Tensorflow-Tiled-Image-Segmentation-Pre-Augmented-DRIVE-Retinal-Vessel (2025/02/05)

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

DRIVE.7z in **Mendeley Data Retinal Vessel**.

On detail of **DRIVE**, please refer to the official site:
DRIVE: Digital Retinal Images for Vessel Extraction , and github repository DRIVE.

Please see also our experiments:
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 Strategies
As demonstrated in our experiments Tensorflow-Tiled-Image-Segmentation-Pre-Augmented-STARE-Retinal-Vessel and Tensorflow-Tiled-Image-Segmentation-IDRiD-HardExudates , the Tiled Image Segmentation based on a simple UNet model trained by a tiledly-splitted images and masks dataset, is an effective method for the large image segmentation over 4K pixels. Furthermore, as mentioned in Tensorflow-Tiled-Image-Segmentation-Pre-Augmented-STARE-Retinal-Vessel, it is difficult to precisely segment Retinal Blood Vessels in small images using a simple UNet model because these vessels are typically very thin. 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.

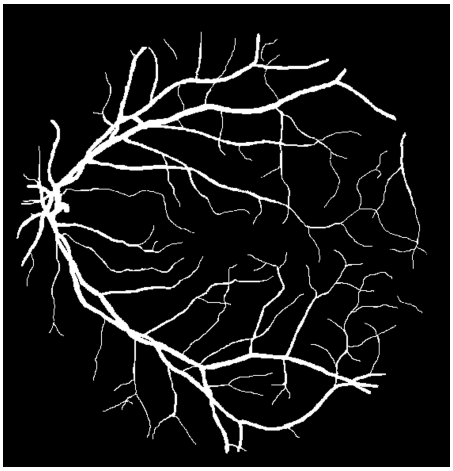
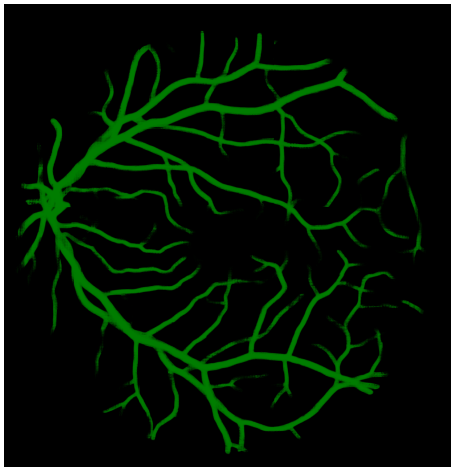
In this experiment, we employed the same strategies in this project as we did in the STARE-Retinal-Vessel.
1. Enlarged Dataset
We generated a 6x enlarged dataset of 40 JPG images and masks, each with 3390x3504 pixels, from the original DRIVE 565x584 pixels TIF image and GIF mask files using bicubic interpolation.

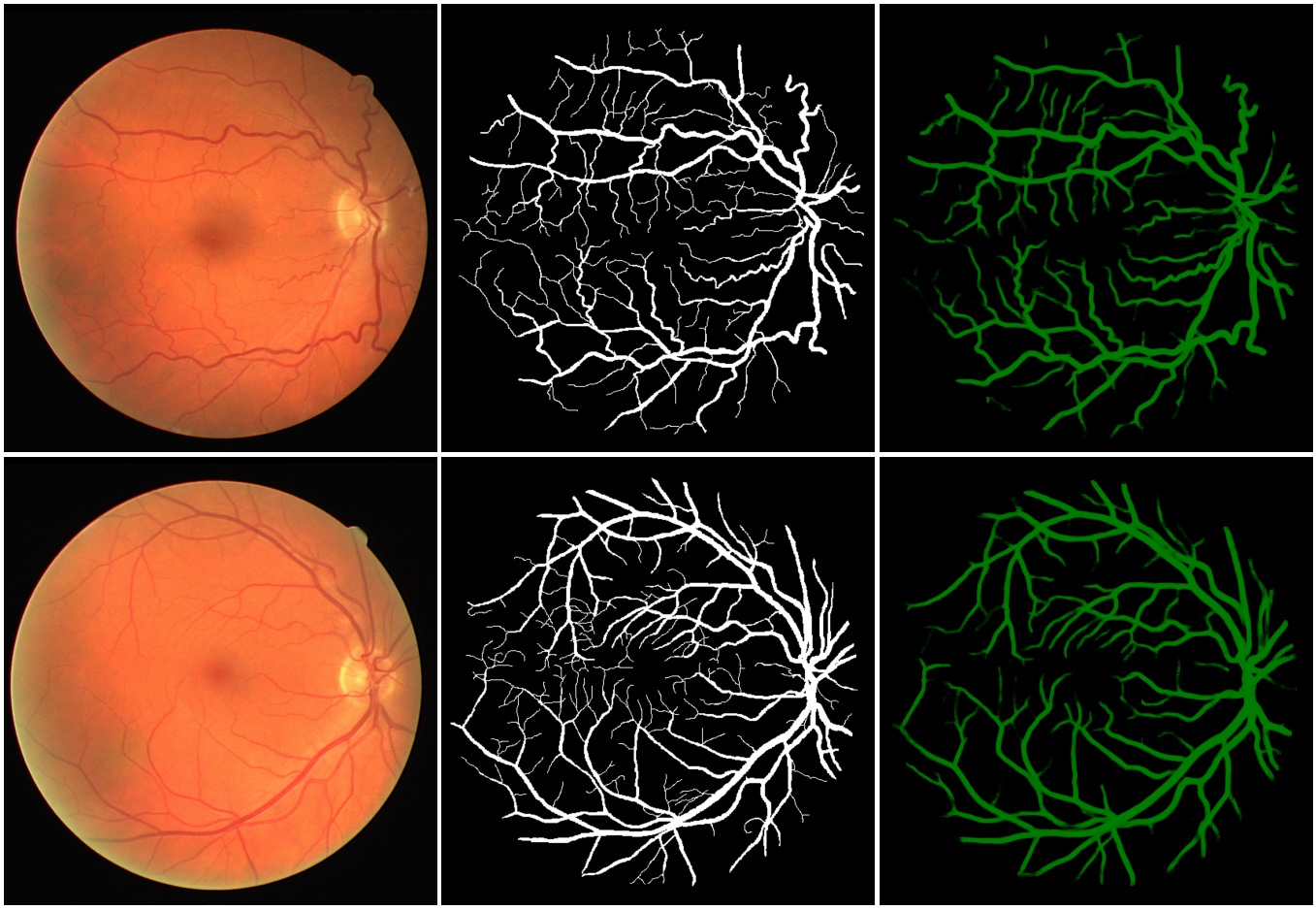
2. Pre Augemtned Tiled DRIVE ImageMask Dataset
We generated a pre-augmented image mask dataset from the enlarged dataset, which was tiledly-splitted to 512x512 pixels and reduced to 512x512 pixels image and mask dataset.

