

Tensorflow-Image-Segmentation-IDRiD-Retinal-Vessel (2025/03/09)

Sarah T. Arai
Software Laboratory antillia.com

This is the first experiment of Image Segmentation for **IDRiD Retinal Vessel** based on the latest [Tensorflow-Image-Segmentation-API](#), and our dataset [Antillia-Augmented-IDRiD-Dataset.zip \(2.06G\)](#).

Please see also our experiments:

- [Tensorflow-Tiled-Image-Segmentation-IDRiD-Retinal-Vessel](#) based on [Indian Diabetic Retinopathy Image Dataset \(IDRiD\)](#).
- [Tensorflow-Tiled-Image-Segmentation-Pre-Augmented-HRF-Retinal-Vessel](#) based on [High-Resolution Fundus \(HRF\) Image Database](#).
- [Tensorflow-Tiled-Image-Segmentation-Pre-Augmented-DRIVE-Retinal-Vessel](#) based on [DRIVE: Digital Retinal Images for Vessel Extraction](#)
- [Tensorflow-Tiled-Image-Segmentation-Pre-Augmented-STARE-Retinal-Vessel](#) based on [STructured Analysis of the Retina](#).
- [Tensorflow-Image-Segmentation-Retinal-Vessel](#) based on [CHASE_DB1 dataset](#).

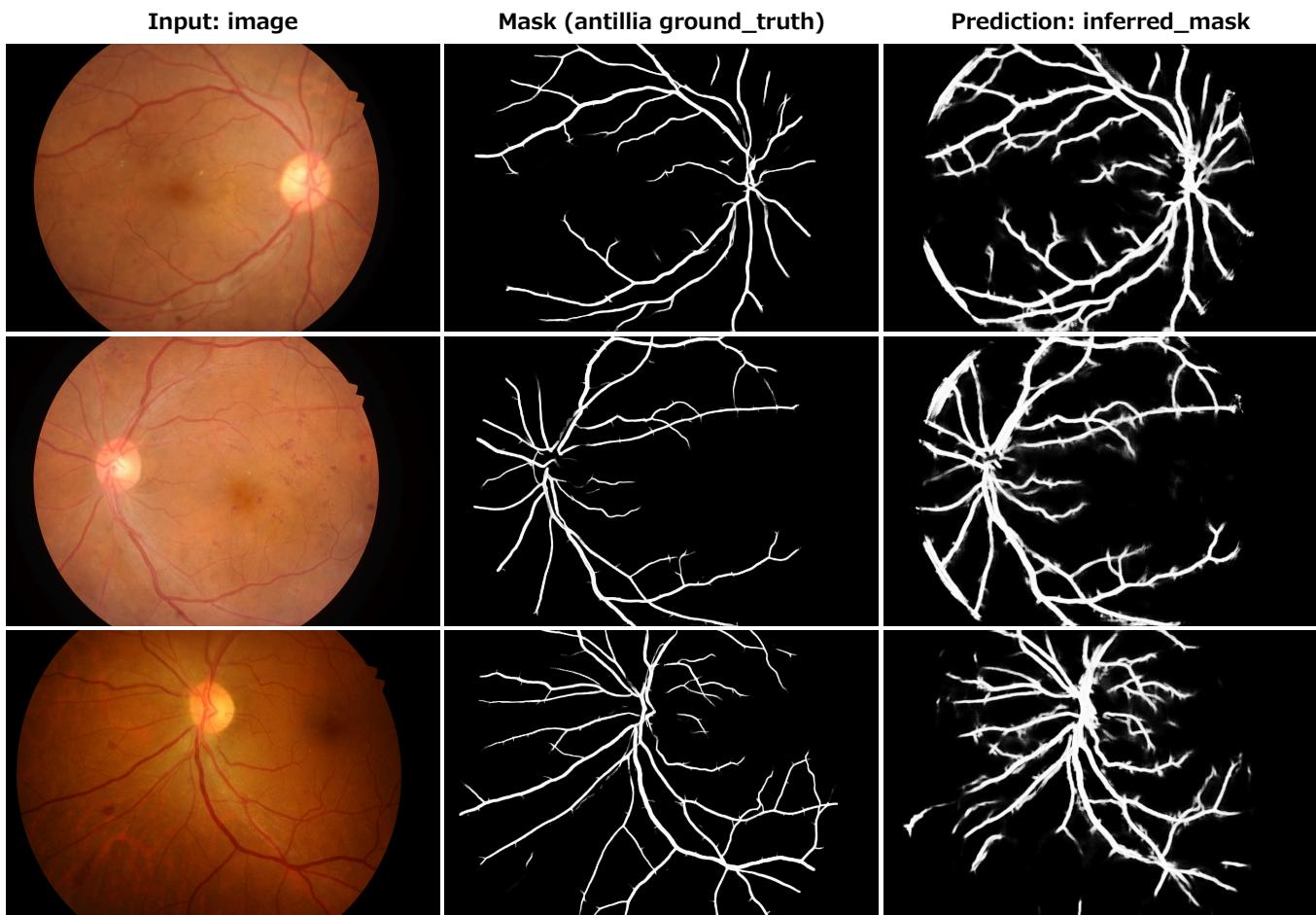
Experiment Strategy

As demonstrated in [Tensorflow-Image-Segmentation-FIVES-Retinal-Vessel](#), COLOR SPACE CONVERSION (cv2.COLOR_BGR2Luv) was effective in Non-Tiled Segmentation for Retinal Vessel. Therefore we employed the same method for this IDRiD Segmentation model.

Actual Image Segmentation for IDRiD images of 4288x2848 pixels

Our segmentation model infers masks that are similar to the ground truth, but it does not produce satisfactory results in detail. The simple UNet model, which takes input images of 512x512 pixels, does not work well for segmenting of high-resolution images.

For segmentation of large images using UNet, employing a divide and conquer algorithm is a good idea. Since large images cannot be processed all at once, they are divided into small tiles. Segmentation processing is then applied to these divided small images, and finally, they are reassembled to their original size. This is our proposed Tiled Image Segmentation method [Tensorflow-Tiled-Image-Segmentation-IDRiD-Retinal-Vessel](#)



In this experiment, we used the simple UNet Model [TensorflowSlightlyFlexibleUNet](#) for this HRF Segmentation Model.

