batch citra
- Y = batch mask
- Keduanya dibaca menggunakan fungsi load_img.

f) Menghasilkan batch ke Keras

yield (X, Y)

Keras akan menerima batch ini sebagai input train step.

g) Geser ke batch berikutnya

batch_start += batch_size

batch_end += batch_size

✅ **Ringkasan fungsi keseluruhan file**

File custom_datagen.py berisi **2 fungsi**:

1. **load_img()**
- Memuat list file .npy dari folder → menjadi array batch.
2. **imageLoader()**
- Generator batch untuk training Keras:
    - baca citra & mask per batch,
    - yield (X,Y) terus menerus.

⚠️ **Catatan penting untuk penggunaan yang benar**

Agar generator ini bekerja mulus saat training:

1. **Urutan img_list & mask_list harus sinkron**
- Kalau img_list tidak di-sort, bisa mismatch.
- Biasanya lakukan:
- img_list.sort()
- mask_list.sort()
2. **Tidak ada shuffle**
- Generator ini mengirim batch berurutan. Kalau ingin training lebih robust, biasanya perlu shuffle setiap epoch.
3. **Batch terakhir bisa lebih kecil**
- Karena pakai min(batch_end, L), batch terakhir bisa kurang dari batch_size. Ini normal, tapi kadang kamu ingin drop remainder.
4. **I/O bisa lambat**
- Load .npy tiap batch dari disk bisa bottleneck. Solusi umum:
    - gunakan cache,
    - gunakan tf.data,
    - gunakan prefetch.

Kalau kamu mau, aku bisa bikinkan versi yang:

- otomatis sort() dan shuffle tiap epoch,
- pakai os.path.join,
- validasi bahwa image dan mask punya pasangan nama yang sama,
- atau migrasi ke tf.data.Dataset biar lebih cepat (prefetch, parallel map).

7 File :hitung_dice_2d.py

Bahasan:

Berikut penjelasan **lengkap** isi file **hitung_dice_2d.py** dan saya pilah berdasarkan **blok fungsinya** (urut sesuai kode).

Sumber kode:

---

1) Header & metadata file

```
# -*- coding: utf-8 -*-
"""
Created on Wed Jan 31 21:28:28 2024
@author: ramad
"""
```

Fungsi blok ini:

- Menetapkan encoding UTF-8.
- Informasi tanggal dibuat & author (komentar saja, tidak memengaruhi eksekusi).

---

2) Import library

import numpy as np

Fungsi:

- Dipakai untuk operasi array (penjumlahan, perkalian elemen, cek NaN).

---

3) Fungsi utama: dice_coefficient(groundtruth, predict)

def dice_coefficient(groundtruth, predict):

Tujuan fungsi

Menghitung **Dice Similarity Coefficient (DSC)** antara:

- groundtruth = mask label asli
- predict = mask hasil prediksi model

Dice dipakai untuk mengukur kualitas segmentasi, terutama di citra medis.

3.1) Hitung intersection

intersection = np.sum(groundtruth * predict)

Makna:

- Jika groundtruth dan predict adalah mask biner (0/1), maka:
  - perkalian elemen (*) akan menghasilkan 1 hanya pada pixel yang sama-sama 1.
  - jumlahnya = jumlah pixel overlap.

📌 Catatan penting:

- Ini mengasumsikan mask sudah biner atau setidaknya kompatibel (misal one-hot per kelas).
- Kalau isinya label 0/1/2/3 tanpa one-hot, maka groundtruth * predict bukan intersection yang benar untuk multi-class.

3.2) Hitung total area groundtruth dan prediksi

sum_groundtruth = np.sum(groundtruth)

sum_predict = np.sum(predict)

Makna:

- Untuk mask biner:
  - sum_groundtruth = jumlah pixel foreground pada GT
  - sum_predict = jumlah pixel foreground pada prediksi

3.3) Cek kasus keduanya kosong

```
if sum_groundtruth + sum_predict == 0:
  return 0
```

Kasus yang ditangani:

- Jika groundtruth kosong **dan** prediksi kosong:
  - rumus dice akan jadi 0/0 (tidak terdefinisi).
- Di kode ini diputuskan mengembalikan **0**.

📌 Catatan evaluasi:

- Ada 2 kebiasaan di komunitas:
  - mengembalikan **1** (karena keduanya sama-sama “benar” kosong)
  - atau **0** (menghindari memberi skor tinggi pada kasus trivial)
- Jadi keputusan return 0 di sini adalah **pilihan desain**.

3.4) Hitung Dice coefficient
dice_coeff = (2 * intersection) / (sum_groundtruth + sum_predict)
Rumus Dice:
[
DSC = \frac{2|A \cap B|}{|A| + |B|}
]

- A = groundtruth foreground
- B = predicted foreground

Nilai:

- 1.0 ⟶ overlap sempurna
- 0.0 ⟶ tidak overlap

3.5) Cek NaN (antisipasi pembagian tak valid)

```
if np.isnan(dice_coeff):
  return 0
```

Fungsi:

- Jika karena alasan tertentu hasilnya NaN, kembalikan 0.
- Sebenarnya kasus NaN utama sudah ditangani di cek sum_groundtruth + sum_predict == 0, tapi ini menjadi guard tambahan.

3.6) Return nilai Dice

```
return dice_coeff
```

✅ **Ringkasan fungsi keseluruhan file**
File ini hanya berisi **1 fungsi** untuk menghitung Dice:

- Input: groundtruth, predict (umumnya 2D, tapi bisa juga 3D asalkan array biner)
- Output: angka float Dice coefficient

⚠️ **Hal penting yang perlu kamu tahu (supaya hasilnya "benar")**
1) Ini Dice untuk mask biner
Kalau kamu ingin Dice untuk multi-class segmentasi (mis. LV, RV, MYO), biasanya kamu lakukan per kelas:

- Ubah ke one-hot atau biner per kelas:
  - gt == kelas_k
  - pred == kelas_k
- Baru panggil dice_coefficient(gt_bin, pred_bin)

2) Jika GT dan pred sama-sama kosong, kode ini return 0
Kalau evaluasi kamu menganggap "kosong-kosong itu benar", kamu mungkin ingin return 1 di kasus itu.

Kalau kamu mau, aku bisa bikinkan versi:

- Dice **per kelas** + rata-rata (mean dice)
- Dice 3D (volume) untuk dataset ACDC
- Dice dengan smoothing (biar stabil untuk objek kecil)

8 File : hitung_evaluasi_metrik.py

Bahasan:

Berikut penjelasan **lengkap** isi file **hitung_evaluasi_metrik.py** dan saya pilah berdasarkan **blok fungsinya** (urut dari atas ke bawah).
Sumber kode:

---

1) Header & metadata

```
# -*- coding: utf-8 -*-
"""
Created on Mon Mar 18 09:17:43 2024
@author: ramad
"""
```

Hanya informasi file (encoding, tanggal, author). Tidak berpengaruh ke proses.

---

2) Import library & modul metrik

```
from keras.models import load_model
import cv2 as cv
import os
import numpy as np
from matplotlib import pyplot as plt
from sklearn.preprocessing import LabelEncoder
from tensorflow.keras.utils import to_categorical
from hitung_iou_2d import iou_2d
from hausdorff.hausdorff import hausdorff_distance
from hitung_dice_2d import dice_coefficient
from hitung_f1score_2d import f1_score_2d
from hitung_surface_distance_2d import surfd
from hitung_mcc_2d import mcc_2d
import csv
```

Fungsi blok ini
Script ini ingin menghitung metrik evaluasi segmentasi 2D:

- **IoU (Jaccard)** → iou_2d
- **Dice** → dice_coefficient
- **Hausdorff Distance** → hausdorff_distance (jarak tepi paling jauh)
- **Surface Distance** → surfd lalu dirata-ratakan
- Ada juga import **F1-score** dan **MCC** tapi penggunaannya dikomentari.

📌 Catatan:

- load_model, LabelEncoder, to_categorical **tidak dipakai** di script ini (bisa sisa dari eksperimen lain).
- Script ini **tidak melakukan inferensi model**, hanya evaluasi gambar hasil prediksi yang sudah tersimpan sebagai PNG.

---

3) Konfigurasi pasien yang dievaluasi + path direktori

```
i = 104
```

Artinya evaluasi fokus ke **Pasien 104**.
Lalu path untuk:

3.1 Direktori prediksi (hasil model)

- direktori_predict_ed_myo
- direktori_predict_ed_rv
- direktori_predict_ed_lv

Ini menunjuk ke folder hasil prediksi model U-Net (diceloss) pada fase **ED** per kelas.

3.2 Direktori ground truth (GT)

- direktori_gt_ed_myo
- direktori_gt_ed_rv
- direktori_gt_ed_lv

Ini menunjuk ke folder groundtruth resize 128 ED per kelas.

3.3 Lokasi simpan CSV & plot

simpan_csv = ".../Pasien {i}/"

Semua output evaluasi (CSV dan boxplot) disimpan di folder pasien tersebut.

---

4) Fungsi evaluasi per kelas

Script membagi evaluasi menjadi **3 fungsi** terpisah:

- hitung_evaluasi_metrik_myo(...)
- hitung_evaluasi_metrik_rv(...)
- hitung_evaluasi_metrik_lv(...)

Strukturnya **sama**, bedanya hanya folder input.

Di bawah ini saya jelaskan pola umumnya, lalu sebut bagian spesifiknya.

4A) hitung_evaluasi_metrik_myo(imgs, masks)

def hitung_evaluasi_metrik_myo(imgs, masks):

4A.1 Inisialisasi list penampung metrik per-slice

iou_list_myo = []

dice_list_myo = []

hausdorff_list_myo = []

surface_distance_list_myo = []

Setiap slice akan menghasilkan 1 nilai IoU, Dice, HD, dan mean surface distance.

4A.2 Membuka file CSV output

```
with open(simpan_csv + f'metrik_evaluasi_unet_myo_ed_Pasien {i}_acdc2017.csv', mode='w',
newline='') as file:
   writer = csv.writer(file, delimiter=';')
   writer.writerow(['Citra Ke', 'IoU', 'Dice','Hausdorff Distance', 'Surface Distance'])
```

- Menulis CSV dengan delimiter ; (umum di Excel Indonesia).
- Header kolom: nama citra, IoU, Dice, HD, Surface Distance.

4A.3 Validasi jumlah file prediksi vs GT

```
if len(imgs) != len(masks):
   print("Jumlah citra predict dan mask tidak sama...")
   return
```

Kalau jumlahnya beda, evaluasi dihentikan agar tidak pairing salah.

4A.4 Loop per-slice: pairing prediksi dan GT

```
for predict_filename, mask_filename in zip(imgs, masks):
   if predict_filename.endswith('.png') and mask_filename.endswith('.png'):
```

(a) Bentuk path lengkap

```
file_path_predict = os.path.join(direktori_predict_ed_myo, predict_filename)
file_path_mask = os.path.join(direktori_gt_ed_myo, mask_filename)
```

(b) Load gambar grayscale

```
img_predict_myo_ed = cv.imread(file_path_predict, 0)
img_gt_myo_ed = cv.imread(file_path_mask, 0)
```

(c) Konversi ke biner 0/1

bin_predict = np.where(img_predict_myo_ed == 255, 1, 0)
bin_gt = np.where(img_gt_myo_ed == 255, 1, 0)
Asumsi penting:

- Mask disimpan sebagai PNG biner: **255 = foreground**, 0 = background.

(d) Hitung metrik
iou_myo_ed = iou_2d(bin_gt, bin_predict)
dice_myo_ed = dice_coefficient(bin_gt, bin_predict)
hd_myo_ed = hausdorff_distance(bin_gt, bin_predict, distance='euclidean')
surface_myo_ed = surfd(bin_gt, bin_predict, sampling=1, connectivity=1)
mean_surface_myo_ed = np.mean(surface_myo_ed)
Makna tiap metrik:

- **IoU**: overlap / union.
- **Dice**: overlap yang “lebih ramah” untuk objek kecil.
- **Hausdorff**: “jarak kesalahan terburuk” pada boundary (semakin kecil semakin baik).
- **Surface distance**: daftar jarak boundary; diambil **rata-rata**.

