

Tensorflow-Tiled-Image-Segmentation-Pre-Augmented-STARE-Retinal-Vessel (2025/02/02)

This is the first experiment of Tiled Image Segmentation for **STARE Retinal Vessel** based on the latest [Tensorflow-Image-Segmentation-API](#), and a **pre-augmented tiled dataset** [Augmented-Tiled-STARE-ImageMask-Dataset.zip](#), which was derived by us from the following images and labels:

[Twenty images used for experiments](#)

[Hand labeled vessel network provided by Adam Hoover](#)

On detail of **STARE(Structured Analysis of the Retina)**, please refer to the official site:

[STructured Analysis of the Retina](#), and github repository [STARE](#)

Please see also our experiment [Tensorflow-Image-Segmentation-Retinal-Vessel](#) based on [CHASE_DB1 dataset](#).

Experiment Strategies

As demonstrated in our experiments [Tensorflow-Tiled-Image-Segmentation-IDRiD-HardExudates](#), and [Tensorflow-Tiled-Image-Segmentation-Augmented-Skin-Cancer](#), the Tiled Image Segmentation based on a simple UNet model trained by a tiledly-split images and masks dataset, is an effective method for the large image segmentation over 4K pixels.

It is difficult to precisely segment Retinal Blood Vessels in small images using a simple UNet model because these vessels are typically very thin and difficult to detect. Therefore, we generate a high-resolution retinal image dataset by upscaling the original images and use it to train the UNet model to improve segmentation performance.

In this experiment, we employed the following strategies to apply the Tiled Image Segmentation method to STARE Retinal Vessel.

1. Enlarged Dataset

We generated a 5x enlarged dataset of 19 JPG images and masks, each with 3500x3025 pixels, from the original STARE 700x605 pixels PPM.GZ image and label files using bicubic interpolation.

2. Pre Augmented Tiled STARE ImageMask Dataset

We generated a pre-augmented image mask dataset from the enlarged dataset, which was tiledly-split to 512x512 pixels and reduced to 512x512 pixels image and mask dataset.

3. Train Segmentation Model

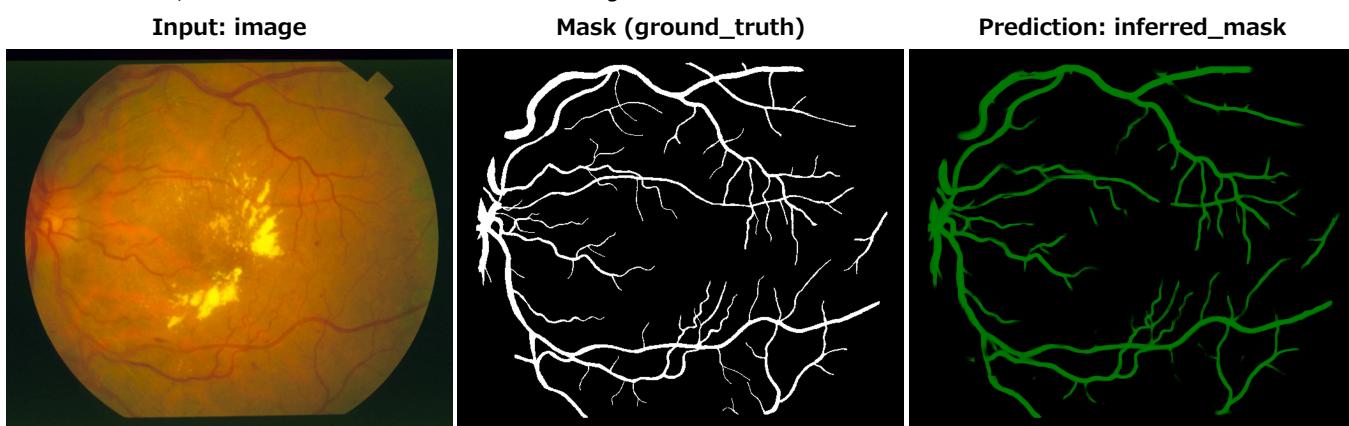
We trained and validated a TensorFlow UNet model by using the **Pre Augmented Tiled STARE ImageMask Dataset**