As shown in [Tensorflow-Image-Segmentation-API](#). you may try other Tensorflow UNet Models:

- [TensorflowSwinUNet.py](#)
- [TensorflowMultiResUNet.py](#)
- [TensorflowAttentionUNet.py](#)
- [TensorflowEfficientUNet.py](#)
- [TensorflowUNet3Plus.py](#)
- [TensorflowDeepLabV3Plus.py](#)

1 Dataset Citation

The dataset used here has been taken from the following **IEEE DataPort** web site

[Indian Diabetic Retinopathy Image Dataset \(IDRiD\)](#)

Please see also [DIABETIC RETINOPATHY: SEGMENTATION AND GRAND CHALLENGE](#)

Citation Author(s):

Prasanna Porwal, Samiksha Pachade, Ravi Kamble, Manesh Kokare, Girish Deshmukh,
Vivek Sahasrabuddhe, Fabrice Meriaudeau,
April 24, 2018, "Indian Diabetic Retinopathy Image Dataset (IDRiD)", IEEE Dataport,

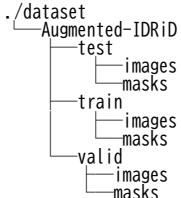
DOI: <https://dx.doi.org/10.21227/H25W98>

License:

[Creative Commons Attribution 4.0 International License.](#)

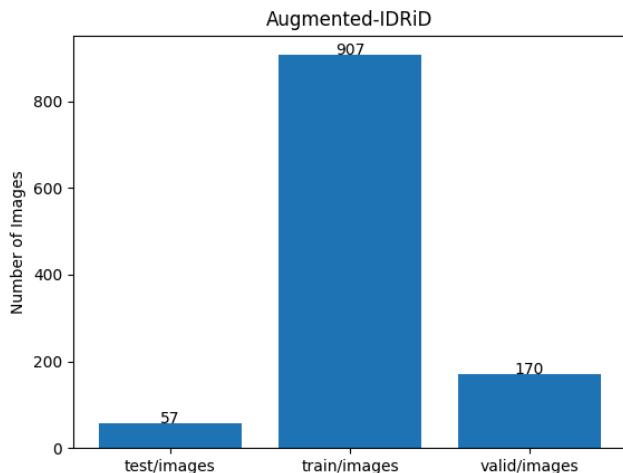
2 Augmented IDRiD Retinal Vessel Dataset

Please download the dataset from the google drive [Antillia-Augmented-IDRiD-Dataset.zip \(2.06G\)](#), expand it and place it under ./dataset to be.

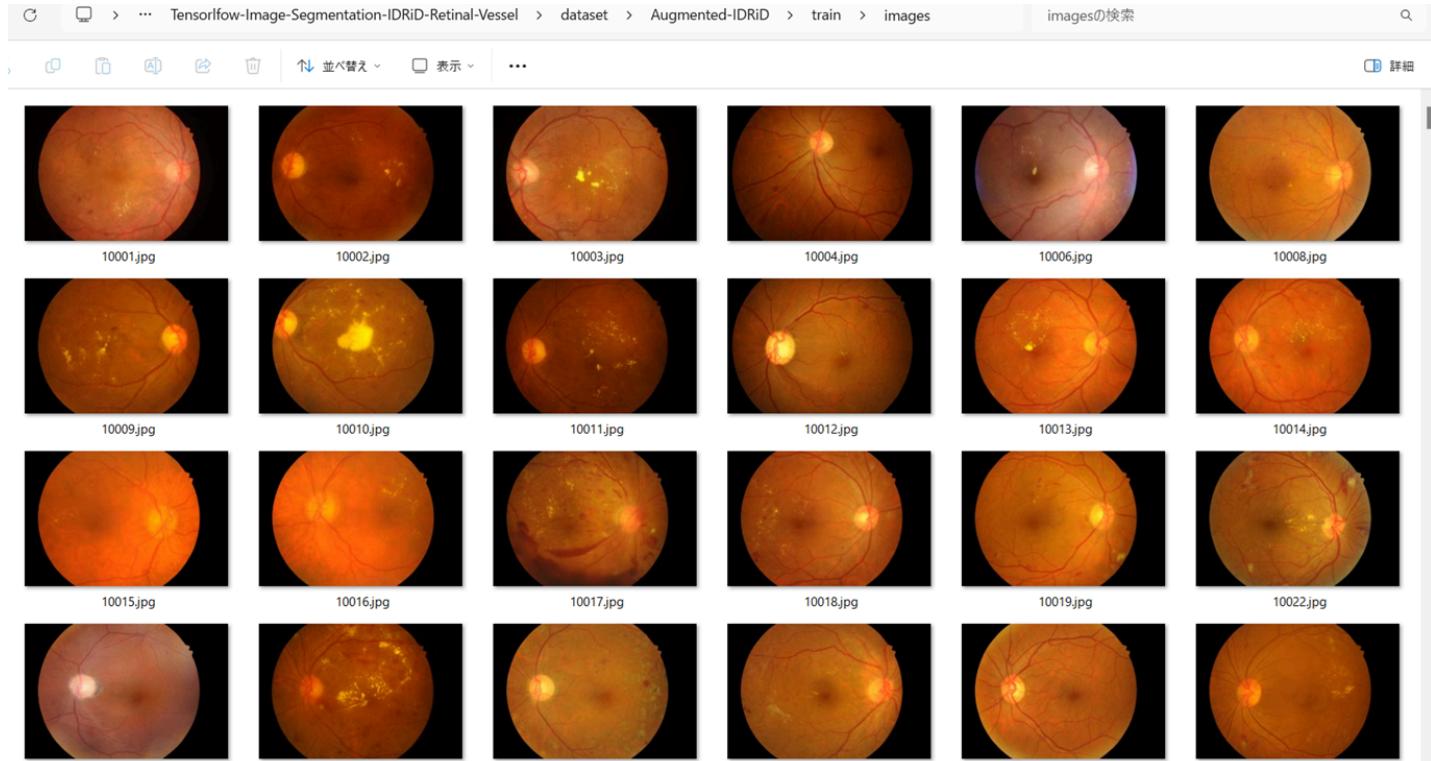
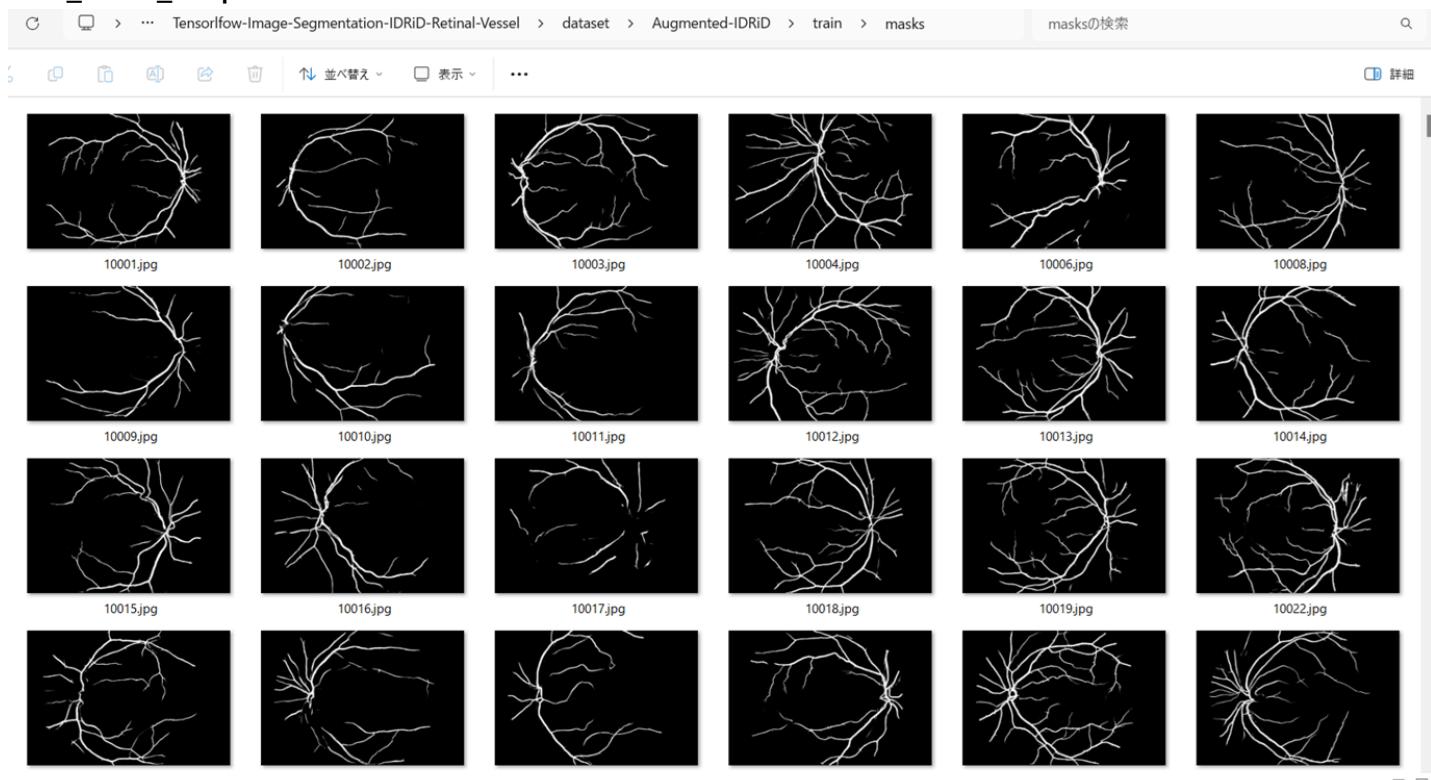


