ImageSumming

1 Getting started

ImageSumming[A] helps to create summed patches from experimental measurements (ImageSumming notebook) and visualizes DeepSTORM prediction results to support parameter adjustments (VisualizeDeepSTORM2DLocs notebook). Some smaller AddOns are collected that have been proven to be helpful for the workflow (DeepSTORM2DAddOns notebook).
ImageSumming is compatible with the DeepSTORM2D Colab notebook in the ZeroCostDL4Mic framework.[1][2]

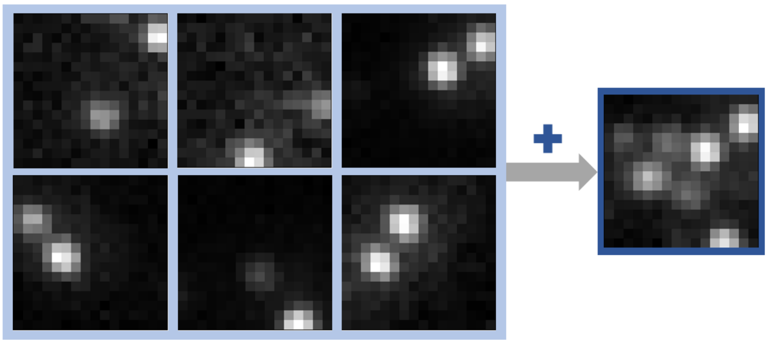


Fig. 1: Sparse emitter density patches are summed to high emitter density patches.

Execute the code listing in an anaconda prompt to:

- create a virtual environment in anaconda and activate it
- go to directory containing the *.whl file and install it
- run the notebook process

Listing 1: Commands to install *ImageSumming* and jupyter extensions in the anaconda prompt.

```
conda create --name ImageSumming
conda activate ImageSumming
cd path_to_file
conda install pip
pip install ImageSumming-XXXXX-py3-none-any.whl
jupyter notebook
```

[A] https://github.com/JohannaRahm/ImageSumming

2 How to start and run a notebook

Once ImageSumming is installed, it can be started by changing the directory to the notebooks and typing *„jupyter notebook“* in the anaconda prompt. The notebooks are opened in a browser. Select a notebook, click on restart the kernel to load the widgets (1), click on show codecell inputs to hide code (2), collapse headings if needed (3) (Fig. 2). Each notebook defines the needed input files, parameters can be adjusted, the process executed and results saved, all via userfriendly widgets.

Listing 2: Commands to run *ImageSumming* in the anaconda prompt.

```
conda activate ImageSumming
cd path_to_notebooks
jupyter notebook
```

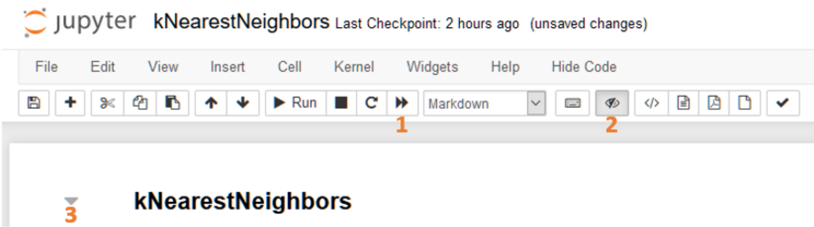


Fig. 2: Start and run a notebook.

3 Adjust default parameters

To adjust the default parameters *„Hide input all“* has to be deactivated to see the code. All adjustable settings are highlighted as comments, just change the values to your needs (Fig. 3).

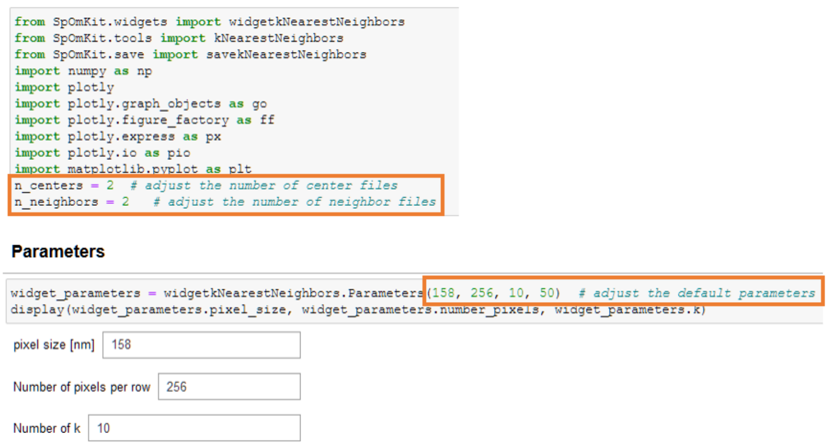


Fig. 3: Two examples of where to adjust the default parameters in the code. The places are marked with # comments.

