# Leveraging Deep Learning for Identification and Segmentation of CAF-1/p60 Positive Nuclei in Oral Squamous Cell Carcinoma Tissue Samples

November 25, 2024

**Supplementary material**

## Supplementary 1

| criterion | 'iou' |
|---|---|
| thresh | 0.3 |
| fp | 1050 |
| tp | 2318 |
| fn | 9989 |
| precision | 0.6882422802850356 |
| recall | 0.18834809458032015 |
| accuracy | 0.17354196301564723 |
| f1 | 0.295757575757575 |
| n_true | 12307 |
| n_pred | 3368 |
| mean_true_score | 0.12096721451404742 |
| mean_matched_score | 0.6422534551442544 |
| panoptic_quality | 0.1899513249153916 |
| by_image | False |

Supplementary Table S 1: Dataset matching values based on the intersection of union (iou) parameter.

## Supplementary 2

```
runPlugin('qupath.imagej.detect.cells.WatershedCellDetection',
                    '{"detectionImageBrightfield":"Hematoxylin OD",
                            "backgroundByReconstruction":true,
                            "backgroundRadius":15.0,
                            "medianRadius":0.0,
                            "sigma":6.0,
                            "minArea":100.0,
                            "maxArea":1000.0,
                            "threshold":0.05,
                            "maxBackground":2.0,
                            "watershedPostProcess":true,
                            "excludeDAB":false,
                            "cellExpansion":0.0,
                            "includeNuclei":true,
                            "smoothBoundaries":true,
                            "makeMeasurements":true}')
setDetectionIntensityClassifications(
    "Nucleus: DAB OD mean", 3)
selectObjectsByClassification("Negative");
clearSelectedObjects(true);
clearSelectedObjects();
.
```

Supplementary Code 1: QuPath Groovy script used for the cell detection, the classification based on the intensity of staining with DAB, and the subsequent cleaning of the detections to obtain a mask that only highlights positive cells.