This is a Non-Tiled 4288x2848 pixels images and their corresponding masks dataset for IDRiD Retinal Vessel.

Augmented-IDRiD Statistics



As shown above, the number of images of train and valid datasets is not so large, but enough to use for a training set of our segmentation model.

Train_images_sample**Train_masks_sample****3 Train TensorflowUNet Model**

We have trained HRF TensorflowUNet Model by using the following [train_eval_infer.config](#) file.
Please move to ./projects/TensorflowSlightlyFlexibleUNet/IDRiD and run the following bat file.

```
>1.train.bat
```

, which simply runs the following command.

```
>python ../../src/TensorflowUNetTrainer.py ./train_eval_infer.config
```

Model parameters

Enabled Batch Normalization.

Defined a small **base_filters=16** and large **base_kernels=(13,13)** for the first Conv Layer of Encoder Block of [TensorflowUNet.py](#) and a large num_layers (including a bridge between Encoder and Decoder Blocks).

```
[model]
base_filters = 16
base_kernels = (13, 13)
num_layers = 8
dilation = (1, 1)
```

Learning rate

Defined a small learning rate.

```
[model]
learning_rate = 0.0001
```

Online augmentation

Disabled our online augmentation tool.

```
[model]
model = "TensorflowUNet"
generator = False
```

Loss and metrics functions

Specified "bce_dice_loss" and "dice_coef".

```
[model]
loss = "bce_dice_loss"
metrics = ["dice_coef"]
```

Learning rate reducer callback

Enabled learning_rate_reducer callback, and a small reducer_patience.

```
[train]
learning_rate_reducer = True
reducer_factor = 0.4
reducer_patience = 4
```

Dataset class

Specified ImageMaskDataset class.

```
[dataset]
datasetclass = "ImageMaskDataset"
resize_interpolation = "cv2.INTER_LINEAR"
```

Early stopping callback

Enabled early stopping callback with patience parameter.

```
[train]
patience = 10
```

Color space conversion

Used cv2.COLOR_BGR2Luv color converter.

```
[image]
color_converter = "cv2.COLOR_BGR2Luv"
```

Inference

Used the original IDRiD IMAGES as a mini_test dataset for our inference images.

```
[tiledinfer]
images_dir = "./mini_test/images"
output_dir = "./mini_test_output_tiled"
```