4 Input filetypes

The notebooks require *.tif files that contain multiple frames of a measurement and corresponding localization files. We use Picasso[3] localization files as ground truth. The files have to be converted to be used in DeepSTORM2D ZeroCostDL4Mic (=DeepSTORM file format) (fig. 4).

Picasso hdf5 format

	frame	x	y	photons	sx	sy	bg	lpx	lpy	ellipticity	net_gradient
0	0	1.4413762	254.61617	23015.182	1.6556486	1.3295391	371.85214	0.01742...	0.02595...	0.19696...	15884.189
1	0	1.6262058	329.20697	12868.999	1.0678529	1.2171849	305.56125	0.01796...	0.01505...	0.12268...	17199.262
2	0	1.8233835	439.0251	4496.8906	1.1016792	1.220813	210.9673	0.03491...	0.03094...	0.09758...	5861.6675

Picasso csv format

```
"id","frame","x [nm]","y [nm]","sigma [nm]","intensity [photon]","offset [photon]","bkgstd [photon]","uncertainty_xy [nm]"
0,0,201.31,6689.55,141.18,67220,557,0,0.85
1,0,224.16,11655.38,130.04,15519,299,0,1.66
2,0,252.65,9010.93,123.40,15052,489,0,1.76
```

DeepSTORM csv format

```
,frame,x [nm],y [nm],Photon #,Sigma [nm]
1,1.0,201.31,6689.55,141.18,67220
2,1.0,224.16,11655.38,130.04,15519
3,1.0,252.65,9010.93,123.4,15052
```

Fig. 4: Example of hdf5 file format of Picasso (top) csv file format of Picasso (middle) csv file format of DeepSTORM (bottom).

5 Notebooks

5.1 ImageSumming

This notebook takes movies and localization files of multiple measurements, cuts patches and randomly summs them. It is compatible with DeepSTORM and Picasso file formats.

5.1.1 Parameters

Following parameters have to be defined. The pixel size in nm. The camera offset in px intensitiy (average px intensity with closed shutter), as the offset is only regarded once and is subtracted from the final summed image (number of summed images - 1 time). The final patch size in px, the DeepSTORM notebook handles input images with a maximum size of final patch size - 1, if a patch size of 26x26 px^2 should be the input in DeepSTORM notebook, the patch size has to be set to 27 in the ImageSumming notebook. The number of patches created defines the number of randomly cropped patches from the input measurement *.tif files. Number of summed patches created defines the final number of summed patches saved as *.tif movies, where each patch is a frame and corresponding localizations are saved as csv in DeepSTORM format, the patches for summed are drawn from the created patches and only used once per summed patch. Define the number of patches to be summed together. Min emitters per patch defines the minimum number of emitters a patch has to contain (before summed) to be used, because empty patches do not contain information for the network.

