DIC-C2DH-HeLa

August 7, 2025

1 U-Net cell segmentation on the DIC-C2DH-HeLa dataset

1.1 Import modules

[3]:
```
from PIL import Image
from skimage.measure import label
import matplotlib.pyplot as plt
import numpy as np
import cv2 as cv
import gc, os
```

[4]:
```
import tensorflow as tf
import tensorflow.keras.backend as K
import tensorflow_datasets as tfds
```

[5]:
```
from unet.utils import UNetHelper
from unet.losses import IoU, dice_loss, unet_sample_weights
from unet.augmentation import elastic_deformation, grid_deformation
```

```
I0000 00:00:1754503500.986878 1224702 gpu_device.cc:2022] Created device
/job:localhost/replica:0/task:0/device:GPU:0 with 21770 MB memory:  -> device:
0, name: NVIDIA GeForce RTX 3090, pci bus id: 0000:65:00.0, compute capability:
8.6
```

[6]:
```
tf.get_logger().setLevel('ERROR')
```

2 Notebook configuration

[7]:
```
train_model = True
tf_dir = "TFData"
batch_size = 8
max_epochs = 280
```

[8]:
```
os.system(f"mkdir -p {tf_dir}/TFCache {tf_dir}/models")
```

[8]: 0

2.1 Random seed

For resetting the seed when running the training loop multiple times

[9]:
```
reset_seed = lambda seed=42: tf.keras.utils.set_random_seed(seed)
reset_seed()
```

2.2 Distributed training strategy

This selection is based off the tools I have at my disposal: either 1 GPU at work or 2 on Kaggle

[10]:
```
gpus = len(tf.config.list_physical_devices("GPU"))

if gpus <= 1:
    strategy = tf.distribute.OneDeviceStrategy(device="/GPU:0")
else:
    strategy = tf.distribute.MirroredStrategy(cross_device_ops=tf.distribute.
 ↪ReductionToOneDevice())

n_devices = strategy.num_replicas_in_sync
print(f"Using {n_devices} device(s).")
print(f"Using {strategy.__class__.__name__}.")
```

```
Using 1 device(s).
Using OneDeviceStrategy.
```

3 Load the dataset

[11]:
```
def process_img(img, mask):
    """
    Contrast Limited Adaptive Histogram Equalization (CLAHE) step,
    followed by sample weight calculation [0.0, 1.0] normalization.
    CLAHE uses the default OpenCV parameters.
    """
    clh = cv.createCLAHE(clipLimit=2.0, tileGridSize=(8, 8))
    clh_img = clh.apply(np.squeeze(img.numpy()))
    sample_weights = unet_sample_weights(mask.numpy(), data_type=np.float32)
    return (tf.constant(np.expand_dims(clh_img / 255.0, -1), dtype=tf.float32,␣
 ↪shape=img.get_shape()),
            mask,
            tf.constant(sample_weights, dtype=tf.float32, shape=mask.
 ↪get_shape()))
```

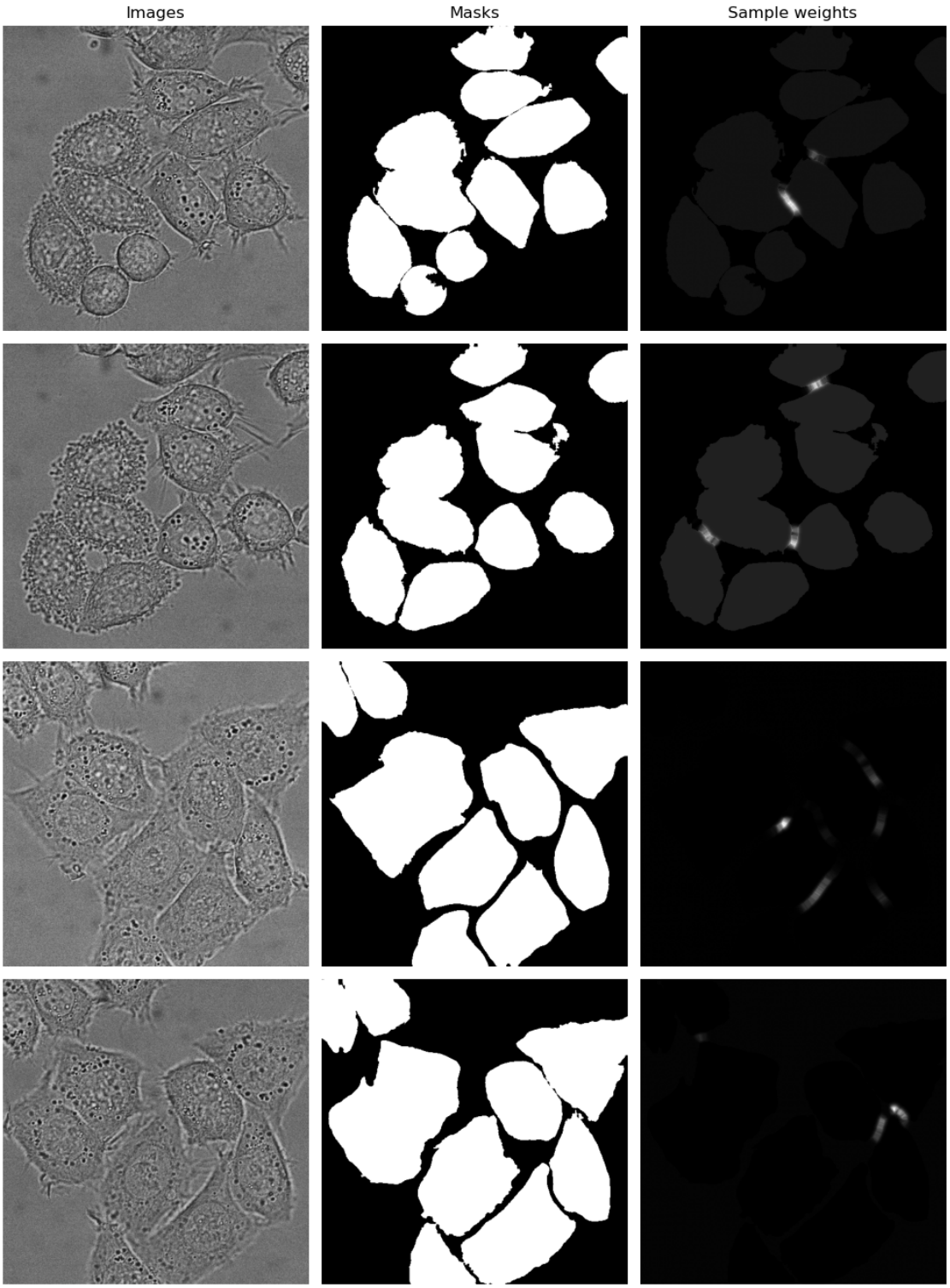
[12]:
```
def min_max(arr):
    arr = np.asarray(arr)
    minimum, maximum = arr.min(), arr.max()
    return (arr - minimum) / (maximum - minimum)
```

[13]:
```
img_shape = (512, 512, 1)
mask_shape = (512, 512)

hela_train = tfds.load("hela_train", data_dir=tf_dir)

# Cache CLAHE-equalized images
# ensure_shape step needed for graph execution

# Cache segment 01
hela_train["01"] = hela_train["01"].map(lambda sample: tf.
 ↪py_function(process_img, inp=[sample['image'], sample['mask']],
                                                                Tout=[tf.
 ↪float32, tf.int32, tf.float32]),
                                     num_parallel_calls=tf.data.AUTOTUNE)\
                                  .map(lambda X, y, sw: (tf.ensure_shape(X,␣
 ↪img_shape),
                                                         tf.ensure_shape(y,␣
 ↪mask_shape),
                                                         tf.ensure_shape(sw,␣
 ↪mask_shape)))\
                                  .cache(f"{tf_dir}/TFCache/01_CLAHE_NORM")
example = list(hela_train["01"].take(2))

# Cache segment 02
hela_train["02"] = hela_train["02"].map(lambda pair: tf.
 ↪py_function(process_img, inp=[pair['image'], pair['mask']],
                                                            Tout=[tf.
 ↪float32, tf.int32, tf.float32]),
                                     num_parallel_calls=tf.data.AUTOTUNE)\
                                  .map(lambda X, y, sw: (tf.ensure_shape(X,␣
 ↪img_shape),
                                                         tf.ensure_shape(y,␣
 ↪mask_shape),
                                                         tf.ensure_shape(sw,␣
 ↪mask_shape)))\
                                  .cache(f"{tf_dir}/TFCache/02_CLAHE_NORM")
example += list(hela_train["02"].take(2))
```

```
2025-08-06 15:05:02.640792: I tensorflow/core/framework/local_rendezvous.cc:405]
Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence
2025-08-06 15:05:02.752407: I tensorflow/core/framework/local_rendezvous.cc:405]
Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence
```

