# Cell Segmentation Pipeline — Full README (Clean Version)

This document explains how to run the project from start to finish. Follow the order. Edit the values in the USER CONFIG block at the top of each script, then run.

## Before you start

1. Python 3.9 or newer

2. A GPU helps for training but is not required

3. Install once:

pip install numpy opencv-python tifffile matplotlib pillow scipy pandas scikit-learn tensorflow tqdm scikit-image

## Data conventions

• Classes in masks: 0- boundary, 1- Hair cell, 2- supporting cell, 3 – undefined.

• Pixel values in PNG or TIFF: 0, 85, 170, 255 map to classes 0, 1, 2, 3 in that order.

• Data TIFF layout: pages are interleaved. Page 0 is ch1. Page 1 is ch2. Then ch1 and ch2 again.

• GT TIFF layout: masks are on odd pages. Pages 1, 3, 5 and so on

• Black frames: a page with all zeros. We skip them in metrics and keep their original index.

• TRANSPOSE\_FIRST\_N: use when the first frames in GT are rotated 90 degrees. If height and width look swapped, set this to the number of leading frames that need transpose.

## Run order

1. Prepare data crops

2. Train the model

3. Predict on the full TIFF

4. Clean the predictions

5. Measure accuracy

## 1) data\_preparation.py — make training crops

**What it does**

• Reads segmentation masks, data with two channels, and optional RGB files.

• Splits each full frame into smaller square crops.

• Tracks black frames in the masks and skips the same frames in data and RGB.

• Saves aligned crops in three folders.

**What you need**

• Segmentation TIFF files with labels encoded as 0, 1, 2

• Data TIFF files with interleaved ch1 and ch2

• Optional RGB or position TIFF files

**User config to set**

• seg\_paths: list of GT TIFF files

• data\_paths: list of data TIFF files

• rgb\_paths: list of RGB or position TIFF files if available

• crop\_size: usually 256

• frames\_per\_file: None for all frames or a number to limit

• num\_crops: a perfect square like 49

• transpose\_first\_n: number of leading GT frames to transpose if needed

• start\_indices: where to start in each file

**How to run**

1. Open the script and set the values in USER CONFIG

2. Run from your IDE or from a terminal with: python data\_preparation.py

**What you get**

• A folder named data with channel crops

• A folder named label with mask crops

• A folder named RGB with optional crops

**Common mistakes**

• num\_crops is not a perfect square

• Wrong file paths

• Shapes do not match because some frames need transpose

## 2) training.py — train the U-Net

**What it does**

• Builds a standard U-Net model

• Streams crops from disk

• Applies simple preprocessing such as normalization

• Trains with class weights and saves weights and logs

**What you need**

• Crops in IMAGE\_DIR and MASK\_DIR created by the previous step

**User config to set**

• SAVE\_DIR: output folder for weights and logs

• IMAGE\_DIR: path to data crops

• MASK\_DIR: path to label crops

• SIZE, IN\_CHANNELS, NUM\_CLASSES

• BATCH\_SIZE, EPOCHS, LEARNING\_RATE

• VAL\_SPLIT, RANDOM\_STATE

• CLASS\_WEIGHTS

• RESUME\_TRAINING, SAVE\_WEIGHTS\_EVERY\_N, SAVE\_FULL\_MODEL\_EVERY\_N

**How to run**

1. Open the script and set the values in USER CONFIG

2. Run from your IDE or from a terminal with: python training.py

What you get

• model\_last.weights.h5 in SAVE\_DIR

• model\_best.weights.h5 in SAVE\_DIR

• training\_log.csv in SAVE\_DIR

• Optional full model files if enabled

**Common mistakes**

• IMAGE\_DIR and MASK\_DIR do not match the file names produced by data\_preparation.py

## 3) prediction\_only.py — run inference

**What it does**

• Loads the U-Net with your weights

• Reads a two-channel interleaved data TIFF

• Uses the GT only to keep the same black frame positions and orientation

• Predicts on non-black frames and writes one multi-page TIFF

**What you need**

• Trained weights file saved by the training step

• Data TIFF with interleaved ch1 and ch2

• GT TIFF for black frame indices and orientation

**User config to set**

• MODEL\_PATH: path to the weights file

• DATA\_TIF\_PATH: path to the data TIFF

• LABEL\_TIF\_PATH: path to the GT TIFF

• OUTPUT\_DIR: output folder for predictions

• MAX\_FRAMES\_TO\_EVAL: None for all frames or a number to limit

• TRANSPOSE\_FIRST\_N: number of first GT frames to transpose

How to run

1. Open the script and set the values in USER CONFIG

2. Run from your IDE or from a terminal with: python prediction\_only.py

**What you get**

• all\_predictions.tif in OUTPUT\_DIR with one page per frame

• Pages are encoded as 0, 85, 170, 255

• Black frames remain zero at their original indices

## 4) post\_processing.py — clean the predictions

**What it does**

• Reads the predictions TIFF

• Maps values to class indices

• Removes small noise and fixes broken borders

• Keeps black frames black and at the same indices

**What you need**

• The all\_predictions.tif file from the previous step

**User config to set**

• PRED\_TIFF\_PATH: path to all\_predictions.tif

• OUTPUT\_DIR: folder for cleaned outputs

• MAX\_FRAMES: None for all frames or a number to limit

• VERBOSE\_PER\_FRAME: True to print per-frame info

**How to run**

1. Open the script and set the values in USER CONFIG

2. Run from your IDE or from a terminal with: python post\_processing.py

**What you get**

• post\_processed\_085.tif with values 0, 85, 170, 255

• post\_processed\_0\_3.tif with class indices 0 to 3

• frames\_0\_3.npz with an array named frames

• post\_processed\_png folder with one PNG per frame if enabled

## 5) accuracy\_calculation.py — measure accuracy

**What it does**

• Compares GT and predictions page by page

• Skips frames that are black on either side

• Transposes GT when height and width are swapped

• Computes IoU, Precision, Recall and F1 for classes 0 to 3

**What you need**

• GT TIFF with masks

• Predictions TIFF from the previous step

**User config to set**

• GT\_TIFF\_PATH: path to the GT TIFF

• PRED\_TIFF\_PATH: path to the predictions TIFF

• OUT\_DIR: output folder for the CSV

• TRANSPOSE\_FIRST\_N: number of GT frames to transpose

• MAX\_FRAMES: None for all frames or a number to limit

**How to run**

1. Open the script and set the values in USER CONFIG

2. Run from your IDE or from a terminal with: python accuracy\_calculation.py

**What you get**

• frame\_metrics\_from\_folders.csv with one row per frame

• Columns: IoU, Precision, Recall and F1 per class and simple averages