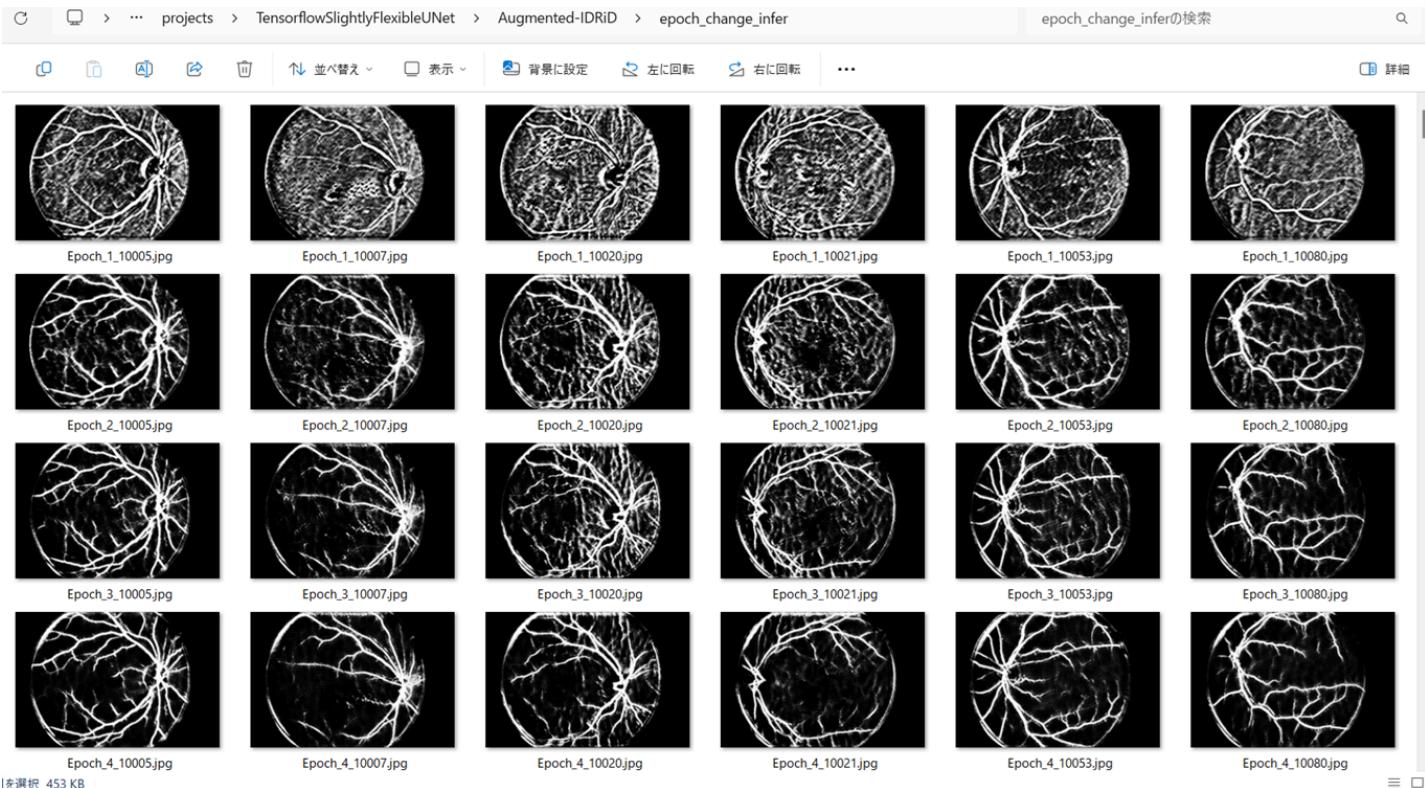
Epoch change inference callbacks

Enabled epoch_change_infer callback.

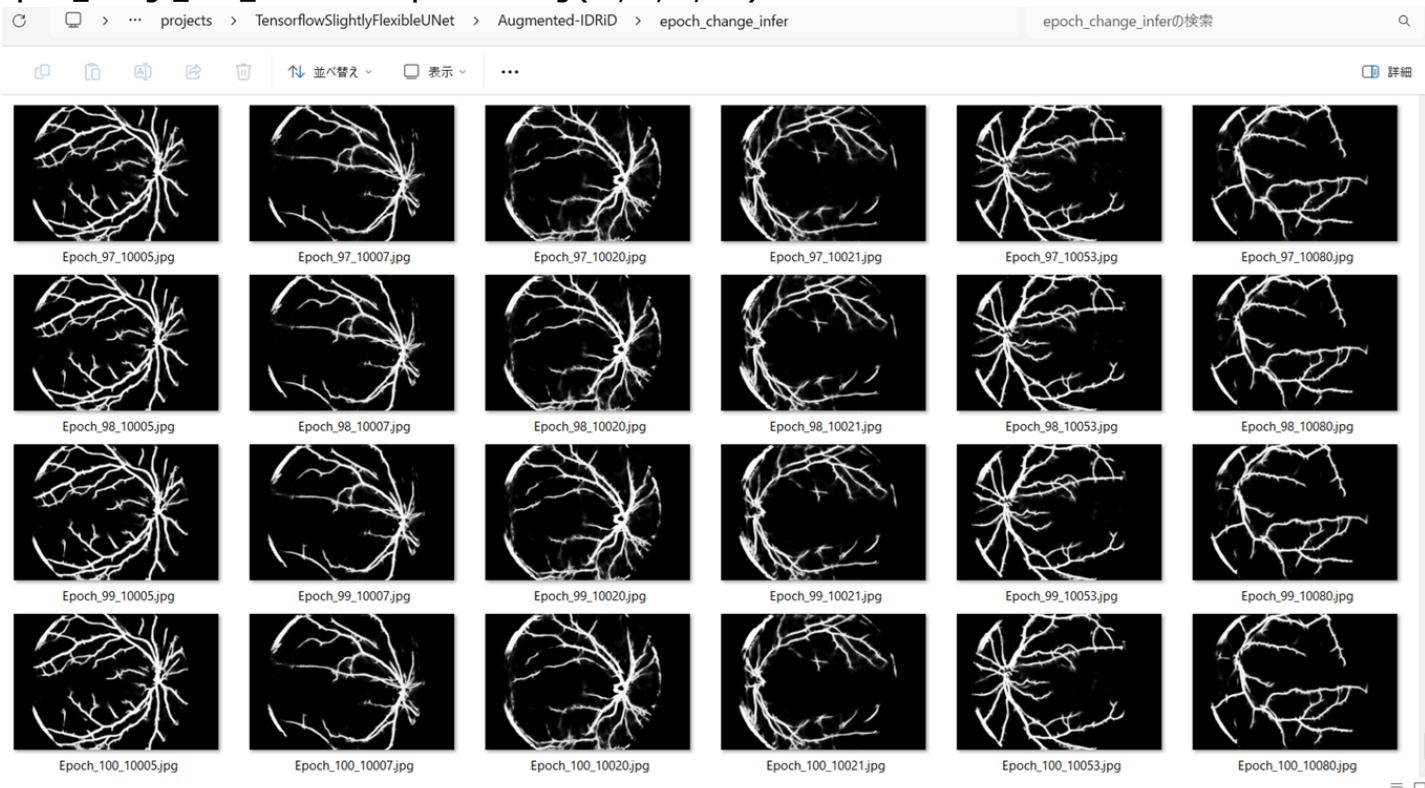
```
[train]
epoch_change_infer = True
epoch_change_infer_dir = "./epoch_change_infer"
epoch_change_tiledinfer = False
epoch_change_tiledinfer_dir = "./epoch_change_tiledinfer"
num_infer_images = 6
```

By using this callback, on every epoch_change, the epoch change tiled inference procedure can be called for 6 images in **mini_test** folder. This will help you confirm how the predicted mask changes at each epoch during your training process.