[14]:
```
fig, axes = plt.subplots(len(example), 3, figsize=(10, 5 * len(example)))

axes[0,0].set_title("Images")
```

```
axes[0,1].set_title("Masks")
axes[0,2].set_title("Sample weights")

for row, ex in zip(axes, example):
    for ax, img in zip(row, ex):
        ax.imshow(min_max(img), cmap="gray")
        ax.axis("off")
        ax.set_aspect('equal')

fig.tight_layout(h_pad=-40, w_pad=1)
plt.show()
```


Images
Masks
Sample weights

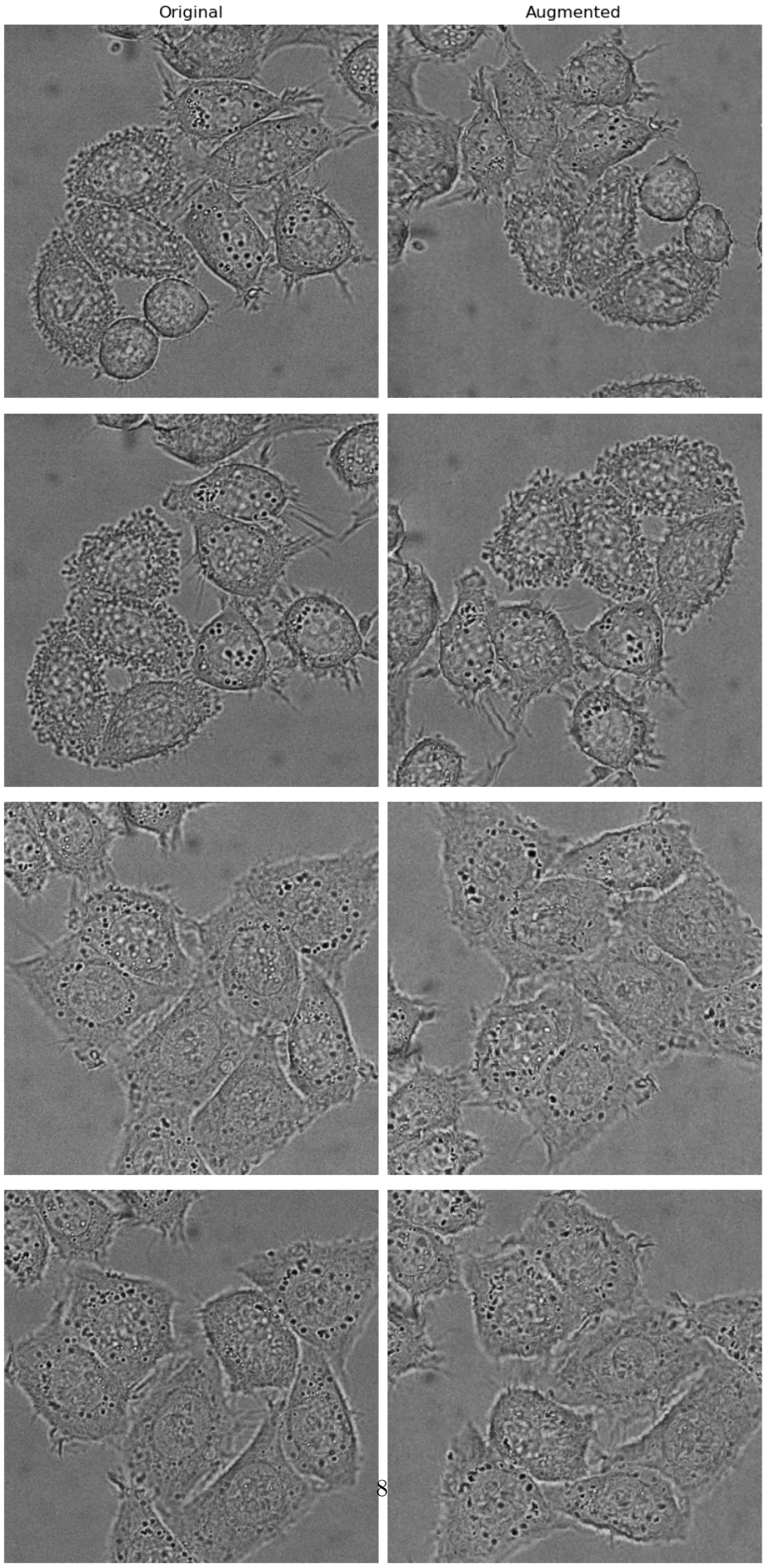
4 Data augmentation

[15]:
```
@tf.function
def pipeline(X, y, w):
    # Add channel axis
    y = tf.expand_dims(y, axis=-1)
    w = tf.expand_dims(w, axis=-1)
    # Horizontal flip
    if tf.random.uniform((), 0.0, 1.0) >= 0.5:
        X = tf.image.flip_left_right(X)
        y = tf.image.flip_left_right(y)
        w = tf.image.flip_left_right(w)
    # Vertical flip
    if tf.random.uniform((), 0.0, 1.0) >= 0.5:
        X = tf.image.flip_up_down(X)
        y = tf.image.flip_up_down(y)
        w = tf.image.flip_up_down(w)
    # Grid deformation
    if tf.random.uniform((), 0.0, 1.0) >= 0.5:
        grid_size = 5
        distort_limits = (-0.35, 0.35)
        X = grid_deformation(X, distort_limits=distort_limits,␣
↪grid_size=grid_size, order=1)
        y = grid_deformation(y, distort_limits=distort_limits,␣
↪grid_size=grid_size, order=0)
        w = grid_deformation(w, distort_limits=distort_limits,␣
↪grid_size=grid_size, order=0)
    # Elastic deformation
    if tf.random.uniform((), 0.0, 1.0) >= 0.5:
        alpha = 100.0
        sigma = 5.0
        auto_kSize = True
        X = elastic_deformation(X, alpha=alpha, sigma=sigma,␣
↪auto_kSize=auto_kSize, order=1)
        y = elastic_deformation(y, alpha=alpha, sigma=sigma,␣
↪auto_kSize=auto_kSize, order=0)
        w = elastic_deformation(w, alpha=alpha, sigma=sigma,␣
↪auto_kSize=auto_kSize, order=0)
    return [X, tf.squeeze(y), tf.squeeze(w)]
```

[16]:
```
fig, axes = plt.subplots(len(example), 2, figsize=(8, 4 * len(example)))

axes[0,0].set_title("Original")
axes[0,1].set_title("Augmented")

reset_seed()
for row, (tmp_X, tmp_y, tmp_w) in zip(axes, example):
```

```
    row[0].imshow(min_max(tmp_X), cmap="gray")
    row[0].axis("off")
    row[1].imshow(min_max(pipeline(tf.expand_dims(tmp_X, 0),
                                   tf.expand_dims(tmp_y, 0),
                                   tf.expand_dims(tmp_w, 0))[0][0]),␣
 ↪cmap="gray")
    row[1].axis("off")

fig.tight_layout()
plt.show()
```

```
I0000 00:00:1754503507.671559 1225481 cuda_dnn.cc:529] Loaded cuDNN version
90300
```