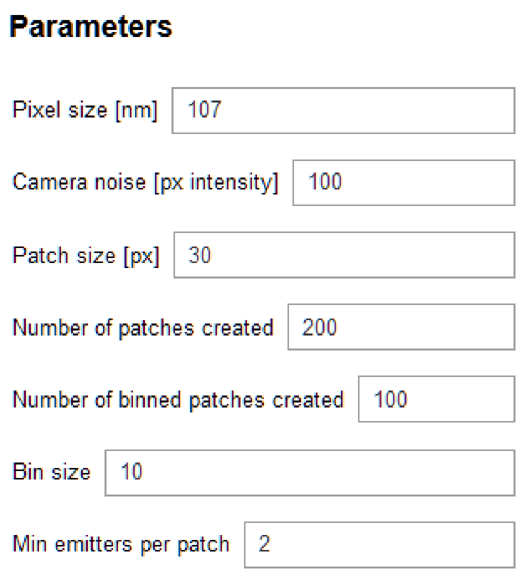
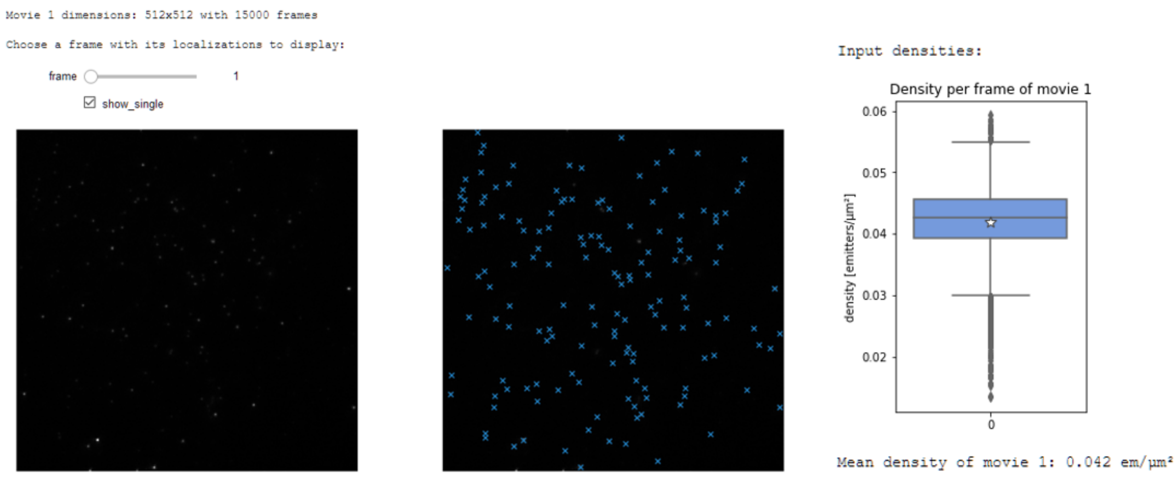


Fig. 5: Parameters of ImageSumming Notebook.

5.1.2 Input

Visualization of the measurement inputs (fig. 6). A chosen measurement is visualized framewise and localizations are marked with crosses. The mean density per frame is calculated to help determining the amount of images to be summed.



id	frame	x [nm]	y [nm]	sigma [nm]	intensity [photon]	offset [photon]	bkgstd [photon]	uncertainty_xy [nm]
0	1	717.65	22039.89	129.04	2774	43	0	3.56
1	1	1201.11	42330.04	119.85	12312	94	0	1.38
2	1	1372.31	39258.48	126.37	1168	39	0	6.32

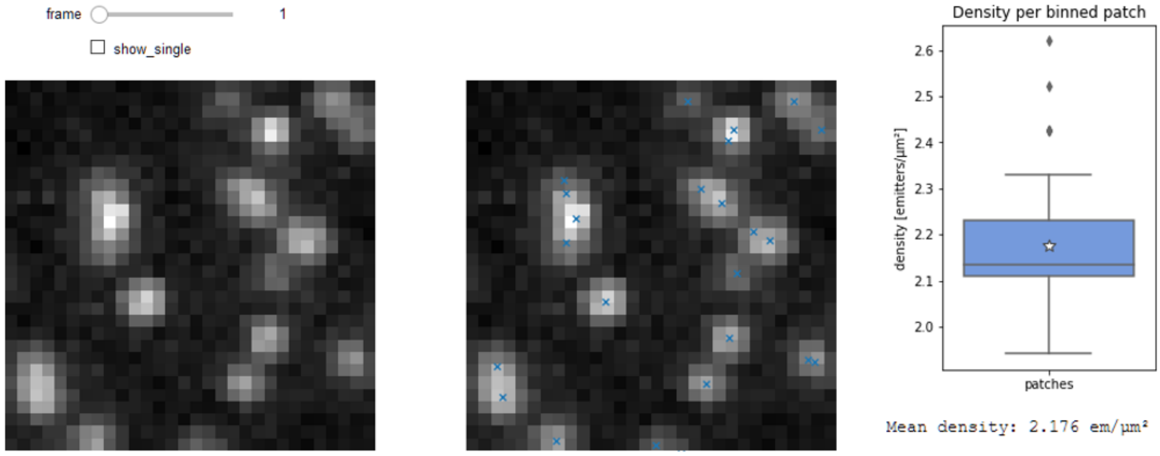
Fig. 6: Visualization of the measurement input in the ImageSumming notebook. A chosen measurement is visualized framewise and the localizations are marked with blue crosses. Average frame density is calculated and displayed as boxplot.