3. Train Segmention Model
We trained and validated a TensorFlow UNet model by using the **Pre Augmented Tiled DRIVE ImageMask Dataset**

4. Tiled Image Segmentation
We applied our Tiled-Image Segmentation inference method to predict the DRIVE Retinal Vessel for the mini_test images with a resolution of 3390x3504 pixels of the Enlarged Dataset.

Actual Tiled Image Segmentation for Images of 3390x3504 pixels
As shown below, the inferred masks look similar to the ground truth masks.

Input: image	Mask (ground_truth)	Prediction: inferred_mask
		



In this experiment, we used the simple UNet Model TensorflowSlightlyFlexibleUNet for this DRIVESegmentation 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 dataset DRIVE.7z in **Retinal Vessel**
On more detail, please refer to the github repository DRIVE

Authors and Institutions
Joes Staal (Image Sciences Institute, University Medical Center Utrecht)
Michael D. Abràmoff (Department of Ophthalmology and Visual Sciences, University of Iowa)
Meindert Niemeijer (Image Sciences Institute, University Medical Center Utrecht)
Max A. Viergever (Image Sciences Institute, University Medical Center Utrecht)
Bram van Ginneken (Image Sciences Institute, University Medical Center Utrecht)

Citation
@ARTICLE{1282003,
author={Staal, J. and Abramoff, M.D. and Niemeijer, M. and Viergever, M.A. and van Ginneken, B.},
journal={IEEE Transactions on Medical Imaging},
title={Ridge-based vessel segmentation in color images of the retina},
year={2004},
volume={23},
number={4},
pages={501-509},
doi={10.1109/TMI.2004.825627}}

2 Augmented-Tiled-DRIVE ImageMask Dataset

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

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

This is a 512x512 pixels pre augmented tiles dataset generated from 3500x3025 pixels 20 **Enlarged-images** and their corresponding **Enlarged-masks**.

.

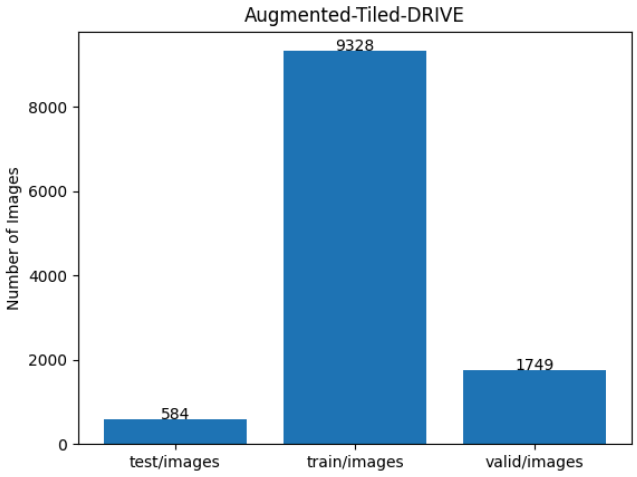
The folder structure of the original **DRIVE/training** dataset is the following.

```
./DRIVE
└─training
    ├─1st_manual
    │  ├─21_manual1.gif
    │  ├─22_manual1.gif
    │  ...
    │  └─40_manual1.gif
    └─images
       ├─21_training.tif
       ├─22_training.tif
       ...
       └─40_training.tif
```

We excluded all black (empty) masks and their corresponding images to generate our dataset from the original DRIVE.
On the derivation of the dataset, please refer to the following Python scripts.

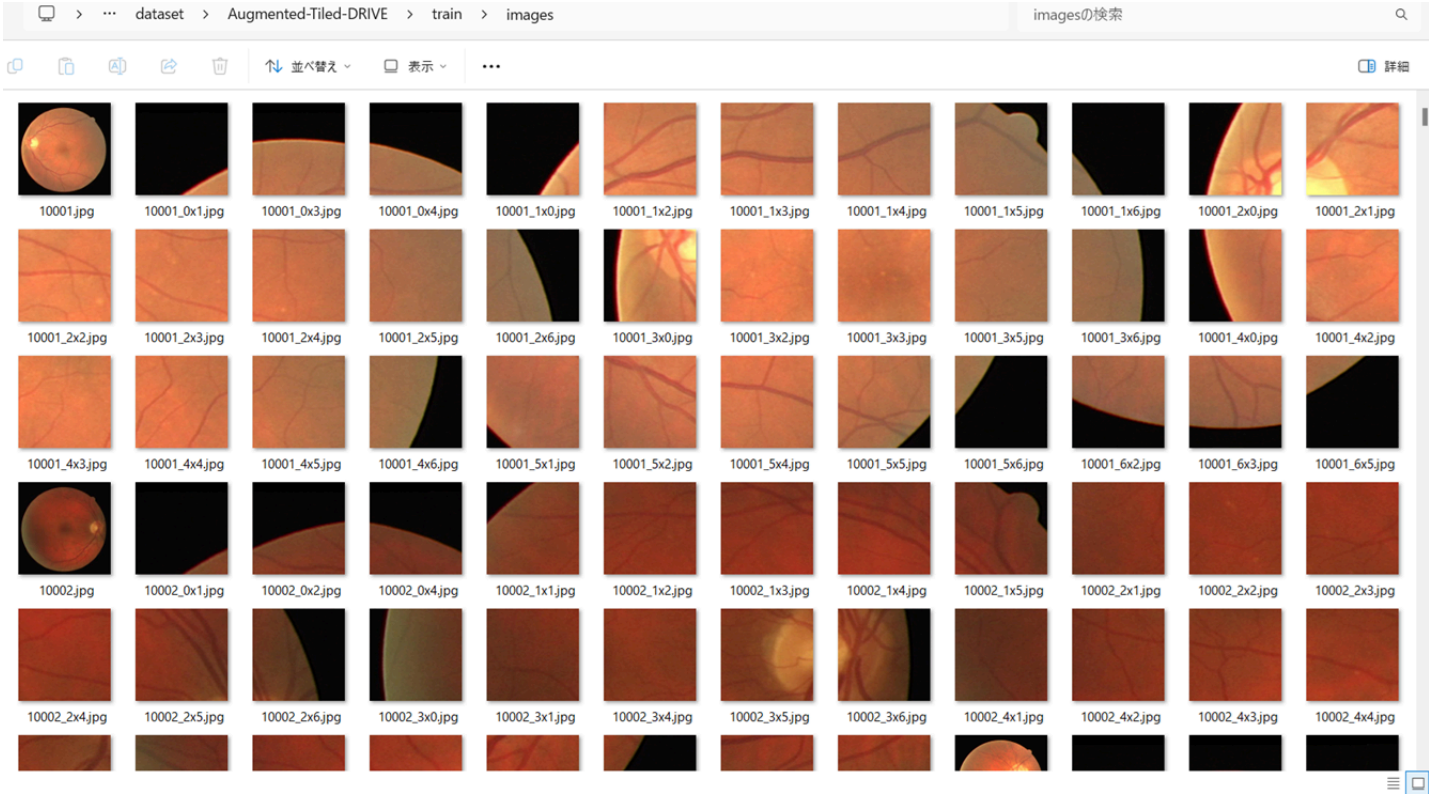
- Preprocessor.py
- TiledImageMaskDatasetGenerator.py
- split_tiled_master.py