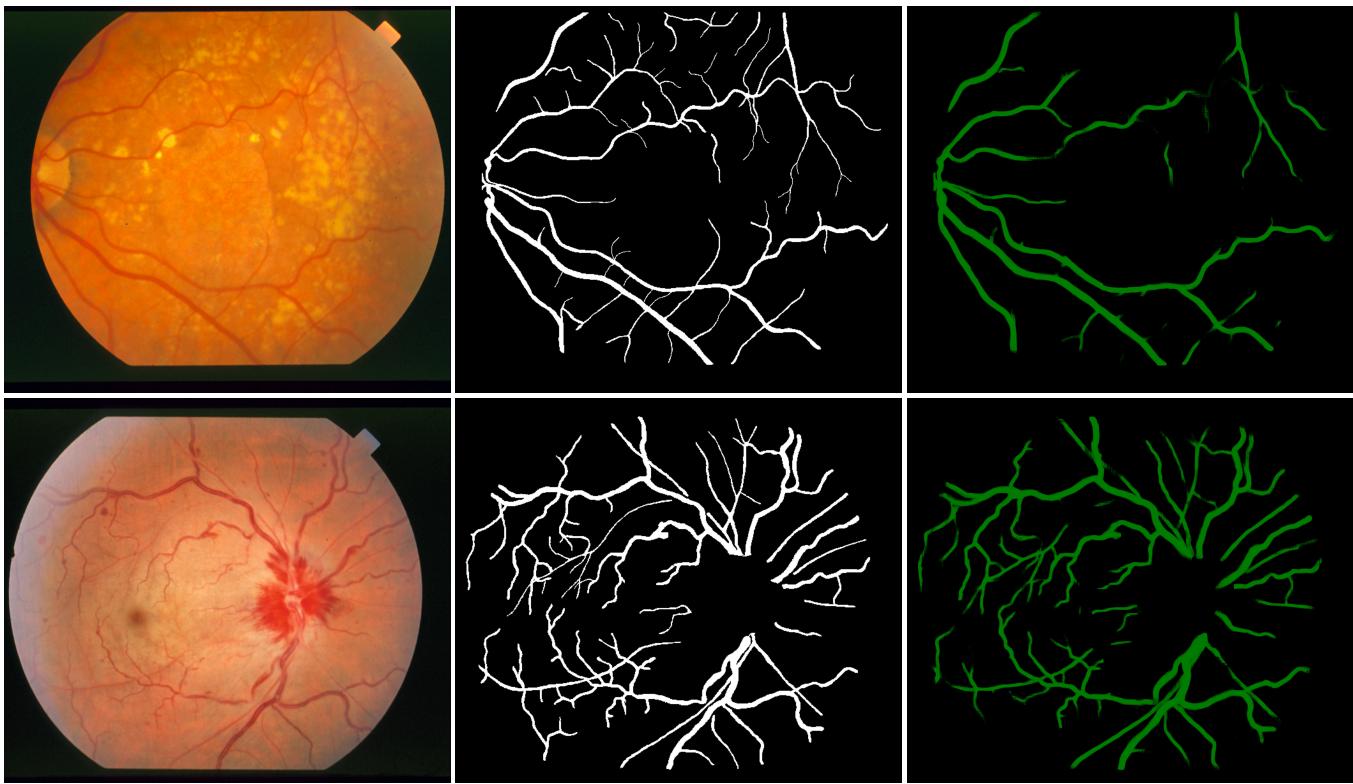
4. Tiled Image Segmentation

We applied our Tiled-Image Segmentation inference method to predict the STARE Retinal Vessel for the mini_test images with a resolution of 3500x3025 pixels of the Enlarged Dataset.

Actual Tiled Image Segmentation for Images of 3500x3025 pixels

As shown below, the inferred masks look similar to the ground truth masks.





In this experiment, we used the simple UNet Model [TensorflowSlightlyFlexibleUNet](#) for this STARESegmentation 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 images and labels in [STructured Analysis of the Retina](#):

[Twenty images used for experiments](#)

[Hand labeled vessel network provided by Adam Hoover](#)

Please see also [STARE](#)

Authors and Institutions

Adam Hoover (Department of Electrical and Computer Engineering, Clemson University)

Valentina Kouznetsova (Vision Computing Lab, Department of Electrical and Computer Engineering, University of California, San Diego, La Jolla)

Michael Goldbaum (Department of Ophthalmology, University of California, San Diego)

Citation

```
@ARTICLE{845178,
author={Hoover, A.D. and Kouznetsova, V. and Goldbaum, M.},
journal={IEEE Transactions on Medical Imaging},
title={Locating blood vessels in retinal images by piecewise threshold probing of a matched filter response},
year={2000},
volume={19},
number={3},
pages={203-210},
doi={10.1109/42.845178})}
```

2 Augmented-Tiled-STARE ImageMask Dataset

If you would like to train this STARE Segmentation model by yourself, please download the pre-augmented dataset from the google drive [Augmented-Tiled-STARE-ImageMask-Dataset.zip](#), expand the downloaded ImageMaskDataset and put it under **./dataset** folder to be

```
./dataset
└── Augmented-Tiled-STARE
    ├── test
    │   ├── images
    │   └── masks
    ├── train
    │   ├── images
    │   └── masks
    └── valid
        ├── images
        └── masks
```

This is a 512x512 pixels pre augmented tiles dataset generated from 3500x3025 pixels 19 **Enlarged-images** and their corresponding **Enlarged-masks**.

- We excluded all black (empty) masks and their corresponding images to generate our dataset from the original one. The folder structure of the original stare-images and labels-ah data is the following.

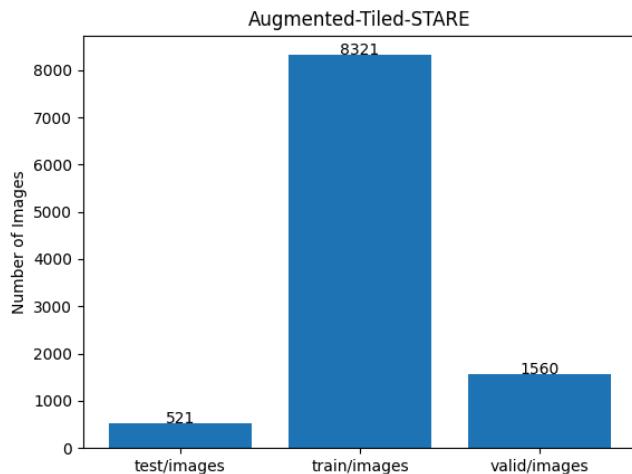
```
./STARE
└── stare-images
    ├── im0001.ppm.gz
    ├── im0002.ppm.gz
    └── im0319.ppm.gz
└── labels-ah
    ├── im0001.ah.ppm.gz
    ├── im0002.ah.ppm.gz
    └── im0319.ah.ppm.gz
```

We excluded im0324.ah.ppm.gz file in the original labels-ah folder, because no corresponding im0324.ppm.gz was in stare-images folder.

On the derivation of this tiled dataset, please refer to the following Python scripts.