5.1.3 Patches

Patches are created from frames of measurement files and summed. Visualization of the created summed patches (fig. 7). Summed patches are visualized framewise and localizations are marked with crosses. The mean density per patch is calculated to show the final density. Summed patches and a corresponding localization list are saved and are ready to use in the DeepSTORM notebook as input.

Patch dimensions: 30x30 with 100 frames

Choose a patch with its localizations to display (1 patch = 1 frame):



Mean density: 2.176 em/µm²

	frame	x [nm]	y [nm]	sigma [nm]	intensity [photon]	offset [photon]	bkgstd [photon]	uncertainty_xy [nm]
0	7940	791.48	810.39	130.01	911	38	0	7.91
1	7940	1858.69	124.33	127.64	2166	46	0	4.22

Show single frames:

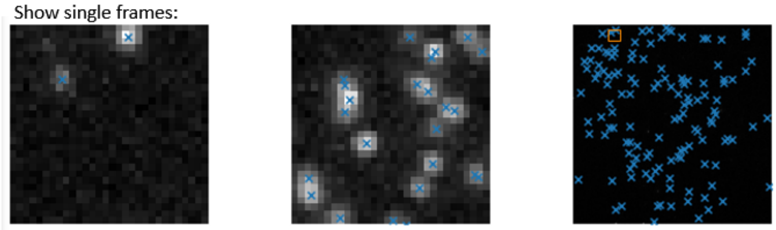


Fig. 7: Visualization of the created patches in the ImageSumming notebook. Summed patches are visualized framewise and the localizations are marked with blue crosses. Average summed patch density is calculated and displayed as boxplot. By enabling show_single, single patches are shown that contribute to the summed image (left) single patch (middle) final summed patch (right) orange region markes the patch on the measurement frame.

5.2 DeepSTORM2DAddOns

DeepSTORM2DAddOns are small add ons that have proven helpful for the workflow.

5.2.1 Convert Picasso csv file to DeepSTORM2D csv format

A Picasso csv file is converted into DeepSTORM csv file format.
Input file: Picasso localization file csv
Output file: DeepSTORM2D file csv

5.2.2 Convert Picasso hdf5 file to DeepSTORM2D csv format

A Picasso hdf5 file is converted into DeepSTORM csv file format.
Input file: Picasso localization file hdf5
Output file: DeepSTORM2D file csv

5.2.3 Split tif movie and corresponding localization file at defined frame

A *.tif movie and corresponding localization file are split into two files at defined frame.
Input file: Tif movie and csv file (Picasso or DeepSTORM2D format).
Parameter split at frame: Number of frames of the first file.
Output file: Two tif movies and csv files, split at defined frame.

5.2.4 Merge multiple movies and localization files

Merge multiple movies and their localization files to one.
Input files: Define multiple paths to movies and their localization files (Picasso or DeepSTORM format).
Output file: Tif movie with frames of movies stacked and localization file with continous frame numbering.

5.2.5 Split into single frames

Split a movie and localization file into single frames.
Input file: Tif movie and csv file (Picasso or DeepSTORM2D format).
Output files: Single frames of tif movies with corresponding csv file saved in defined directory.

5.3 VisualizeDeepSTORM2DLocs

This notebooks visualizes the found localizations after the post-processing in the Deep-STORM notebook and helps to tune post-processing parameters (fig. 8-10).

Prediction directory: Define the directory to test file single frames and localization files (prediciton output of DeepSTORM2D, sec 6.1), optionally this directory contains the ground truth localization file.

Confidence threshold: All localizations below this threshold will be filtered out. If no filtering should be applied, set the value to 0.