Original
Augmented

5 Main training loop

[17]:
```
def train(helper, train_dataset, val_dataset=None, examples=None, epochs=100,␣
↪ckpt_every=10, plot_every=1, verbose=True): # A helper function I wrote in a␣
↪hurry.
    history = []
    ds_card = train_dataset.cardinality
    for epoch in range(1, epochs + 1):
        # Learning rate schedule
        if helper.opt_schedule is not None:
            helper.optimizer.learning_rate = helper.opt_schedule(epoch)

        # Create progress bar
        if verbose:
            print(f"\nEpoch {epoch}/{epochs}")
            progbar = tf.keras.utils.Progbar(target=ds_card)

        # Run the training steps
        for i, batch in enumerate(train_dataset):
            loss, acc = helper.dist_train_step(batch)
            # Update prog bar
            if verbose:
                progbar.update(i + 1, zip(['loss', 'acc'], [loss, acc]),␣
↪finalize=False)

        # Run for the validation set (if any) and append to history
        if val_dataset is not None:
            val_loss, val_acc = 0.0, 0.0
            for j, batch in enumerate(val_dataset):
                vloss, vacc = helper.dist_val_step(batch)
                val_loss += vloss
                val_acc += vacc
            val_loss /= (j + 1)
            val_acc /= (j + 1)
            history.append([loss, acc, val_loss, val_acc])
            if verbose: progbar.update(i, zip(['loss', 'acc', 'val_loss',␣
↪'val_acc', 'lr'],
                                              [loss, acc, val_loss, val_acc,␣
↪helper.optimizer.learning_rate.numpy()]), finalize=True)
        else:
            history.append([loss, acc])
            if verbose: progbar.update(i, zip(['loss', 'acc', 'lr'], [loss,␣
↪acc, helper.optimizer.learning_rate.numpy()]), finalize=True)
```

```
        # Training checkpoint
        if type(ckpt_every) is int:
            if epoch % ckpt_every == 0:
                helper.checkpoint.save(helper.checkpoint_dir)

        # Plot training progression with the selected examples
        if type(plot_every) is int:
            if epoch % plot_every == 0 and examples is not None:
                plt.close()
                X, y = list(examples.take(1))[0]
                image_list = [X.numpy()[0], y.numpy()[0], helper.model(X).
↪numpy().argmax(axis=-1)[0]]
                image_list = [(255.0 * img).astype('uint8') if img.dtype !
↪='uint8' else img for img in image_list]
                fig, ax = plt.subplots(1, 3, figsize=(12, 24))
                fig.suptitle(f"\nEpoch {epoch}/{epochs}", y=0.6)
                ax[0].set_title("Image")
                ax[1].set_title("Mask")
                ax[2].set_title("Predicted Mask")
                for k in range(3):
                    ax[k].imshow(image_list[k], cmap="gray")
                    ax[k].axis("off")
                fig.tight_layout()
                plt.show()
    return history
```

5.1 Cross-validation

Nothing too fancy: GroupKFold with each of the recordings as a group

[ ]:
```
max_lr = 1.E-3

lr_decay_start, lr_decay_rate, lr_decay_step = (2, 0.1, 3)

model_param = {"input_shape": img_shape,
               "dropout": 0.2}

oof_dice = []
oof_IoU = []

fold = [["01", "02"], ["02", "01"]]

for i in range(2):
    # In case we're running this cell over and over again when searching␣
↪hyperparameters
    try:
```

```
        del helper
    except:
        pass

    # Restore the random seed and clear the current TF graph
    reset_seed()
    K.clear_session()

    # Set the augmentation, batching and distribution of the dataset.
    # The augmentation .map() should come after both the .batch() and .cache()
    # for increased variety of augmented samples.
    training_size = hela_train[fold[i][0]].cardinality().numpy()
    train_ds = hela_train[fold[i][0]].shuffle(training_size,␣
↪reshuffle_each_iteration=True)\
                                     .repeat(np.lcm(batch_size, training_size) /
↪/ (training_size))\
                                     .batch(batch_size, drop_remainder=False,␣
↪num_parallel_calls=tf.data.AUTOTUNE)\
                                     .map(pipeline, num_parallel_calls=tf.data.
↪AUTOTUNE)\
                                     .prefetch(tf.data.AUTOTUNE)
    dist_train = strategy.experimental_distribute_dataset(train_ds)

    # Same thing for the validation split
    validation_size = hela_train[fold[i][1]].cardinality().numpy()
    val_ds = hela_train[fold[i][1]].map(lambda X, y, sw: (X, y))\
                                   .cache()\
                                   .batch(2 * batch_size, drop_remainder=False,␣
↪num_parallel_calls=tf.data.AUTOTUNE)
    dist_val = strategy.experimental_distribute_dataset(val_ds)

    # GPU training
    gc.collect()
    with strategy.scope():
        gc.collect()
        helper = UNetHelper(strategy=strategy,
                            model_param=model_param,
                            loss_func=tf.keras.losses.
↪sparse_categorical_crossentropy,
                            optimizer=tf.keras.optimizers.
↪SGD(learning_rate=max_lr, momentum=0.99),
                            #opt_schedule=tf.keras.optimizers.schedules.
↪PiecewiseConstantDecay(boundaries=[5,], values=[1e-2, 1e-3]),
                            )
        if train_model:
```

```
            train(helper, dist_train, dist_val, val_ds.rebatch(1), max_epochs,␣
↪ckpt_every=60, plot_every=70, verbose=False)#, max_epochs, ckpt_every=60,␣
↪plot_every=70)
            helper.model.save(f"{tf_dir}/models/model_fold{i + 1}.keras")
        else:
            helper.model.load(f"{tf_dir}/models/model_fold{i + 1}.keras")

    # Out-of-fold results
    pred = helper.model.predict(val_ds.map(lambda X, y: X))
    oof_true = list(val_ds.map(lambda X, y: y).rebatch(validation_size).
↪take(1))[0]
    oof_dice.append(dice_loss(oof_true, pred).numpy().mean())
    oof_IoU.append(IoU(oof_true, pred).numpy().mean())
```

```
WARNING: All log messages before absl::InitializeLog() is called are written to
STDERR
W0000 00:00:1754503513.981665 1224702 auto_shard.cc:553] The
`assert_cardinality` transformation is currently not handled by the auto-shard
rewrite and will be removed.
W0000 00:00:1754503514.089969 1224702 auto_shard.cc:553] The
`assert_cardinality` transformation is currently not handled by the auto-shard
rewrite and will be removed.
2025-08-06 15:05:20.643849: E tensorflow/core/util/util.cc:131] oneDNN supports
DT_INT32 only on platforms with AVX-512. Falling back to the default Eigen-based
implementation if present.
E0000 00:00:1754503530.117998 1224702 meta_optimizer.cc:966] layout failed:
INVALID_ARGUMENT: Size of values 0 does not match size of permutation 4 @ fanin
shape inStatefulPartitionedCall/sequential_1/u_net_1/spatial_dropout2d_1/statele
ss_dropout/SelectV2-2-TransposeNHWCToNCHW-LayoutOptimizer
2025-08-06 15:06:02.925652: I tensorflow/core/framework/local_rendezvous.cc:405]
Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence
         [[{{node MultiDeviceIteratorGetNextFromShard}}]]
         [[RemoteCall]]
2025-08-06 15:06:14.019825: I tensorflow/core/framework/local_rendezvous.cc:405]
Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence
2025-08-06 15:06:56.849174: I tensorflow/core/framework/local_rendezvous.cc:405]
Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence
         [[{{node MultiDeviceIteratorGetNextFromShard}}]]
         [[RemoteCall]]
2025-08-06 15:07:46.888806: I tensorflow/core/framework/local_rendezvous.cc:405]
Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence
2025-08-06 15:10:04.231824: I tensorflow/core/framework/local_rendezvous.cc:405]
Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence
         [[{{node MultiDeviceIteratorGetNextFromShard}}]]
         [[RemoteCall]]
2025-08-06 15:13:59.394652: I tensorflow/core/framework/local_rendezvous.cc:405]
Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence
```

[ ]:
```
print("Average out-of-fold IoU: {:.6f}".format(np.mean(oof_IoU)))
print("Average out-of-fold dice loss: {:.6f}".format(np.mean(oof_dice)))
```

```
Average out-of-fold IoU: 0.860822
Average out-of-fold dice loss: 0.115691
```