(e) Simpan ke list
iou_list_myo.append(iou_myo_ed)
dice_list_myo.append(dice_myo_ed)
hausdorff_list_myo.append(hd_myo_ed)
surface_distance_list_myo.append(mean_surface_myo_ed)
(f) Format angka koma untuk CSV
iou_myo_ed_str = str(iou_myo_ed).replace('.', ',')
...
Supaya Excel locale Indonesia tidak salah baca desimal.
(g) Tulis 1 baris ke CSV
writer.writerow([f"citra {predict_filename}", iou_myo_ed_str, dice_myo_ed_str, hd_myo_ed_str, mean_surface_myo_ed_str])
(h) Print log ke console
Untuk monitoring progress.
4A.5 Hitung rata-rata semua slice & tulis ke CSV
avg_iou = np.mean(iou_list_myo)
avg_dice = np.mean(dice_list_myo)
avg_hausdorff = np.mean(hausdorff_list_myo)
avg_surface = np.mean(surface_distance_list_myo)
...
writer.writerow(["Mean IoU", "Mean Dice","Mean Hausdorff Distance", "Mean Surface Distance"])
writer.writerow([avg_iou_myo_ed_str, avg_dice_myo_ed_str, avg_hd_myo_ed_str, avg_surface_myo_ed_str])
4A.6 Return list metrik
return iou_list_myo, dice_list_myo, hausdorff_list_myo, surface_distance_list_myo
📌 Catatan bug kecil:
Setelah return, ada print("Perhitungan ...") tapi **tidak pernah dieksekusi** karena fungsi sudah keluar.

4B) hitung_evaluasi_metrik_rv(imgs, masks)
Isinya sama persis dengan MYO, hanya beda:

- folder input prediksi: direktori_predict_ed_rv
- folder GT: direktori_gt_ed_rv
- nama file CSV: ..._rv_...

Output list:

- iou_list_rv, dice_list_rv, hausdorff_list_rv, surface_distance_list_rv

---

4C) hitung_evaluasi_metrik_lv(imgs, masks)

Sama persis lagi, hanya beda folder LV.

---

5) Fungsi visualisasi boxplot: plot_visualize(...)

def plot_visualize(iou_list_myo, dice_list_myo, hausdorff_list_myo, surface_list_myo,
iou_list_rv, dice_list_rv, hausdorff_list_rv, surface_list_rv,
iou_list_lv, dice_list_lv, hausdorff_list_lv, surface_list_lv):

Tujuan

Membuat **boxplot per metrik** untuk membandingkan distribusi nilai per kelas (MYO vs RV vs LV) pada pasien i.

Ada 4 boxplot yang dibuat:

1. boxplot IoU
2. boxplot Dice
3. boxplot Hausdorff Distance
4. boxplot Surface Distance

Contoh IoU:

boxplot = ax.boxplot([iou_list_myo, iou_list_rv, iou_list_lv], labels=[...], patch_artist=True)
ax.set_title('Mean IoU ACDC 2017')
plt.savefig(...'boxplot_mean_iou_pasien {i}_acdc2017.png')

📌 Catatan penting:

- Di dalam fungsi ini, warna boxplot ditentukan manual:
- colors = ['red', 'yellow', 'green']
- patch.set_facecolor(color)
- Boxplot menyimpan gambar ke folder simpan_csv.

---

6) main() — mengambil file, urutkan, panggil evaluasi + plot

```
def main():
  get_masks_myo = os.listdir(direktori_gt_ed_myo)
  sorted_masks_myo = sorted(get_masks_myo, key=lambda x: x.lower())
  ...
```

Langkah main:

1. Ambil daftar file GT per kelas (myo, rv, lv)
2. Sort lower-case (supaya urut konsisten)
3. Ambil daftar file prediksi per kelas
4. Sort juga
5. Panggil evaluasi:
6. iou_list_myo, dice_list_myo, ... = hitung_evaluasi_metrik_myo(sorted_predicts_myo, sorted_masks_myo)
7. Panggil plot_visualize(...)

---

7) Entry point

```
if __name__ == "__main__":
  main()
```

Agar script langsung berjalan ketika dieksekusi.

✅ **Ringkasan fungsi keseluruhan script**

Script ini melakukan **evaluasi segmentasi 2D** untuk **Pasien tertentu (i=104)**, fase **ED**, dan menghitung metrik **per-slice** untuk tiap kelas:

- Myocardium
- Right ventricle
- Left ventricle

Output yang dihasilkan:

1. 3 file CSV (MYO, RV, LV) berisi nilai metrik per slice + mean
2. 4 gambar boxplot untuk membandingkan distribusi metrik antar kelas

---

⚠️ **Catatan penting / potensi masalah (biar hasil evaluasi akurat)**

1. **Pairing prediksi–GT hanya berdasarkan urutan list**
   - Kalau nama file tidak benar-benar matching, bisa tertukar slice.
   - Lebih aman pairing berdasarkan nama file yang sama.
2. **Asumsi mask biner 255**
   - Jika mask tidak benar-benar 255/0 (mis. 1/0), konversinya perlu disesuaikan.
3. **Kode duplikat**
   - Fungsi MYO/RV/LV hampir sama → bisa disederhanakan jadi 1 fungsi umum.
4. load_model, LabelEncoder, to_categorical tidak dipakai → bisa dibersihkan.

Kalau kamu mau, aku bisa bantu refactor script ini supaya:

- cukup 1 fungsi evaluasi untuk semua kelas (lebih ringkas),
- pairing file berdasarkan nama slice (lebih aman),
- menghasilkan 1 CSV gabungan untuk MYO/RV/LV sekaligus.

9 File : hitung_evaluasi_metrik_acdc2017.py

Bahasan:

Berikut penjelasan **lengkap** untuk file **hitung_evaluasi_metrik_acdc2017.py** (yang baru) dan **beda utamanya** dibanding **hitung_evaluasi_metrik.py** (file sebelumnya). Saya pilah berdasarkan **blok fungsi/komponen** sesuai urutan kode.

Sumber kode: dan pembanding:

A) Penjelasan hitung_evaluasi_metrik_acdc2017.py per blok fungsinya

1) Header & metadata

```
# -*- coding: utf-8 -*-
"""
Created on Wed Jun 12 21:55:30 2024
@author: ramad
"""
```

Hanya metadata file (tidak memengaruhi eksekusi).

---

2) Import library & modul metrik

```
from keras.models import load_model
import cv2 as cv
import os
import numpy as np
from matplotlib import pyplot as plt
from sklearn.preprocessing import LabelEncoder
```

from tensorflow.keras.utils import to_categorical
from hitung_iou_2d import iou_2d
from hausdorff.hausdorff import hausdorff_distance
from hitung_dice_2d import dice_coefficient
from hitung_f1score_2d import f1_score_2d
from hitung_surface_distance_2d import surfd
from hitung_mcc_2d import mcc_2d
import csv

Fungsi blok ini

Menyiapkan semua yang diperlukan untuk menghitung metrik evaluasi segmentasi 2D (berbasis file PNG prediksi vs ground truth):

- **IoU**: iou_2d(bin_gt, bin_pred)
- **Dice**: dice_coefficient(bin_gt, bin_pred)
- **Hausdorff Distance**: hausdorff_distance(...)
- **Surface Distance**: surfd(...) lalu diambil mean

📌 Catatan:

- load_model, LabelEncoder, to_categorical **tidak dipakai** di script ini (kemungkinan sisa dari eksperimen lain).
- f1_score_2d dan mcc_2d diimport tapi pemakaiannya dikomentari.

---

3) Loop utama untuk semua pasien test (101–150)

for i in range(101, 151):

Fungsi blok ini

Script akan melakukan evaluasi untuk **setiap pasien** dari **Pasien 101 sampai 150**.
Jadi output CSV + boxplot dibuat **per pasien**.
Ini perbedaan paling besar dari file sebelumnya yang hanya fokus 1 pasien.

---

4) Konfigurasi path input-output per pasien

Di dalam loop for i ..., kode membuat path:

4.1 Path prediksi per kelas (ED)

direktori_predict_ed_myo = f".../Pasien {i}/prediksi myocardium/"
direktori_predict_ed_rv = f".../Pasien {i}/prediksi right ventricel/"
direktori_predict_ed_lv = f".../Pasien {i}/prediksi left ventricel/"

4.2 Path ground truth per kelas (ED)

direktori_gt_ed_myo = f".../Pasien {i}/groundtruth myocardium/"
direktori_gt_ed_rv = f".../Pasien {i}/groundtruth right ventricel/"
direktori_gt_ed_lv = f".../Pasien {i}/groundtruth left ventricel/"

4.3 Folder output

simpan_csv = f".../Prediksi Pasien/Pasien {i}/"

Makna:

- Semua CSV & boxplot disimpan di folder pasien tersebut.

---

5) Fungsi evaluasi per kelas (MYO / RV / LV)

Di file ini ada **3 fungsi**, masing-masing untuk 1 kelas:

- hitung_evaluasi_metrik_myo(imgs, masks)
- hitung_evaluasi_metrik_rv(imgs, masks)
- hitung_evaluasi_metrik_lv(imgs, masks)

Strukturnya **hampir identik**, bedanya hanya folder input & nama output.

Pola kerja di setiap fungsi:

1. Buat list penampung metrik per-slice:
2. iou_list = []
3. dice_list = []
4. hausdorff_list = []
5. surface_distance_list = []
6. Buat CSV dan tulis header:
7. with open(simpan_csv + f'...Pasien {i}....csv', 'w') as file:
8. writer = csv.writer(file, delimiter=';')
9. writer.writerow(['Citra Ke','IoU','Dice','Hausdorff Distance','Surface Distance'])
10. Validasi jumlah file prediksi vs GT:
11. if len(imgs) != len(masks): return
12. Loop per slice (zip(imgs, masks)):
    - Baca prediksi & GT (grayscale):
    - img_pred = cv.imread(path_pred, 0)
    - img_gt = cv.imread(path_gt, 0)
    - Konversi biner (asumsi foreground = 255):
    - bin_pred = np.where(img_pred == 255, 1, 0)
    - bin_gt = np.where(img_gt == 255, 1, 0)
    - Hitung metrik:
    - iou = iou_2d(bin_gt, bin_pred)
    - dice = dice_coefficient(bin_gt, bin_pred)
    - hd = hausdorff_distance(bin_gt, bin_pred, distance='euclidean')
    - surface = surfd(bin_gt, bin_pred, sampling=1, connectivity=1)
    - mean_surface = np.mean(surface)
    - Simpan hasil ke list dan CSV (desimal diganti koma):
    - writer.writerow([..., str(iou).replace('.',','), ...])
13. Setelah loop, hitung rata-rata semua slice:
14. avg_iou = np.mean(iou_list)
15. ...
16. writer.writerow(["Mean IoU", "Mean Dice", ...])
17. writer.writerow([avg_iou_str, avg_dice_str, ...])

📌 Catatan bug kecil yang sama seperti file sebelumnya:

- Ada print("Perhitungan ... selesai") setelah return ... → **tidak pernah dieksekusi**.