Save: The localizations are filtered by the confidence threshold and saved as new csv file.

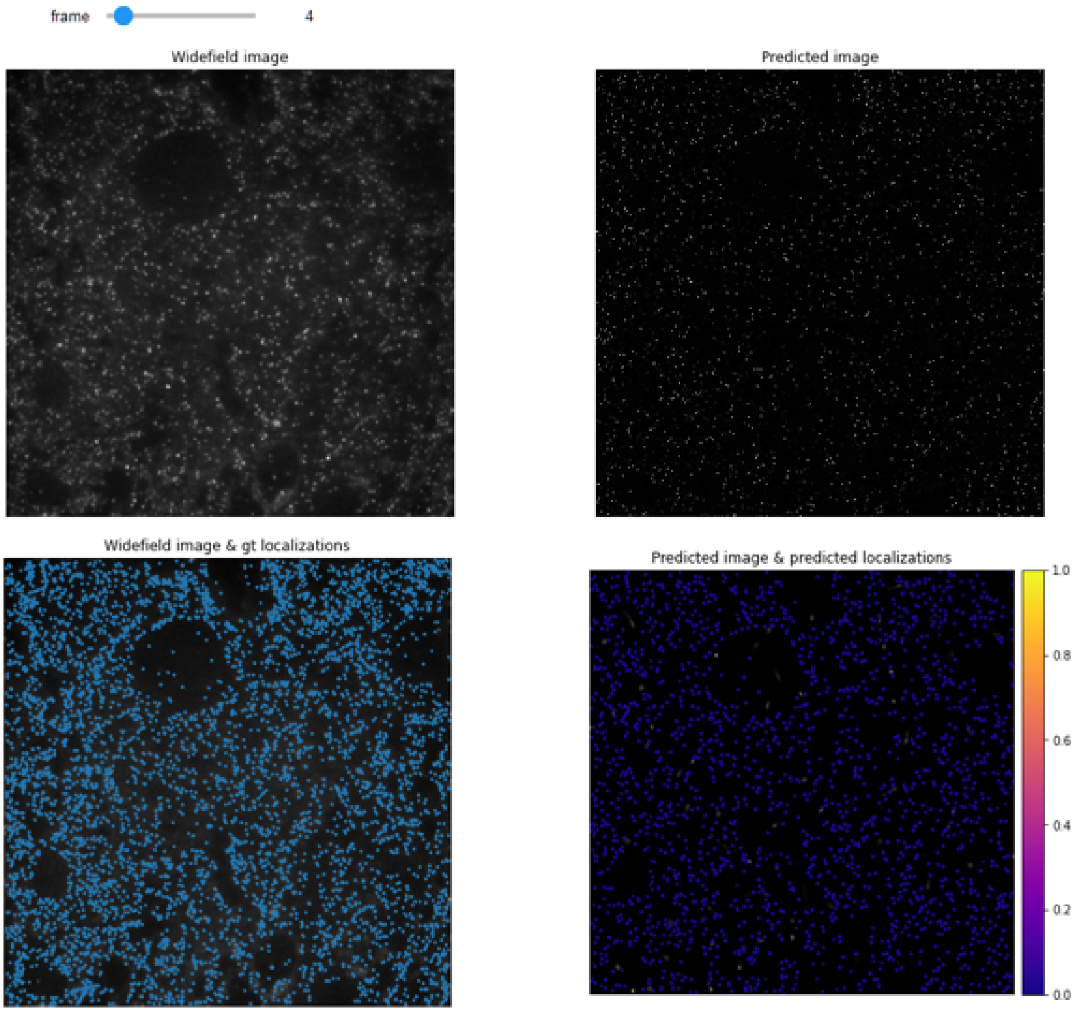


Fig. 8: Framewise visualization of recorded widefield image, predicted image, widefield image and ground truth localizations (if available) as blue crosses and predicted image and predicted localizations as crosses color coded by their confidence.