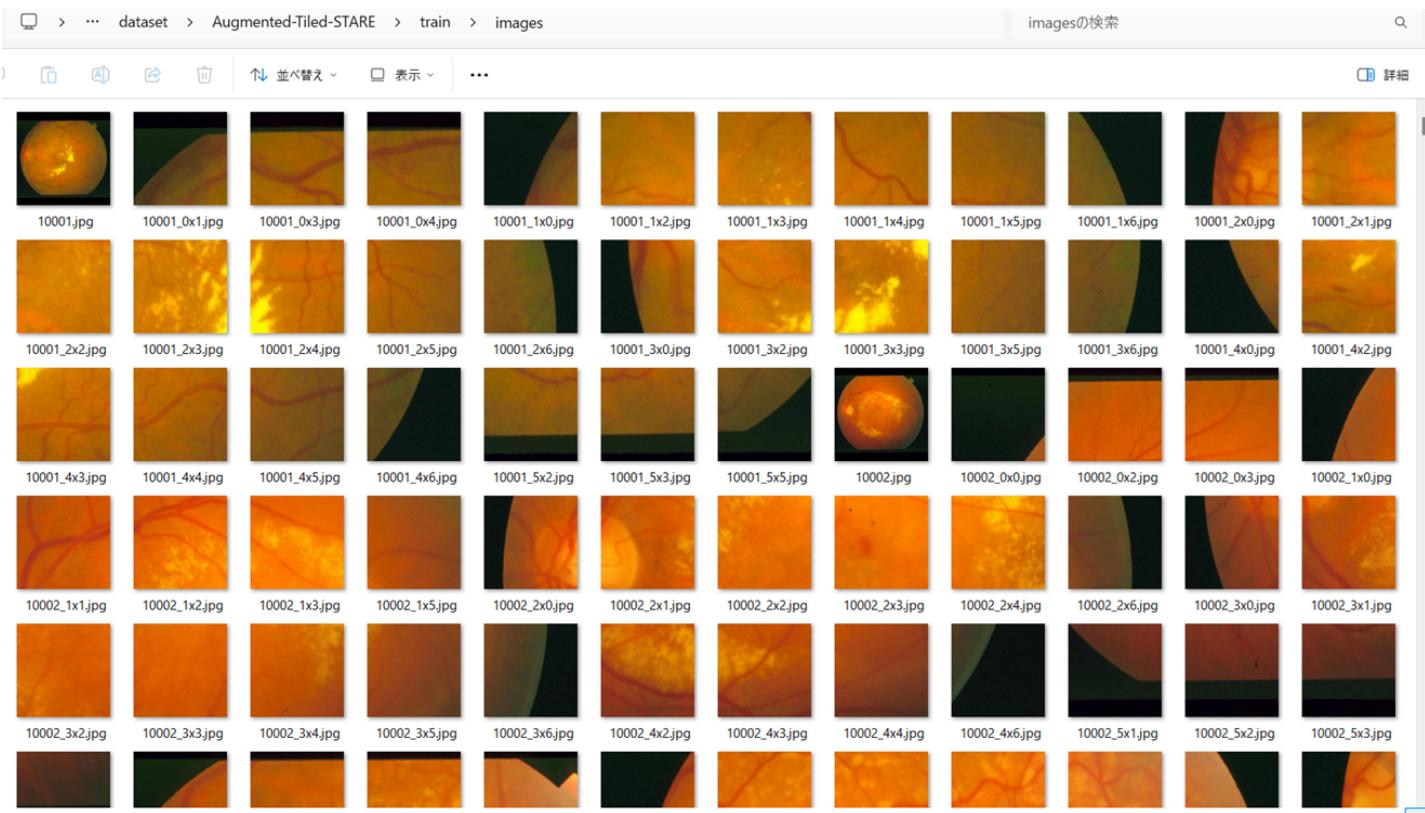
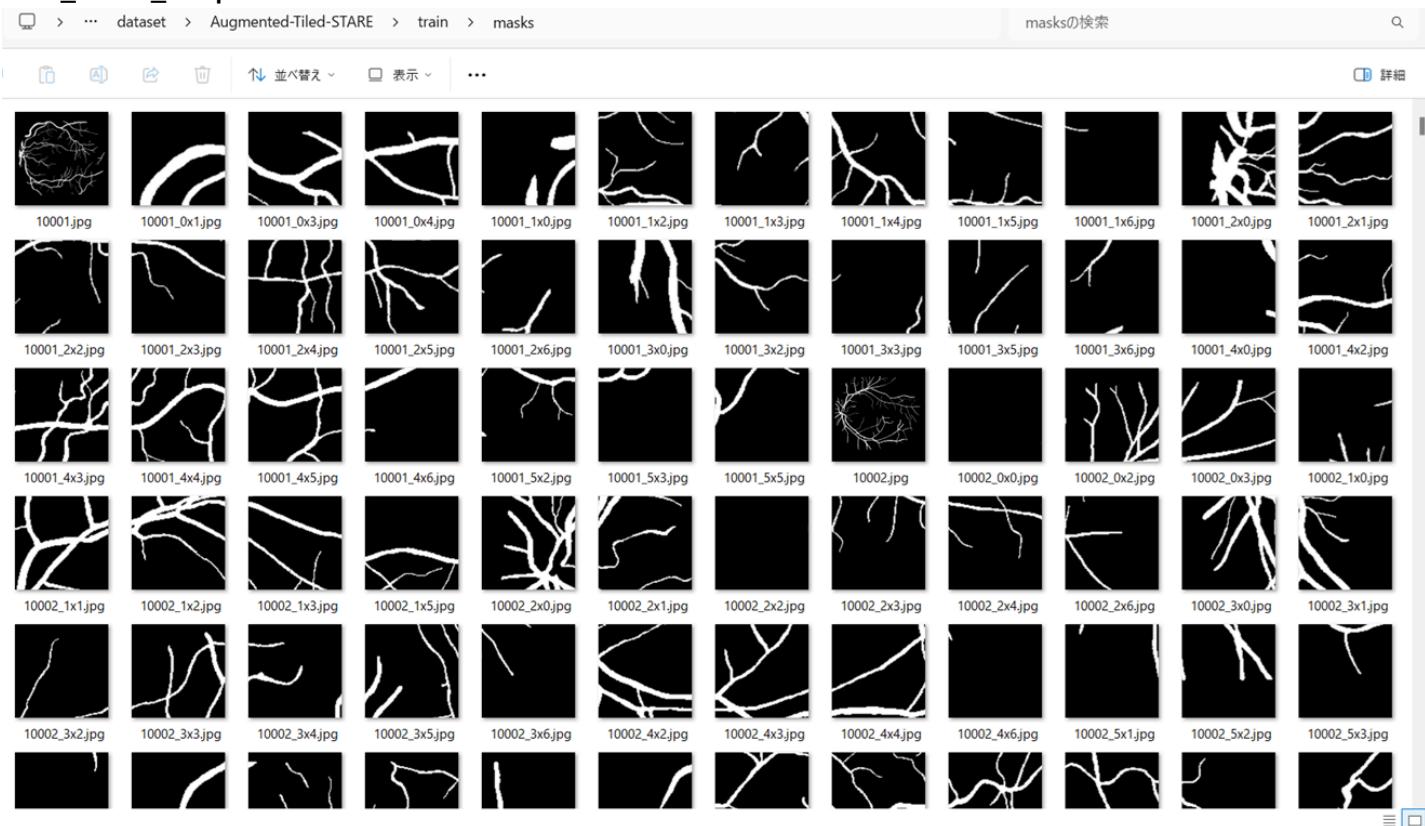
- [Preprocessor.py](#)
- [TiledImageMaskDatasetGenerator.py](#)
- [split_tiled_master.py](#)

Augmented-Tiled-STARE Statistics



As shown above, the number of images of train and valid datasets is enough to use for a training set of our segmentation model.

Train_images_sample

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

We have trained STARE TensorflowUNet Model by using the following [train_eval_infer.config](#) file.

Please move to `./projects/TensorflowSlightlyFlexibleUNet/Augmented-Tiled-STARE` 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=(9,9)** 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   = (9, 9)
num_layers     = 8
dilation       = (3, 3)
```

Learning rate

Defined a small learning rate.

