APR 01, 2024

OPEN ACCESS

DOI:
dx.doi.org/10.17504/protocols.io.81wgbz1x3gpk/v1

Protocol Citation: Kyu Sang Han, Pei-Hsun Wu, Joel Sunshine, Ashley Kiemen, Sashank Reddy, Denis Wirtz 2024. CODA (part 3): deep learning tissue structures labeling | HuBMAP | JHU-TMC. **protocols.io** https://dx.doi.org/10.17504/protocols.io.81wgbz1x3gpk/v1

MANUSCRIPT CITATION: Kiemen, A.L., Braxton, A.M., Grahn, M.P. *et al.* CODA: quantitative 3D reconstruction of large tissues at cellular resolution. *Nat Methods* **19**, 1490–1499 (2022). https://doi.org/10.1038/s41592-022-01650-9

CODA (part 3): deep learning tissue structures labeling | HuBMAP | JHU-TMC

Kyu Sang Han[1], Pei-Hsun Wu[1], Joel Sunshine[2], Ashley Kiemen[2], Sashank Reddy[2], Denis Wirtz[1,2]

[1]Johns Hopkins University; [2]Johns Hopkins Medicine

Human BioMolecular Atlas Program (HuBMAP) Method Development Community

TMC - Johns Hopkins University

Kyu Sang Han
Johns Hopkins University

ABSTRACT

In this section, we describe steps to create a basic semantic segmentation algorithm using CODA. CODA uses a modified resnet50 network adapted for semantic segmentation using an implementation of DeepLab. Here, we describe how to generate training datasets of manual annotations for a set of images, how to format the data for deep learning, and how to train and apply a CODA deep learning model.

Protocol status: Working
We use this protocol and it's working

Created: Mar 28, 2024

Last Modified: Apr 01, 2024

PROTOCOL integer ID: 97509

Funders Acknowledgement:
National Cancer Institute
Grant ID: U54CA143868
Institute of Arthritis and Musculoskeletal and Skin Diseases
Grant ID: U54AR081774

Deep learning multi-labelling of tissue structures using training on manual…

1 Choose the biological structures you wish to segment in your images. Semantic segmentation algorithms must classify every pixel of every image with a label, so your list must be exhaustive. For example, in lung histology you could choose to annotate:

a. Bronchioles
b. Alveoli
c. Vasculature
d. Metastases
e. Nonexpanded lung
f. Background
g. Stroma

Here, populations like fibroblasts and immune cells may be annotated inside of the ‘stroma’ layer. Background can encompass nontissue space as well as non-target noise within the histological images including red blood cells, shadows, and dust visible in the scanned images.

2 Next, select some original resolution (.ndpi or .svs) images to annotate. Try first choosing 7 training images and 1 testing image, then add images as necessary until your model performs acceptably (>90% quantitative accuracy + passes visual inspection) on an independent (not seen in training) testing image. Put images you wish to annotate in a separate folder named 'pthannotations':

pthannotations=[pth,'annotations']; % put images here to annotate for training

3 Inside of pthannotations, create a subfolder named pthim where you copy a corresponding high-resolution (here 10x) tif image of each image that you are annotating.

pthim=[pthannotations,'10x']; % copy a tif image here for each training image

Notes on the resolution you choose to train your model at: This choice highly depends on the resolution of structures you wish to label. If you want to quantify 'bulk' structures, low resolution (5x or lower) should be sufficient. If you want small structures such as small vasculature, small tubules, a medium resolution (~10x) should be sufficient. If you want to label individual cellular-sized strucutres, try ≥20x (you will need a computer with very high RAM and will have to very tediously annotate for this).

4 Inside of pthannotations, copy one additional 20x (.ndpi or .svs) image to a subfolder named 'pthtest,' with a corresponding high-resolution tif image saved in 'pthtestim'.

pthannotations_test=[pthannotations,'testing image']; % put image here to annotate for testing
pthtestim=[pthannotations_test,'10x']; % copy a tif image here for each testing image

5 For each image you wish to annotate, open the file in **Aperio ImageScope**. To generate the xml files that will contain annotation data, select the Annotations button. This will open the Annotations window

6 Generate annotation layers by pressing the plus button, and rename layers by hovering over the text that says 'Layer 1', and clicking left, then right, then left on the mouse in quick succession (I don't know why this works).