5.2 Training with the entire dataset

Same as before, but this time for the entire training dataset

[ ]:
```
try:
    del helper
except:
    pass

reset_seed()
K.clear_session()

train_ds = hela_train["01"].concatenate(hela_train["02"])
training_size = train_ds.cardinality()
train_ds = train_ds.shuffle(training_size, reshuffle_each_iteration=True)\
                   .repeat(2 * np.lcm(batch_size, training_size) //␣
 ↪(training_size))\
                   .batch(batch_size, drop_remainder=False,␣
 ↪num_parallel_calls=tf.data.AUTOTUNE)\
                   .map(pipeline, num_parallel_calls=tf.data.AUTOTUNE)\
                   .prefetch(tf.data.AUTOTUNE)
dist_train = strategy.experimental_distribute_dataset(train_ds)

gc.collect()
with strategy.scope():
    gc.collect()
    helper = UNetHelper(strategy=strategy,
                        model_param=model_param,
                        loss_func=tf.keras.losses.
 ↪sparse_categorical_crossentropy,
                        optimizer=tf.keras.optimizers.SGD(learning_rate=max_lr,␣
 ↪momentum=0.99),
                        #opt_schedule=tf.keras.optimizers.schedules.
 ↪PiecewiseConstantDecay(boundaries=[5,], values=[1e-2, 1e-3]),
                        )
    if train_model:
        train(helper, dist_train, None, None, max_epochs, ckpt_every=60,␣
 ↪plot_every=None, verbose=True)
        helper.model.save(f"{tf_dir}/models/model_all.keras")
    else:
        helper.model.load(f"{tf_dir}/models/model_all.keras")
```

```
W0000 00:00:1754452284.630207  495847 auto_shard.cc:553] The
`assert_cardinality` transformation is currently not handled by the auto-shard
rewrite and will be removed.
W0000 00:00:1754452284.630408  495847 auto_shard.cc:553] The
`assert_cardinality` transformation is currently not handled by the auto-shard
rewrite and will be removed.


Epoch 1/280

E0000 00:00:1754452299.372376  495847 meta_optimizer.cc:966] layout failed:
INVALID_ARGUMENT: Size of values 0 does not match size of permutation 4 @ fanin
shape inStatefulPartitionedCall/sequential_1/u_net_1/spatial_dropout2d_1/statele
ss_dropout/SelectV2-2-TransposeNHWCToNCHW-LayoutOptimizer

41/42            49s 876ms/step -
loss: 0.9627 - acc: 0.5273 - lr: 0.0010

Epoch 2/280
41/42            41s 882ms/step -
loss: 0.7138 - acc: 0.6263 - lr: 0.0010

Epoch 3/280
41/42            41s 885ms/step -
loss: 0.5912 - acc: 0.6728 - lr: 0.0010

Epoch 4/280
41/42            41s 885ms/step -
loss: 0.5660 - acc: 0.7029 - lr: 0.0010

Epoch 5/280
41/42            41s 884ms/step -
loss: 0.5119 - acc: 0.7257 - lr: 0.0010

Epoch 6/280
41/42            41s 886ms/step -
loss: 0.4980 - acc: 0.7420 - lr: 0.0010

Epoch 7/280
41/42            41s 883ms/step -
loss: 0.4505 - acc: 0.7549 - lr: 0.0010

Epoch 8/280
41/42            41s 885ms/step -
loss: 0.4821 - acc: 0.7653 - lr: 0.0010

Epoch 9/280
41/42            41s 888ms/step -
loss: 0.4245 - acc: 0.7729 - lr: 0.0010
```

```
Epoch 10/280
41/42            41s 885ms/step -
loss: 0.4388 - acc: 0.7808 - lr: 0.0010

Epoch 11/280
41/42            41s 885ms/step -
loss: 0.4438 - acc: 0.7873 - lr: 0.0010

Epoch 12/280
41/42            41s 887ms/step -
loss: 0.4030 - acc: 0.7932 - lr: 0.0010

Epoch 13/280
41/42            41s 883ms/step -
loss: 0.3476 - acc: 0.7996 - lr: 0.0010

Epoch 14/280
41/42            41s 888ms/step -
loss: 0.3959 - acc: 0.8059 - lr: 0.0010

Epoch 15/280
41/42            41s 885ms/step -
loss: 0.4186 - acc: 0.8103 - lr: 0.0010

Epoch 16/280
41/42            41s 884ms/step -
loss: 0.4131 - acc: 0.8139 - lr: 0.0010

Epoch 17/280
41/42            41s 886ms/step -
loss: 0.3673 - acc: 0.8174 - lr: 0.0010

Epoch 18/280
41/42            41s 884ms/step -
loss: 0.3582 - acc: 0.8214 - lr: 0.0010

Epoch 19/280
41/42            41s 883ms/step -
loss: 0.2944 - acc: 0.8254 - lr: 0.0010

Epoch 20/280
41/42            41s 882ms/step -
loss: 0.3139 - acc: 0.8297 - lr: 0.0010

Epoch 21/280
41/42            41s 888ms/step -
loss: 0.3694 - acc: 0.8331 - lr: 0.0010
```

```
Epoch 22/280
41/42            41s 884ms/step -
loss: 0.3414 - acc: 0.8358 - lr: 0.0010

Epoch 23/280
41/42            41s 885ms/step -
loss: 0.3675 - acc: 0.8383 - lr: 0.0010

Epoch 24/280
41/42            41s 882ms/step -
loss: 0.3018 - acc: 0.8406 - lr: 0.0010

Epoch 25/280
41/42            41s 884ms/step -
loss: 0.4039 - acc: 0.8431 - lr: 0.0010

Epoch 26/280
41/42            41s 885ms/step -
loss: 0.2667 - acc: 0.8451 - lr: 0.0010

Epoch 27/280
41/42            41s 883ms/step -
loss: 0.4134 - acc: 0.8475 - lr: 0.0010

Epoch 28/280
41/42            41s 885ms/step -
loss: 0.2880 - acc: 0.8491 - lr: 0.0010

Epoch 29/280
41/42            41s 884ms/step -
loss: 0.3126 - acc: 0.8513 - lr: 0.0010

Epoch 30/280
41/42            41s 882ms/step -
loss: 0.3457 - acc: 0.8531 - lr: 0.0010

Epoch 31/280
41/42            41s 887ms/step -
loss: 0.3313 - acc: 0.8547 - lr: 0.0010

Epoch 32/280
41/42            41s 884ms/step -
loss: 0.3563 - acc: 0.8561 - lr: 0.0010

Epoch 33/280
41/42            40s 870ms/step -
loss: 0.3580 - acc: 0.8571 - lr: 0.0010
```

```
Epoch 34/280
41/42            40s 856ms/step -
loss: 0.2814 - acc: 0.8586 - lr: 0.0010

Epoch 35/280
41/42            41s 893ms/step -
loss: 0.2328 - acc: 0.8604 - lr: 0.0010

Epoch 36/280
41/42            42s 902ms/step -
loss: 0.2418 - acc: 0.8624 - lr: 0.0010

Epoch 37/280
41/42            40s 843ms/step -
loss: 0.3096 - acc: 0.8640 - lr: 0.0010

Epoch 38/280
41/42            40s 846ms/step -
loss: 0.4781 - acc: 0.8651 - lr: 0.0010

Epoch 39/280
41/42            41s 887ms/step -
loss: 0.3749 - acc: 0.8654 - lr: 0.0010

Epoch 40/280
41/42            42s 907ms/step -
loss: 0.3092 - acc: 0.8662 - lr: 0.0010

Epoch 41/280
41/42            42s 904ms/step -
loss: 0.2492 - acc: 0.8676 - lr: 0.0010

Epoch 42/280
41/42            40s 863ms/step -
loss: 0.2816 - acc: 0.8691 - lr: 0.0010

Epoch 43/280
41/42            43s 916ms/step -
loss: 0.3840 - acc: 0.8701 - lr: 0.0010

Epoch 44/280
41/42            39s 846ms/step -
loss: 0.3495 - acc: 0.8708 - lr: 0.0010

Epoch 45/280
41/42            40s 868ms/step -
loss: 0.3236 - acc: 0.8715 - lr: 0.0010
```

```
Epoch 46/280
41/42            42s 907ms/step -
loss: 0.3000 - acc: 0.8723 - lr: 0.0010

Epoch 47/280
41/42            42s 911ms/step -
loss: 0.2297 - acc: 0.8733 - lr: 0.0010

Epoch 48/280
41/42            42s 911ms/step -
loss: 0.2884 - acc: 0.8746 - lr: 0.0010

Epoch 49/280
41/42            42s 914ms/step -
loss: 0.2086 - acc: 0.8756 - lr: 0.0010

Epoch 50/280
41/42            42s 919ms/step -
loss: 0.2374 - acc: 0.8768 - lr: 0.0010

Epoch 51/280
41/42            42s 928ms/step -
loss: 0.2887 - acc: 0.8777 - lr: 0.0010

Epoch 52/280
41/42            42s 917ms/step -
loss: 0.3497 - acc: 0.8785 - lr: 0.0010

Epoch 53/280
41/42            42s 914ms/step -
loss: 0.2826 - acc: 0.8790 - lr: 0.0010

Epoch 54/280