```
[model]
learning_rate = 0.00007
```

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     = 20
```

Tiled inference

We used 3500x3025 pixels enlarged images and masks generated by [Preprocessor.py](#) as a mini_test dataset for our TiledInference.

```
[tiledinfer]
overlapping   = 64
images_dir    = "./mini_test/images"
output_dir    = "./mini_test_output_tiled"
```

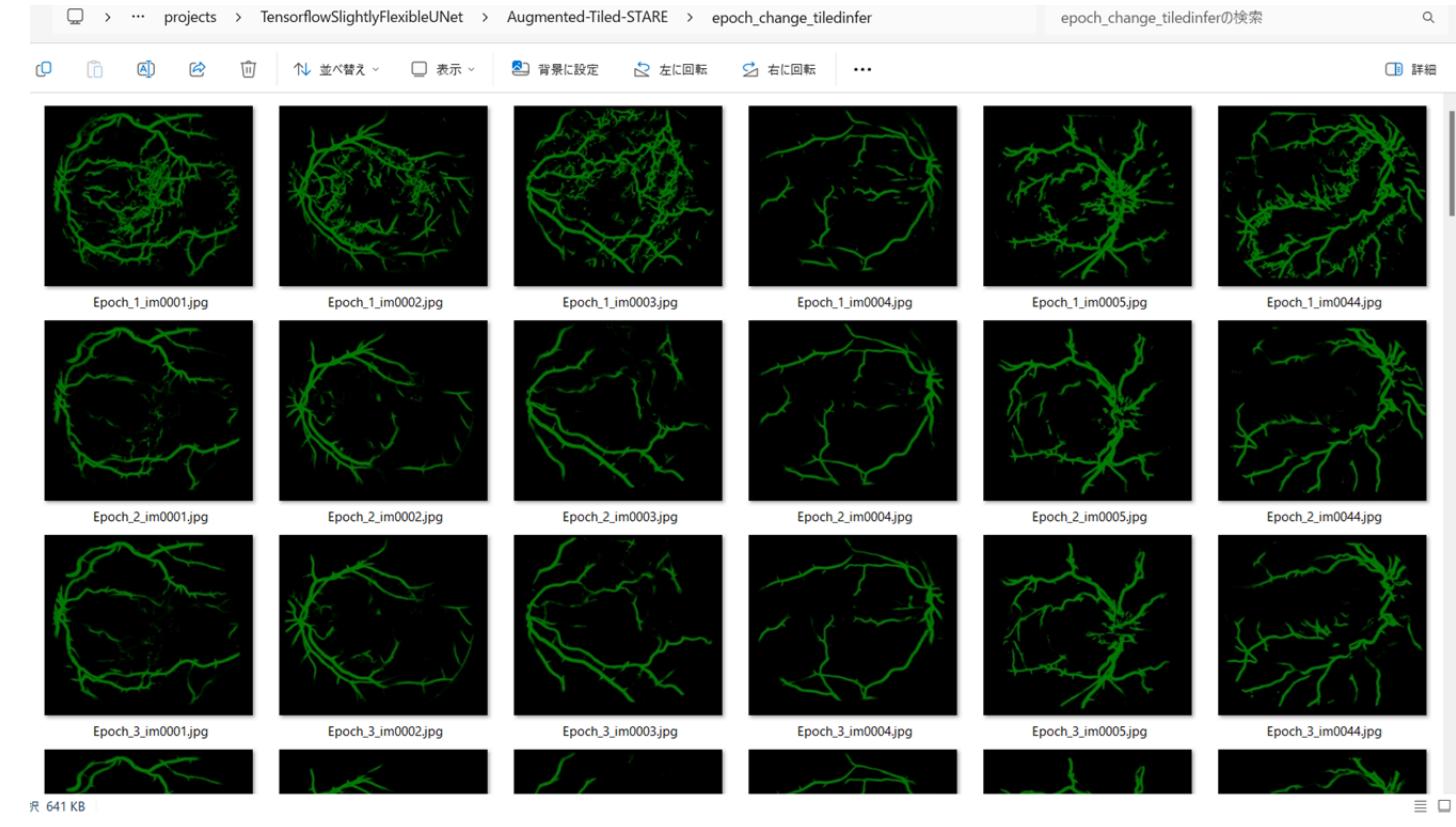
Epoch change inference callbacks

Enabled epoch_change_infer callback.

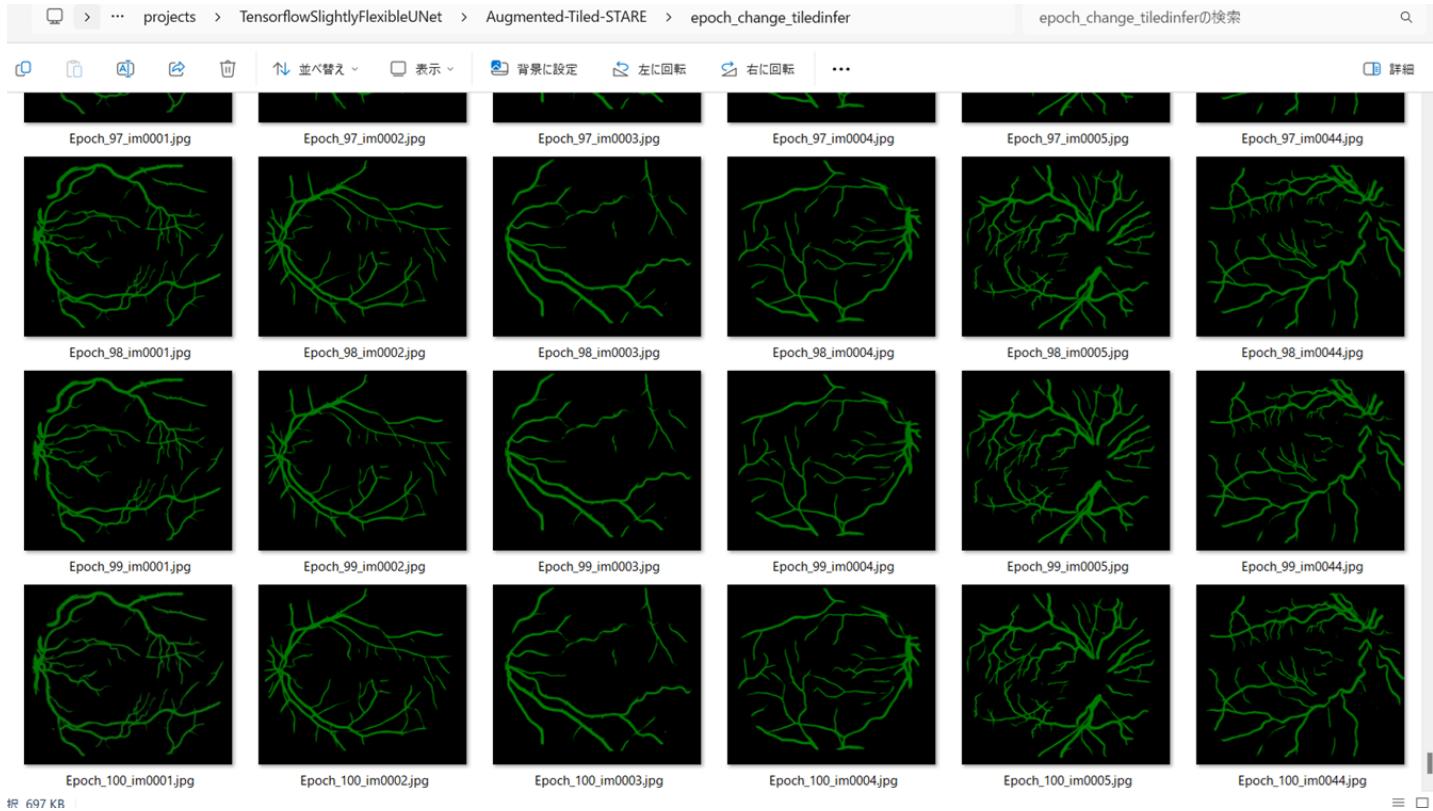
```
[train]
epoch_change_infer     = False
epoch_change_infer_dir = "./epoch_change_infer"
epoch_change_tiledinfer = True
epoch_change_tiledinfer_dir = "./epoch_change_tiledinfer"
num_infer_images       = 6
```