7 Make all layers for all structures you want to annotate in your sample. It is VERY IMPORTANT that all layers exist in the exact same order in all training and testing images you annotate. Create a layer even for structures that are not present in all images you annotate. If a layer is present in one image, it must be present in all images.

8 Next, create your annotations by selecting the pen tool in the ImageScope taskbar. Aim for ~25 annotations of each structure on each image. This is not always possible for rare structures.

Notes on annotating

9 The quality of your segmentation model depends on the quality of your annotations. Zoom-in to high magnification to annotate, and try to annotate very cleanly along the edges of structures.

10 Try to make all of annotations roughly the same size (do not make huge background annotations and tiny tissue annotations).

11 If your annotations are overlapping, they must follow consistent 'laws.' One annotation layer must always dominate the other layer, so that MATLAB understands how to interpret overlapping masks. Consider creating a list of 'nesting order' before you begin annotating to organize which annotations are the 'bottom layer' up to the 'top layer.' For example, to annotate bronchiole inside of alveoli in lung histology, the alveolar annotation (yellow) will encircle the bronchiole annotation (green). A background annotation (blue) will circle noise inside the bronchiole. Here, the background layer is 'dominant' over the bronchiole layer, and the bronchiole layer is 'dominant' over the alveolar layer, in cases of overlap.

After annotation

12 When you've finished making annotations, you are ready to set up your MATLAB training function. This package requires several variable definitions that are listed inside the top section of the function '**train_image_segmentation**.'
For the sample lung dataset, this function is filled out and saved as **train_image_segmentation_lung**.

13 To create the inputs for this function you will need the followings

14 The subfolder containing '.xml' files with your training annotation information (created by ImageScope)

pthannotations=[pth,'annotations']; % put images here to annotate for training

15 The subfolder containing '.xml' files with your testing annotation information (created by ImageScope)

pthannotations_test=[pthannotations,'testing image']; % put image here to annotate for testing

16 The subfolder containing high-resolution tif images corresponding to each annotated training image

pthim=[pthannotations,'10x']; % copy a tif image here for each training image

17 The subfolder containing high-resolution tif images corresponding to each annotated testing image

pthtestim=[pthannotations_test,'10x']; % copy a tif image here for each testing image

18 The subfolder containing the full dataset of high-resolution tif images that you want to classify with your model after training is finished

pthclassify=[pth,'10x'];

19 The downsample factor of your high-resolution tif images compared to the images you annotated in ImageScope (this is the same number you gave as input to the function **create_downsampled_tif_images** to create your high-resolution tif images).

umpix=2;

20 The date that your model was trained. This will define the output folder name where the trained model will be saved, such that unique folders are generated for different iterations of models made.

nm='12_15_2023';

21 The size (in pixels) of the tiles you want to create for model training. By default this is set at 700, resulting in creation of 700 x 700 x 3 sized RGB tiles. Depending on your GPU memory, you may be able to increase this (model performance will be better for larger tiles) or you may need to decrease this.

sxy=700;

22 The number of large training images you want to create. Each of these large images will be chopped into ~100 training tiles. A good number is around 15 but go lower or higher if you have very few (~5 images) or very many (>20 images) annotations.

ntrain=15;

The variable ‘WS’ - how to create it

23 The variable ‘**WS**’ is the most complicated to create, as it requires you to think critically about the annotation layers you created. Inside of this variable, you will define how to order your layers, whether to combine layers, and whether to keep or remove whitespace from your layers. For simplicity, **WS** is split into four components:

24 **annotation_whitespace** is a matrix of size [1 N] where N is the number of annotation layers you created in ImageScope. For each position within **annotation_whitespace**, indicate with 0 if you wish to remove the whitespace from your annotation (for example if you annotated a blood vessel and with to remove the luminal space from your annotation), indicate with 1 if you wish to keep only the whitespace in your annotation (for example if you annotated fat and wish to remove the nonwhite lines dividing separate fat cells from your annotation), and indicate with 2 if you wish to keep both whitespace and nonwhite space within your annotation.