---

6) Fungsi visualisasi plot_visualize(...)

def plot_visualize(iou_list_myo, dice_list_myo, ... surface_list_lv):

Fungsi blok ini

Membuat **4 boxplot** untuk membandingkan **MYO vs RV vs LV** pada pasien itu:

1. Mean IoU
2. Mean Dice
3. Mean Hausdorff Distance
4. Mean Surface Distance

Setiap plot:

- diberi warna box (red, yellow, green)
- disimpan ke file .png di folder simpan_csv

---

7) Fungsi main() + entry point

```
def main():
  get_masks_myo = os.listdir(direktori_gt_ed_myo)
  sorted_masks_myo = sorted(get_masks_myo, key=lambda x: x.lower())
  ...
  iou_list_myo, ... = hitung_evaluasi_metrik_myo(sorted_predicts_myo, sorted_masks_myo)
  ...
  plot_visualize(...)
```

Apa yang dilakukan:

- Mengambil daftar file mask GT dan prediksi dari tiap folder (MYO/RV/LV)
- Mengurutkan dengan lower() supaya konsisten
- Memanggil 3 fungsi evaluasi
- Memanggil plot_visualize

Entry point:

```
if __name__ == "__main__":
  main()
```

Karena main() berada di dalam loop for i ..., maka untuk tiap pasien, main dipanggil dan menghasilkan output pasien tersebut.

B) Perbedaan dengan file sebelumnya hitung_evaluasi_metrik.py

Berikut perbedaan yang benar-benar "berdampak" (bukan sekadar kosmetik).

Pembanding:

1) Scope evaluasi: banyak pasien vs satu pasien

- **File lama (hitung_evaluasi_metrik.py)**: i = 104 ⟶ hanya evaluasi **1 pasien** (pasien 104).
- **File baru (hitung_evaluasi_metrik_acdc2017.py)**: for i in range(101, 151) ⟶ evaluasi **50 pasien** (101–150), otomatis per pasien.

✅ Dampak: file baru menghasilkan **banyak CSV & boxplot** (setiap pasien satu set).

---

2) Model/eksperimen yang dievaluasi (path berbeda)

- File lama menunjuk hasil prediksi **U-Net** (terlihat dari nama CSV: ...unet... dan path folder unet diceloss).
- File baru menunjuk hasil prediksi **TransUNet** dengan konfigurasi folder transunet/msefocalctc dan nama CSV ...transunet....

✅ Dampak: file baru dipakai untuk evaluasi eksperimen/model yang berbeda.

---

3) Penamaan output

- File lama: metrik_evaluasi_unet_...
- File baru: metrik_evaluasi_transunet_...

---

4) Struktur program: loop di luar vs fokus tunggal

- File lama lebih "skrip satu kasus" (satu pasien), definisi fungsi berada di level global lalu dipanggil sekali.
- File baru: fungsi-fungsi didefinisikan **di dalam loop pasien**, lalu main() dipanggil per pasien.

📌 Dampak teknis:

- Ini valid di Python, tapi kurang efisien (fungsi didefinisikan ulang 50x).
- Namun tidak mengubah hasil akhir.

---

5) Isi metrik yang dihitung: sama secara inti

Keduanya sama-sama menghitung:

- IoU
- Dice
- Hausdorff Distance
- Mean Surface Distance

Dan sama-sama:

- mengubah mask PNG (255) → biner 0/1,
- melakukan evaluasi per-slice,
- menyimpan CSV per kelas,
- membuat 4 boxplot per pasien.

C) Ringkasan cepat

- hitung_evaluasi_metrik.py = evaluasi **U-Net** untuk **1 pasien** (contoh/cek detail).
- hitung_evaluasi_metrik_acdc2017.py = evaluasi **TransUNet** untuk **semua pasien test 101–150** + output CSV & boxplot per pasien.

Kalau kamu mau, aku bisa bantu refactor file baru supaya:

- fungsi tidak didefinisikan berulang di dalam loop,
- ada pengecekan folder exist (biar tidak crash jika pasien tertentu belum ada output),
- pairing prediksi-GT pakai **nama slice** (lebih aman daripada hanya urutan list).

10	File : hitung_f1score_2d.py

Bahasan:

Berikut penjelasan **lengkap** isi file **hitung_f1score_2d.py** dan saya pilah **berdasarkan blok fungsinya** (urut sesuai kode).
Sumber kode:

---

1) Header & metadata file

```
# -*- coding: utf-8 -*-
"""
Created on Fri May 17 21:26:53 2024
@author: ramad
"""
```

Fungsi blok ini:

- Menentukan encoding UTF-8.
- Metadata tanggal & author (komentar, tidak memengaruhi program).

---

2) Import library

```
import numpy as np
```

Fungsi:

- numpy dipakai untuk operasi array (walau di fungsi ini operasi numpy yang dipakai minimal).
- Ada baris konversi nilai yang dikomentari menggunakan np.where.

---

3) Fungsi utama: f1_score_2d(groundtruth, predict)

```
def f1_score_2d(groundtruth, predict):
```

Tujuan fungsi

Menghitung **F1-score (Dice/F1 untuk biner)** pada mask 2D antara:

- groundtruth (mask GT)
- predict (mask prediksi)

F1-score untuk segmentasi biner biasanya:

[
F1 = \frac{2TP}{2TP + FP + FN}
]

Formula yang dipakai di kode ini ekuivalen:

[
F1 = \frac{TP}{TP + 0.5(FP+FN)}
]

3.1) (Dikomentari) Konversi 255→1

```
# groundtruth = np.where(groundtruth == 255, 1, 0)
# predict = np.where(predict == 255, 1, 0)
```

Maksudnya:

- Kalau input mask masih bentuk PNG biner (0 dan 255), ini mengubahnya jadi 0 dan 1.
- Karena dikomentari, fungsi ini **mengasumsikan input sudah 0/1**.

📌 Implikasi:

- Kalau kamu memasukkan mask 0/255 tanpa mengubah, perbandingan == 1 akan gagal → TP bisa jadi 0 semua → F1 salah.

3.2) Inisialisasi variabel confusion counts

```
tp = 0
fp = 0
fn = 0
```

Arti:

- tp (True Positive): GT=1, Pred=1
- fp (False Positive): GT=0, Pred=1
- fn (False Negative): GT=1, Pred=0

3.3) Print shape untuk debugging

```
print(groundtruth.shape)
print(predict.shape)
```

Tujuan:

- Debugging memastikan ukuran GT dan Pred sama (misalnya (128,128)).
- Tapi side effect: kalau dipakai batch banyak slice, console akan sangat ramai.

3.4) Loop pixel-per-pixel menghitung TP/FP/FN

```
for i in range(groundtruth.shape[0]):
   for j in range(groundtruth.shape[1]):
      try:
         if groundtruth[i][j] == 1 and predict[i][j] == 1:
            tp += 1
         elif groundtruth[i][j] == 0 and predict[i][j] == 1:
            fp += 1
         elif groundtruth[i][j] == 1 and predict[i][j] == 0:
            fn += 1
```

```
    except IndexError as e:
        print(f"Error at index ({i}, {j}): {e}")
        break
```

Fungsi blok ini:

- Melakukan iterasi semua pixel 2D.
- Menambah counter berdasarkan kondisi nilai pixel.

Tentang try/except IndexError:

- Sebenarnya jika groundtruth.shape dipakai sebagai batas loop, groundtruth[i][j] aman.
- Potensi IndexError hanya jika predict lebih kecil dari groundtruth.
- Namun cara yang lebih tepat adalah **cek shape di awal** dan stop, bukan try/except di dalam loop (ini membuat loop lebih lambat).

3.5) Menghitung F1-score

```
if tp + fp + fn > 0:
    f1_score = tp / (tp + 0.5*(fp + fn))
else:
    f1_score = 0
```

Makna:

- Jika tidak ada kasus positif sama sekali (tp=fp=fn=0), F1 diset 0 agar tidak division by zero.
- Rumus yang dipakai ekuivalen dengan Dice/F1 biner:
    - tp / (tp + 0.5*(fp + fn)) = 2tp / (2tp + fp + fn)

📌 Catatan interpretasi:

- Jika GT kosong dan pred kosong, banyak literatur menganggap “perfect” (F1=1), tapi kode ini mengembalikan 0 karena denom = 0.
- Ini sama pola keputusan desain seperti yang kamu pakai di dice_coefficient (versi lain).

3.6) Return hasil

```
return f1_score
```

✅ **Ringkasan fungsi keseluruhan file**

File ini membuat **fungsi F1-score untuk mask 2D biner** dengan cara:

1. hitung TP/FP/FN secara pixel-by-pixel,
2. gunakan rumus F1,
3. jika tidak ada foreground sama sekali → return 0.

⚠️ **Catatan penting / perbedaan dengan implementasi umum**

1) Input harus 0/1 (karena konversi 255→1 dikomentari)

Kalau input kamu PNG hasil cv.imread(...,0) biasanya 0/255, maka perlu aktifkan baris konversi.

2) Performa lambat (loop Python dua tingkat)

Untuk gambar 128×128 mungkin masih oke, tapi kalau banyak slice/pasien, ini jauh lebih lambat daripada versi numpy-vectorized.

Versi cepat biasanya:

- tp = np.sum((gt==1) & (pred==1)), dst.

3) Penamaan komentar “Perhitungan IoU” keliru

```
# Perhitungan IoU untuk gambar saat ini
```

Padahal yang dihitung adalah **F1-score**, bukan IoU.

Kalau kamu mau, aku bisa bantu buat versi yang:

- otomatis handle 0/255 atau 0/1,
- lebih cepat (tanpa loop),
- dan bisa hitung F1 per kelas untuk multi-class segmentation.

11	File :hitung_iou_2d.py

Bahasan:
Berikut penjelasan **lengkap** isi file **hitung_iou_2d.py** dan saya pilah **berdasarkan blok fungsinya** (urut sesuai kode).
Sumber kode:

---

1 Header & metadata

```
# -*- coding: utf-8 -*-
"""
Created on Sun Jan 28 15:47:03 2024
@author: ramad
"""
```

Fungsi blok ini:

- Encoding UTF-8 + komentar metadata (tanggal/author).
- Tidak memengaruhi jalannya program.

---

2 Import library

```
from keras.metrics import MeanIoU
import numpy as np
```

Fungsi & catatan

- numpy dipakai untuk operasi array (walau implementasi IoU yang aktif tidak memakai operasi numpy yang kompleks).
- MeanIoU **diimport tetapi tidak digunakan** dalam kode aktif. Biasanya MeanIoU dipakai untuk menghitung IoU multi-class atau rata-rata IoU, tapi di sini implementasi IoU dibuat manual.

---

3 Fungsi utama: iou_2d(groundtruth, predict)

```
def iou_2d(groundtruth, predict):
```

Tujuan fungsi
Menghitung **Intersection over Union (IoU)** untuk **segmentasi biner 2D** antara:

- groundtruth (mask GT)
- predict (mask prediksi)

IoU (Jaccard index) untuk biner:

[
IoU = \frac{TP}{TP + FP + FN}
]

3.1) (Dikomentari) Konversi nilai 255 → 1

```
# groundtruth = np.where(groundtruth == 255, 1, 0)
# predict = np.where(predict == 255, 1, 0)
```

Maksudnya:

- Jika mask berasal dari PNG (umumnya 0 dan 255), blok ini mengubahnya ke 0 dan 1.

- Karena dikomentari, fungsi mengasumsikan input groundtruth dan predict **sudah 0/1**.

📌 Jika input masih 0/255:

- kondisi groundtruth[i][j] == 1 akan hampir selalu **False**
- IoU bisa salah (0 atau sangat kecil)

3.2) Inisialisasi confusion counts

```
tp = 0
fp = 0
fn = 0
```

Arti:

- **TP (True Positive)**: GT=1 dan Pred=1
- **FP (False Positive)**: GT=0 dan Pred=1
- **FN (False Negative)**: GT=1 dan Pred=0

3.3) (Debug) Cetak shape input

```
print(groundtruth.shape)
print(predict.shape)
```

Fungsi:

- Memastikan ukuran array sesuai.
- Tapi kalau dipakai evaluasi banyak slice/pasien, console akan sangat penuh output.