By using this callback, on every epoch_change, the epoch change tiledinfer 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_inference output at starting (1,2,3)



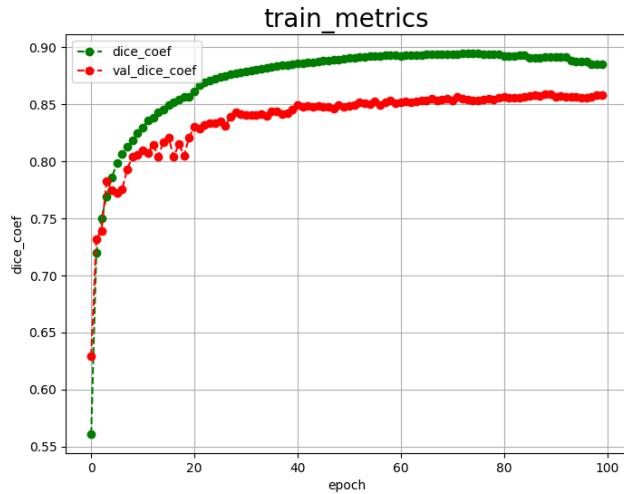
Epoch_change_inference output at ending (98,99,100)



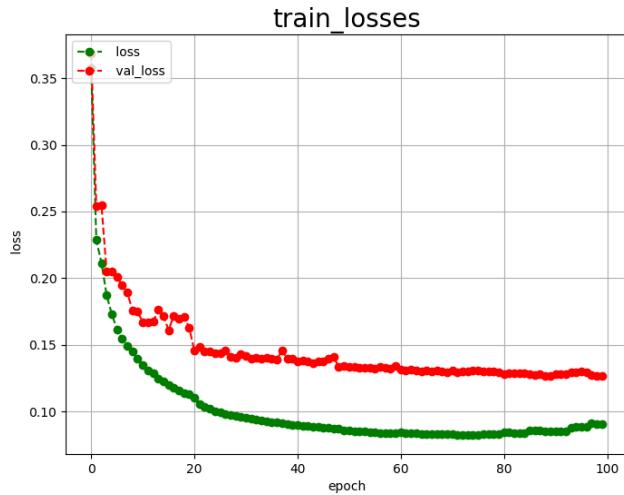
In this experiment, the training process was terminated at epoch 100.

```
PowerShell 7 (x64) + - x
Epoch 89: val_loss improved from 0.12723 to 0.12656, saving model to ./models/best_model.h5
Epoch 90/100
832/8321 [=====] - ETA: 0s - loss: 0.0853 - dice_coef: 0.8913 - val_loss: 0.1266 - val_dice_coef: 0.8593 - lr: 2.8672e-07
832/8321 [=====] - ETA: 0s - loss: 0.0853 - dice_coef: 0.8914
832/8321 [=====] - ETA: 0s - loss: 0.0853 - dice_coef: 0.8914
Epoch 91/100
832/8321 [=====] - ETA: 0s - loss: 0.0852 - dice_coef: 0.8914
Epoch 91: val_loss did not improve from 0.12656
832/8321 [=====] - ETA: 0s - loss: 0.0852 - dice_coef: 0.8914
832/8321 [=====] - ETA: 0s - loss: 0.0852 - dice_coef: 0.8914
832/8321 [=====] - ETA: 0s - loss: 0.0852 - dice_coef: 0.8914
Epoch 92/100
832/8321 [=====] - ETA: 0s - loss: 0.0852 - dice_coef: 0.8914
Epoch 92: val_loss did not improve from 0.12656
832/8321 [=====] - ETA: 0s - loss: 0.0852 - dice_coef: 0.8914
832/8321 [=====] - ETA: 0s - loss: 0.0852 - dice_coef: 0.8914
832/8321 [=====] - ETA: 0s - loss: 0.0852 - dice_coef: 0.8914
Epoch 93/100
832/8321 [=====] - ETA: 0s - loss: 0.0851 - dice_coef: 0.8916
Epoch 93: val_loss did not improve from 0.12656
832/8321 [=====] - ETA: 0s - loss: 0.0851 - dice_coef: 0.8916
832/8321 [=====] - ETA: 0s - loss: 0.0851 - dice_coef: 0.8916
832/8321 [=====] - ETA: 0s - loss: 0.0851 - dice_coef: 0.8916
Epoch 94/100
832/8321 [=====] - ETA: 0s - loss: 0.0851 - dice_coef: 0.8916
Epoch 94: val_loss did not improve from 0.12656
832/8321 [=====] - ETA: 0s - loss: 0.0851 - dice_coef: 0.8916
832/8321 [=====] - ETA: 0s - loss: 0.0851 - dice_coef: 0.8916
832/8321 [=====] - ETA: 0s - loss: 0.0851 - dice_coef: 0.8916
Epoch 95/100
832/8321 [=====] - ETA: 0s - loss: 0.0851 - dice_coef: 0.8916
Epoch 95: val_loss did not improve from 0.12656
832/8321 [=====] - ETA: 0s - loss: 0.0851 - dice_coef: 0.8916
832/8321 [=====] - ETA: 0s - loss: 0.0851 - dice_coef: 0.8916
832/8321 [=====] - ETA: 0s - loss: 0.0851 - dice_coef: 0.8916
Epoch 96/100
832/8321 [=====] - ETA: 0s - loss: 0.0882 - dice_coef: 0.8880
Epoch 96: val_loss did not improve from 0.12656
832/8321 [=====] - ETA: 0s - loss: 0.0882 - dice_coef: 0.8880
832/8321 [=====] - ETA: 0s - loss: 0.0882 - dice_coef: 0.8880
832/8321 [=====] - ETA: 0s - loss: 0.0882 - dice_coef: 0.8880
Epoch 97/100
832/8321 [=====] - ETA: 0s - loss: 0.0882 - dice_coef: 0.8880
Epoch 97: val_loss did not improve from 0.12656
832/8321 [=====] - ETA: 0s - loss: 0.0882 - dice_coef: 0.8880
832/8321 [=====] - ETA: 0s - loss: 0.0882 - dice_coef: 0.8880
832/8321 [=====] - ETA: 0s - loss: 0.0882 - dice_coef: 0.8880
Epoch 98/100
832/8321 [=====] - ETA: 0s - loss: 0.0912 - dice_coef: 0.8849
Epoch 98: val_loss did not improve from 0.12656
832/8321 [=====] - ETA: 0s - loss: 0.0912 - dice_coef: 0.8849
832/8321 [=====] - ETA: 0s - loss: 0.0912 - dice_coef: 0.8849
832/8321 [=====] - ETA: 0s - loss: 0.0912 - dice_coef: 0.8849
Epoch 99/100
832/8321 [=====] - ETA: 0s - loss: 0.0907 - dice_coef: 0.8853
Epoch 99: val_loss did not improve from 0.12656
832/8321 [=====] - ETA: 0s - loss: 0.0907 - dice_coef: 0.8853
832/8321 [=====] - ETA: 0s - loss: 0.0907 - dice_coef: 0.8853
832/8321 [=====] - ETA: 0s - loss: 0.0907 - dice_coef: 0.8853
Epoch 100/100
832/8321 [=====] - ETA: 0s - loss: 0.0905 - dice_coef: 0.8854
Epoch 100: val_loss did not improve from 0.12656
832/8321 [=====] - ETA: 0s - loss: 0.0905 - dice_coef: 0.8854
832/8321 [=====] - ETA: 0s - loss: 0.0905 - dice_coef: 0.8854
832/8321 [=====] - ETA: 0s - loss: 0.0905 - dice_coef: 0.8854
Save history.json
```