For example, for the four classes:

1. Bronchioles
   (remove whitespace [contains lumen])
2. Alveoli
   (remove whitespace [keep only the webbing])
3. Vasculature
   (remove whitespace [lumen of vasculature])
4. Metastases
   (remove whitespace [whitespace around the edges of the cancer cells])
5. Nonexpanded
   lung (remove whitespace [keep only the webbing])
6. Background
   (keep both [this class contains whitespace + noise like shadows])
7. Stroma
   (remove whitespace [keep only the thin collagen fibers])

annotation_whitespace=[0 0 0 0 0 2 0];

25 **add_whitespace_to** is a matrix of size [1 2]. The first number defines the annotation layer to add whitespace to when it was removed from another class (where **annotation_whitespace** = 0) – this is usually the background class. The second number defines the annotation layer to add nonwhitespace to when it is removed from another class (where **annotation_whitespace**=1) – this is usually the stroma class. For

example, for the four classes listed above, add whitespace to class 6 (background), and add nonwhitespace to class 6 (also background as this is not applicable to this model).

add_whitespace_to=[6 6];

26 **nesting_order** is a matrix of size [1 N] where N is the number of annotation layers you created in ImageScope. This variable allows you to define how overlapping annotations should be processed. Numbers on the left side of this variable are 'below' annotations listed to the right of them. For example, in the example above, we annotated a bronchiole inside of an alveolar annotation. This means bronchioles must be 'above' alveoli in the nesting order. Let's assume that stroma is the 'bottom layer,' followed by alveoli, nonexpanded tissue, cancer, vasculature, bronchioles, and background.

nesting_order=[7 2 5 4 3 1 6]; % stroma, alveoli, nonexpanded, cancer, vessels, bronch., backgrd

In contrast, if none of your annotations overlap, nesting_order can be sequential:
nesting_order=[1 2 3 4 5]; % no particular nesting order

27 **combine_classes** is a matrix of size [1 N] where N is the number of annotation layers you created in ImageScope. This variable allows you to re-order or combine multiple classes. For example, if you originally annotate the seven classes listed above but decide to combine the alveoli and nonexpanded tissue classes your variable would look like:

combine_classes=[1 2 3 4 2 5 6];

You now will create a deep learning model that has only 6 classes.

28 These four defined variables will be combined into the variable '**WS**' in MATLAB.

29 Finally, create variables defining the names and RGB colors for the final tissue structures in your model. In our sample model, we have four classes (after we combined the classes fat and background using the combine_classes variable). For this, we define classNames and cmap. ClassNames is a string variable containing the names of each class. Note this variable cannot contain spaces in names, use underscore instead ("blood_vessels" instead of "blood vessels").

classNames=["bronchioles","alveoli","vasculature","cancer","nonexpanded","whitespace","stroma"];

30 cmap is a matrix of size [N 3] for N final classes in your deep learning model. Each row of this matrix defines the RGB color of one class of your deep learning model in 8-bit space.

For example:
cmap=[150 099 023;... % 1 bronchioles (brown)

023 080 150;... % 2 alveoli (dark blue)

150 031 023;... % 3 vasculature (dark red)

199 196 147;... % 4 cancer (v dark purple)

023 080 150;... % 5 nonexpanded (dark blue)

255 255 255;... % 6 whitespace (white)

242 167 227];... % 7 collagen (light pink)

31 With these inputs, you are ready to train your model. If you call the function **train_image_segmentation**, this will train a deep learning model (saved in a folder inside **pthannotations**), test that model using the annotations inside **pthannotations_test** and will classify the images in the folder **pthclassify**.

32 If you do not receive satisfactory results, add additional annotations (either entirely new training images or additional annotations on the same training images) until satisfactory results are achieved.

33 The workflow in this section will create a subfolder inside **pthannotations** containing the trained deep learning model and training tiles. These tiles can take up large amounts of disk space, so delete them (but keep your model inside the folder 'net.mat').

pthmodel=[pthannotations,nm];

34 This workflow will also create a subfolder inside the high-resolution tif image path named **['classification_',nm]**, where nm is the date of the deep learning model training. Inside this subfolder will be the segmented, high-resolution tif images.

pthclassified=[pth10x,'classification_',nm];

35 Inside of this subfolder will be a second folder named '**check_classification**,' containing color-labelled versions of the classified files. These colorized classified images are handy for visualization of the results and qualitative validation of model performance across the dataset.

```
pthcheckclassification=[pth10x,'classification_',nm,'check_classification'];
```