Augmented-Tiled-DRIVE Statistics

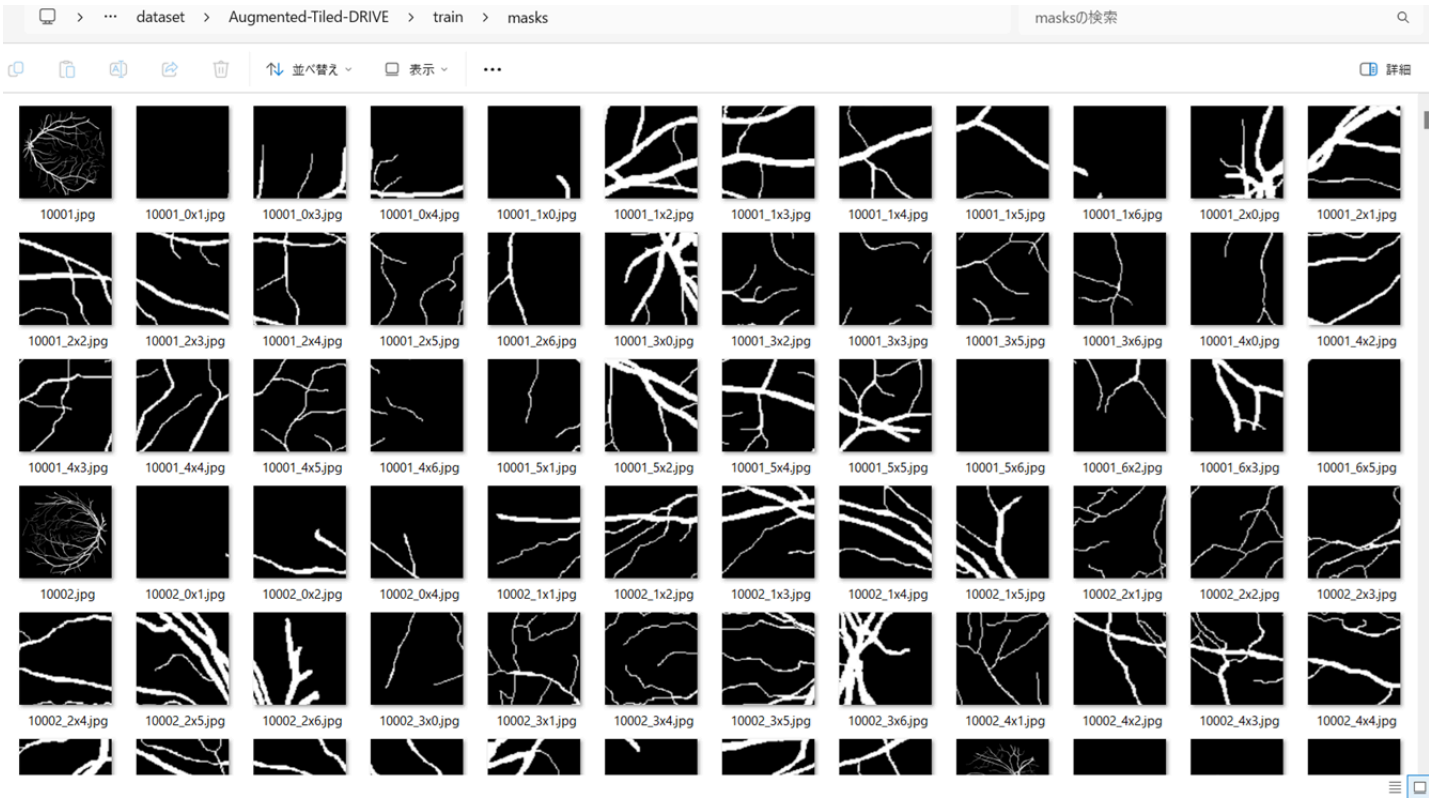


As shown above, the number of images of train and valid datasets is enough to use for a training set of our segmentation model.

