kaggle

Kaggle Winner Presentation

vesuvius-challenge-ink-detection

9-th place

HENG CHER KENG

Agenda

1. Self introduction
2. Solution
3. Important Findings

---

4. Reproduce the Submission

Background

Background

I am a contract computer vision and deep learning algorithm engineer.

- discovering fractures in x-ray bone images
- implementing visual slam for robotic navigation.

Kaggle competition master and had participated in previous Kaggle competitions related in image segmentation.

Solution

Summary

- My solution is an ensemble of **two models**, each model has 2 deep nets trained from different fold splits.

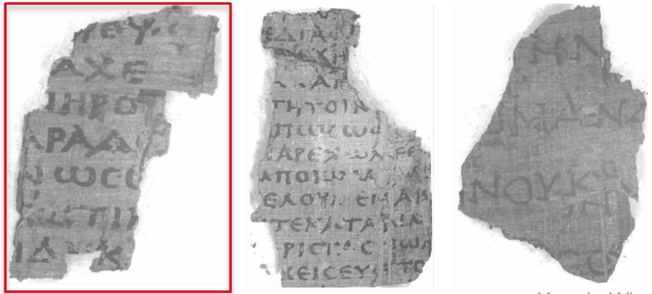
- Each net consists of : encoder + mean-pool + decoder

model 1: encoder = resnet34d CNN,
decoder = unet

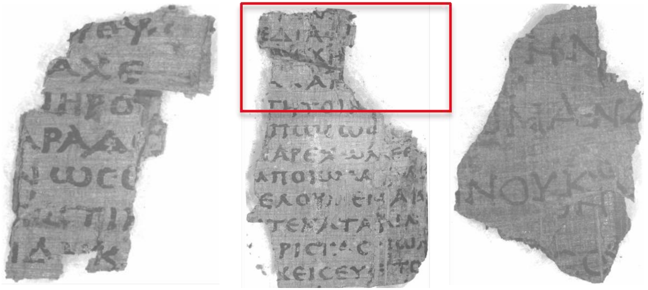
model 2: encoder = pvtv2-b3 pVIT,
decoder = daformer

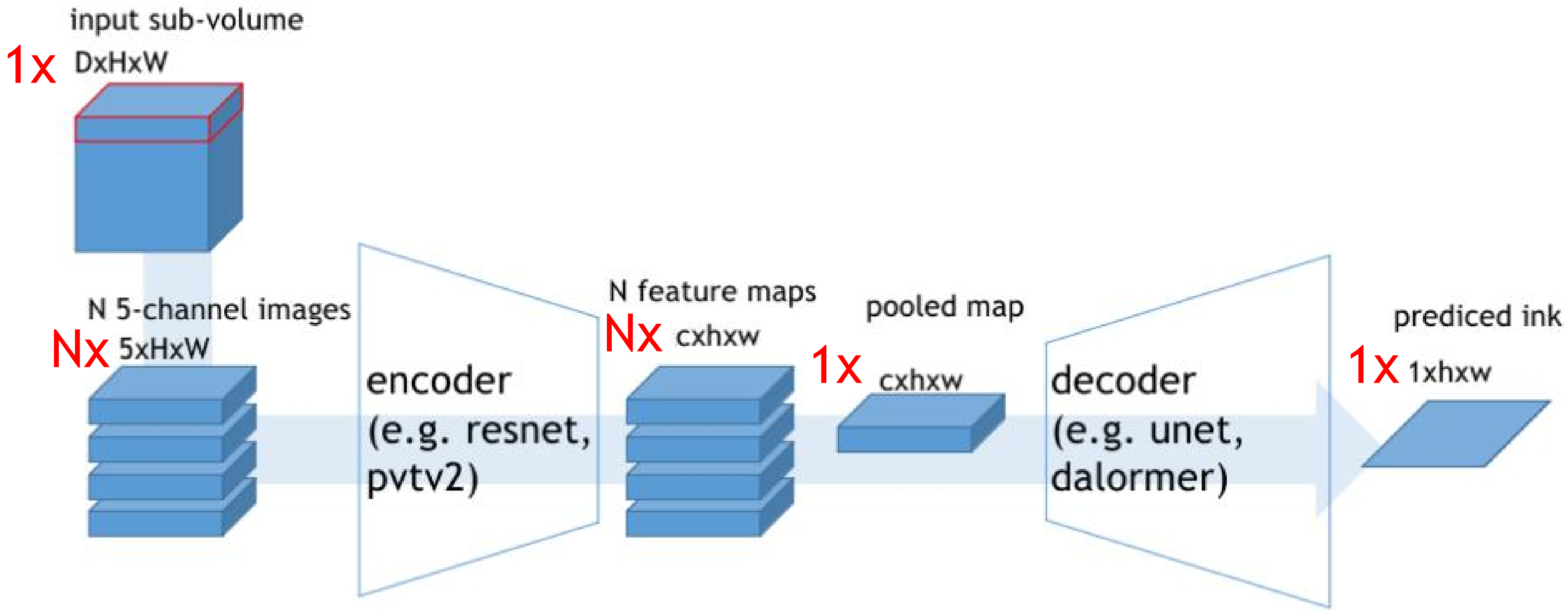
red: validation
non-red: training

fold1:



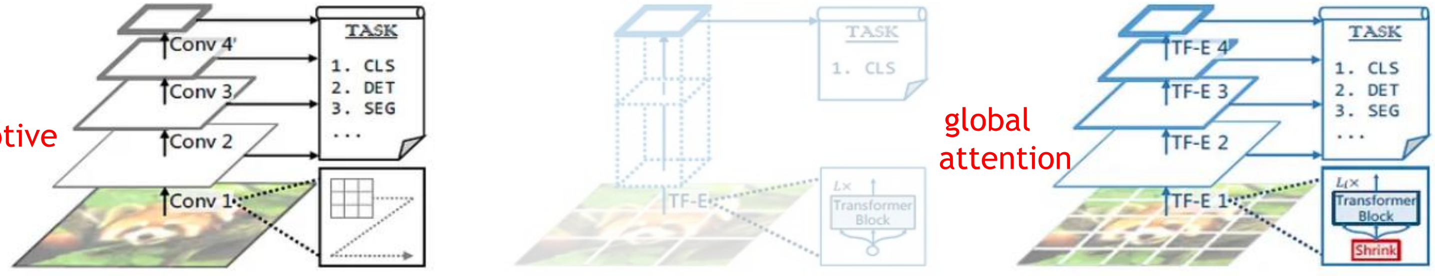
fold2aa:





- Input sub-volume is divided into several 5-channel images.
- The images overlap in channels, e.g. image1 uses z slice [20 to 25], image2 uses [22 to 27], etc...
- At the encoder, the feature maps are mean pooled into single feature map

- choice of backbone:

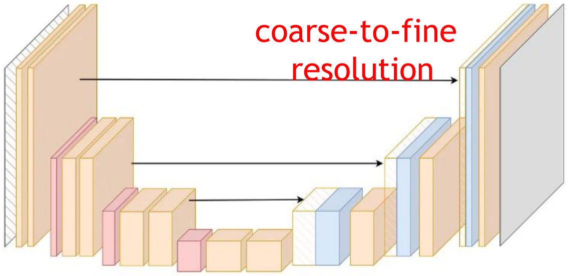


(a) CNNs: VGG [53], ResNet [21], *etc.* (b) Vision Transformer [12] (c) Pyramid Vision Transformer (ours)

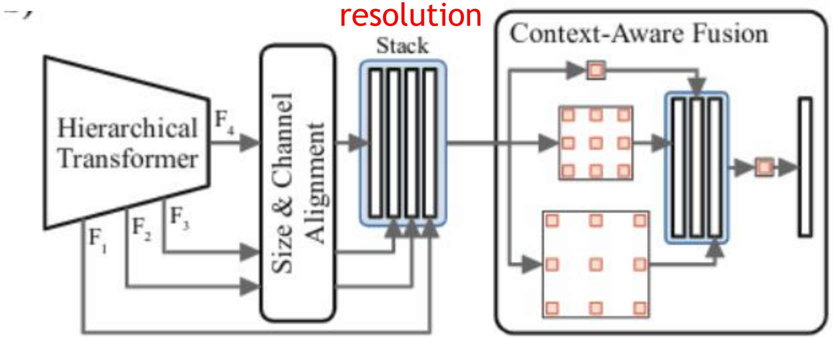
CNN : convolution net

pVIT: pyramid vision transformer

- choice of encoder-decoder:



Unet



Daformer

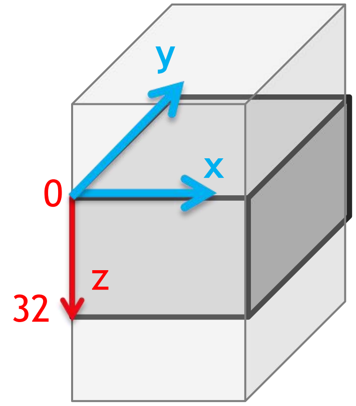
[1] “PVT v2: Improved Baselines with Pyramid Vision Transformer” - Wenhai Wang
https://arxiv.org/abs/2106.13797