41/42            42s 922ms/step -
loss: 0.3554 - acc: 0.8797 - lr: 0.0010

Epoch 55/280
41/42            42s 918ms/step -
loss: 0.3075 - acc: 0.8800 - lr: 0.0010

Epoch 56/280
41/42            42s 913ms/step -
loss: 0.2842 - acc: 0.8807 - lr: 0.0010

Epoch 57/280
41/42            42s 915ms/step -
loss: 0.2123 - acc: 0.8815 - lr: 0.0010
```

```
Epoch 58/280
41/42            43s 926ms/step -
loss: 0.2611 - acc: 0.8824 - lr: 0.0010

Epoch 59/280
41/42            42s 914ms/step -
loss: 0.3345 - acc: 0.8831 - lr: 0.0010

Epoch 60/280
41/42            43s 926ms/step -
loss: 0.2102 - acc: 0.8837 - lr: 0.0010

Epoch 61/280
41/42            43s 926ms/step -
loss: 0.2080 - acc: 0.8846 - lr: 0.0010

Epoch 62/280
41/42            43s 925ms/step -
loss: 0.3870 - acc: 0.8852 - lr: 0.0010

Epoch 63/280
41/42            43s 924ms/step -
loss: 0.3202 - acc: 0.8855 - lr: 0.0010

Epoch 64/280
41/42            42s 925ms/step -
loss: 0.2797 - acc: 0.8859 - lr: 0.0010

Epoch 65/280
41/42            43s 913ms/step -
loss: 0.2161 - acc: 0.8866 - lr: 0.0010

Epoch 66/280
41/42            42s 914ms/step -
loss: 0.2572 - acc: 0.8874 - lr: 0.0010

Epoch 67/280
41/42            42s 914ms/step -
loss: 0.2441 - acc: 0.8880 - lr: 0.0010

Epoch 68/280
41/42            42s 915ms/step -
loss: 0.2446 - acc: 0.8887 - lr: 0.0010

Epoch 69/280
41/42            42s 915ms/step -
loss: 0.1920 - acc: 0.8894 - lr: 0.0010
```

```
Epoch 70/280
41/42            42s 914ms/step -
loss: 0.2038 - acc: 0.8901 - lr: 0.0010

Epoch 71/280
41/42            42s 925ms/step -
loss: 0.2599 - acc: 0.8908 - lr: 0.0010

Epoch 72/280
41/42            42s 924ms/step -
loss: 0.3592 - acc: 0.8911 - lr: 0.0010

Epoch 73/280
41/42            43s 925ms/step -
loss: 0.2024 - acc: 0.8915 - lr: 0.0010

Epoch 74/280
41/42            43s 926ms/step -
loss: 0.3753 - acc: 0.8920 - lr: 0.0010

Epoch 75/280
41/42            42s 925ms/step -
loss: 0.2117 - acc: 0.8922 - lr: 0.0010

Epoch 76/280
41/42            43s 925ms/step -
loss: 0.2503 - acc: 0.8929 - lr: 0.0010

Epoch 77/280
41/42            43s 926ms/step -
loss: 0.1893 - acc: 0.8934 - lr: 0.0010

Epoch 78/280
41/42            42s 924ms/step -
loss: 0.3243 - acc: 0.8940 - lr: 0.0010

Epoch 79/280
41/42            43s 927ms/step -
loss: 0.2262 - acc: 0.8943 - lr: 0.0010

Epoch 80/280
41/42            43s 925ms/step -
loss: 0.3952 - acc: 0.8947 - lr: 0.0010

Epoch 81/280
41/42            43s 924ms/step -
loss: 0.3014 - acc: 0.8947 - lr: 0.0010
```

```
Epoch 82/280
41/42            42s 914ms/step -
loss: 0.2911 - acc: 0.8949 - lr: 0.0010

Epoch 83/280
41/42            42s 915ms/step -
loss: 0.2735 - acc: 0.8953 - lr: 0.0010

Epoch 84/280
41/42            42s 915ms/step -
loss: 0.3100 - acc: 0.8956 - lr: 0.0010

Epoch 85/280
41/42            42s 913ms/step -
loss: 0.2528 - acc: 0.8959 - lr: 0.0010

Epoch 86/280
41/42            42s 915ms/step -
loss: 0.1892 - acc: 0.8963 - lr: 0.0010

Epoch 87/280
41/42            43s 926ms/step -
loss: 0.2014 - acc: 0.8968 - lr: 0.0010

Epoch 88/280
41/42            43s 925ms/step -
loss: 0.3113 - acc: 0.8973 - lr: 0.0010

Epoch 89/280
41/42            43s 926ms/step -
loss: 0.3294 - acc: 0.8975 - lr: 0.0010

Epoch 90/280
41/42            43s 927ms/step -
loss: 0.2753 - acc: 0.8977 - lr: 0.0010

Epoch 91/280
41/42            43s 925ms/step -
loss: 0.2598 - acc: 0.8979 - lr: 0.0010

Epoch 92/280
41/42            43s 926ms/step -
loss: 0.2692 - acc: 0.8983 - lr: 0.0010

Epoch 93/280
41/42            43s 928ms/step -
loss: 0.1850 - acc: 0.8986 - lr: 0.0010
```

```
Epoch 94/280
41/42            42s 914ms/step -
loss: 0.1826 - acc: 0.8992 - lr: 0.0010

Epoch 95/280
41/42            42s 914ms/step -
loss: 0.2472 - acc: 0.8996 - lr: 0.0010

Epoch 96/280
41/42            42s 914ms/step -
loss: 0.2478 - acc: 0.9000 - lr: 0.0010

Epoch 97/280
41/42            42s 914ms/step -
loss: 0.3764 - acc: 0.9002 - lr: 0.0010

Epoch 98/280
41/42            43s 926ms/step -
loss: 0.1943 - acc: 0.9004 - lr: 0.0010

Epoch 99/280
41/42            43s 926ms/step -
loss: 0.2451 - acc: 0.9008 - lr: 0.0010

Epoch 100/280
41/42            43s 926ms/step -
loss: 0.2337 - acc: 0.9011 - lr: 0.0010

Epoch 101/280
41/42            43s 924ms/step -
loss: 0.3741 - acc: 0.9012 - lr: 0.0010

Epoch 102/280
41/42            42s 924ms/step -
loss: 0.2565 - acc: 0.9014 - lr: 0.0010

Epoch 103/280
41/42            42s 929ms/step -
loss: 0.3091 - acc: 0.9016 - lr: 0.0010

Epoch 104/280
41/42            42s 916ms/step -
loss: 0.2723 - acc: 0.9017 - lr: 0.0010

Epoch 105/280
41/42            43s 926ms/step -
loss: 0.2499 - acc: 0.9019 - lr: 0.0010
```

```
Epoch 106/280
41/42            43s 928ms/step -
loss: 0.2566 - acc: 0.9022 - lr: 0.0010

Epoch 107/280
41/42            43s 927ms/step -
loss: 0.1784 - acc: 0.9025 - lr: 0.0010

Epoch 108/280
41/42            43s 927ms/step -
loss: 0.1892 - acc: 0.9029 - lr: 0.0010

Epoch 109/280
41/42            43s 928ms/step -
loss: 0.3461 - acc: 0.9032 - lr: 0.0010

Epoch 110/280
41/42            43s 928ms/step -
loss: 0.3324 - acc: 0.9033 - lr: 0.0010

Epoch 111/280
41/42            43s 928ms/step -
loss: 0.1943 - acc: 0.9034 - lr: 0.0010

Epoch 112/280
41/42            43s 927ms/step -
loss: 0.1861 - acc: 0.9038 - lr: 0.0010

Epoch 113/280
41/42            43s 930ms/step -
loss: 0.1770 - acc: 0.9043 - lr: 0.0010

Epoch 114/280
41/42            43s 928ms/step -
loss: 0.2878 - acc: 0.9046 - lr: 0.0010

Epoch 115/280
41/42            43s 928ms/step -
loss: 0.2076 - acc: 0.9048 - lr: 0.0010

Epoch 116/280
41/42            43s 927ms/step -
loss: 0.2665 - acc: 0.9051 - lr: 0.0010

Epoch 117/280
41/42            43s 926ms/step -
loss: 0.3286 - acc: 0.9052 - lr: 0.0010
```

```
Epoch 118/280
41/42            43s 926ms/step -
loss: 0.3013 - acc: 0.9053 - lr: 0.0010

Epoch 119/280
41/42            43s 927ms/step -
loss: 0.2746 - acc: 0.9053 - lr: 0.0010

Epoch 120/280
41/42            43s 926ms/step -
loss: 0.3077 - acc: 0.9055 - lr: 0.0010

Epoch 121/280
41/42            43s 927ms/step -
loss: 0.2429 - acc: 0.9056 - lr: 0.0010

Epoch 122/280
41/42            43s 927ms/step -
loss: 0.2708 - acc: 0.9057 - lr: 0.0010

Epoch 123/280
41/42            43s 927ms/step -
loss: 0.2257 - acc: 0.9060 - lr: 0.0010

Epoch 124/280
41/42            43s 926ms/step -
loss: 0.3190 - acc: 0.9062 - lr: 0.0010

Epoch 125/280
41/42            43s 926ms/step -
loss: 0.2701 - acc: 0.9063 - lr: 0.0010

Epoch 126/280
41/42            42s 926ms/step -
loss: 0.2600 - acc: 0.9064 - lr: 0.0010

Epoch 127/280
41/42            43s 927ms/step -
loss: 0.3022 - acc: 0.9066 - lr: 0.0010

Epoch 128/280
41/42            43s 928ms/step -
loss: 0.2287 - acc: 0.9067 - lr: 0.0010

Epoch 129/280
41/42            43s 928ms/step -
loss: 0.1758 - acc: 0.9070 - lr: 0.0010
```

```
Epoch 130/280
41/42            43s 927ms/step -