Train_images_sample



Train_masks_sample



3 Train TensorflowUNet Model

We have trained DRIVE TensorflowUNet Model by using the following **train_eval_infer.config** file.
Please move to ./projects/TensorflowSlightlyFlexibleUNet/Augmented-Tiled-DRIVE 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 learing_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
```

Tiled inference

We used 3390x3504 pixels enlarged images and masks generated by Preprocessor.pys as a mini_test dataset for our TiledInference.

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

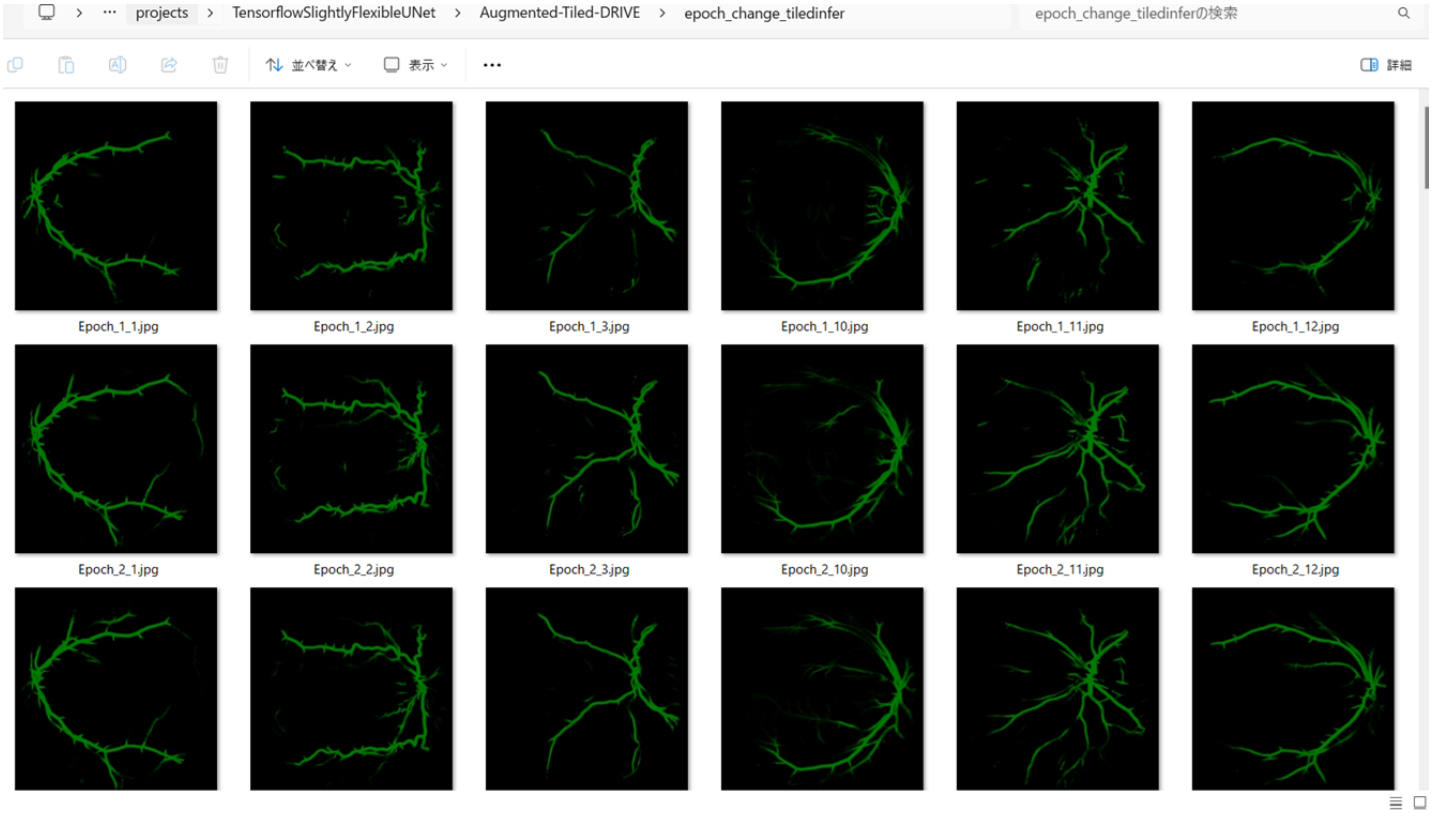
Epoch change inference callbacks

Enabled epoch_change_infer callback.

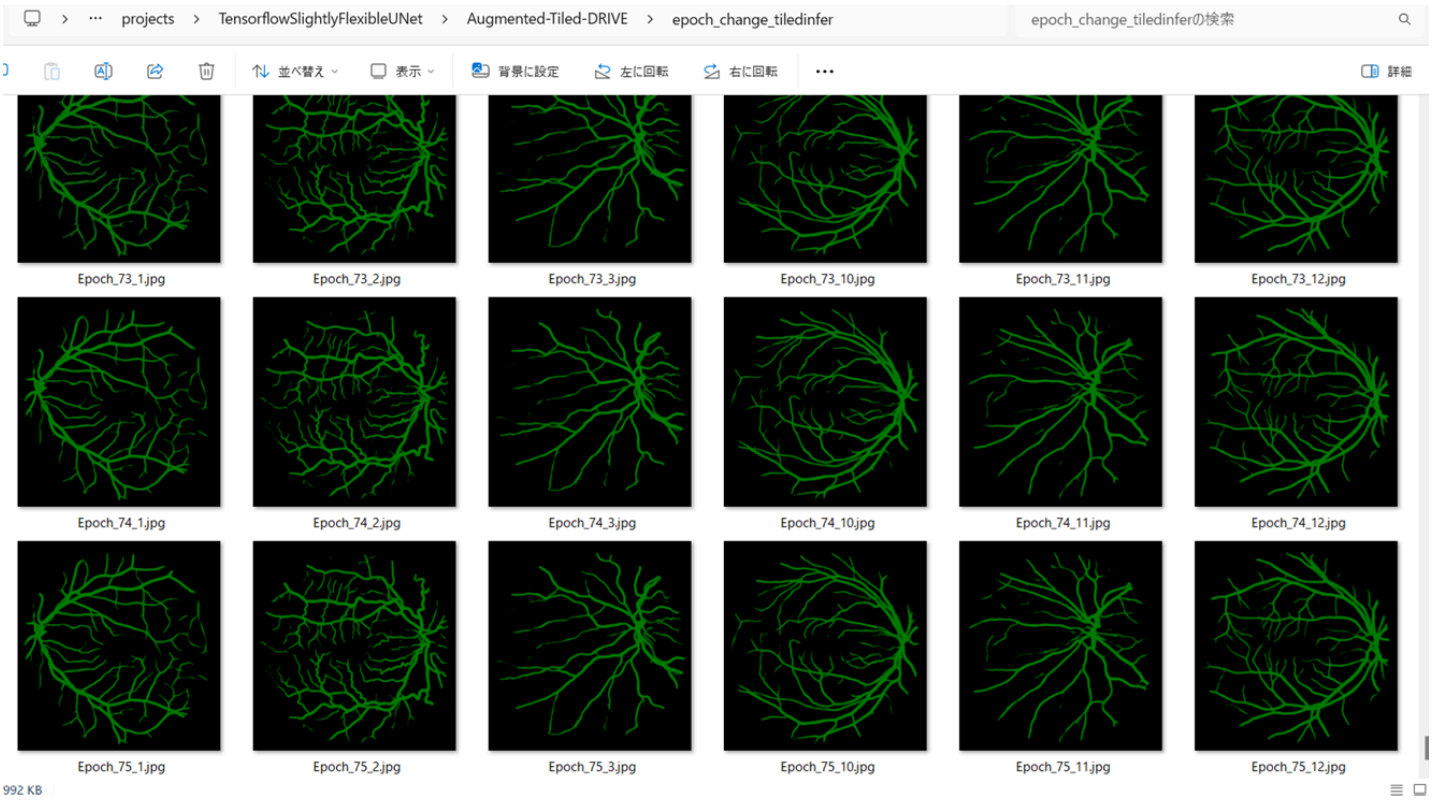
```
[train]
epoch_change_infer      = False
epoch_change_infer_dir  = "./epoch_change_infer"
epoch_change_tiledinfer = True
epoch_change_tiledinfer_dir = "./epoch_change_tiledinfer"
num_infer_images        = 6
```

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)



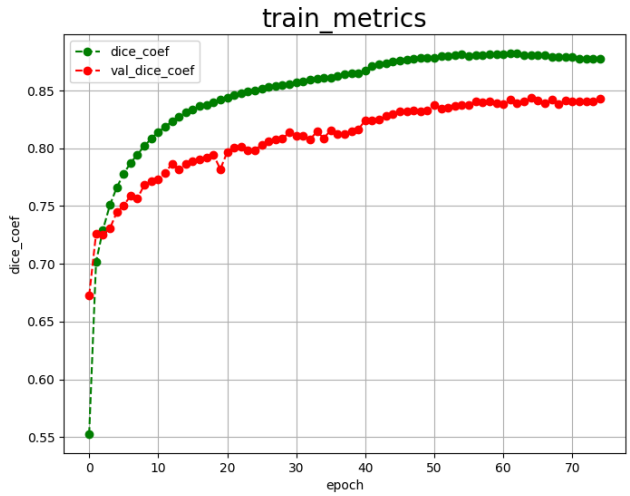
Epoch_change_inference output at ending (73,74,75)



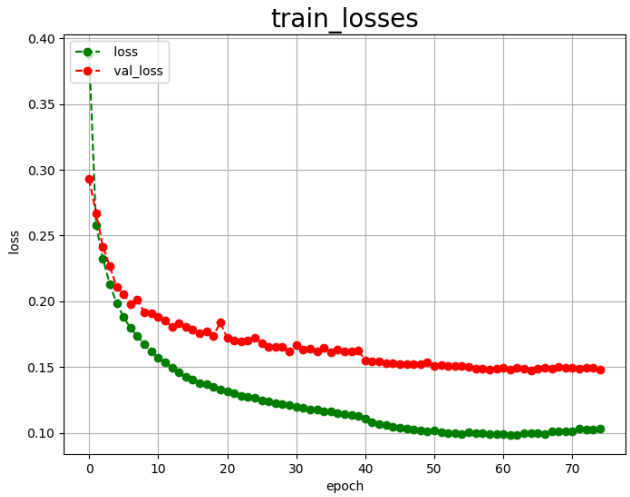
In this experiment, the training process was stopped at epoch 75 by EarlyStopping Callback.