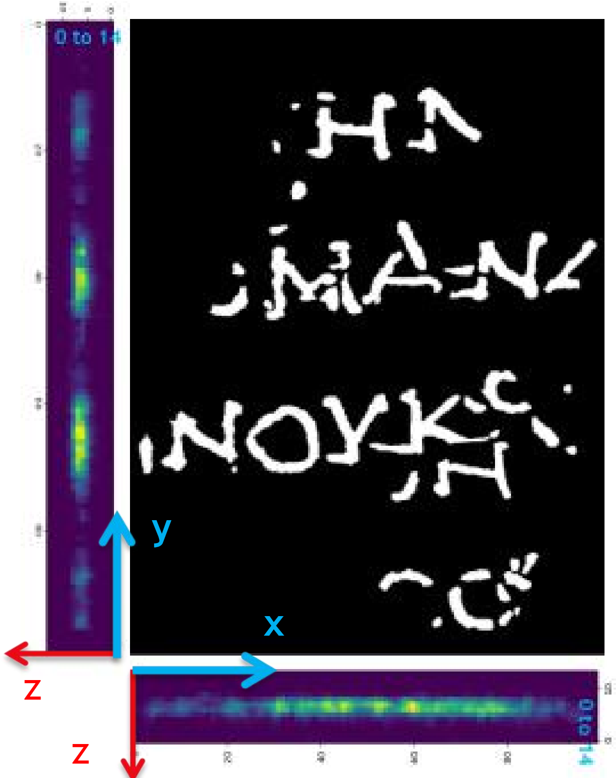
Important Findings

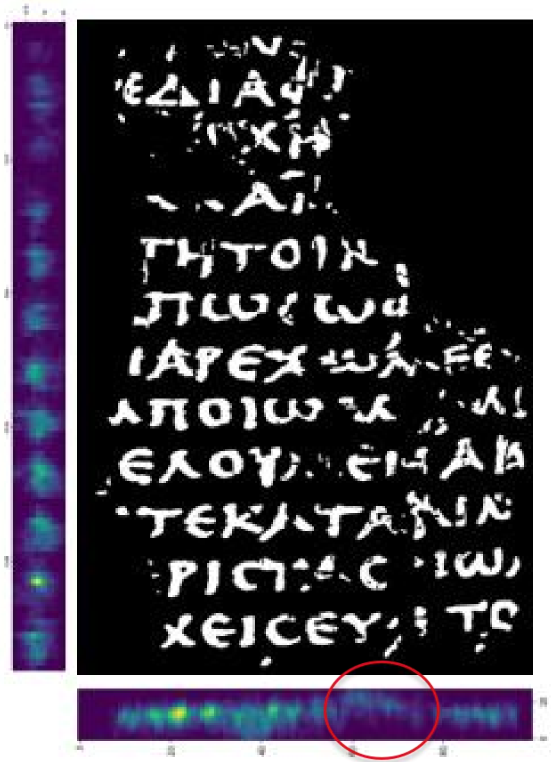
Unique tricks that works:

1. Mean pooling as effective slice selection

The central 16 or 32 z slices are used in subvolume. Here, the CAM activation heatmap shows which slices are activated.