[train_metrics.csv](#)



[train_losses.csv](#)



4 Evaluation

Please move to a **./projects/TensorflowSlightlyFlexibleUNet/Augmented-Tiled-STARE** folder, and run the following bat file to evaluate TensorflowUNet model for STARE.

`./evaluate.bat`

This bat file simply runs the following command.

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

Evaluation console output:

```
PowerShell 7 (x64) + - x
==== ConfigParser ./train_eval_infer.config
==== WARNING: Not found [train] best_model_file, return default value best_model.h5
==== Loaded a weight file ./models/best_model.h5
==== DatasetClass <class '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] rgbd_mask, return default value False
==== WARNING: Not found [dataset] color_order, return default value bgr
-- contrast adjuster False
==== WARNING: Not found [image] contrast alphas, 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] grayscale, 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
-- Not found [model] evaluation, return default value test
BaseImageMaskDataset.create dataset test
create ./../../dataset/Augmented-Tiled-STARE/test/images/ ../../dataset/Augmented-Tiled-STARE/test/masks/
==== WARNING: Not found [mask] mask_channels, return default value 1
num classes 1 image data type <class 'numpy.uint8'>
num images 521 512 512
100% | 521/521 [00:04<00:00, 118.58it/s]
X: shape (521, 512, 512, 3) type uint8
-- Create X: len(521) Y: len(512)
==== WARNING: Not found [eval] batch_size, return default value 4
==== evaluate batch size 4
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 :0.1236
Test accuracy:0.8544
Evaluation metric:bce_loss score:0.1236
Evaluation metric:dice_coef score:0.8544
-- Saved ./evaluation.csv
```

Image-Segmentation-STARE [evaluation.csv](#)

The loss (bce_dice_loss) to this Augmented-Tiled-STARE/test was low, and dice_coef high as shown below.

```
loss,0.1236
dice_coef,0.8544
```

5 Tiled inference

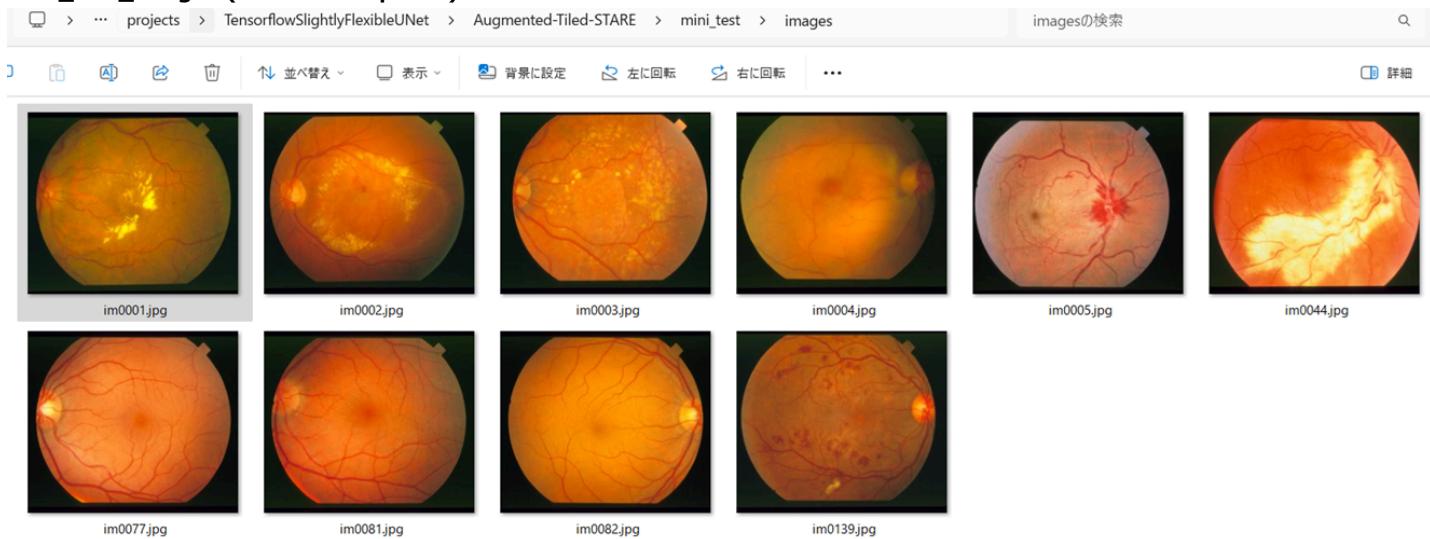
Please move to a **./projects/TensorflowSlightlyFlexibleUNet/Augmented-Tiled-STARE** folder ,and run the following bat file to infer segmentation regions for images by the Trained-TensorflowUNet model for STARE.