Epoch_change_tiled_inference output at starting (1,2,3,4)



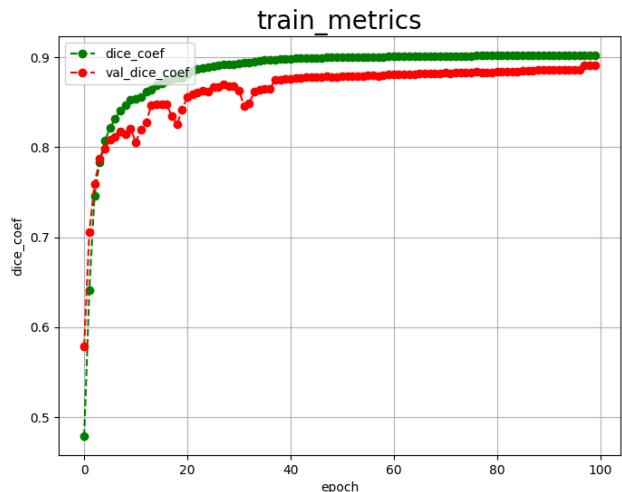
Epoch_change_tiled_inference output at ending (97,98,99,100)



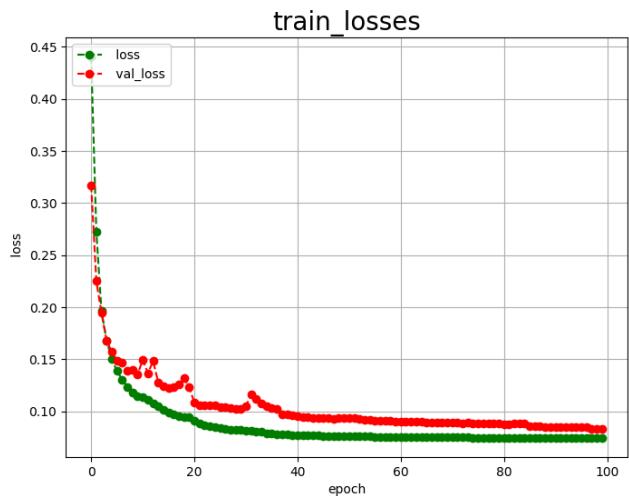
In this experiment, the training process was terminated at epoch 100.

```
PowerShell 7 (x64) x + - x
Epoch 90/100
907/907 [=====] - ETA: 0s - loss: 0.0747 - dice_coef: 0.9020
Epoch 90: val loss improved from 0.08534 to 0.08518, saving model to ./models/best_model.h5
907/907 [=====] - 331s 365ms/sample - loss: 0.0747 - dice_coef: 0.9020 - val_loss: 0.0852 - val_dice_coef: 0.8863 - lr: 1.0240e-06
907/907 [=====] - ETA: 0s - loss: 0.0747 - dice_coef: 0.9020
907/907 [=====] - ETA: 0s - loss: 0.0747 - dice_coef: 0.9020
907/907 [=====] - 331s 365ms/sample - loss: 0.0747 - dice_coef: 0.9020 - val_loss: 0.0851 - val_dice_coef: 0.8864 - lr: 1.0240e-06
Epoch 92/100
907/907 [=====] - ETA: 0s - loss: 0.0747 - dice_coef: 0.9020
907/907 [=====] - ETA: 0s - loss: 0.0747 - dice_coef: 0.9020
907/907 [=====] - 331s 365ms/sample - loss: 0.0747 - dice_coef: 0.9020 - val_loss: 0.0852 - val_dice_coef: 0.8862 - lr: 1.0240e-06
Epoch 93/100
907/907 [=====] - ETA: 0s - loss: 0.0746 - dice_coef: 0.9021
907/907 [=====] - ETA: 0s - loss: 0.0746 - dice_coef: 0.9021
907/907 [=====] - 331s 365ms/sample - loss: 0.0747 - dice_coef: 0.9021 - val_loss: 0.0851 - val_dice_coef: 0.8864 - lr: 1.0240e-06
Epoch 94/100
907/907 [=====] - ETA: 0s - loss: 0.0746 - dice_coef: 0.9021
907/907 [=====] - ETA: 0s - loss: 0.0746 - dice_coef: 0.9021
907/907 [=====] - 329s 363ms/sample - loss: 0.0746 - dice_coef: 0.9021 - val_loss: 0.0852 - val_dice_coef: 0.8862 - lr: 1.0240e-06
Epoch 95/100
907/907 [=====] - ETA: 0s - loss: 0.0746 - dice_coef: 0.9021
907/907 [=====] - ETA: 0s - loss: 0.0746 - dice_coef: 0.9021
907/907 [=====] - 328s 360ms/sample - loss: 0.0746 - dice_coef: 0.9021 - val_loss: 0.0852 - val_dice_coef: 0.8863 - lr: 1.0240e-06
907/907 [=====] - ETA: 0s - loss: 0.0746 - dice_coef: 0.9022
907/907 [=====] - ETA: 0s - loss: 0.0746 - dice_coef: 0.9022
907/907 [=====] - 325s 359ms/sample - loss: 0.0746 - dice_coef: 0.9022 - val_loss: 0.0852 - val_dice_coef: 0.8862 - lr: 1.0240e-06
Epoch 97/100
907/907 [=====] - ETA: 0s - loss: 0.0746 - dice_coef: 0.9022
907/907 [=====] - ETA: 0s - loss: 0.0746 - dice_coef: 0.9022
907/907 [=====] - 325s 359ms/sample - loss: 0.0746 - dice_coef: 0.9022 - val_loss: 0.0850 - val_dice_coef: 0.8865 - lr: 1.0240e-06
907/907 [=====] - ETA: 0s - loss: 0.0748 - dice_coef: 0.9018
907/907 [=====] - ETA: 0s - loss: 0.0748 - dice_coef: 0.9018
907/907 [=====] - 329s 362ms/sample - loss: 0.0748 - dice_coef: 0.9018 - val_loss: 0.0837 - val_dice_coef: 0.8911 - lr: 4.0960e-07
Epoch 99/100
907/907 [=====] - ETA: 0s - loss: 0.0747 - dice_coef: 0.9020
907/907 [=====] - ETA: 0s - loss: 0.0747 - dice_coef: 0.9020
907/907 [=====] - 329s 363ms/sample - loss: 0.0747 - dice_coef: 0.9020 - val_loss: 0.0836 - val_dice_coef: 0.8913 - lr: 4.0960e-07
Epoch 100/100
907/907 [=====] - ETA: 0s - loss: 0.0746 - dice_coef: 0.9021
907/907 [=====] - ETA: 0s - loss: 0.0746 - dice_coef: 0.9021
907/907 [=====] - 325s 358ms/sample - loss: 0.0746 - dice_coef: 0.9021 - val_loss: 0.0837 - val_dice_coef: 0.8912 - lr: 4.0960e-07
== Save history.json
```

[train_metrics.csv](#)



[train_losses.csv](#)



4 Evaluation

Please move to a **./projects/TensorflowSlightlyFlexibleUNet/IDRiD** folder, and run the following bat file to evaluate TensorflowUNet model for IDRiD/test.