3.4) Hitung TP/FP/FN pixel-by-pixel (nested loop)

```
for i in range(groundtruth.shape[0]):
  for j in range(groundtruth.shape[1]):
    try:
      if groundtruth[i][j] == 1 and predict[i][j] == 1:
        tp += 1
      elif groundtruth[i][j] == 0 and predict[i][j] == 1:
        fp += 1
      elif groundtruth[i][j] == 1 and predict[i][j] == 0:
        fn += 1
    except IndexError as e:
      print(f"Error at index ({i}, {j}): {e}")
      break
```

Fungsi blok ini

- Loop seluruh piksel (2D).
- Tambah counter sesuai kondisi pixel.

Tentang try/except IndexError

- Error bisa terjadi kalau predict ukurannya lebih kecil dari groundtruth.
- Tetapi kalau ukuran sama, IndexError seharusnya tidak terjadi.
- Cara yang lebih bersih biasanya: cek groundtruth.shape == predict.shape di awal, lalu stop kalau beda.

📌 Kekurangan performa:

- Nested loop Python lambat dibanding operasi vektorisasi numpy.

3.5) Menghitung IoU (rumus utama)

```
if tp + fp + fn > 0:
  iou = tp / (tp + fp + fn)
```

else:
  iou = 0
return iou

Makna

- Jika tidak ada foreground sama sekali (tp=fp=fn=0), IoU jadi tidak terdefinisi (0/0), maka dikembalikan 0.
- Ini keputusan desain—beberapa implementasi mengembalikan 1 untuk kasus "keduanya kosong", tapi kode ini memilih 0.

4️⃣ Blok alternatif (dikomentari): IoU pakai logical_and / logical_or

Di bagian bawah ada implementasi yang lebih "numpy style" tapi dikomentari:

```
# intersection = np.logical_and(groundtruth, predict)
# union = np.logical_or(groundtruth, predict)
# if np.sum(union) == 0:
#    iou = 0
# else:
#    iou = np.sum(intersection) / np.sum(union)
# return iou
```

Tujuan blok ini

- Cara lebih ringkas dan biasanya lebih cepat:
  - intersection = area overlap
  - union = area gabungan
- Sama-sama butuh input biner.

✅ Ringkasan fungsi keseluruhan file

File hitung_iou_2d.py menyediakan fungsi:

- iou_2d(groundtruth, predict)
  yang menghitung IoU biner 2D dengan cara:

1. hitung TP/FP/FN secara pixel-by-pixel
2. terapkan rumus tp / (tp + fp + fn)
3. jika semua nol → return 0

⚠️ Catatan penting untuk penggunaan yang benar

1. **Pastikan input sudah biner 0/1**
- Kalau input 0/255, aktifkan konversi yang dikomentari atau konversi sebelum memanggil fungsi.
2. **Pastikan shape sama**
- Kalau tidak sama, IoU bisa salah atau memicu IndexError.
3. **Kalau dipakai massal, print shape sebaiknya dimatikan**
- Agar evaluasi tidak "banjir log".
4. **Bisa dipercepat dengan numpy**
- Implementasi di komentar (logical_and/or) jauh lebih cepat dan sederhana.

Kalau kamu mau, aku bisa buatkan versi IoU yang:

- otomatis menerima input 0/255 atau 0/1,
- tanpa nested loop,
- bisa hitung IoU per kelas (multi-class) sekaligus.

1 2	File : hitung_mcc_2d.py

Bahasan:
Berikut penjelasan **lengkap** isi file **hitung_mcc_2d.py** dan saya pilah **berdasarkan blok fungsinya** (urut sesuai kode).
Sumber kode:

---

1 Header & metadata file

```
# -*- coding: utf-8 -*-
"""
Created on Fri May 17 23:21:33 2024
@author: ramad
"""
```

Fungsi blok ini:

- Menentukan encoding UTF-8.
- Metadata tanggal dan penulis (komentar saja, tidak memengaruhi eksekusi program).

---

2 Import library

```
import numpy as np
```

Fungsi:

- numpy digunakan untuk operasi logika elemen-per-elemen (==, &) dan perhitungan statistik (sum, sqrt).

---

3 Fungsi utama: mcc_2d(groundtruth, predict)

```
def mcc_2d(groundtruth, predict):
```

Tujuan fungsi
Menghitung **Matthews Correlation Coefficient (MCC)** untuk **segmentasi biner 2D** antara:

- groundtruth (mask GT)
- predict (mask prediksi)

MCC adalah metrik evaluasi yang kuat untuk data **tidak seimbang (imbalanced)**, karena memperhitungkan:

- True Positive (TP)
- True Negative (TN)
- False Positive (FP)
- False Negative (FN)

Nilai MCC:

- +1 → prediksi sempurna
- 0 → prediksi acak
- -1 → prediksi sepenuhnya salah

3.1) Menghitung confusion matrix elements (TP, TN, FP, FN)

```
tp = np.sum((groundtruth == 1) & (predict == 1)).astype(np.float64)
tn = np.sum((groundtruth == 0) & (predict == 0)).astype(np.float64)
fp = np.sum((groundtruth == 0) & (predict == 1)).astype(np.float64)
fn = np.sum((groundtruth == 1) & (predict == 0)).astype(np.float64)
```

Penjelasan:

- Operasi (groundtruth == 1) & (predict == 1) menghasilkan array boolean:
    - True jika pixel benar-benar TP
- np.sum(...) menghitung jumlah pixel True
- .astype(np.float64):
    - memastikan tipe float
    - mencegah overflow integer pada perkalian besar

📌 **Asumsi penting**:

- Input groundtruth dan predict **harus biner 0/1**
- Jika masih 0/255 (PNG), harus dikonversi dulu.

3️⃣2) Menghitung pembilang (numerator) MCC

numerator = (tp * tn) - (fp * fn)

Ini adalah bagian atas dari rumus MCC:

[
\text{numerator} = TP \times TN - FP \times FN
]

3️⃣3) Menghitung penyebut (denominator) MCC

denominator = np.sqrt((tp + fp) * (tp + fn) * (tn + fp) * (tn + fn) + 1e-7)

Penjelasan:

Rumus matematis penyebut MCC:

[
\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}
]

- + 1e-7 (epsilon kecil):
    - mencegah pembagian nol
    - menjaga stabilitas numerik

📌 Walaupun epsilon sudah ditambahkan, kode **tetap melakukan pengecekan denominator == 0** sebagai pengaman tambahan.

3️⃣4) Menghindari pembagian nol

```
if denominator == 0:
   mcc = 0
else:
   mcc = numerator / denominator
```

Keputusan desain:

- Jika penyebut nol ⟶ MCC diset 0
- Alternatif di literatur:
    - kadang MCC didefinisikan 0 pada kasus tidak terdefinisi
    - keputusan ini konsisten dengan implementasi Dice & IoU di kode lain milikmu

---

3️⃣5) Return nilai MCC

return mcc

✅ **Ringkasan fungsi keseluruhan file**

File ini hanya berisi **1 fungsi**:

Fungsi	Keterangan

mcc_2d(gt, pred) Menghitung Matthews Correlation Coefficient untuk mask biner 2D

⚠️ **Catatan penting agar hasil benar**

1. **Pastikan input biner 0/1**
    - Jika input PNG (0/255), konversi dulu:
    - gt = (gt == 255).astype(int)
    - pred = (pred == 255).astype(int)
2. **MCC cocok untuk dataset tidak seimbang**
    - Lebih informatif daripada akurasi ketika background jauh lebih banyak daripada foreground (umum di segmentasi medis).
3. **Tidak ada loop pixel**
    - Implementasi ini **sudah vektorisasi numpy**, jauh lebih cepat dibanding versi IoU/F1 yang memakai nested loop.

Kalau kamu mau, aku bisa:

- menggabungkan IoU, Dice, F1, MCC ke **satu modul evaluasi umum**,
- menambahkan **handling otomatis 0/255 vs 0/1**,
- atau membuat fungsi **multi-class MCC (one-vs-rest)** untuk LV/RV/MYO.

13	File :hitung_surface_distance_2d.py

Bahasan:

Berikut penjelasan **lengkap** isi file **hitung_surface_distance_2d.py** dan saya pilah **berdasarkan blok fungsinya** (urut sesuai kode).
Sumber kode:

---

1 **Header & metadata**

```
# -*- coding: utf-8 -*-
"""
Created on Fri May 17 21:31:12 2024
@author: ramad
"""
```

Fungsi blok ini:

- Encoding UTF-8 + metadata (tanggal/author).
- Tidak memengaruhi eksekusi.

---

2 **Import library**

```
import numpy as np
from scipy.ndimage import morphology
```

Peran library

- **NumPy**: operasi logika array, konversi tipe, concatenate.
- **scipy.ndimage.morphology**: operasi morfologi biner dan distance transform, khususnya:
    - generate_binary_structure
    - binary_erosion
    - distance_transform_edt

📌 Catatan: modul scipy.ndimage.morphology di SciPy versi baru cenderung dianggap legacy (fungsi-fungsinya dipindah/diakses juga lewat scipy.ndimage), tapi tetap bekerja pada banyak environment.

3️⃣ **Fungsi utama surfd(input1, input2, sampling=1, connectivity=1)**

def surfd(input1, input2, sampling=1, connectivity=1):

Tujuan fungsi

Menghitung **Surface Distance** (jarak permukaan/boundary) antara dua mask biner:

- input1 = ground truth (atau salah satu mask)
- input2 = prediksi (atau mask pembanding)

Outputnya adalah **daftar jarak** (array) dari permukaan mask A ke permukaan mask B dan sebaliknya.

Biasanya dari daftar jarak ini dihitung ringkasan seperti:

- mean surface distance
- median surface distance
- max surface distance
- percentile (misal 95%)

Di script evaluasi kamu, hasil surfd kemudian diambil rata-ratanya (np.mean(...)).

3.1) Konversi input ke boolean dan minimal 1D

input_1 = np.atleast_1d(input1.astype(bool))

input_2 = np.atleast_1d(input2.astype(bool))

Fungsinya:

- astype(bool):
  - semua nilai non-zero ⟶ True (foreground)
  - 0 ⟶ False (background)
- np.atleast_1d:
  - memastikan minimal berdimensi 1 (jaga-jaga jika input scalar atau bentuk tidak sesuai)

📌 Implikasi bagus:

- Fungsi ini **tidak harus input 0/1** secara ketat.
- Selama foreground non-zero, akan dianggap True.

3.2) Membuat struktur konektivitas (kernel morfologi)

conn = morphology.generate_binary_structure(input_1.ndim, connectivity)

Makna:

- Membuat kernel tetangga untuk operasi morfologi sesuai dimensi data:
  - Kalau 2D: konektivitas 1 biasanya 4-neighbor, konektivitas 2 biasanya 8-neighbor.
  - Kalau 3D: konektivitas 1 biasanya 6-neighbor, konektivitas 2 bisa 18/26-neighbor tergantung.

Dalam konteks file ini (2D), parameter ini menentukan cara mendefinisikan “permukaan/batas”.

3.3) Ekstraksi “surface/boundary” dari masing-masing mask

S = np.logical_xor(input_1, morphology.binary_erosion(input_1, conn))

Sprime = np.logical_xor(input_2, morphology.binary_erosion(input_2, conn))

Penjelasan konsep:

- binary_erosion(input_1, conn) mengecilkan area foreground.
- input_1 XOR eroded(input_1) menghasilkan piksel yang “hilang saat erosi” ⟶ itu adalah **boundary/surface**.

Jadi:

- S = permukaan mask 1
- Sprime = permukaan mask 2

📌 Ini teknik standar untuk mengambil tepi objek pada mask biner.