loss: 0.2881 - acc: 0.9073 - lr: 0.0010

Epoch 131/280
41/42            43s 928ms/step -
loss: 0.1678 - acc: 0.9075 - lr: 0.0010

Epoch 132/280
41/42            43s 928ms/step -
loss: 0.2365 - acc: 0.9077 - lr: 0.0010

Epoch 133/280
41/42            43s 926ms/step -
loss: 0.2541 - acc: 0.9080 - lr: 0.0010

Epoch 134/280
41/42            43s 927ms/step -
loss: 0.2288 - acc: 0.9081 - lr: 0.0010

Epoch 135/280
41/42            43s 927ms/step -
loss: 0.3106 - acc: 0.9083 - lr: 0.0010

Epoch 136/280
41/42            43s 925ms/step -
loss: 0.2557 - acc: 0.9083 - lr: 0.0010

Epoch 137/280
41/42            43s 927ms/step -
loss: 0.2458 - acc: 0.9085 - lr: 0.0010

Epoch 138/280
41/42            43s 929ms/step -
loss: 0.2048 - acc: 0.9087 - lr: 0.0010

Epoch 139/280
41/42            43s 924ms/step -
loss: 0.2429 - acc: 0.9089 - lr: 0.0010

Epoch 140/280
41/42            43s 927ms/step -
loss: 0.1945 - acc: 0.9091 - lr: 0.0010

Epoch 141/280
41/42            43s 927ms/step -
loss: 0.2071 - acc: 0.9094 - lr: 0.0010
```

```
Epoch 142/280
41/42            43s 924ms/step -
loss: 0.2241 - acc: 0.9096 - lr: 0.0010

Epoch 143/280
41/42            43s 927ms/step -
loss: 0.2950 - acc: 0.9097 - lr: 0.0010

Epoch 144/280
41/42            43s 928ms/step -
loss: 0.1685 - acc: 0.9099 - lr: 0.0010

Epoch 145/280
41/42            43s 925ms/step -
loss: 0.2170 - acc: 0.9102 - lr: 0.0010

Epoch 146/280
41/42            42s 922ms/step -
loss: 0.2514 - acc: 0.9103 - lr: 0.0010

Epoch 147/280
41/42            42s 915ms/step -
loss: 0.2179 - acc: 0.9105 - lr: 0.0010

Epoch 148/280
41/42            42s 915ms/step -
loss: 0.2203 - acc: 0.9107 - lr: 0.0010

Epoch 149/280
41/42            42s 914ms/step -
loss: 0.3323 - acc: 0.9108 - lr: 0.0010

Epoch 150/280
41/42            42s 915ms/step -
loss: 0.2517 - acc: 0.9108 - lr: 0.0010

Epoch 151/280
41/42            43s 925ms/step -
loss: 0.2849 - acc: 0.9109 - lr: 0.0010

Epoch 152/280
41/42            43s 924ms/step -
loss: 0.2153 - acc: 0.9110 - lr: 0.0010

Epoch 153/280
41/42            43s 925ms/step -
loss: 0.2882 - acc: 0.9112 - lr: 0.0010
```

```
Epoch 154/280
41/42            43s 927ms/step -
loss: 0.3528 - acc: 0.9112 - lr: 0.0010

Epoch 155/280
41/42            43s 924ms/step -
loss: 0.2780 - acc: 0.9112 - lr: 0.0010

Epoch 156/280
41/42            43s 926ms/step -
loss: 0.3165 - acc: 0.9112 - lr: 0.0010

Epoch 157/280
41/42            43s 926ms/step -
loss: 0.2258 - acc: 0.9113 - lr: 0.0010

Epoch 158/280
41/42            43s 925ms/step -
loss: 0.2179 - acc: 0.9115 - lr: 0.0010

Epoch 159/280
41/42            42s 915ms/step -
loss: 0.2200 - acc: 0.9117 - lr: 0.0010

Epoch 160/280
41/42            43s 928ms/step -
loss: 0.2540 - acc: 0.9118 - lr: 0.0010

Epoch 161/280
41/42            43s 925ms/step -
loss: 0.2124 - acc: 0.9120 - lr: 0.0010

Epoch 162/280
41/42            43s 926ms/step -
loss: 0.2737 - acc: 0.9121 - lr: 0.0010

Epoch 163/280
41/42            43s 926ms/step -
loss: 0.1964 - acc: 0.9122 - lr: 0.0010

Epoch 164/280
41/42            43s 924ms/step -
loss: 0.2106 - acc: 0.9124 - lr: 0.0010

Epoch 165/280
41/42            43s 924ms/step -
loss: 0.2580 - acc: 0.9126 - lr: 0.0010
```

```
Epoch 166/280
41/42            43s 925ms/step -
loss: 0.3192 - acc: 0.9126 - lr: 0.0010

Epoch 167/280
41/42            43s 923ms/step -
loss: 0.3017 - acc: 0.9127 - lr: 0.0010

Epoch 168/280
41/42            43s 926ms/step -
loss: 0.2435 - acc: 0.9127 - lr: 0.0010

Epoch 169/280
41/42            43s 926ms/step -
loss: 0.2224 - acc: 0.9128 - lr: 0.0010

Epoch 170/280
41/42            43s 923ms/step -
loss: 0.2896 - acc: 0.9129 - lr: 0.0010

Epoch 171/280
41/42            43s 925ms/step -
loss: 0.1838 - acc: 0.9131 - lr: 0.0010

Epoch 172/280
41/42            45s 995ms/step -
loss: 0.2045 - acc: 0.9133 - lr: 0.0010

Epoch 173/280
41/42            42s 921ms/step -
loss: 0.3133 - acc: 0.9134 - lr: 0.0010

Epoch 174/280
41/42            43s 925ms/step -
loss: 0.2783 - acc: 0.9134 - lr: 0.0010

Epoch 175/280
41/42            43s 926ms/step -
loss: 0.2162 - acc: 0.9135 - lr: 0.0010

Epoch 176/280
41/42            43s 921ms/step -
loss: 0.2279 - acc: 0.9137 - lr: 0.0010

Epoch 177/280
41/42            43s 926ms/step -
loss: 0.2818 - acc: 0.9137 - lr: 0.0010
```

```
Epoch 178/280
41/42            43s 926ms/step -
loss: 0.1681 - acc: 0.9139 - lr: 0.0010

Epoch 179/280
41/42            43s 924ms/step -
loss: 0.1631 - acc: 0.9141 - lr: 0.0010

Epoch 180/280
41/42            43s 924ms/step -
loss: 0.1997 - acc: 0.9143 - lr: 0.0010

Epoch 181/280
41/42            43s 925ms/step -
loss: 0.2202 - acc: 0.9145 - lr: 0.0010

Epoch 182/280
41/42            43s 923ms/step -
loss: 0.1969 - acc: 0.9146 - lr: 0.0010

Epoch 183/280
41/42            43s 925ms/step -
loss: 0.1535 - acc: 0.9148 - lr: 0.0010

Epoch 184/280
41/42            43s 926ms/step -
loss: 0.2359 - acc: 0.9150 - lr: 0.0010

Epoch 185/280
41/42            43s 923ms/step -
loss: 0.3188 - acc: 0.9151 - lr: 0.0010

Epoch 186/280
41/42            43s 924ms/step -
loss: 0.3237 - acc: 0.9151 - lr: 0.0010

Epoch 187/280
41/42            43s 925ms/step -
loss: 0.2545 - acc: 0.9151 - lr: 0.0010

Epoch 188/280
41/42            43s 924ms/step -
loss: 0.3081 - acc: 0.9151 - lr: 0.0010

Epoch 189/280
41/42            43s 924ms/step -
loss: 0.2091 - acc: 0.9152 - lr: 0.0010
```

```
Epoch 190/280
41/42            43s 926ms/step -
loss: 0.2202 - acc: 0.9153 - lr: 0.0010

Epoch 191/280
41/42            43s 924ms/step -
loss: 0.2334 - acc: 0.9154 - lr: 0.0010

Epoch 192/280
41/42            43s 925ms/step -
loss: 0.2371 - acc: 0.9155 - lr: 0.0010

Epoch 193/280
41/42            43s 926ms/step -
loss: 0.1914 - acc: 0.9157 - lr: 0.0010

Epoch 194/280
41/42            43s 927ms/step -
loss: 0.1552 - acc: 0.9158 - lr: 0.0010

Epoch 195/280
41/42            43s 925ms/step -
loss: 0.2768 - acc: 0.9160 - lr: 0.0010

Epoch 196/280
41/42            43s 926ms/step -
loss: 0.3225 - acc: 0.9160 - lr: 0.0010

Epoch 197/280
41/42            43s 926ms/step -
loss: 0.2581 - acc: 0.9160 - lr: 0.0010

Epoch 198/280
41/42            43s 925ms/step -
loss: 0.1758 - acc: 0.9161 - lr: 0.0010

Epoch 199/280
41/42            43s 926ms/step -
loss: 0.2587 - acc: 0.9163 - lr: 0.0010

Epoch 200/280
41/42            43s 926ms/step -
loss: 0.2646 - acc: 0.9163 - lr: 0.0010

Epoch 201/280
41/42            43s 926ms/step -
loss: 0.1991 - acc: 0.9164 - lr: 0.0010
```

```
Epoch 202/280
41/42            43s 925ms/step -
loss: 0.2990 - acc: 0.9165 - lr: 0.0010

Epoch 203/280
41/42            43s 925ms/step -
loss: 0.1981 - acc: 0.9166 - lr: 0.0010

Epoch 204/280
41/42            43s 923ms/step -
loss: 0.2584 - acc: 0.9167 - lr: 0.0010

Epoch 205/280
41/42            43s 926ms/step -
loss: 0.2945 - acc: 0.9167 - lr: 0.0010

Epoch 206/280
41/42            43s 926ms/step -
loss: 0.2740 - acc: 0.9168 - lr: 0.0010