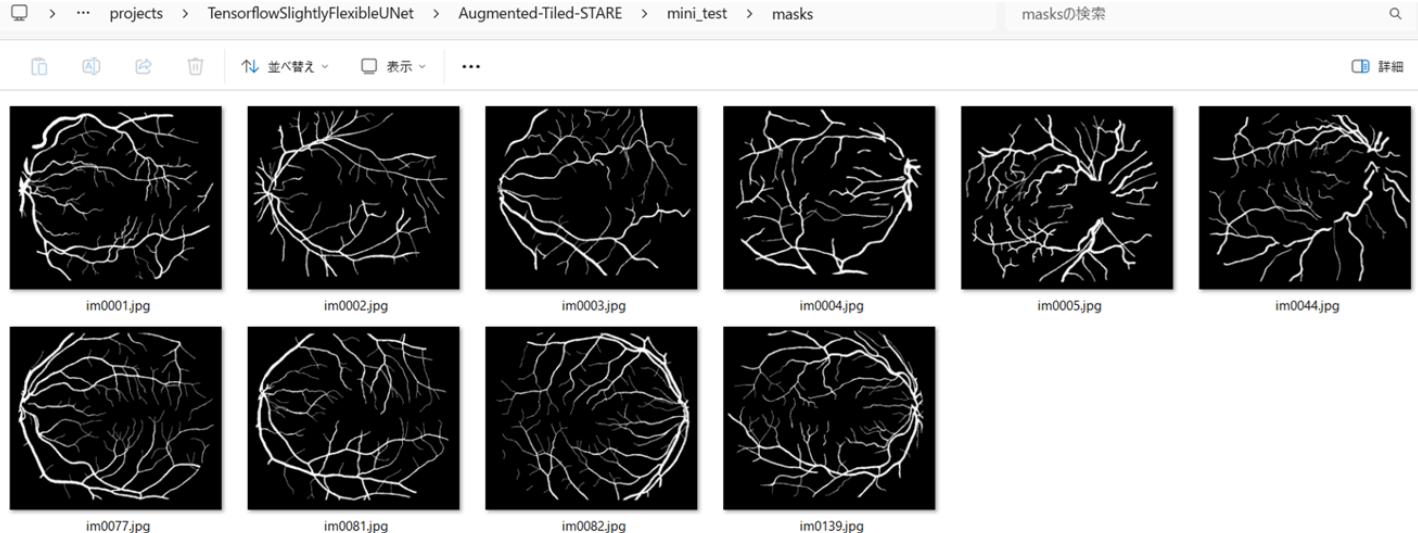
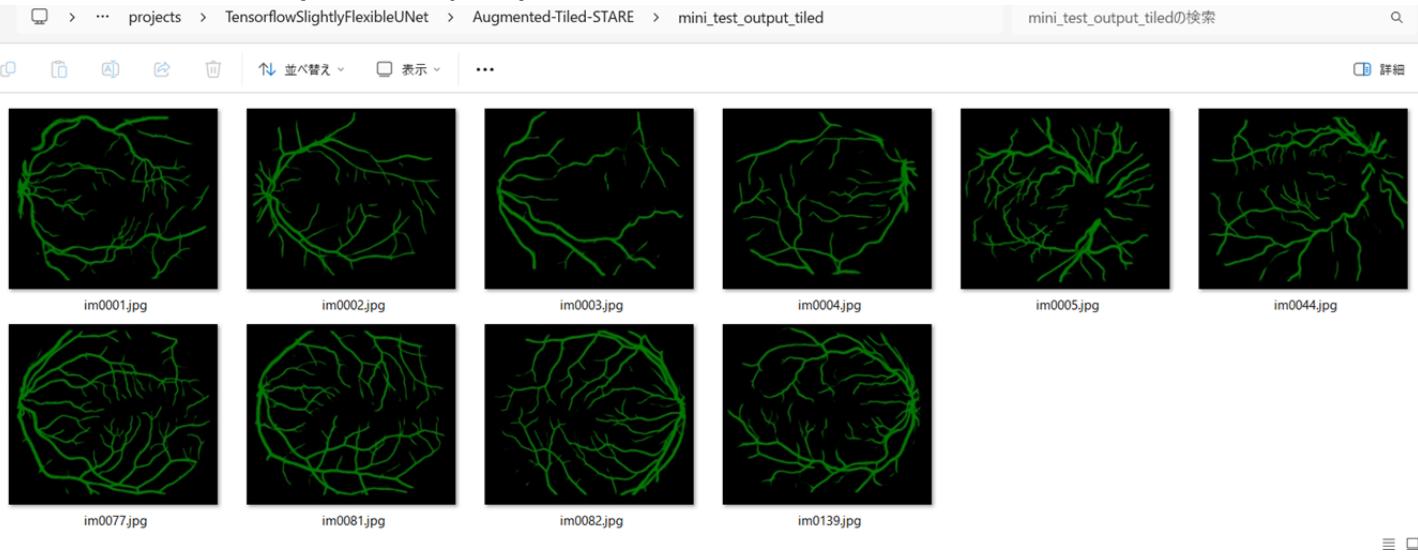
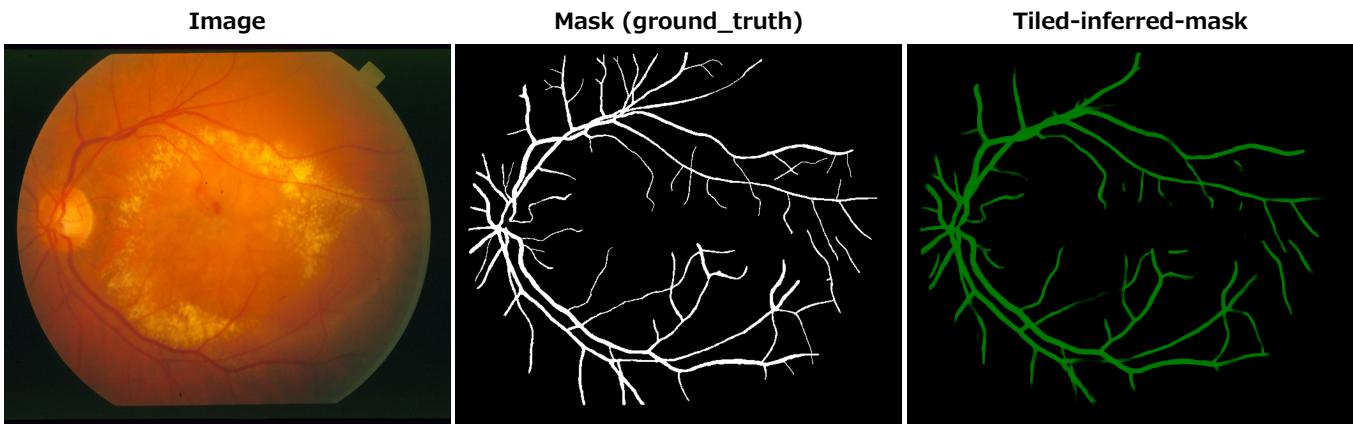
```
./4.tiled_infer.bat
```