## Supplementary 3

```
from google.colab import drive
drive.mount('/content/drive')

!pip install StarDist

!pip install gputools

!pip install scikit-tensor-py3


# Commented out IPython magic to ensure Python compatibility.
from __future__ import print_function,
unicode_literals, absolute_import, division
import sys
import numpy as np
import matplotlib
matplotlib.rcParams["image.interpolation"] = "None"
import matplotlib.pyplot as plt
# %matplotlib inline
# %config InlineBackend.figure_format = 'retina'

from glob import glob
from tqdm import tqdm
from tifffile import imread
from csbdeep.utils import Path, normalize, download_and_extract_zip_file
```

```
58
59 from stardist import fill_label_holes,
60 random_label_cmap, calculate_extents, gputools_available
61 from stardist.matching import matching, matching_dataset
62 from stardist.models import Config2D, StarDist2D, StarDistData2D
63
64 np.random.seed(42)
65 lbl_cmap = random_label_cmap()
66
67 gputools_available()
68
69
70 !unzip /content/drive/MyDrive/HE_p60_stardist/
71 images_masks_0909.zip -d /content/data
72
73 X = sorted(glob('/content/data/images_masks_0909/*.tif'))
74 Y = sorted(glob('/content/data/masks/*.tif'))
75 assert all(Path(x).name==Path(y).name for x,y in zip(X,Y))
76
77 X = list(map(imread,X))
78 Y = list(map(imread,Y))
79 n_channel = 1 if X[0].ndim == 2 else X[0].shape[-1]
80
81 """Normalize images and fill small label holes."""
82
83 axis_norm = (0,1)   # normalize channels independently
84 # axis_norm = (0,1,2) # normalize channels jointly
```

```
85 if n_channel > 1:
86     print("Normalizing image channels %s." %
87     ('jointly' if axis_norm is None or 2 in axis_norm else 'independently'))
88     sys.stdout.flush()
89
90 X = [normalize(x,1,99.8,axis=axis_norm) for x in tqdm(X)]
91 Y = [fill_label_holes(y) for y in tqdm(Y)]
92
93 """Split into train and validation datasets."""
94
95 assert len(X) > 1, "not enough training data"
96 rng = np.random.RandomState(42)
97 ind = rng.permutation(len(X))
98 n_val = max(1, int(round(0.15 * len(ind))))
99 ind_train, ind_val = ind[:-n_val], ind[-n_val:]
100 X_val, Y_val = [X[i] for i in ind_val]  , [Y[i] for i in ind_val]
101 X_trn, Y_trn = [X[i] for i in ind_train], [Y[i] for i in ind_train]
102 print('number of images: %3d' % len(X))
103 print('- training:       %3d' % len(X_trn))
104 print('- validation:     %3d' % len(X_val))
105
106 """Training data consists of pairs of input image and label instances."""
107
108 def plot_img_label(img, lbl, img_title="image", lbl_title="label", **kwargs):
109     fig, (ai,al) = plt.subplots(1,2, figsize=(12,5), gridspec_kw=
110     dict(width_ratios=(1.25,1)))
111     im = ai.imshow(img, cmap='gray', clim=(0,1))
```

```
112         ai.set_title(img_title)
113         fig.colorbar(im, ax=ai)
114         al.imshow(lbl, cmap=lbl_cmap)
115         al.set_title(lbl_title)
116         plt.tight_layout()
117 
118 i = min(9, len(X)-1)
119 img, lbl = X[i], Y[i]
120 assert img.ndim in (2,3)
121 img = img if (img.ndim==2 or img.shape[-1]==3) else img[...,0]
122 plot_img_label(img,lbl)
123 None;
124 
125 """# Configuration
126 
127 A `StarDist2D` model is specified via a `Config2D` object.
128 """
129 
130 print(Config2D.__doc__)
131 
132 # 32 is a good default choice (see 1_data.ipynb)
133 n_rays = 32
134 
135 # Use OpenCL-based computations for data generator during training (requires 'gp
136 use_gpu = True and gputools_available()
137 
138 # Predict on subsampled grid for increased efficiency and larger field of view
```

```
grid = (2,2)

conf = Config2D (
    n_rays       = n_rays,
    grid         = grid,
    use_gpu      = use_gpu,
    n_channel_in = n_channel,
)
print(conf)
vars(conf)

if use_gpu:
    from csbdeep.utils.tf import limit_gpu_memory
    # adjust as necessary: limit GPU memory to be used by TensorFlow to leave so
    #limit_gpu_memory(0.8)
    # alternatively, try this:
    limit_gpu_memory(None, allow_growth=True)

"""**Note:** The trained `StarDist2D` model will *not* predict completed shapes

model = StarDist2D(conf, name='stardist_1009', basedir='/content/model')

"""Check if the neural network has a large enough field of view to see up to the

model.callbacks.append(earlystop)

median_size = calculate_extents(list(Y), np.median)
```

```
fov = np.array(model._axes_tile_overlap('YX'))
print(f"median object size:      {median_size}")
print(f"network field of view :  {fov}")
if any(median_size > fov):
    print("WARNING: median object size larger than field of view of the neural n

"""# Data Augmentation

You can define a function/callable that applies augmentation to each batch of th
We here use an 'augmenter' that applies random rotations, flips, and intensity c
"""

def random_fliprot(img, mask):
    assert img.ndim >= mask.ndim
    axes = tuple(range(mask.ndim))
    perm = tuple(np.random.permutation(axes))
    img = img.transpose(perm + tuple(range(mask.ndim, img.ndim)))
    mask = mask.transpose(perm)
    for ax in axes:
        if np.random.rand() > 0.5:
            img = np.flip(img, axis=ax)
            mask = np.flip(mask, axis=ax)
    return img, mask

def random_intensity_change(img):
    img = img*np.random.uniform(0.6,2) + np.random.uniform(-0.2,0.2)
    return img
```

```
193
194
195 def augmenter(x, y):
196     """Augmentation of a single input/label image pair.
197     x is an input image
198     y is the corresponding ground-truth label image
199     """
200     x, y = random_fliprot(x, y)
201     x = random_intensity_change(x)
202     # add some gaussian noise
203     sig = 0.02*np.random.uniform(0,1)
204     x = x + sig*np.random.normal(0,1,x.shape)
205     return x, y
206
207 # plot some augmented examples
208 img, lbl = X[60],Y[60]
209 plot_img_label(img, lbl)
210 for _ in range(3):
211     img_aug, lbl_aug = augmenter(img,lbl)
212     plot_img_label(img_aug, lbl_aug, img_title="image augmented", lbl_title="lab
213
214 """# Training"""
215
216 import tensorflow as tf
217
218 reduce_lr = tf.keras.callbacks.ReduceLROnPlateau(monitor='val_loss', factor=0.2,
219                                 patience=5, min_lr=0.001)
```

```
'''
earlystop = tf.keras.callbacks.EarlyStopping(
                monitor="val_loss",
                min_delta=0.01,
                patience=20,
                verbose=1,
                mode="auto",
                baseline=None,
                restore_best_weights=True,
                start_from_epoch=100,
            )
#model.train(X_trn, Y_trn, callbacks=[reduce_lr])
'''

model.prepare_for_training()

model.callbacks.append(reduce_lr)


# Commented out IPython magic to ensure Python compatibility.
# %load_ext tensorboard
# %tensorboard --logdir /content/model/stardist_1009/logs/

!pip install pyopencl

'''
quick_demo = False
```

```
247
248 if quick_demo:
249     print (
250         "NOTE: This is only for a quick demonstration!\n"
251         "      Please set the variable 'quick_demo = False' for proper (long) tr
252         file=sys.stderr, flush=True
253     )
254     model.train(X_trn, Y_trn, validation_data=(X_val,Y_val), augmenter=augmenter
255                 epochs=2, steps_per_epoch=10)
256
257     print("====> Stopping training and loading previously trained demo model fro
258     model = StarDist2D.from_pretrained('2D_demo')
259 else:
260     model.train(X_trn, Y_trn, validation_data=(X_val,Y_val), augmenter=augmenter
261 None;
262 '''
263
264 model.train(X_trn, Y_trn, validation_data=(X_val,Y_val), augmenter=augmenter, ep
265
266
267 model.optimize_thresholds(X_val, Y_val)
268
269 help(matching)
270
271 Y_val_pred = [model.predict_instances(x, n_tiles=model._guess_n_tiles(x), show_
272               for x in tqdm(X_val)]
273
```

```
plot_img_label(X_val[10],Y_val[10], lbl_title="label GT")
plot_img_label(X_val[10],Y_val_pred[10], lbl_title="label Pred")

taus = [0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9]
stats = [matching_dataset(Y_val, Y_val_pred, thresh=t, show_progress=False) for

stats[taus.index(0.3)]

fig, (ax1,ax2) = plt.subplots(1,2, figsize=(15,5))

for m in ('precision', 'recall', 'accuracy', 'f1', 'mean_true_score', 'mean_matc
    ax1.plot(taus, [s._asdict()[m] for s in stats], '.-', lw=2, label=m)
ax1.set_xlabel(r'IoU threshold $\tau$')
ax1.set_ylabel('Metric value')
ax1.grid()
ax1.legend()

for m in ('fp', 'tp', 'fn'):
    ax2.plot(taus, [s._asdict()[m] for s in stats], '.-', lw=2, label=m)
ax2.set_xlabel(r'IoU threshold $\tau$')
ax2.set_ylabel('Number #')
ax2.grid()
ax2.legend();
```