`./2.evaluate.bat`

This bat file simply runs the following command.

```
python ../../src/TensorflowUNetEvaluator.py ./train_eval_infer.config
```

Evaluation console output:

```
== WARNING: Not found [train] best_model_file, return default value best_model.h5
== Loaded weight file, /model/best_model.h5
== Dataset class <class 'ImageMaskDataset.ImageMaskDataset'>
== BaseImageMaskDataset constructor
== ConfigParser ./train_eval_infer.config
== WARNING: Not found [mask] algorithm, return default value None
== WARNING: Not found [dataset] image_format, return default value rgb
== WARNING: Not found [dataset] input_normalize, return default value True
== WARNING: Not found [dataset] debug, return default value True
== WARNING: Not found [dataset] rgb_mask, return default value False
== WARNING: Not found [dataset] color_order, return default value bgr
== contrast adjuster False
== WARNING: Not found [image] contrast_alpha, return default value 1.5
== WARNING: Not found [image] contrast_best, return default value 40
== WARNING: Not found [dataset] mask_format, return default value gray
== WARNING: Not found [mask] binarize, return default value False
== WARNING: Not found [mask] grayscaling, return default value True
== WARNING: Not found [dataset] image_normalize, return default value False
== WARNING: Not found [dataset] debug, return default value False
== WARNING: Not found [mask] mask_colors, return default value None
mask colors None
num classes 1
image normalize False
binarize algorithm None
ImageMaskDataset constructor
self.resize_interpolation_1
WARNING: Not found [model] evaluation, return default value test
BaseImageMaskDataset.create_dataset test
E:\py310-efficientdet\lib\site-packages\keras\engine\training_v1.py:2332: UserWarning: `Model.state_updates` will be removed in a future version. This property should not be used in TensorFlow 2.0, as `updates` are applied automatically.
    updates = self.state_updates
Test loss : 0.08851
Test accuracy:0.8887
Evaluation metric:loss score:0.08851
Evaluation metric:dice_coef score:0.8887
Saved ./evaluation.csv
```

[evaluation.csv](#)

The loss (bce_dice_loss) to this IDRiD/test was low, and dice_coef high as shown below.

```
loss, 0.0851
dice_coef, 0.8887
```

5 Inference

Please move to a **./projects/TensorflowSlightlyFlexibleUNet/IDRiD** folder

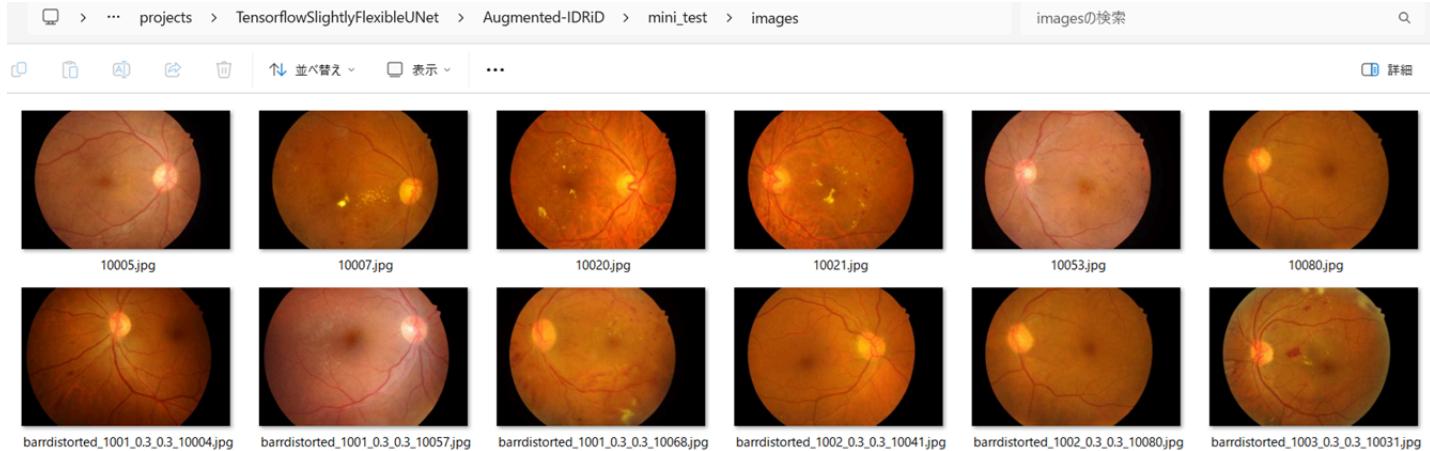
,and run the following bat file to infer segmentation regions for images by the Trained-TensorflowUNet model for IDRiD.