Epoch 207/280
41/42            43s 924ms/step -
loss: 0.2915 - acc: 0.9168 - lr: 0.0010

Epoch 208/280
41/42            43s 926ms/step -
loss: 0.2179 - acc: 0.9168 - lr: 0.0010

Epoch 209/280
41/42            43s 926ms/step -
loss: 0.3015 - acc: 0.9169 - lr: 0.0010

Epoch 210/280
41/42            43s 925ms/step -
loss: 0.2483 - acc: 0.9169 - lr: 0.0010

Epoch 211/280
41/42            43s 926ms/step -
loss: 0.1851 - acc: 0.9170 - lr: 0.0010

Epoch 212/280
41/42            43s 926ms/step -
loss: 0.2609 - acc: 0.9172 - lr: 0.0010

Epoch 213/280
41/42            43s 923ms/step -
loss: 0.2543 - acc: 0.9172 - lr: 0.0010
```

```
Epoch 214/280
41/42            43s 927ms/step -
loss: 0.1828 - acc: 0.9173 - lr: 0.0010

Epoch 215/280
41/42            43s 929ms/step -
loss: 0.2879 - acc: 0.9174 - lr: 0.0010

Epoch 216/280
41/42            43s 923ms/step -
loss: 0.3457 - acc: 0.9174 - lr: 0.0010

Epoch 217/280
41/42            43s 926ms/step -
loss: 0.2202 - acc: 0.9174 - lr: 0.0010

Epoch 218/280
41/42            43s 926ms/step -
loss: 0.2615 - acc: 0.9175 - lr: 0.0010

Epoch 219/280
41/42            43s 924ms/step -
loss: 0.2118 - acc: 0.9175 - lr: 0.0010

Epoch 220/280
41/42            43s 925ms/step -
loss: 0.1615 - acc: 0.9177 - lr: 0.0010

Epoch 221/280
41/42            43s 927ms/step -
loss: 0.2201 - acc: 0.9178 - lr: 0.0010

Epoch 222/280
41/42            43s 924ms/step -
loss: 0.1537 - acc: 0.9179 - lr: 0.0010

Epoch 223/280
41/42            46s 997ms/step -
loss: 0.3344 - acc: 0.9181 - lr: 0.0010

Epoch 224/280
41/42            43s 924ms/step -
loss: 0.2117 - acc: 0.9181 - lr: 0.0010

Epoch 225/280
41/42            43s 924ms/step -
loss: 0.1663 - acc: 0.9182 - lr: 0.0010
```

```
Epoch 226/280
41/42            43s 928ms/step -
loss: 0.3004 - acc: 0.9183 - lr: 0.0010

Epoch 227/280
41/42            43s 925ms/step -
loss: 0.2364 - acc: 0.9183 - lr: 0.0010

Epoch 228/280
41/42            43s 924ms/step -
loss: 0.2973 - acc: 0.9184 - lr: 0.0010

Epoch 229/280
41/42            43s 924ms/step -
loss: 0.2217 - acc: 0.9184 - lr: 0.0010

Epoch 230/280
41/42            43s 926ms/step -
loss: 0.1916 - acc: 0.9185 - lr: 0.0010

Epoch 231/280
41/42            43s 924ms/step -
loss: 0.2234 - acc: 0.9186 - lr: 0.0010

Epoch 232/280
41/42            43s 927ms/step -
loss: 0.2336 - acc: 0.9187 - lr: 0.0010

Epoch 233/280
41/42            43s 926ms/step -
loss: 0.2462 - acc: 0.9188 - lr: 0.0010

Epoch 234/280
41/42            43s 926ms/step -
loss: 0.2920 - acc: 0.9188 - lr: 0.0010

Epoch 235/280
41/42            43s 926ms/step -
loss: 0.2520 - acc: 0.9188 - lr: 0.0010

Epoch 236/280
41/42            43s 927ms/step -
loss: 0.2204 - acc: 0.9189 - lr: 0.0010

Epoch 237/280
41/42            43s 924ms/step -
loss: 0.2978 - acc: 0.9189 - lr: 0.0010
```

```
Epoch 238/280
41/42            43s 927ms/step -
loss: 0.2690 - acc: 0.9189 - lr: 0.0010

Epoch 239/280
41/42            43s 926ms/step -
loss: 0.2143 - acc: 0.9190 - lr: 0.0010

Epoch 240/280
41/42            43s 925ms/step -
loss: 0.2665 - acc: 0.9191 - lr: 0.0010
2025-08-06 03:40:47.224764: I tensorflow/core/framework/local_rendezvous.cc:405]
Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence


Epoch 241/280
41/42            43s 927ms/step -
loss: 0.1777 - acc: 0.9191 - lr: 0.0010

Epoch 242/280
41/42            43s 926ms/step -
loss: 0.2527 - acc: 0.9192 - lr: 0.0010

Epoch 243/280
41/42            43s 924ms/step -
loss: 0.2587 - acc: 0.9193 - lr: 0.0010

Epoch 244/280
41/42            43s 925ms/step -
loss: 0.3313 - acc: 0.9193 - lr: 0.0010

Epoch 245/280
41/42            43s 927ms/step -
loss: 0.2921 - acc: 0.9192 - lr: 0.0010

Epoch 246/280
41/42            42s 912ms/step -
loss: 0.2583 - acc: 0.9192 - lr: 0.0010

Epoch 247/280
41/42            42s 910ms/step -
loss: 0.2702 - acc: 0.9193 - lr: 0.0010

Epoch 248/280
41/42            42s 898ms/step -
loss: 0.1728 - acc: 0.9193 - lr: 0.0010
```

```
Epoch 249/280
41/42             42s 899ms/step -
loss: 0.1705 - acc: 0.9195 - lr: 0.0010

Epoch 250/280
41/42             42s 898ms/step -
loss: 0.2093 - acc: 0.9196 - lr: 0.0010

Epoch 251/280
41/42             42s 900ms/step -
loss: 0.2110 - acc: 0.9197 - lr: 0.0010

Epoch 252/280
41/42             42s 901ms/step -
loss: 0.1528 - acc: 0.9198 - lr: 0.0010

Epoch 253/280
41/42             42s 901ms/step -
loss: 0.3192 - acc: 0.9199 - lr: 0.0010

Epoch 254/280
41/42             42s 896ms/step -
loss: 0.2566 - acc: 0.9199 - lr: 0.0010

Epoch 255/280
41/42             42s 899ms/step -
loss: 0.2075 - acc: 0.9200 - lr: 0.0010

Epoch 256/280
41/42             42s 898ms/step -
loss: 0.1963 - acc: 0.9200 - lr: 0.0010

Epoch 257/280
41/42             42s 902ms/step -
loss: 0.3312 - acc: 0.9201 - lr: 0.0010

Epoch 258/280
41/42             42s 901ms/step -
loss: 0.1960 - acc: 0.9201 - lr: 0.0010

Epoch 259/280
41/42             42s 901ms/step -
loss: 0.1682 - acc: 0.9202 - lr: 0.0010

Epoch 260/280
41/42             42s 903ms/step -
loss: 0.1515 - acc: 0.9204 - lr: 0.0010
```

```
Epoch 261/280
41/42             42s 902ms/step -
loss: 0.2098 - acc: 0.9205 - lr: 0.0010

Epoch 262/280
41/42             42s 904ms/step -
loss: 0.3021 - acc: 0.9205 - lr: 0.0010

Epoch 263/280
41/42             42s 900ms/step -
loss: 0.2523 - acc: 0.9205 - lr: 0.0010

Epoch 264/280
41/42             42s 900ms/step -
loss: 0.2202 - acc: 0.9206 - lr: 0.0010

Epoch 265/280
41/42             42s 901ms/step -
loss: 0.2558 - acc: 0.9206 - lr: 0.0010

Epoch 266/280
41/42             42s 900ms/step -
loss: 0.1639 - acc: 0.9207 - lr: 0.0010

Epoch 267/280
41/42             42s 898ms/step -
loss: 0.3699 - acc: 0.9207 - lr: 0.0010

Epoch 268/280
41/42             42s 902ms/step -
loss: 0.2559 - acc: 0.9207 - lr: 0.0010

Epoch 269/280
41/42             42s 898ms/step -
loss: 0.2551 - acc: 0.9207 - lr: 0.0010

Epoch 270/280
41/42             42s 898ms/step -
loss: 0.1696 - acc: 0.9208 - lr: 0.0010

Epoch 271/280
41/42             42s 901ms/step -
loss: 0.2071 - acc: 0.9209 - lr: 0.0010

Epoch 272/280
41/42             42s 901ms/step -
loss: 0.1511 - acc: 0.9210 - lr: 0.0010
```

```
Epoch 273/280
41/42              42s 898ms/step -
loss: 0.2632 - acc: 0.9211 - lr: 0.0010

Epoch 274/280
41/42              42s 901ms/step -
loss: 0.2245 - acc: 0.9212 - lr: 0.0010

Epoch 275/280
41/42              42s 900ms/step -
loss: 0.2867 - acc: 0.9212 - lr: 0.0010

Epoch 276/280
41/42              42s 900ms/step -
loss: 0.2643 - acc: 0.9212 - lr: 0.0010

Epoch 277/280
41/42              42s 899ms/step -
loss: 0.1697 - acc: 0.9212 - lr: 0.0010

Epoch 278/280
41/42              42s 901ms/step -
loss: 0.2722 - acc: 0.9213 - lr: 0.0010

Epoch 279/280
41/42              46s 995ms/step -
loss: 0.1886 - acc: 0.9214 - lr: 0.0010

Epoch 280/280
41/42              42s 898ms/step -
loss: 0.2138 - acc: 0.9215 - lr: 0.0010
```