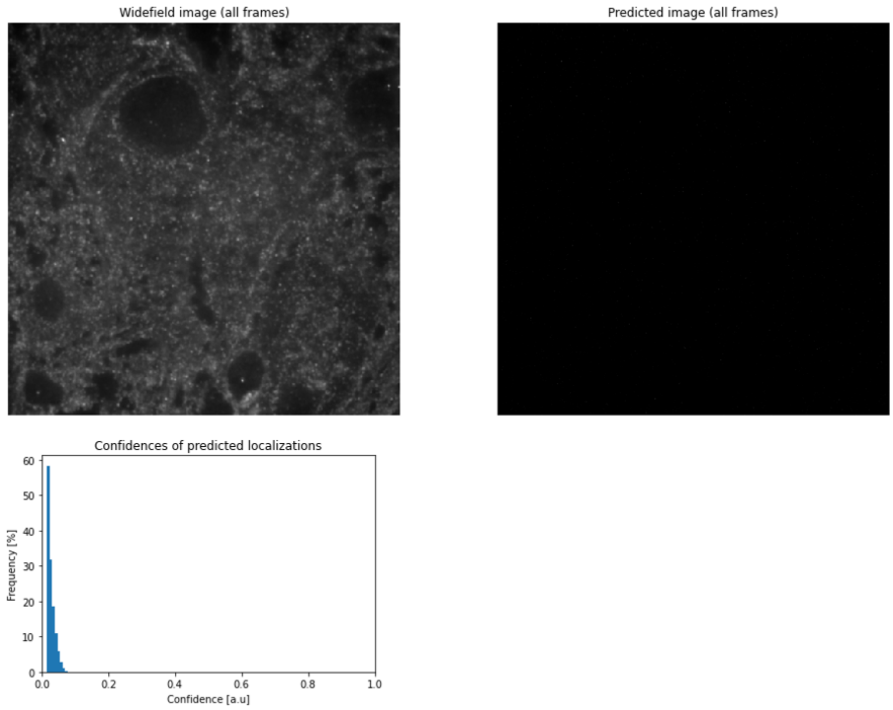


Fig. 9: Visualization of the merged widefield image (all frames are displayed as once) and the merged prediction image. Histogram of confidences of found localizations.

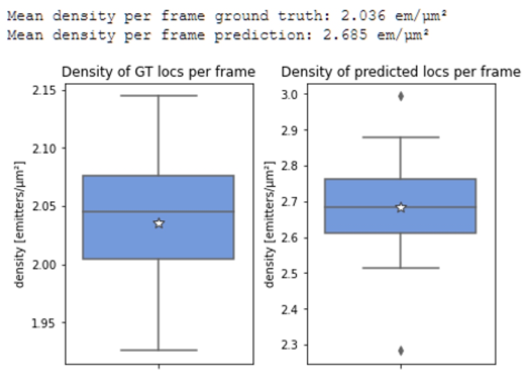


Fig. 10: Mean density of ground truth localizations and predicted localizations.

6 DeepSTORM2D notebook workflow and code adjustments

In Section 3.2 Generate training patches, patch_size is set to the patch_size created in ImageSumming - 1 (smaller patch sizes are possible as well). To consider each summed patch once, the parameter num_patches_per_frame is set to 1.

In Section 5.2 Error mapping and quality metrics estimation, the upsampling factor has to be hard coded, because image sizes are calculated based on the inputs in Section 3.2 that differ from the test image dimensions.

```
if pixel_size_INPUT == None:
  pixel_size, N, M = getPixelSizeTIFFmetadata(os.path.join(QC_image_folder,imageFilename))

#upsampling_factor = int(Mhr/M)
upsampling_factor = 8
print('Upsampling factor: '+str(upsampling_factor))
pixel_size_hr = pixel_size/upsampling_factor # in nm
```

Fig. 11: Code change in section 5.3: Hardcode the upsampling factor in this section to run quality control on differently shaped test images.