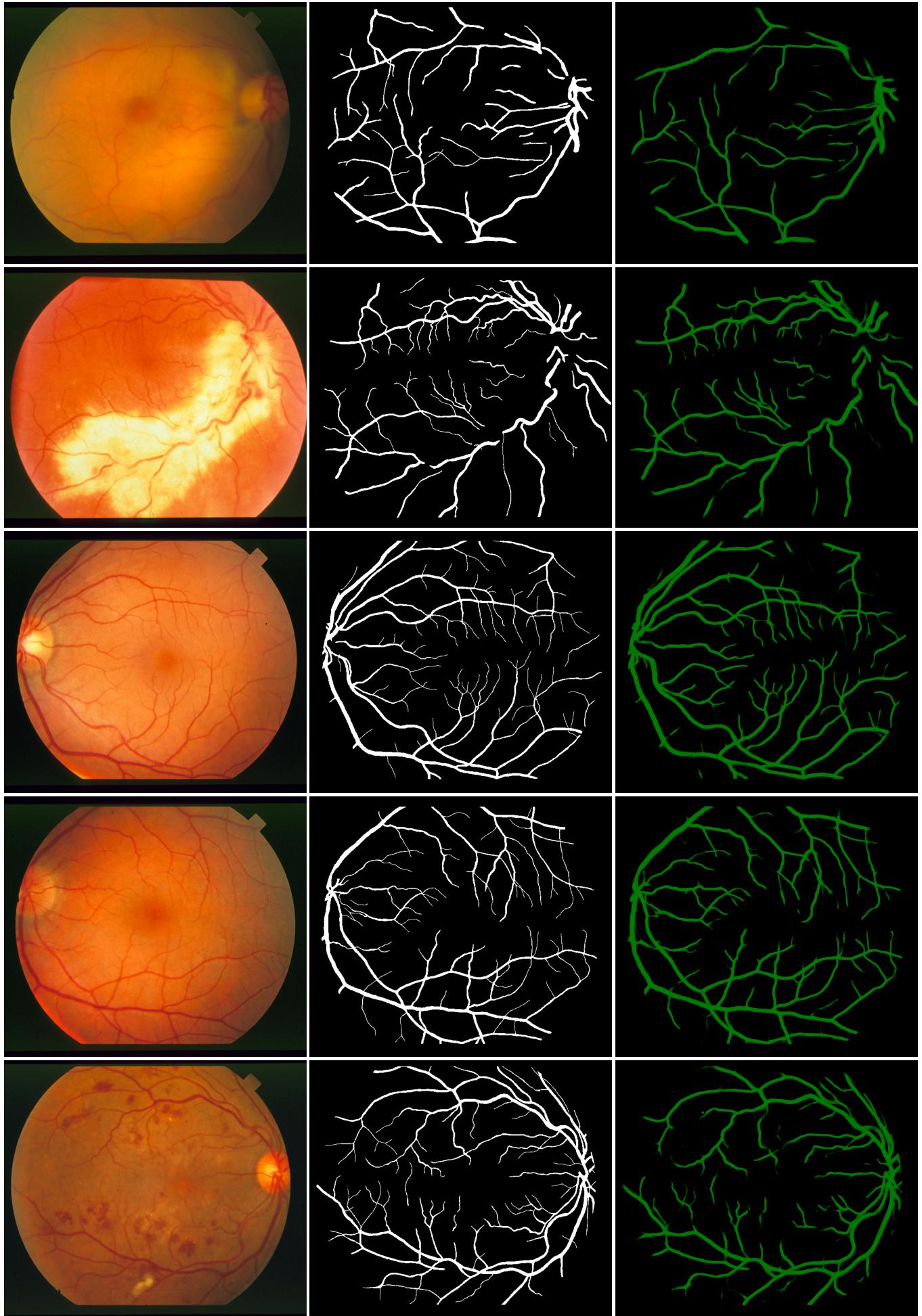
This simply runs the following command.

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

mini_test_images (3500x3025 pixels)



mini_test_mask(ground_truth)**Tiled inferred test masks (3500x3025 pixels)****Enlarged images and masks of 3500x3025 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. STructured Analysis of the Retina

<https://cecas.clemson.edu/~ahoover/stare/>

3. STARE

<https://github.com/openmedlab/Awesome-Medical-Dataset/blob/main/resources/STARE.md>

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

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

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

6, Tensorflow-Image-Segmentation-Retinal-Vessel

Toshiyuki Arai @antillia.com

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

7. Tensorflow-Tiled-Image-Segmentation-Augmented-Skin-Cancer

Toshiyuki Arai @antillia.com

<https://github.com/sarah-antillia/Tensorflow-Tiled-Image-Segmentation-Augmented-Skin-Cancer>

8. Tensorflow-Tiled-Image-Segmentation-Augmented-MultipleMyeloma

Toshiyuki Arai @antillia.com

<https://github.com/sarah-antillia/Tensorflow-Tiled-Image-Segmentation-Augmented-MultipleMyeloma>

9. Tiled-ImageMask-Dataset-Breast-Cancer

Toshiyuki Arai @antillia.com

<https://github.com/sarah-antillia/Tiled-ImageMask-Dataset-Breast-Cancer>