```
PowerShell 7 (x64)
9328/9328 [==============================] - 2523s 270ms/sample - loss: 0.0998 - dice_coef: 0.8805 - val_loss: 0.1490 - val_dice_coef: 0.8405 - lr: 1.7920e-06
Epoch 65/100
9328/9328 [==============================] - ETA: 0s - loss: 0.0996 - dice_coef: 0.8807
Epoch 65: val_loss improved from 0.14778 to 0.14771, saving model to ./models¥best_model.h5
9328/9328 [==============================] - 2515s 270ms/sample - loss: 0.0996 - dice_coef: 0.8807 - val_loss: 0.1477 - val_dice_coef: 0.8434 - lr: 1.7920e-06
Epoch 66/100
9328/9328 [==============================] - ETA: 0s - loss: 0.0995 - dice_coef: 0.8808
Epoch 66: val_loss did not improve from 0.14771
9328/9328 [==============================] - 2493s 267ms/sample - loss: 0.0995 - dice_coef: 0.8808 - val_loss: 0.1487 - val_dice_coef: 0.8411 - lr: 1.7920e-06
Epoch 67/100
9328/9328 [==============================] - ETA: 0s - loss: 0.0993 - dice_coef: 0.8810
Epoch 67: val_loss did not improve from 0.14771
9328/9328 [==============================] - 2524s 271ms/sample - loss: 0.0993 - dice_coef: 0.8810 - val_loss: 0.1496 - val_dice_coef: 0.8391 - lr: 1.7920e-06
Epoch 68/100
9328/9328 [==============================] - ETA: 0s - loss: 0.1010 - dice_coef: 0.8790
Epoch 68: val_loss did not improve from 0.14771
9328/9328 [==============================] - 2524s 271ms/sample - loss: 0.1010 - dice_coef: 0.8790 - val_loss: 0.1489 - val_dice_coef: 0.8419 - lr: 7.1680e-07
Epoch 69/100
9328/9328 [==============================] - ETA: 0s - loss: 0.1010 - dice_coef: 0.8792
Epoch 69: val_loss did not improve from 0.14771
9328/9328 [==============================] - 2561s 275ms/sample - loss: 0.1010 - dice_coef: 0.8792 - val_loss: 0.1504 - val_dice_coef: 0.8386 - lr: 7.1680e-07
Epoch 70/100
9328/9328 [==============================] - ETA: 0s - loss: 0.1008 - dice_coef: 0.8794
Epoch 70: val_loss did not improve from 0.14771
9328/9328 [==============================] - 2596s 278ms/sample - loss: 0.1008 - dice_coef: 0.8794 - val_loss: 0.1491 - val_dice_coef: 0.8412 - lr: 7.1680e-07
Epoch 71/100
9328/9328 [==============================] - ETA: 0s - loss: 0.1008 - dice_coef: 0.8794
Epoch 71: val_loss did not improve from 0.14771
9328/9328 [==============================] - 2513s 269ms/sample - loss: 0.1008 - dice_coef: 0.8794 - val_loss: 0.1492 - val_dice_coef: 0.8408 - lr: 7.1680e-07
Epoch 72/100
9328/9328 [==============================] - ETA: 0s - loss: 0.1029 - dice_coef: 0.8772
Epoch 72: val_loss did not improve from 0.14771
9328/9328 [==============================] - 2588s 277ms/sample - loss: 0.1029 - dice_coef: 0.8772 - val_loss: 0.1490 - val_dice_coef: 0.8410 - lr: 2.8672e-07
Epoch 73/100
9328/9328 [==============================] - ETA: 0s - loss: 0.1026 - dice_coef: 0.8774
Epoch 73: val_loss did not improve from 0.14771
9328/9328 [==============================] - 2519s 270ms/sample - loss: 0.1026 - dice_coef: 0.8774 - val_loss: 0.1491 - val_dice_coef: 0.8408 - lr: 2.8672e-07
Epoch 74/100
9328/9328 [==============================] - ETA: 0s - loss: 0.1027 - dice_coef: 0.8774
Epoch 74: val_loss did not improve from 0.14771
9328/9328 [==============================] - 2504s 268ms/sample - loss: 0.1027 - dice_coef: 0.8774 - val_loss: 0.1493 - val_dice_coef: 0.8405 - lr: 2.8672e-07
Epoch 75/100
9328/9328 [==============================] - ETA: 0s - loss: 0.1028 - dice_coef: 0.8773
Epoch 75: val_loss did not improve from 0.14771
9328/9328 [==============================] - 2497s 268ms/sample - loss: 0.1028 - dice_coef: 0.8773 - val_loss: 0.1480 - val_dice_coef: 0.8429 - lr: 2.8672e-07
Epoch 75: early stopping
=== Save history.json
```

[train_metrics.csv]



[train_losses.csv]



4 Evaluation

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