3.4) Distance transform untuk menghitung jarak ke surface

dta = morphology.distance_transform_edt(~S, sampling)
dtb = morphology.distance_transform_edt(~Sprime, sampling)

Arti:

- distance_transform_edt menghitung untuk setiap pixel background, jarak Euclidean ke pixel foreground terdekat.
- Karena kita ingin jarak **ke boundary**, maka:
  - boundary dianggap sebagai "foreground"
  - maka kita pakai ~S (kebalikan) agar distance_transform_edt menghasilkan peta jarak yang sesuai terhadap boundary.

sampling:

- Bisa berupa:
  - angka skalar (misal 1)
  - atau tuple (sy, sx) untuk spacing piksel non-isotropik.
- Ini penting kalau jarak ingin dalam satuan fisik.

3.5) Mengambil jarak surface A ke surface B, dan sebaliknya

sds = np.concatenate([dta[Sprime != 0].ravel(), dtb[S != 0].ravel()])

Ini bagian inti:

- dta[Sprime != 0]:
  - ambil jarak dari semua titik boundary milik mask2 (Sprime) ke boundary mask1 (karena dta dibangun dari ~S)
- dtb[S != 0]:
  - ambil jarak dari semua titik boundary milik mask1 (S) ke boundary mask2

Lalu digabung (concatenate) jadi satu array jarak gabungan dua arah.

📌 Hasilnya adalah daftar jarak "symmetric surface distance".

3.6) Penanganan kasus kosong

```
if sds.size == 0:
   return 0
```

Kapan bisa kosong?

- Jika salah satu atau kedua mask tidak punya boundary (mis. mask kosong seluruhnya).
- Agar tidak error saat di-mean, fungsi mengembalikan 0.

📌 Catatan evaluasi:

- Secara definisi metrik, "kosong-kosong" kadang dianggap jarak 0 (masuk akal), tapi kalau salah satu kosong dan satu tidak, jarak seharusnya besar—di sini bisa saja tetap menghasilkan daftar jarak non-kosong tergantung kondisi boundary mana yang ada.

3.7) Return hasil

return sds

Output:

- Array 1D yang berisi jarak-jarak permukaan.
- Biasanya pemakai menghitung statistiknya:

- np.mean(sds)
- np.max(sds)
- np.percentile(sds, 95)

✅ **Ringkasan fungsi keseluruhan file**
File ini menyediakan **1 fungsi**:
surfd(maskA, maskB, sampling=..., connectivity=...)

- Mencari boundary kedua mask
- Menghitung distance transform
- Mengambil jarak boundary A→B dan B→A
- Mengembalikan kumpulan jarak surface distance (simetris)

⚠️ **Hal penting agar interpretasi benar**

1. **Input sebaiknya mask biner**
- Fungsi memang mengubah ke boolean, tapi kalau input label multi-class (0,1,2,3), semua label non-zero akan dianggap foreground jadi "kelas bercampur".
2. **sampling penting kalau ingin jarak fisik**
- Kalau pixel spacing bukan 1×1, sebaiknya isi sampling=(sy, sx).
3. **Return 0 saat kosong**
- Karena fungsi kadang mengembalikan integer 0 (bukan array), pemanggil perlu hati-hati.
  - Di kode evaluasi kamu, np.mean(surface) aman untuk array, tapi kalau surface=0, np.mean(0)=0 masih aman.

Kalau kamu mau, aku bisa bantu bikin versi yang:

- selalu mengembalikan array (meski kosong) agar tipe konsisten,
- bisa langsung mengembalikan mean/median/hd95 (95th percentile) agar evaluasi lebih standar,
- mendukung multi-class dengan cara "one-vs-rest" per kelas.

14	File :pisah_gt_myo.py

Bahasan:
Menurut sebuah dokumen dari **Thu Mar 28 2024** (sesuai header file), script **pisah_gt_myo.py** dipakai untuk **memisahkan (extract) kelas Myocardium (MYO)** dari mask ground truth multiclass (PNG) menjadi **mask biner (0/255)** lalu menyimpannya ke folder khusus gt_myo.
Di bawah ini penjelasan **lengkap dan dipilah per blok fungsinya**.

---

1) Header & Import Library
import cv2 as cv
import numpy as np
import os
Fungsi blok ini:

- **cv2**: baca mask PNG grayscale (cv.imread) dan simpan hasil mask biner (cv.imwrite)
- **numpy**: cari nilai unik kelas pada mask (np.unique), buat mask biner (np.where)
- **os**: akses folder (os.listdir), join path, ambil waktu modifikasi file (os.path.getmtime)

---

2) Konfigurasi Path Input & Output

```
direktori_gt = ".../groundtruth/"
simpan_gt_myo = ".../gt_myo/"
```

Fungsi blok ini:

- direktori_gt: folder berisi **groundtruth mask** (PNG) yang masih **belum dipisah per kelas**
- simpan_gt_myo: folder tujuan untuk menyimpan **mask myocardium** hasil pemisahan

📌 Catatan: script ini **tidak membuat folder output** (os.makedirs) — jadi folder gt_myo harus sudah ada, kalau tidak cv.imwrite bisa gagal menyimpan.

---

3) Fungsi Sorting Berdasarkan Waktu Modifikasi

```
def get_mask_modification_time(item):
    item_path = os.path.join(direktori_gt, item)
    return os.path.getmtime(item_path)
```

Tujuan blok ini:

- Mengambil **timestamp modifikasi terakhir** dari tiap file mask.
- Dipakai sebagai key untuk sorting supaya urutan file mengikuti urutan perubahan/penyimpanan file di OS.

📌 Ini berbeda dari sorting “nama file”; kalau kamu butuh urutan slice berdasarkan angka/filename, pendekatan ini bisa kurang stabil.

---

4) Fungsi Inti: pisah_kelas_myo(sorted_masks)

```
def pisah_kelas_myo(sorted_masks):
    for filenames in sorted_masks:
        if filenames.endswith('.png'):
            file_path = os.path.join(direktori_gt, filenames)
            mask = cv.imread(file_path, 0)
            unique_values = np.unique(mask)
```

4.1 Loop semua file PNG

- Mengiterasi list sorted_masks
- Hanya memproses file yang berekstensi .png
- Membaca mask sebagai grayscale (0) sehingga mask adalah array 2D berisi nilai kelas.

4.2 Debug nilai unik

```
print("Unique values in the mask:", unique_values)
```

- Untuk memastikan kelas apa saja yang muncul pada mask itu (misalnya [0, 1, 2] atau [0, 1, 2, 3]).

4.3 Logika pemilihan “kelas MYO” berdasarkan jumlah nilai unik

```
if len(unique_values) == 3:
    kelas_target = np.where(mask == unique_values[1], 255, 0).astype(np.uint8)
elif len(unique_values) == 4:
    kelas_target = np.where(mask == unique_values[2], 255, 0).astype(np.uint8)
else:
    kelas_target = np.zeros_like(mask)
```

Ini inti pemisahan kelasnya.

Maknanya:

- Jika mask hanya punya **3 nilai unik** (contoh umum: [0, 1, 2]):
    - script menganggap kelas target = unique_values[1] (nilai unik ke-2)
- Jika mask punya **4 nilai unik** (contoh umum: [0, 1, 2, 3]):
    - script menganggap kelas target = unique_values[2] (nilai unik ke-3)

- Selain itu, hasilnya dibuat **mask kosong** (semua 0)

📌 Interpretasi yang tersirat:

- Script ini mencoba "menebak" kelas myocardium berdasarkan **jumlah kelas yang muncul** pada slice tersebut.
- Karena pada sebagian slice bisa saja hanya muncul 2 struktur (misal RV tidak tampak), maka jumlah nilai unik bisa 3; pada slice lain bisa 4.

⚠️ Risiko penting:

- Ini **bergantung pada urutan unique_values**, bukan mapping kelas yang eksplisit (mis. MYO selalu label 2).
- Kalau encoding mask kamu berubah (mis. nilai 85/170/255) atau urutan label tidak sesuai asumsi, maka MYO bisa **tertukar** dengan kelas lain.

4.4 Menentukan nama output & menyimpan hasil

```
base = os.path.splitext(filenames)[0]
name_file_mask = os.path.join(simpan_gt_myo, str(base) + ".png")
cv.imwrite(name_file_mask, kelas_target)
```

- Nama file output **sama dengan input** (hanya memastikan ekstensi .png)
- Disimpan ke folder gt_myo
- Isi output adalah mask biner: **255 untuk kelas target**, 0 untuk background.

4.5 Logging selesai

```
print(f"Citra {filenames} sudah diproses\n")
...
print("Proses Resize Citra mask Myocardium Sudah Selesai \n")
```

Catatan: teks "Resize" di print, tapi script ini **tidak melakukan resize**—hanya pemisahan kelas dan penyimpanan mask biner.

---

5) Fungsi main() (Alur Eksekusi Utama)

```
def main():
  get_masks = os.listdir(direktori_gt)
  sorted_masks = sorted(get_masks, key=get_mask_modification_time)
  pisah_kelas_myo(sorted_masks)
```

Fungsi blok ini:

1. Mengambil semua nama file dalam direktori_gt
2. Mengurutkan file berdasarkan **waktu modifikasi**
3. Memanggil proses pemisahan kelas MYO

---

6) Entry Point

```
if __name__ == "__main__":
  main()
```

Artinya: saat file dijalankan langsung, main() dieksekusi.

Kalau kamu ingin versi yang lebih "aman"

Dari sisi akurasi, biasanya lebih aman kalau MYO ditentukan eksplisit, misalnya:

- kalau ACDC standard label: MYO = 2 → mask == 2
  daripada menebak dari unique_values[1] atau [2].

Kalau kamu kasih tahu "label MYO itu angka berapa" di dataset PNG-mu (mis. 2 atau 170), aku bisa bantu tuliskan versi script yang lebih robust (tanpa risiko tertukar).

15	File :predict_trans_unet.py

Bahasan:

GAMBARAN UMUM SCRIPT

Script ini digunakan untuk:

Melakukan prediksi segmentasi citra jantung 2D fase ED menggunakan model TransUNet, kemudian **menyimpan hasil prediksi** dalam:

- citra prediksi gabungan (visualisasi)
- mask biner per kelas:
  - Right Ventricle (RV)
  - Myocardium (MYO)
  - Left Ventricle (LV)

Script ini adalah **script inferensi**, bukan training.

BLOK 1 — HEADER & IMPORT LIBRARY

```
from keras.models import load_model
import cv2 as cv
import os
import numpy as np
import glob
import re
...
import tensorflow as tf
...
from tensorflow.image import extract_patches
from tensorflow.nn import depth_to_space
```

Fungsi blok ini

Menyiapkan **semua dependensi** yang diperlukan untuk:

- membaca & menyimpan citra (cv2)
- manipulasi array (numpy)
- pencarian file dataset (glob, os)
- ekstraksi ID pasien (re)
- memuat model deep learning (keras, tensorflow)
- mendefinisikan **custom layer TransUNet / Swin Transformer**

📌 **Kenapa import-nya sangat banyak?**

Karena **model TransUNet menggunakan banyak custom layer** yang **HARUS didefinisikan ulang saat inferensi**, kalau tidak load_model() akan gagal.

BLOK 2 — KONFIGURASI PATH DATASET

```
direktori_citra = ".../Data Test Resize 128 ED/"
direktori_prediksi_ed = ".../Hasil Predict Versi 2/transunet/focalctc/"
```

Fungsi

- direktori_citra
  → lokasi **citra ED 2D** yang akan diprediksi (resize 128×128)
- direktori_prediksi_ed
  → folder utama untuk menyimpan **hasil prediksi**

Struktur input yang diasumsikan:
Pasien 101/images/Pasien101_1_ed.png

BLOK 3 — FUNGSI UTILITAS WAKTU MODIFIKASI FILE
def get_image_modification_time(item):
Fungsi
Mengambil waktu modifikasi file (os.path.getmtime).
📌 **Catatan penting**

- Fungsi ini **TIDAK dipakai** di alur utama script
- Kemungkinan sisa eksperimen lama (sorting berdasarkan waktu)

Aman untuk dihapus jika ingin merapikan kode.