6 Predictions with the trained neural network

[ ]:
```
def process_test(img):
    clh = cv.createCLAHE(clipLimit=2.0, tileGridSize=(8, 8))
    clh_img = clh.apply(np.squeeze(img.numpy()))
    return tf.constant(np.expand_dims(clh_img / 255.0, -1), dtype=tf.float32,␣
 ↪shape=img.get_shape())
```

6.1 Create movies using the training dataset

[ ]:
```
os.system(f"mkdir -p figures")
```

[ ]:
```
pred_train = tfds.load("hela_train", data_dir=tf_dir)
```

[ ]:
```
def make_movie(model, ds):
    movie_arr = np.zeros((ds.cardinality().numpy(), img_shape[0], img_shape[1]␣
 ↪* 2), dtype="uint8")
    for i, sample in enumerate(ds.map(lambda x: x["image"])):
        tmp = tf.expand_dims(process_test(sample), 0)
        pred = (model.predict(tmp, verbose=0).argmax(axis=-1)[0] * 255.0).
 ↪astype("uint8")
        movie_arr[i] = np.hstack((sample[...,0].numpy(), pred))
    return [Image.fromarray(frame) for frame in movie_arr]
```

[ ]:
```
# Segment #01
movie = make_movie(helper.model, pred_train["01"])
movie[0].save(f"figures/train01.gif", save_all=True, append_images=movie[1:],␣
 ↪duration=150, loop=0)
```

[ ]:
```
# Segment #02
movie = make_movie(helper.model, pred_train["02"])
movie[0].save(f"figures/train02.gif", save_all=True, append_images=movie[1:],␣
 ↪duration=150, loop=0)
```

6.2 Write the test predictions to dir

Use the CTC format (16-bit with labeled cells)

[462]:
```
os.system("mkdir -p predictions/01 predictions/02")
```

[462]: 0

[463]:
```
hela_test = tfds.load("hela_test", data_dir=tf_dir)
```

[ ]:
```
hela_sub = hela_test['01'].map(lambda pair: tf.py_function(process_test,␣
 ↪inp=[pair['image']], Tout=[tf.float32]),
                               num_parallel_calls=tf.data.AUTOTUNE)\
                           .map(lambda X: tf.ensure_shape(X, img_shape))\
                           .batch(2 * batch_size, num_parallel_calls=tf.data.
 ↪AUTOTUNE)

sub_pred = helper.model.predict(hela_sub).argmax(axis=-1)

for i in range(sub_pred.shape[0]):
    img = (sub_pred[i] * 255.0).astype("uint8")
    labeled_img = label(img).astype("uint16")
    cv.imwrite(f"predictions/01/p{str(i).zfill(3)}.tif", labeled_img)
```

[ ]:
```
hela_sub = hela_test['02'].map(lambda pair: tf.py_function(process_test,␣
 ↪inp=[pair['image']], Tout=[tf.float32]),
                               num_parallel_calls=tf.data.AUTOTUNE)\
```

```
                            .map(lambda X: tf.ensure_shape(X, img_shape))\
                            .batch(2 * batch_size, num_parallel_calls=tf.data.
 ↪AUTOTUNE)

sub_pred = helper.model.predict(hela_sub).argmax(axis=-1)

for i in range(sub_pred.shape[0]):
    img = (sub_pred[i] * 255.0).astype("uint8")
    labeled_img = label(img).astype("uint16")
    cv.imwrite(f"predictions/02/p{str(i).zfill(3)}.tif", labeled_img)
```

```
W0000 00:00:1754503152.303960  495847 auto_shard.cc:553] The
`assert_cardinality` transformation is currently not handled by the auto-shard
rewrite and will be removed.
```

```
8/8             5s 570ms/step
```

[24]:
```
# https://www.youtube.com/watch?v=dQw4w9WgXcQ
```