In Section 2 save the prediciton not only as a merged image over all frames, but also as a stacked *.tif with each frame accessible. This helps to adjust post-processing parameters, but needs a lot of disk space.

```python
# Initialise the results
Prediction = np.zeros((M*upsampling_factor, N*upsampling_factor), dtype=np.float32)
Widefield = np.zeros((M, N), dtype=np.float32)
Prediction_stacked = np.asarray([np.zeros((M*upsampling_factor, N*upsampling_factor), dtype=np.float32) for i in range(nFrames)])

# run model in batches
n_batches = math.ceil(nFrames/batch_size)
for b in tqdm(range(n_batches)):

  nF = min(batch_size, nFrames - b*batch_size)
  Images_norm = np.zeros((nF, M, N),dtype=np.float32)
  Images_upsampled = np.zeros((nF, M*upsampling_factor, N*upsampling_factor), dtype=np.float32)

  # Upsampling using a simple nearest neighbor interp and calculating - MULTI-THREAD this?
  for f in range(nF):
    Images_norm[f,:,:] = project_01(Images[b*batch_size+f,:,:])
    Images_norm[f,:,:] = normalize_im(Images_norm[f,:,:], test_mean, test_std)
    Images_upsampled[f,:,:] = np.kron(Images_norm[f,:,:], np.ones((upsampling_factor,upsampling_factor)))
    Widefield += Images[b*batch_size+f,:,:]

  # Reshaping
  Images_upsampled = np.expand_dims(Images_upsampled,axis=3)

  # Run prediction and local amxima finding
  predicted_density = model.predict_on_batch(Images_upsampled)
  predicted_density[predicted_density < 0] = 0
  Prediction_stacked[b] = predicted_density.sum(axis = 3).sum(axis = 0)
  Prediction += predicted_density.sum(axis = 3).sum(axis = 0)

  bind, xind, yind, confidence = max_layer(predicted_density)

  # normalizing the confidence by the L2_weighting_factor
  confidence /= L2_weighting_factor

  # turn indices to nms and append to the results
  xind, yind = xind*pixel_size_hr, yind*pixel_size_hr
  frmind = (bind.numpy() + b*batch_size + 1).tolist()
  xind = xind.numpy().tolist()
  yind = yind.numpy().tolist()
  confidence = confidence.numpy().tolist()
  frame_number_list += frmind
  x_nm_list += xind
  y_nm_list += yind
  confidence_au_list += confidence

# Open and create the csv file that will contain all the localisations
if use_local_avg:
  ext = '_avg'
else:
  ext = '_max'
with open(os.path.join(savePath, 'Localisations_' + os.path.splitext(filename)[0] + ext + '.csv'), "w", newline='') as file:
  writer = csv.writer(file)
  writer.writerow(['frame', 'x [nm]', 'y [nm]', 'confidence [a.u]'])
  locs = list(zip(frame_number_list, x_nm_list, y_nm_list, confidence_au_list))
  writer.writerows(locs)

# Save the prediction and widefield image
Widefield = np.kron(Widefield, np.ones((upsampling_factor,upsampling_factor)))
Widefield = np.float32(Widefield)

# io.imsave(os.path.join(savePath, 'Predicted_'+os.path.splitext(filename)[0]+'.tif'), Prediction)
# io.imsave(os.path.join(savePath, 'Widefield_'+os.path.splitext(filename)[0]+'.tif'), Widefield)
import tifffile
tifffile.imwrite(savePath + '/' + 'Predicted_stacked_'+os.path.splitext(filename)[0] + ".tif", Prediction_stacked)
saveAsTIF(savePath, 'Predicted_'+os.path.splitext(filename)[0], Prediction, pixel_size_hr)
saveAsTIF(savePath, 'Widefield_'+os.path.splitext(filename)[0], Widefield, pixel_size_hr)

return
```

Fig. 12: Code change in section 2: Save prediciton as stacked tif with single frames additionally to merged prediction output.

7 Literatur

[1] E. Nehme, L. E. Weiss, T. Michaeli, Y. Shechtman, *Optica* **2018**, *5*, 458–464.

[2] L. Chamier, R. F. Laine, J. Jukkala, C. Spahn, D. Krentzel, E. Nehme, M. Lerche, S. Herández-Pérez, P. K. Mattila, E. Karinou, S. Holden, A. C. Solak, A. Krull, T.-O. Buchholz, M. L. Jones, L. Royer, C. Leterrier, Y. Shechtman, F. Jug, M. Heilemann, G. Jacquemet, R. Henriques, *Nat. Comm.* **2021**, 12:2276.

[3] J. Schnitzbauer, M. T. Strauss, T. Schlichthaerle, F. Schueder, R. Jungmann, *Nat. Protoc.* **2017**, 1198–1228.