```
./2.evaluate.bat
```

This bat file simply runs the following command.

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

Evaluation console output:

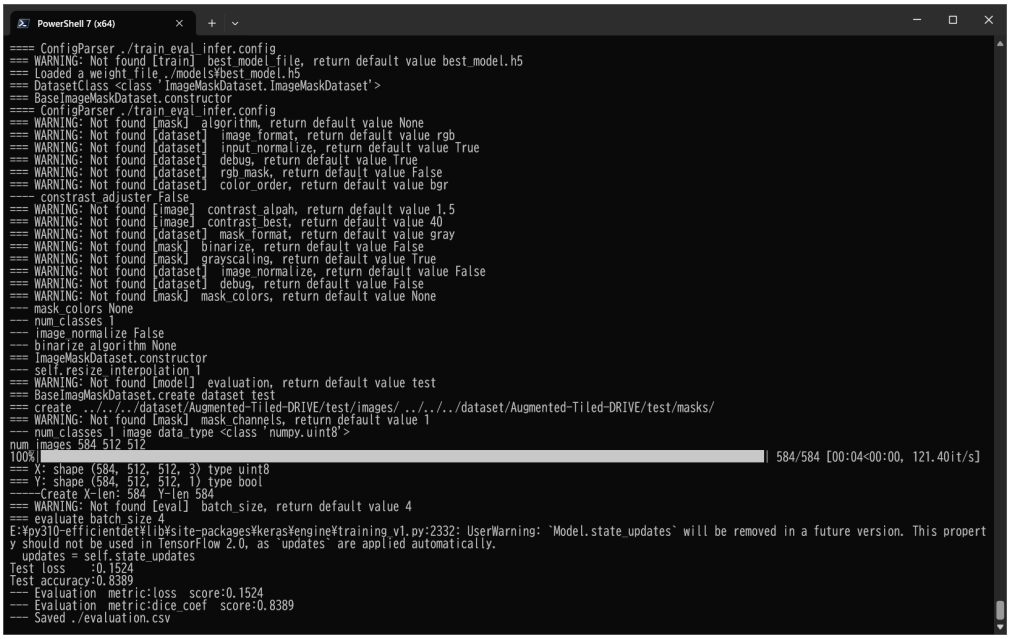


Image-Segmentation-DRIVE evaluation.csv
The loss (bce_dice_loss) to this Augmented-Tiled-DRIVE/test was low, and dice_coef high as shown below.