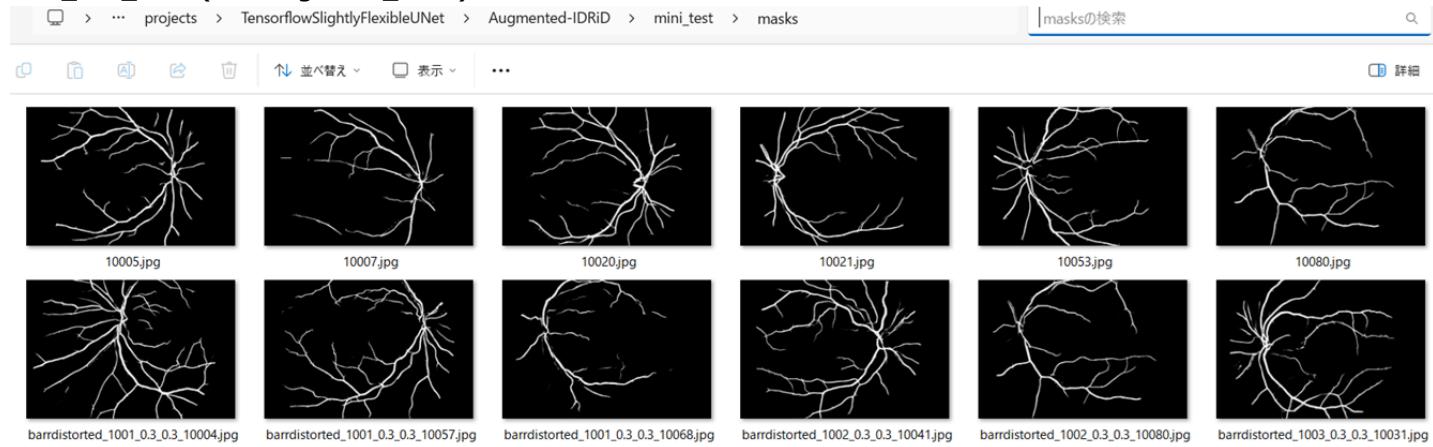
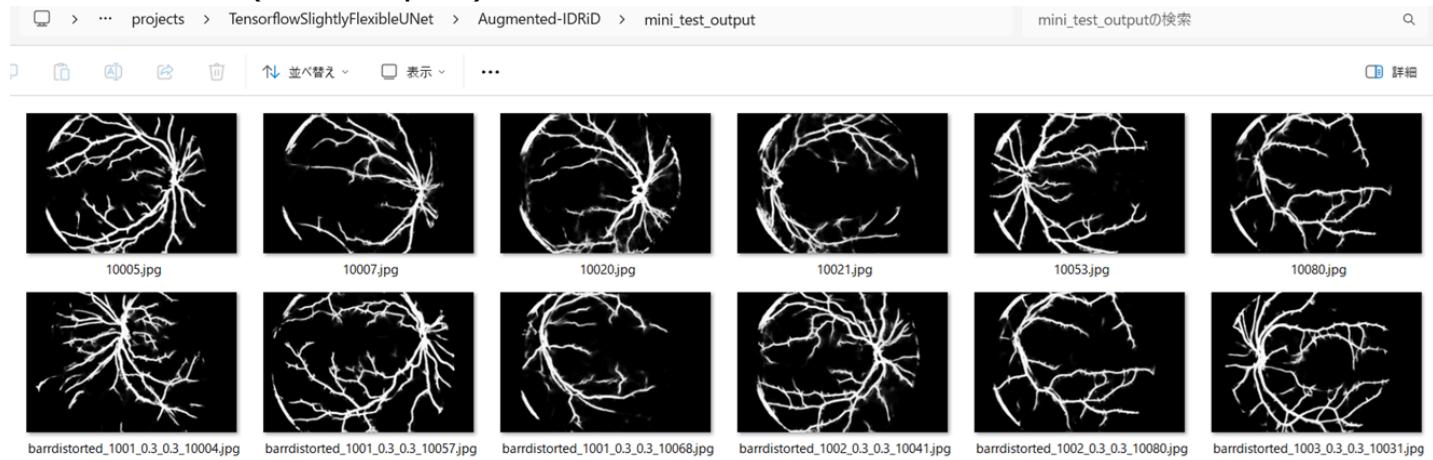
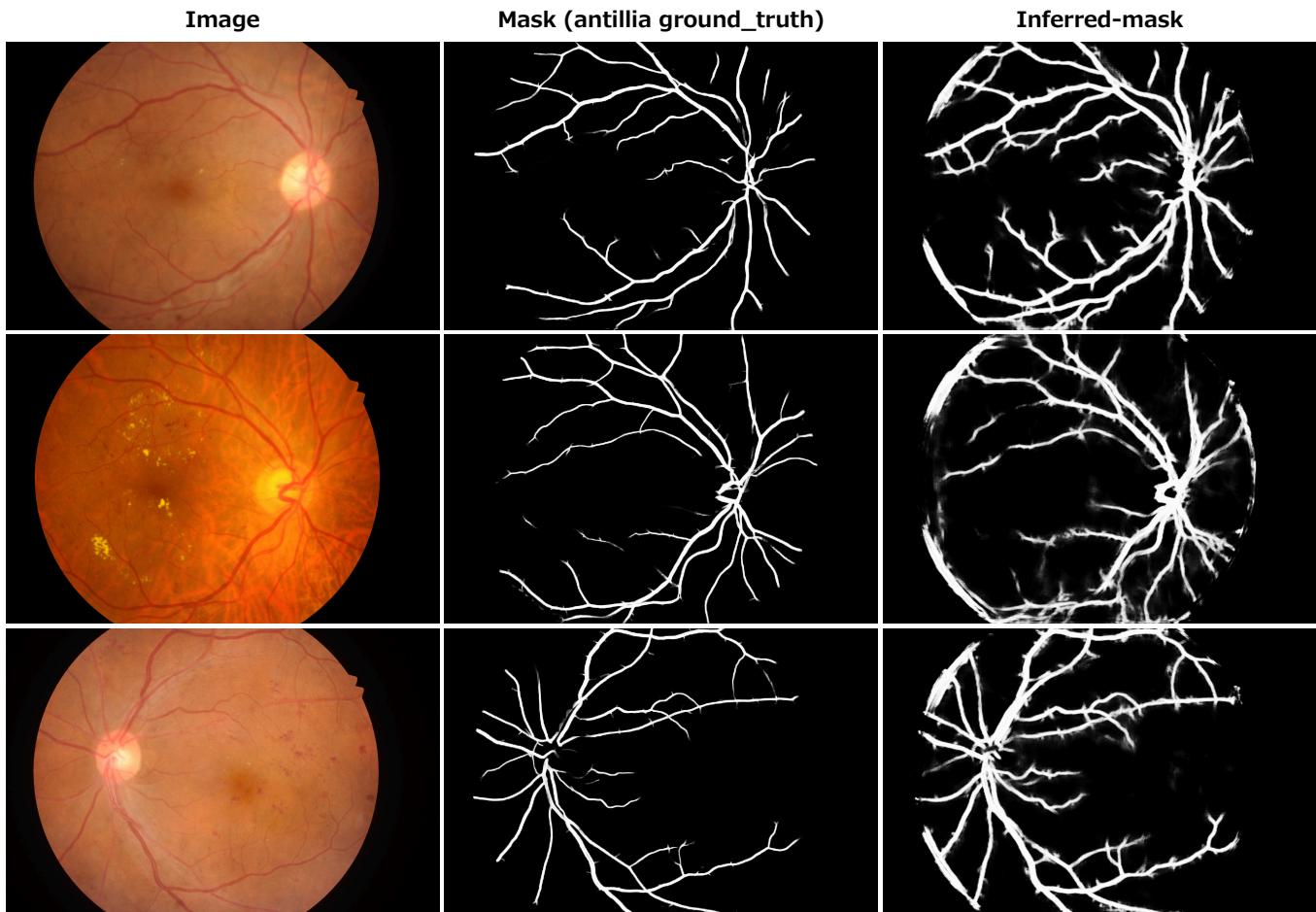
```
./3.infer.bat
```