BLOK 4 — CUSTOM LAYER: PATCH EXTRACT
class patch_extract(Layer):
Fungsi
Mengubah feature map CNN menjadi **patch-token** (ViT style).
Cara kerja
Input:
(B, H, W, C)
Output:
(B, N_patch, patch_dim)
Teknik:

- tf.image.extract_patches
- setiap patch ⟶ satu token transformer

📌 Ini adalah **fondasi Vision Transformer / TransUNet**.

BLOK 5 — CUSTOM LAYER: PATCH EMBEDDING
class patch_embedding(Layer):
Fungsi

- Mengubah patch menjadi **embedding vektor**
- Menambahkan **positional embedding**

Tanpa ini:

- Transformer **tidak tahu posisi spasial patch**

BLOK 6 — PATCH MERGING (DOWNSAMPLING)
class patch_merging(tf.keras.layers.Layer):
Fungsi

- Menggabungkan 4 patch tetangga
- Resolusi ↓, channel ↑

Perannya mirip:

- pooling / encoder U-Net

BLOK 7 — PATCH EXPANDING (UPSAMPLING)
class patch_expanding(tf.keras.layers.Layer):
Fungsi

- Upsampling token
- Menggunakan depth_to_space (pixel shuffle)
- Kebalikan patch_merging

Perannya mirip:

- decoder / upsampling U-Net

BLOK 8 — SWIN TRANSFORMER UTILITIES

8.1 Window Partition & Reverse

def window_partition(...)
def window_reverse(...)

Fungsi

- Memecah feature map menjadi **window kecil**
- Attention dilakukan **lokal**, lebih efisien

8.2 Drop Path (Stochastic Depth)

class drop_path(Layer):

Fungsi

- Regularisasi
- Mematikan jalur residual secara acak (saat training)

Saat inferensi:

- tidak berdampak signifikan

8.3 MLP Block

class Mlp(tf.keras.layers.Layer):

Fungsi

Blok standar transformer:
Dense ⟶ GELU ⟶ Dropout ⟶ Dense ⟶ Dropout

8.4 Window Attention

class WindowAttention(tf.keras.layers.Layer):

Fungsi

- Self-attention dalam window
- Menggunakan **relative position bias**
- Mendukung shifted window

Ini inti dari **Swin Transformer**.

8.5 SwinTransformerBlock

class SwinTransformerBlock(tf.keras.layers.Layer):

Fungsi

Blok transformer lengkap:

1. LayerNorm
2. Window Attention
3. Residual
4. LayerNorm
5. MLP
6. Residual

📌 Ini adalah **otak utama model**.

BLOK 9 — CUSTOM ACTIVATION: GELU & SNAKE

class GELU(Layer)
class Snake(Layer)

Fungsi

- **GELU** ⟶ aktivasi smooth (umum di transformer)
- **Snake** ⟶ aktivasi periodik (menangkap detail tekstur)

📌 Karena **bukan aktivasi standar Keras**, harus didefinisikan ulang saat load model.

BLOK 10 — FUNGSI MEMBUAT STRUKTUR FOLDER OUTPUT

def membuat_direktori_2d_ed_prediksi(id_pasien, path_direktori):

Fungsi

Membuat struktur folder:

Prediksi Pasien/
└── Pasien X/
  ├── citra prediksi
  ├── prediksi right ventricel
  ├── prediksi myocardium
  └── prediksi left ventricel

📌 Tanpa ini cv.imwrite() bisa gagal.

BLOK 11 — FUNGSI INTI PREDIKSI

def prediksi_citra(imgs, model):

Ini adalah **blok TERPENTING**.

11.1 Loop citra input

- Ambil ID pasien dari nama file (regex)
- Buat folder output pasien

11.2 Preprocessing citra

```
img = cv.imread(img_path, 0)
train_images = np.repeat(img[..., np.newaxis], 3, axis=-1)
```

Kenapa?

- Model TransUNet dilatih dengan **input 3-channel**
- Padahal citra medis grayscale ⟶ perlu direplikasi

11.3 Inferensi model

```
prediction = model.predict(test_img_input)
predicted_img = np.argmax(prediction, axis=3)[0,:,:]
```

Hasil:

(H, W) dengan label {0,1,2,3}

11.4 Simpan mask gabungan (visualisasi)

```
predicted_img_norm = (predicted_img / predicted_img.max()) * 255
```

📌 **HANYA UNTUK VISUALISASI**, bukan evaluasi numerik.

11.5 Simpan mask per kelas

Kelas	Label	Output
RV	1	binary PNG
MYO	2	binary PNG
LV	3	binary PNG

Jika kelas tidak muncul → simpan citra kosong (0).

BLOK 12 — FUNGSI MAIN
def main():
Alur

1. Definisikan custom_objects
2. Load model TransUNet (compile=False)
3. Ambil semua citra ED
4. Jalankan prediksi

BLOK 13 — ENTRY POINT

```
if __name__ == "__main__":
   main()
```

RINGKASAN ALUR KESELURUHAN
Load model TransUNet
↓
Load citra ED 2D
↓
Grayscale → 3 channel
↓
Inferensi per slice
↓
Argmax → label map
↓
Simpan:
- mask gabungan
- RV
- MYO
- LV

CATATAN KRITIS (PENTING)

1. **Mask gabungan (0–255) jangan dipakai untuk evaluasi**
2. Evaluasi IoU/Dice/HD harus pakai **mask biner per kelas**
3. Custom layer WAJIB didefinisikan saat load model
4. Jika predicted_img.max() == 0, normalisasi bisa error (division by zero) → perlu guard

16	File : predict_unet_2d.py

Bahasan:
GAMBARAN UMUM SCRIPT
Script ini berfungsi untuk:
Melakukan prediksi segmentasi citra jantung 2D (fase ED) menggunakan model U-Net, kemudian **menyimpan hasil prediksi** dalam bentuk:

- citra prediksi gabungan (label → grayscale)
- citra biner per kelas:

- Right Ventricle (RV)
- Myocardium (MYO)
- Left Ventricle (LV)

Script ini adalah **script inferensi (testing)**, bukan training.

BLOK 1 — HEADER & IMPORT LIBRARY

```
from keras.models import load_model
import cv2 as cv
import os
import numpy as np
import glob
import re
```

Fungsi blok ini

Menyiapkan library yang dibutuhkan untuk:

Library	Fungsi
load_model	Memuat model U-Net terlatih
cv2	Baca & simpan citra PNG
numpy	Manipulasi array & masking
glob	Mengambil daftar file citra
os	Manajemen folder
re	Ekstraksi ID pasien dari nama file

📌

LabelEncoder dan to_categorical **di-import tapi tidak digunakan** → sisa eksperimen lama, aman dihapus.

BLOK 2 — KONFIGURASI PATH DATASET

```
direktori_citra = ".../Data Test Resize 128 ED/"
direktori_prediksi_ed = ".../Hasil Predict 2D versi 3/unet/msefocalctc/"
```

Fungsi

- direktori_citra
  → lokasi citra **test ED 2D** (ukuran 128×128)
- direktori_prediksi_ed
  → folder utama tempat menyimpan **hasil prediksi U-Net**

Struktur input yang diasumsikan:

Pasien 101/images/Pasien101_1_ed.png

BLOK 3 — FUNGSI UTILITAS (TIDAK DIPAKAI)

```
def get_image_modification_time(...)
def get_mask_modification_time(...)
def get_predict_modification_time(...)
```

Fungsi

Mengambil waktu modifikasi file (os.path.getmtime).

📌 **Catatan penting**

- Ketiga fungsi ini **tidak pernah dipanggil**
- direktori_mask dan direktori_prediksi bahkan **tidak didefinisikan**

- Ini **sisa kode lama**, tidak memengaruhi prediksi

➡️ Aman dihapus jika ingin merapikan kode.

BLOK 4 — PEMBUATAN STRUKTUR DIREKTORI OUTPUT

def membuat_direktori_2d_ed_prediksi(id_pasien, path_direktori):

Fungsi

Membuat struktur folder output **per pasien**:

```
Prediksi Pasien/
└── Pasien <id>/
    ├── citra prediksi/
    ├── prediksi right ventricle/
    ├── prediksi myocardium/
    └── prediksi left ventricle/
```

Kenapa ini penting?

- cv.imwrite() **akan gagal** jika folder belum ada
- exist_ok=True → aman walau folder sudah dibuat sebelumnya

BLOK 5 — FUNGSI INTI PREDIKSI U-NET

def prediksi_citra(imgs, model):

Ini adalah **blok TERPENTING** dalam script.

5.1 Loop semua citra input

for img_path in imgs:

- imgs berisi daftar semua citra ED
- Diproses satu per satu (slice-wise)

5.2 Ekstraksi ID pasien dari nama file

match = re.search(r'Pasien(\d+)_', ...)

Fungsi

Mengambil nomor pasien (misalnya 101) dari nama file:

Pasien101_1_ed.png

Jika gagal → file dilewati.

5.3 Pembuatan folder output pasien

membuat_direktori_2d_ed_prediksi(...)

Menjamin semua folder penyimpanan tersedia sebelum menyimpan hasil prediksi.

5.4 Preprocessing citra (KHUSUS U-NET)

```
img = cv.imread(img_path, 0)
test_images = np.expand_dims(img_array, axis=-1)
test_images = test_images / 255.0
```

Penjelasan

- Baca citra **grayscale**
- Tambah **channel = 1**
- Normalisasi ke **[0–1]**

📌 **Berbeda dengan Trans-UNet**

U-Net ini **dilatih dengan input 1 channel**, bukan 3 channel.

5.5 Penyiapan input batch
test_img_input = np.expand_dims(test_img_norm, 0)
Shape akhir input:
(1, 128, 128, 1)
Ini **wajib** agar sesuai input model.predict().

5.6 Eksekusi prediksi U-Net
prediction = model.predict(test_img_input)
predicted_img = np.argmax(prediction, axis=3)[0,:,:]
Penjelasan

- prediction ⟶ probabilitas kelas per pixel
- argmax ⟶ memilih kelas dengan probabilitas tertinggi

Output:
(128, 128) dengan label:
0 = background
1 = RV
2 = MYO
3 = LV
📌 **Inilah titik prediksi U-Net benar-benar terjadi**

5.7 Normalisasi untuk visualisasi
predicted_img_norm = (predicted_img / predicted_img.max()) * 255
📌 **Catatan KRITIS**

- Ini **hanya untuk visualisasi**
- **JANGAN dipakai untuk evaluasi IoU/Dice**

5.8 Penyimpanan hasil prediksi
a) Citra prediksi gabungan
cv.imwrite(name_file_predict, predicted_img_uint8)

---

b) Mask biner per kelas

Kelas	Label	Output
RV	1	predicted_img == 1
MYO	2	predicted_img == 2
LV	3	predicted_img == 3

Jika suatu kelas **tidak muncul**, disimpan citra kosong (semua 0)
⟶ **format output konsisten untuk evaluasi**

BLOK 6 — FUNGSI MAIN
def main():
Alur utama

1. Load model U-Net:

model = load_model(..., compile=False)

2. Ambil semua citra ED:

glob.glob(direktori_citra + '*/images/Pasien*_*_ed.png')

3. Jalankan prediksi:

prediksi_citra(list_img_ed_2d, model)

📌 compile=False → cukup untuk inferensi

BLOK 7 — ENTRY POINT

```
if __name__ == "__main__":
   main()
```

Menandakan script dijalankan langsung.