```
loss,0.1524
dice_coef,0.8389
```

5 Tiled inference

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

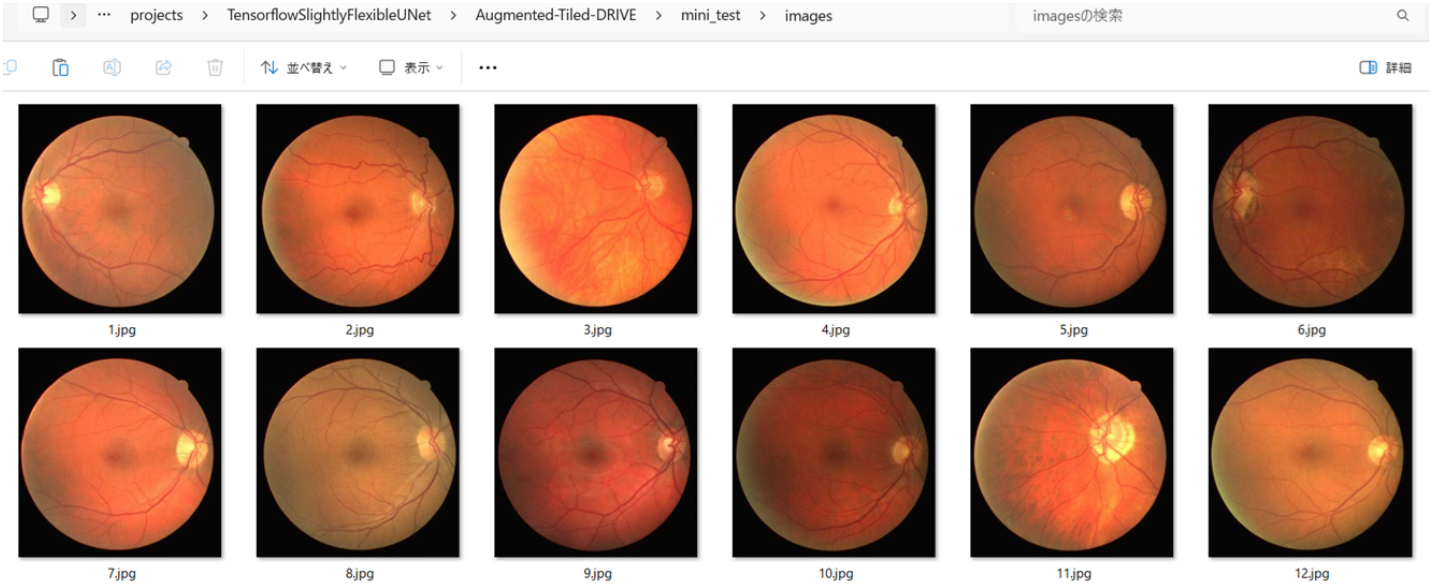
```
./4.tiled_infer.bat
```

This simply runs the following command.

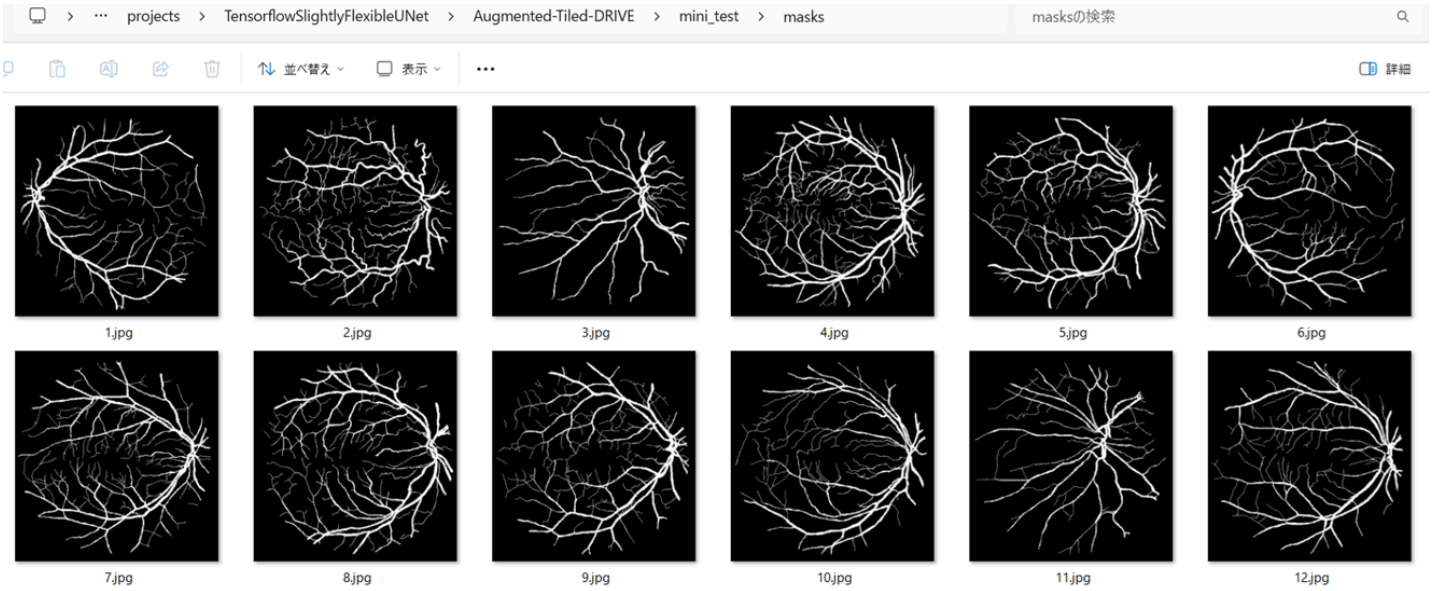
```
python ../../../src/TensorflowUNetTiledInferencer.py ./train_eval_infer.config
```

---

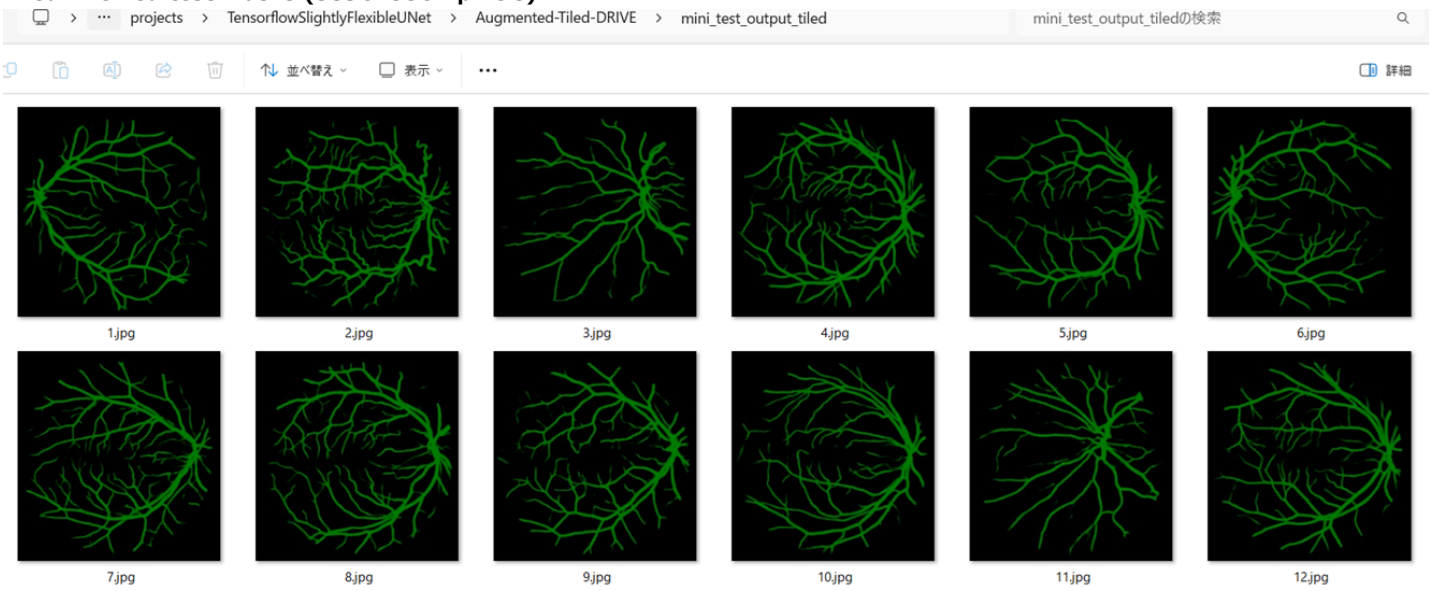
mini_test_images (3390x3504 pixels)



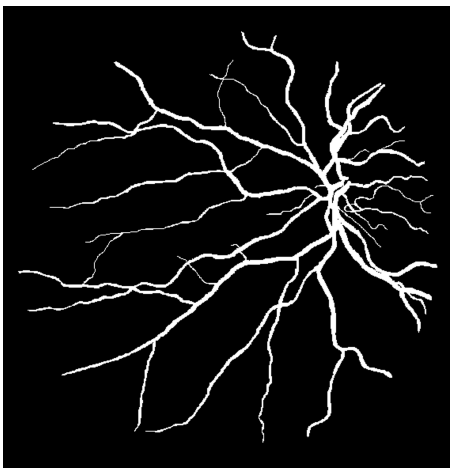
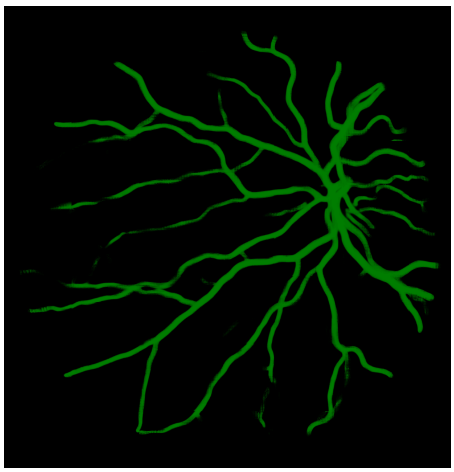
mini_test_mask(ground_truth)

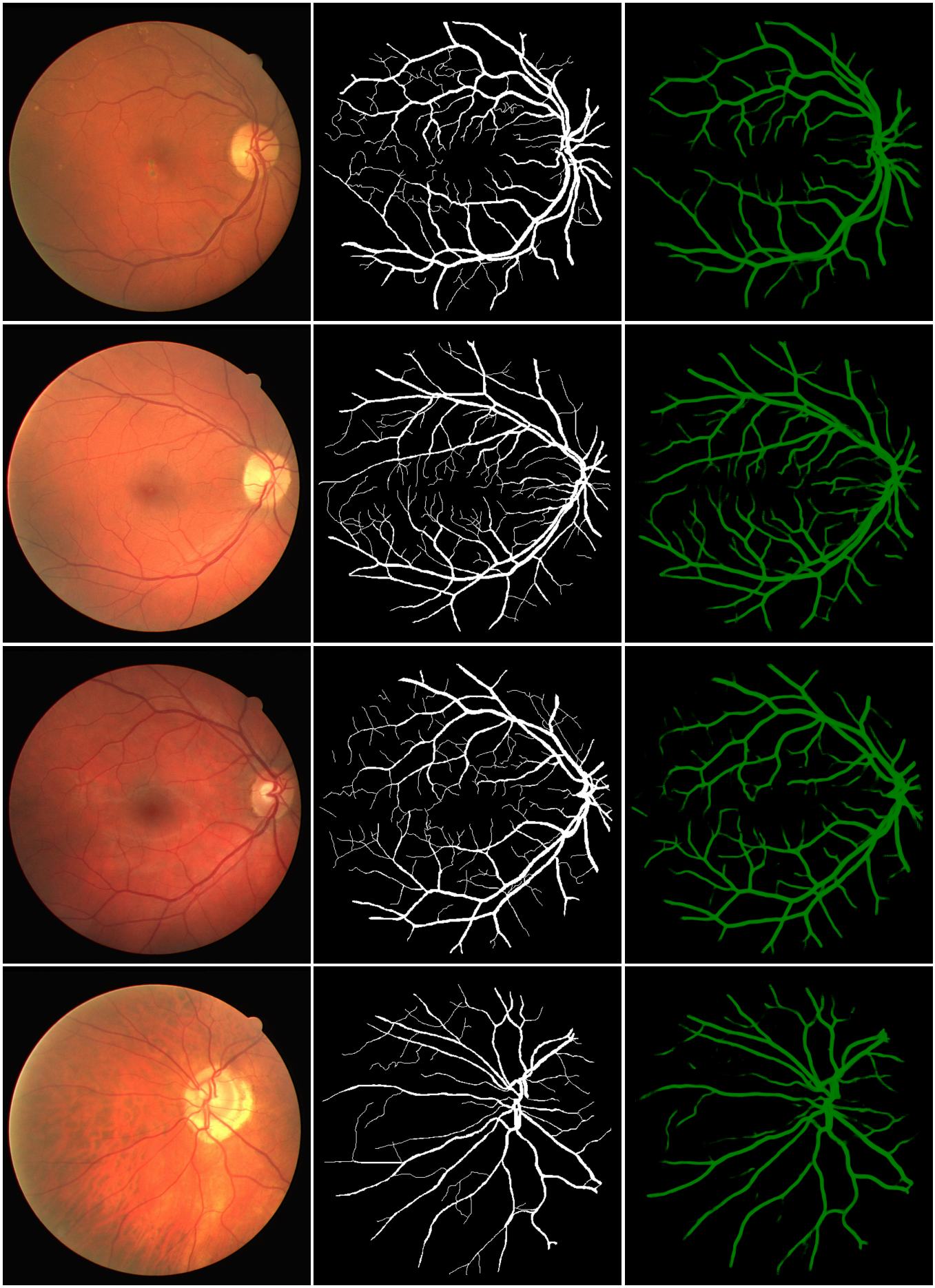


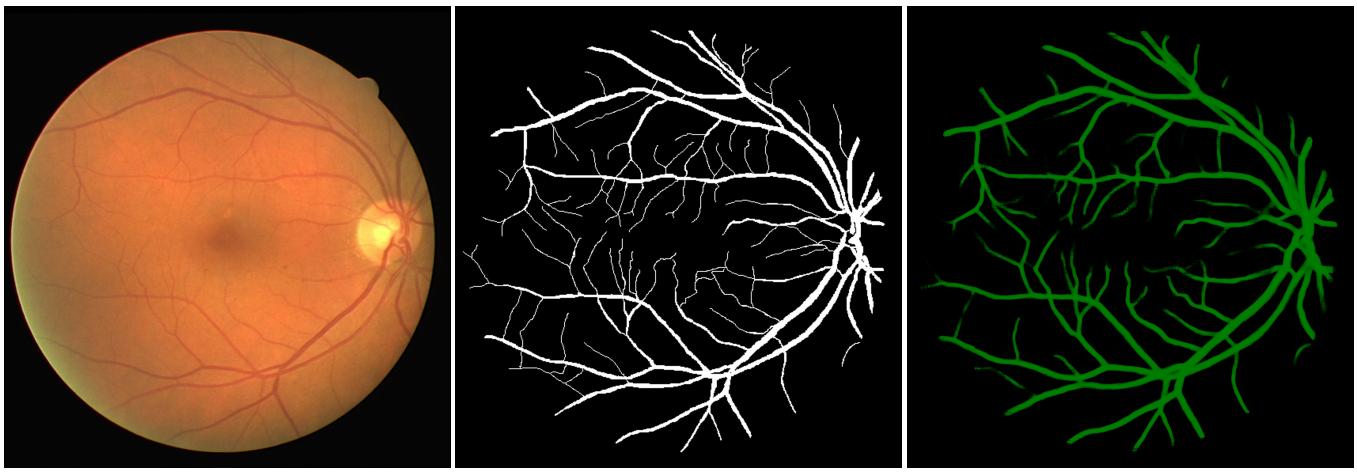
Tiled inferred test masks (3390x3504 pixels)



Enlarged images and masks of 3390x3504 pixels

Image	Mask (ground_truth)	Tiled-inferred-mask
		





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. DRIVE: Digital Retinal Images for Vessel Extraction
https://drive.grand-challenge.org/

3. DRIVE
https://github.com/openmedlab/Awesome-Medical-Dataset/blob/main/resources/DRIVE.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
Sanjeewani, 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-Image-Segmentation-Retinal-Vessel
Toshiyuki Arai @antillia.com
https://github.com/sarah-antillia/Tensorflow-Image-Segmentation-Retinal-Vessel

8. Tensorflow-Tiled-Image-Segmentation-Augmented-Skin-Cancer
Toshiyuki Arai @antillia.com
https://github.com/sarah-antillia/Tensorflow-Tiled-Image-Segmentation-Augmented-Skin-Cancer

9. Tensorflow-Tiled-Image-Segmentation-Augmented-MultipleMyeloma
Toshiyuki Arai @antillia.com
https://github.com/sarah-antillia/Tensorflow-Tiled-Image-Segmentation-Augmented-MultipleMyeloma

10. Tiled-ImageMask-Dataset-Breast-Cancer
Toshiyuki Arai @antillia.com
https://github.com/sarah-antillia/Tiled-ImageMask-Dataset-Breast-Cancer