## Supplementary 4

```
Config2D(n_dim=2, axes='YXC', n_channel_in=3, n_channel_out=33, train_checkpoint
```

```
{'n_dim': 2,
 'axes': 'YXC',
 'n_channel_in': 3,
 'n_channel_out': 33,
 'train_checkpoint': 'weights_best.h5',
 'train_checkpoint_last': 'weights_last.h5',
 'train_checkpoint_epoch': 'weights_now.h5',
 'n_rays': 32,
 'grid': (2, 2),
 'backbone': 'unet',
 'n_classes': None,
 'unet_n_depth': 3,
 'unet_kernel_size': (3, 3),
 'unet_n_filter_base': 32,
 'unet_n_conv_per_depth': 2,
 'unet_pool': (2, 2),
 'unet_activation': 'relu',
 'unet_last_activation': 'relu',
 'unet_batch_norm': False,
 'unet_dropout': 0.0,
 'unet_prefix': '',
 'net_conv_after_unet': 128,
 'net_input_shape': (None, None, 3),
 'net_mask_shape': (None, None, 1),
 'train_shape_completion': False,
 'train_completion_crop': 32,
 'train_patch_size': (256, 256),
```

```
'train_background_reg': 0.0001,
'train_foreground_only': 0.9,
'train_sample_cache': True,
'train_dist_loss': 'mae',
'train_loss_weights': (1, 0.2),
'train_class_weights': (1, 1),
'train_epochs': 800,
'train_steps_per_epoch': 100,
'train_learning_rate': 0.0003,
'train_batch_size': 4,
'train_n_val_patches': None,
'train_tensorboard': True,
'train_reduce_lr': {'factor': 0.5, 'patience': 40, 'min_delta': 0},
'use_gpu': True}
```