RINGKASAN ALUR KERJA

Load model U-Net
↓
Load citra ED (grayscale)
↓
Normalisasi & reshape
↓
model.predict()
↓
argmax → label map
↓
Simpan:
- citra prediksi
- RV
- MYO
- LV

CATATAN TEKNIS PENTING (KRITIS)

1. **Input 1 channel → khas U-Net**
2. **Mask gabungan hanya untuk visualisasi**
3. **Evaluasi IoU/Dice harus pakai mask biner**
4. Jika predicted_img.max() == 0, normalisasi bisa error → sebaiknya ditambahkan guard

17	File : slice3dto2d.py

Bahasan:

GAMBARAN UMUM SCRIPT

Script ini berfungsi untuk:
Mengubah data citra jantung 3D (format NIfTI .nii/.nii.gz) menjadi citra 2D (PNG) untuk **fase ED dan ES**, lalu **melakukan resize ke 128×128**, dan (khusus testing ED) **memisahkan ground truth menjadi kelas RV, MYO, dan LV**.
Script ini adalah **fondasi dataset 2D** yang nantinya dipakai oleh:

- predict_unet_2d.py
- predict_trans_unet.py
- script evaluasi (Dice, IoU, HD, Surface Distance)

BLOK 1 — HEADER & IMPORT LIBRARY

```
import cv2 as cv
import numpy as np
import os
import nibabel as nib
import glob
import re
```

Fungsi blok ini

Menyiapkan library utama:

Library	Fungsi
nibabel	Membaca data MRI 3D (.nii, .nii.gz)
cv2	Normalisasi, resize, simpan PNG
numpy	Manipulasi array & masking
glob	Mengambil daftar file
os	Manajemen direktori
re	Ekstraksi nomor pasien

BLOK 2 — KONFIGURASI PATH DATASET

```
dataset_path = ".../testingACDC17_baru/"
```

Direktori utama

- **Input 3D (NIfTI)**
  → testingACDC17_baru/patientXXX_frameXX.nii.gz
- **Output 2D per pasien**
  - ED → Data 2D/ED/Data Per Pasien Testing 2D/
  - ES → Data 2D/ES/Data Per Pasien Testing 2D/
- **Output resize 128×128**
  - ED → Data 2D/ED/
  - ES → Data 2D/ES/

BLOK 3 — FUNGSI PEMBUATAN DIREKTORI (UTILITY)

3.1 membuat_direktori_2d_ed()

Membuat struktur dasar ED:

```
Pasien X/
 ├── images/
 └── groundtruth/
```

Dipakai untuk:

- slicing 3D → 2D (ED)

3.2 membuat_direktori_2d_ed_testing()

Versi **khusus testing ED**, lebih lengkap:

```
Pasien X/
 ├── images/
 ├── groundtruth/
 ├── groundtruth right ventricel/
 ├── groundtruth myocardium/
 └── groundtruth left ventricel/
```

Dipakai untuk:

- testing
- evaluasi per kelas (RV, MYO, LV)

3.3 membuat_direktori_2d_es()
Struktur sederhana ES:
Pasien X/
├── images/
└── groundtruth/

BLOK 4 — SLICE 3D → 2D FASE ED
def slice_ed(list_img_ed, list_mask_ed):
Fungsi utama
Mengubah **volume MRI 3D ED** menjadi **sekumpulan citra 2D slice-wise**.

Alur kerja

1. Loop tiap pasien (enumerate(..., start=101))
2. Load citra & GT:
3. imgs = nib.load(img_path).get_fdata()
4. masks = nib.load(mask_path).get_fdata()
5. Ambil jumlah slice:
6. num_slices = imgs.shape[2]
7. Loop setiap slice:
    - Ambil irisan axial
    - Normalisasi ke 0–255
    - Konversi ke uint8
8. Simpan:
9. PasienX_sliceY_ed.png
10. PasienX_sliceY_ed_gt.png

📌 **Output tahap ini:**

- Data 2D **belum resize**
- Masih resolusi asli MRI

BLOK 5 — SLICE 3D → 2D FASE ES
def slice_es(list_img_es, list_mask_es):
Perbedaan dengan ED

- Citra dan GT **diproses terpisah**
- Loop pertama → citra
- Loop kedua → ground truth

📌 Digunakan jika:

- ingin memisahkan proses citra & GT ES
- kontrol lebih fleksibel

BLOK 6 — RESIZE DATA TRAIN ED (128×128)
def resize_img_ed(sorted_imgs, sorted_masks):
Fungsi

- Resize citra & GT ED ke **128×128**
- Interpolasi GT: INTER_NEAREST (WAJIB agar label tidak rusak)

Output:

Data Train Resize 128 ED/
├── images/
└── groundtruth/

BLOK 7 — RESIZE DATA TRAIN ES (128×128)
def resize_img_es(sorted_imgs, sorted_masks):
Identik dengan ED, hanya beda fase (ES).

BLOK 8 — RESIZE DATA TESTING ED + PEMISAHAN KELAS
def resize_img_ed_test(sorted_imgs, sorted_masks):
INI BLOK PALING PENTING UNTUK TESTING & EVALUASI

Alur kerja

1. Ekstrak **nomor pasien** dari nama file
2. Buat folder testing ED lengkap (dengan RV/MYO/LV)
3. Resize citra & mask ke 128×128
4. Cek nilai unik mask:
5. unique_values = np.unique(mask_resize)

Pemisahan kelas berdasarkan intensitas

Nilai mask	Kelas
85	RV
170	MYO
255	LV

kelas1 = mask_resize == 85
kelas2 = mask_resize == 170
kelas3 = mask_resize == 255
Disimpan sebagai **mask biner 0/255**.
📌 Ini **krusial** agar:

- evaluasi Dice / IoU per kelas valid
- konsisten dengan output model

BLOK 9 — FUNGSI MAIN (PIPELINE EKSEKUSI)
def main():
Tahapan pipeline

1. Ambil daftar file 3D:
    - ED → frame01
    - ES → selain frame01
2. (Opsional, dikomentari)
    - slice_ed()
    - slice_es()
3. Ambil daftar slice 2D
4. (Opsional)
    - resize train ED / ES
5. **Eksekusi aktif:**
6. resize_img_ed_test(...)

📌 Artinya:
Script ini **saat ini dipakai untuk menyiapkan DATA TESTING ED**.

BLOK 10 — ENTRY POINT

```
if __name__ == "__main__":
    main()
```

RINGKASAN ALUR DATA KESELURUHAN

MRI 3D (.nii)
↓
Slice axial → PNG 2D
↓
Resize 128×128
↓
Pisah ground truth:
RV / MYO / LV
↓
Siap untuk:
- prediksi U-Net / Trans-UNet
- evaluasi Dice / IoU / HD

CATATAN TEKNIS PENTING (KRITIS)

1. **Normalisasi GT ke 0–255**
   - Aman karena konsisten
   - Tapi label numerik asli hilang (tidak masalah untuk PNG)
2. **Resize GT wajib INTER_NEAREST**
   - Sudah benar ✅
3. **Pemisahan kelas berbasis intensitas**
   - HARUS konsisten dengan data asli ACDC
4. **Script ini adalah fondasi**
   - Jika salah di sini → semua hasil prediksi & evaluasi ikut salah

18	File :split_data.py

Bahasan:

GAMBARAN UMUM SCRIPT

Script ini digunakan untuk:
Membagi dataset citra ke dalam folder Training dan Validation secara otomatis
berdasarkan **rasio tertentu (90% : 10%)**, dengan struktur folder yang rapi dan konsisten.
Biasanya script ini dipakai **sebelum training model** (U-Net, Trans-UNet, CNN, dsb).

BLOK 1 — HEADER FILE

```
# -*- coding: utf-8 -*-
"""
Created on Wed Feb 14 09:36:55 2024
@author: ramad
```

"""

Fungsi

- Menandai encoding file (utf-8)
- Metadata waktu pembuatan & penulis
- Tidak memengaruhi eksekusi kode

BLOK 2 — IMPORT LIBRARY

import splitfolders

Fungsi

Mengimpor library **splitfolders**, yang digunakan untuk:

- membagi dataset berbasis folder
- mempertahankan struktur subfolder kelas

📌 Library ini sangat cocok untuk:

- dataset klasifikasi citra
- dataset segmentasi (images / masks)
- dataset dengan banyak kelas

BLOK 3 — KONFIGURASI PATH DATASET

input_folder = '.../Dataset Olah Pertama/'
output_folder = '.../Dataset Olah Kedua/Citra Asli/'

Fungsi

Menentukan:

- **input_folder** → lokasi dataset awal (belum terbagi)
- **output_folder** → lokasi dataset hasil split

📌 Struktur input biasanya seperti:

Dataset Olah Pertama/
├── kelas_1/
├── kelas_2/
└── kelas_3/

BLOK 4 — PROSES SPLIT DATASET

```
splitfolders.ratio(
   input_folder,
   output=output_folder,
   seed=42,
   ratio=(.90, .10),
   group_prefix=None
)
```

Fungsi Utama

Melakukan **pembagian dataset secara otomatis**.

Penjelasan Parameter

Parameter	Fungsi
input_folder	Folder sumber dataset
output	Folder tujuan hasil split
seed=42	Seed random agar split **reproducible**
ratio=(.90, .10)	90% train, 10% validation

group_prefix=None Tidak mengelompokkan file berdasarkan prefix

---

Struktur Output yang Dihasilkan
Setelah dijalankan, folder output akan berisi:
Citra Asli/
 ├── train/
 │   ├── kelas_1/
 │   ├── kelas_2/
 │   └── kelas_3/
 └── val/
     ├── kelas_1/
     ├── kelas_2/
     └── kelas_3/
📌 **Struktur kelas tetap terjaga**
Setiap kelas di-split secara proporsional.

BLOK 5 — CATATAN PENTING (BERDASARKAN KOMENTAR KODE)
To only split into training and validation set, set a tuple to `ratio`, i.e, `(.8, .2)`
Penjelasan

- Jika ingin:
    - **Train + Validation** → (train_ratio, val_ratio)
    - **Train + Validation + Test** → (train, val, test)

Contoh:
ratio=(0.7, 0.2, 0.1)

PERAN SCRIPT INI DALAM PIPELINE KAMU
Script ini biasanya dijalankan **SEBELUM**:

- training U-Net / Trans-UNet
- training CNN / Vision Transformer
- data augmentation

Pipeline umum:
Dataset mentah
↓
splitfolders (train / val)
↓
Training model
↓
Prediksi & evaluasi

KELEBIHAN MENGGUNAKAN splitfolders
✅ Otomatis
✅ Reproducible (seed)
✅ Struktur kelas konsisten
✅ Tidak perlu script manual

CATATAN TEKNIS PENTING

1. **Pastikan jumlah data cukup**

- Kelas kecil bisa jadi sangat sedikit di validation

2. **Segmentasi (image-mask)**
   - Jika images & masks terpisah → split **harus konsisten**
3. **Seed jangan diubah**
   - Agar hasil training bisa direproduksi

RINGKASAN SINGKAT

Import splitfolders
↓
Tentukan folder input & output
↓
Split dataset (90% train, 10% val)
↓
Dataset siap untuk training

19	File : test_gambar.py

Bahasan:

GAMBARAN UMUM SCRIPT

Script ini memiliki **3 tujuan utama**:

1. **Melakukan prediksi segmentasi citra jantung 3D (ED) menggunakan model U-Net 3D**
2. **Mengonversi hasil prediksi dan ground truth menjadi citra 2D per slice**
3. **Menghitung metrik evaluasi (Dice & Hausdorff Distance) untuk kelas Myocardium**

Script ini **bukan hanya prediksi**, tetapi **end-to-end pipeline evaluasi**:
Data 3D → Prediksi → Slice 2D → Simpan → Hitung metrik → Simpan CSV

BLOK 1 — HEADER & IMPORT LIBRARY

```
from keras.models import load_model
import numpy as np
import random
from matplotlib import pyplot as plt
import os
import cv2 as cv
from hausdorff.hausdorff import hausdorff_distance
from hitung_dice_2d import dice_coefficient
import csv
```

Fungsi blok ini

Menyiapkan seluruh dependensi:

Library	Fungsi
keras.load_model	Memuat model U-Net 3D
numpy	Manipulasi array 3D/2D
matplotlib	Menyimpan slice sebagai PNG
cv2	Membaca citra hasil prediksi
hausdorff_distance	Menghitung Hausdorff Distance
dice_coefficient	Menghitung Dice Score

csv Menyimpan hasil evaluasi

📌 Script ini **fokus pada evaluasi Myocardium**, bukan semua kelas.