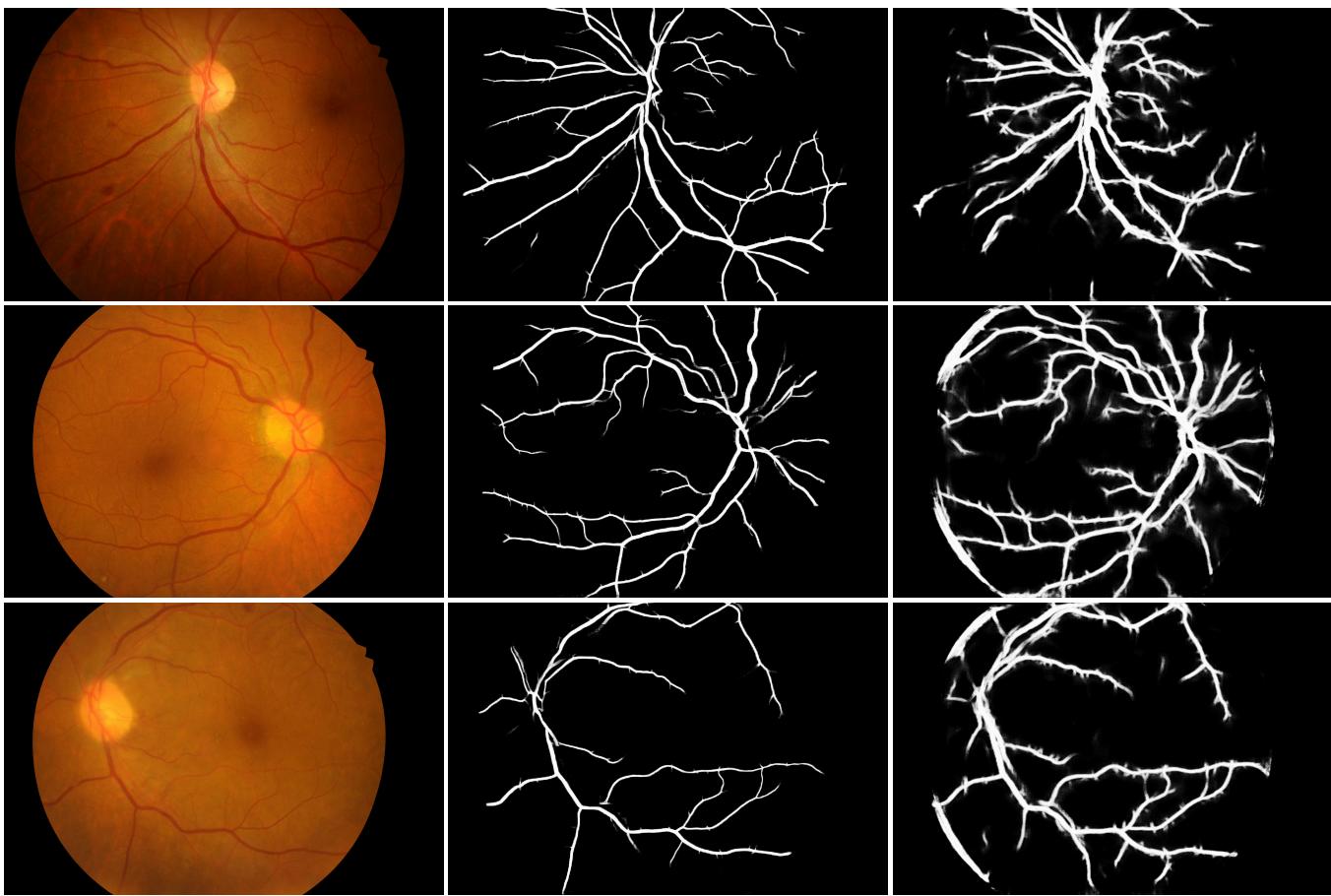
This simply runs the following command.

```
python ../../src/TensorflowUNetInferencer.py ./train_eval_infer.config
```

mini_test_images (4288x2848 pixels)



mini_test_mask(antillia ground_truth)**Inferred test masks (4288x2848 pixels)****Enlarged images and masks of 4288x2848 pixels**



References

1. Locating Blood Vessels in Retinal Images by Piecewise Threshold Probing of a Matched Filter Response

Adam Hoover, Valentina Kouznetsova, and Michael Goldbaum

<https://www.uhu.es/retinopathy/General/000301IEEETransMedImag.pdf>

2. High-Resolution Fundus (HRF) Image Database

Budai, Attila; Bock, Rüdiger; Maier, Andreas; Hornegger, Joachim; Michelson, Georg.

<https://www5.cs.fau.de/research/data/fundus-images/>.

3. Robust Vessel Segmentation in Fundus Images

Budai, Attila; Bock, Rüdiger; Maier, Andreas; Hornegger, Joachim; Michelson, Georg.

<https://onlinelibrary.wiley.com/doi/10.1155/2013/154860>

4. State-of-the-art retinal vessel segmentation with minimalistic models

Adrian Galdran, André Anjos, José Dolz, Hadi Chakor, Hervé Lombaert & Ismail Ben Ayed

<https://www.nature.com/articles/s41598-022-09675-y>

5. Retinal blood vessel segmentation using a deep learning method based on modified U-NET model

Sanjeeewani, Arun Kumar Yadav, Mohd Akbar, Mohit Kumar, Divakar Yadav

<https://www.semanticscholar.org/reader/f5cb3b1c69a2a7e97d1935be9d706017af8cc1a3>

6. Tensorflow-Tiled-Image-Segmentation-Pre-Augmented-STARE-Retinal-Vessel

Toshiyuki Arai @antillia.com

<https://github.com/sarah-antillia/Tensorflow-Tiled-Image-Segmentation-Pre-Augmented-STARE-Retinal-Vessel>

7. Tensorflow-Tiled-Image-Segmentation-Pre-Augmented-HRF-Retinal-Vessel

Toshiyuki Arai @antillia.com

<https://github.com/sarah-antillia/Tensorflow-Tiled-Image-Segmentation-Pre-Augmented-HRF-Retinal-Vessel>