BLOK 2 — KONFIGURASI PATH DATASET
direktori_citra_ed
direktori_gt_ed
simpan_prediksi_ed
simpan_gt_ed
direktori_prediksi_ed_myo
direktori_gt_ed_myo
simpan_csv
Fungsi
Menentukan **lokasi data**:

Path	Isi
direktori_citra_ed	Data citra 3D ED (.npy)
direktori_gt_ed	Ground truth 3D ED (.npy)
simpan_prediksi_ed	Output prediksi 2D
simpan_gt_ed	Output GT 2D
direktori_prediksi_ed_myo	Mask Myo prediksi
direktori_gt_ed_myo	Mask Myo GT
simpan_csv	File hasil evaluasi

BLOK 3 — FUNGSI SORTING BERDASARKAN WAKTU FILE
get_image_modification_time
get_mask_modification_time
get_predict_myo_modification_time
get_gt_myo_modification_time
Fungsi
Mengambil **timestamp file** untuk:

- memastikan urutan slice **sinkron** antara prediksi & ground truth

📌 Ini penting agar:
slice ke-n prediksi dibandingkan dengan slice ke-n ground truth

---

BLOK 4 — PEMBUATAN STRUKTUR DIREKTORI
4.1 Ground Truth
membuat_direktori_gt()
Struktur:
Pasien X/
├── Groundtruth Keseluruhan
└── Groundtruth Myo
📌 Hanya Myocardium yang dipakai untuk evaluasi.

4.2 Prediksi
membuat_direktori_prediksi()

Struktur:
Pasien X/
├── Predict Keseluruhan
├── Predict RV
├── Predict Myo
└── Predict LV

BLOK 5 — SLICE GROUND TRUTH 3D → 2D
def slice_gt(mask_ed):
Fungsi

- Membaca GT 3D .npy
- Mengambil argmax kelas
- Menyimpan setiap slice sebagai PNG

Proses:

1. Load GT 3D
2. argmax(axis=3) → label map
3. Loop slice:
    - simpan GT keseluruhan
    - simpan **GT Myocardium (label = 2)**

📌 Ini **menyiapkan ground truth 2D** untuk evaluasi.

BLOK 6 — PREDIKSI U-NET 3D (ED)
def prediksi_ed(img_ed, model):
Fungsi inti prediksi model
Alur:

1. Load citra 3D .npy
2. Tambah dimensi batch
3. model.predict()
4. argmax(axis=4) → label 3D
5. Loop slice:
    - simpan prediksi keseluruhan
    - simpan prediksi RV
    - simpan prediksi Myo
    - simpan prediksi LV

📌 Inilah **bagian utama yang menjalankan U-Net 3D**.

BLOK 7 — PERHITUNGAN METRIK EVALUASI
def hitung_evaluasi_metrik(imgs, masks):
Fungsi
Menghitung **Dice Score & Hausdorff Distance** untuk **kelas Myocardium**.

Proses detail

1. Buka file CSV
2. Loop pasangan prediksi & GT
3. Baca citra PNG
4. Konversi ke biner:
    - 255 → 1

- lainnya → 0

5. Hitung:
   - dice_coefficient
   - hausdorff_distance
6. Simpan hasil per slice ke CSV
7. Hitung & simpan **rata-rata**

📌 Evaluasi **slice-wise**, bukan volume-wise.

BLOK 8 — FUNGSI MAIN (PIPELINE EKSEKUSI)

def main():

Alur yang AKTIF:

1. Load model U-Net 3D
2. Ambil daftar citra & GT
3. Sort berdasarkan timestamp
4. Ambil prediksi Myo & GT Myo
5. Hitung evaluasi metrik

Alur yang DIKOMENTARI:

- prediksi_ed()
- slice_gt()

Artinya:

Script ini saat dijalankan **hanya menghitung evaluasi**, bukan prediksi ulang.

BLOK 9 — ENTRY POINT

```
if __name__ == "__main__":
  main()
```

BLOK 10 — KODE EKSPERIMEN (DIKOMENTARI)

Bagian akhir berisi:

- visualisasi slice
- debugging prediksi
- perbandingan GT vs prediksi

📌 Tidak dieksekusi, hanya untuk analisis manual.

RINGKASAN FUNGSI UTAMA PER BLOK

Blok	Fungsi
Import	Persiapan library
Path	Konfigurasi dataset
Sorting	Sinkronisasi slice
Prediksi	U-Net 3D inference
Slice	3D → 2D
Evaluasi	Dice & Hausdorff
Output	CSV hasil

CATATAN TEKNIS PENTING (KRITIS)

1. **Evaluasi hanya Myocardium**
2. **Argmax dilakukan pada axis kelas**

3. **Sinkronisasi file sangat penting**
4. **Dice & Hausdorff dihitung per slice**
5. **CSV menggunakan delimiter ;**

KESIMPULAN
Script ini adalah **pipeline evaluasi U-Net 3D ED ACDC 2017**, yang:

- memprediksi citra 3D
- mengekstrak slice 2D
- mengevaluasi Myocardium dengan Dice & Hausdorff
- menyimpan hasil numerik ke CSV

Jika kamu mau, saya bisa:

- menyederhanakan script ini menjadi **modular**
- mengubah evaluasi menjadi **3 kelas (RV, MYO, LV)**
- menjelaskan **alur matematis Dice & Hausdorff**
- membandingkan versi **2D vs 3D**

Tinggal bilang mau lanjut ke bagian mana 👍

20	# File : distances.py
	Bahasan: **GAMBARAN UMUM KODE** File ini berisi **implementasi berbagai fungsi perhitungan jarak (distance metrics)** antar dua vektor numerik menggunakan **Numba JIT** agar **sangat cepat dieksekusi**. Fungsi-fungsi ini umum dipakai untuk: ● pengukuran jarak antar titik / fitur ● evaluasi berbasis boundary (Hausdorff, surface distance) ● komputasi numerik intensif (loop besar) --- **BLOK 1 — IMPORT LIBRARY** import numba import numpy as np from math import sqrt, pow, cos, sin, asin **Fungsi masing-masing import** **Library** **Fungsi** numba Just-In-Time compilation (mempercepat loop Python) numpy Operasi numerik dasar math Fungsi matematika presisi tinggi 📌 **Kenapa Numba dipakai?** Karena perhitungan jarak sering dilakukan **jutaan kali** (misalnya antar boundary pixel), Python murni akan lambat. --- **BLOK 2 — DECORATOR NUMBA JIT** Semua fungsi memakai: @numba.jit(nopython=True, fastmath=True)

Arti parameter

Parameter	Arti
nopython=True	Kode dikompilasi penuh ke machine code (tanpa Python interpreter)
fastmath=True	Mengizinkan optimasi floating-point agresif

📌 Konsekuensi:

- ⚡ Sangat cepat
- ⚠️ Sedikit kompromi presisi numerik (aman untuk jarak)

BLOK 3 — MANHATTAN DISTANCE (L1)

def manhattan(array_x, array_y):

Fungsi matematis

[
d(x,y) = \sum_{i=1}^{n} |x_i - y_i|
]

Cara kerja

- Loop setiap dimensi
- Hitung selisih absolut
- Jumlahkan semua selisih

Karakteristik

- Sensitif terhadap perbedaan absolut
- Banyak dipakai di:
  - grid-based distance
  - image processing

BLOK 4 — EUCLIDEAN DISTANCE (L2)

def euclidean(array_x, array_y):

Fungsi matematis

[
d(x,y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}
]

Cara kerja

- Hitung selisih kuadrat per dimensi
- Jumlahkan
- Akar kuadrat di akhir

Karakteristik

- Jarak geometris standar
- Digunakan di:
  - Hausdorff Distance
  - Surface Distance
  - clustering

📌 **Ini jarak paling umum untuk evaluasi segmentasi.**

BLOK 5 — CHEBYSHEV DISTANCE (L∞)

def chebyshev(array_x, array_y):

Fungsi matematis

[
d(x,y) = \max_i |x_i - y_i|
]

Cara kerja

- Hitung selisih absolut
- Ambil nilai maksimum

Karakteristik

- Mengukur **perbedaan terburuk**
- Dipakai untuk:
  - bounding box
  - toleransi error maksimum

BLOK 6 — COSINE DISTANCE

def cosine(array_x, array_y):

Fungsi matematis

[
d(x,y) = 1 - \frac{x \cdot y}{|x||y|}
]

Cara kerja

1. Hitung dot product
2. Hitung norma masing-masing vektor
3. Hitung cosine similarity
4. Konversi ke cosine distance

Karakteristik

- Mengukur **arah**, bukan magnitudo
- Umum dipakai untuk:
  - feature embedding
  - text/image similarity

📌 **Tidak umum untuk evaluasi mask**, tapi berguna untuk feature-based analysis.

BLOK 7 — HAVERSINE DISTANCE (GEODESIC)

def haversine(array_x, array_y):

Fungsi matematis

Menghitung **jarak permukaan bumi** antara dua titik latitude–longitude.

[
d = 2R \cdot \arcsin(\sqrt{a})
]

Cara kerja

- Konversi derajat ⟶ radian
- Terapkan rumus haversine
- Gunakan radius bumi R = 6378 km

Karakteristik

- Khusus data geografis
- Tidak relevan langsung untuk citra medis

📌 Kemungkinan hanya bagian dari library umum jarak.

BLOK 8 — STRUKTUR INPUT & OUTPUT

Input semua fungsi

array_x = np.array([...])
array_y = np.array([...])

- Panjang **harus sama**
- Biasanya:
  - [x, y] (2D)
  - [x, y, z] (3D)

Output

- **float**
- Nilai jarak sesuai metrik

BLOK 9 — PERAN FILE INI DALAM PIPELINE KAMU

Dalam konteks **ACDC + evaluasi segmentasi**, fungsi ini biasanya dipakai oleh:

- **Hausdorff Distance**
- **Surface Distance**
- **Boundary-based metric**

Contoh:

Boundary GT pixel ↔ Boundary Pred pixel
↓
euclidean()
↓
min / max / mean

📌 **Dice / IoU TIDAK memakai file ini**
(karena Dice & IoU berbasis area, bukan jarak).

RINGKASAN FUNGSI PER METRIK

Fungsi	Jenis	Umum dipakai untuk
manhattan	L1	grid distance
euclidean	L2	Hausdorff, surface
chebyshev	L∞	worst-case
cosine	angular	feature similarity
haversine	geodesic	koordinat bumi

CATATAN TEKNIS PENTING (KRITIS)

1. **Semua fungsi dioptimasi Numba**
2. **Input harus numeric & fixed shape**
3. **Tidak ada pengecekan error**
   - array beda panjang → crash
4. **Presisi floating-point dikompromikan**
   - aman untuk evaluasi

KESIMPULAN

File ini adalah **modul utilitas jarak berperforma tinggi**, dirancang untuk:

- komputasi jarak intensif
- evaluasi boundary
- dipanggil berulang kali dalam loop besar

Ia **tidak berdiri sendiri**, tetapi menjadi **fondasi metrik seperti Hausdorff dan Surface Distance**.

	Jika kamu mau, saya bisa: • menjelaskan **bagaimana fungsi ini dipakai di Hausdorff Distance** • mengaitkan langsung ke **script evaluasi kamu** • membandingkan **Euclidean vs Hausdorff vs Surface Distance** • menuliskan **contoh visual 2D jarak boundary** Tinggal bilang mau lanjut ke mana 👍