2. Small-to-large crop training

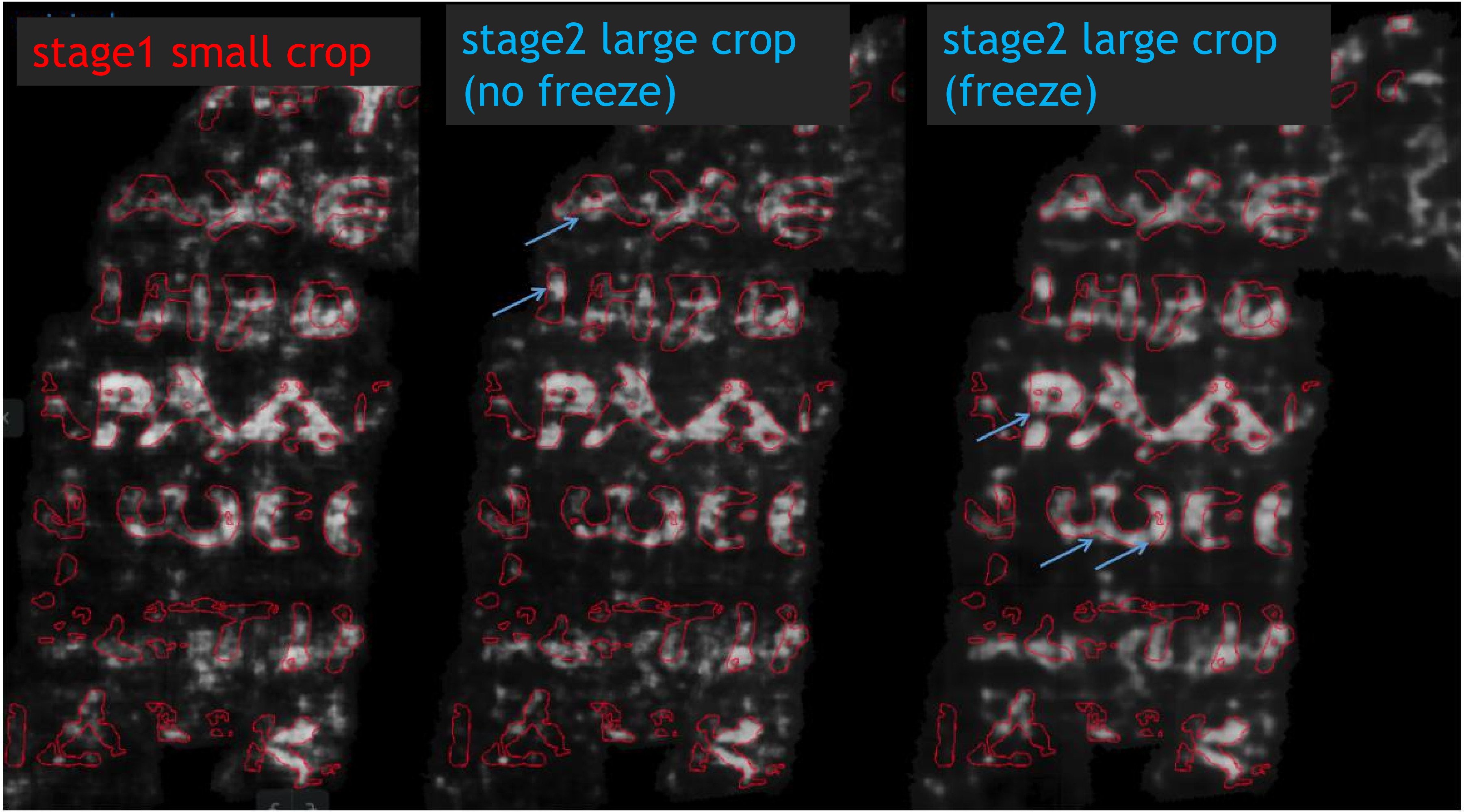
stage1 : train at small crop
stage2 : finetune at large crop (small learning rate, freeze if required)

CNN: 128 -> 256, pVIT: 128 -> 384*

* We had to reduce depth from 32 to 16 for pVIT at larger crop

r050_resnet34-unet-mean32-pool-05								
		crop_size	crop_depth	augmentation	freeze_encode	learning rate	epoch	output model file
fold-1	stage1_0	128	32	train_augment_v2	FALSE	1.00E-03	0 to 13	00008788.model.pth
	stage1_1	128	32	train_augment_v2	FALSE	5.00E-04	13 to 21	00014196.model.pth
	stage1_2	128	32	train_augment_v2	FALSE	1.00E-04	21 to 23	00015548.model.pth
	stage2_0	256	32	train_augment_v2	TRUE	1.00E-04	23 to 44	fold1-Resnet34MeanPool.00018924.model.pth
fold-2aa	stage1_0	128	32	train_augment_v2	FALSE	1.00E-03	0 to 15	00009210.model.pth
	stage1_1	128	32	train_augment_v2	FALSE	5.00E-04	15 to 23	00014122.model.pth
	stage2_0	256	32	train_augment_v2	TRUE	5.00E-04	23 to 28	fold-2aa-Resnet34MeanPool-00014850.model.pth

r091_pvt_v2_b3-daformer-mean32-aug2-00								
		crop_size	crop_depth		freeze_encode	learning rate	epoch	output model file
fold-1	stage1_0	128	32	train_augment_v2	FALSE	1.00E-04	0 to 13	00017576.model.pth
	stage1_1	224	32	train_augment_v2	FALSE	1.00E-04	13 to 37	00028080.model.pth
	stage2_0	384	16	train_augment_v2f	FALSE	1.00E-04	37 to 47	fold1-Pvt2b3MeanPoolDaformer-00029376.model.pth
fold-2aa	stage1_0	224	32	train_augment_v2	FALSE	1.00E-04	0 to 22	00008624.model.pth
	stage2_0	384	16	train_augment_v2f	FALSE	1.00E-04	22 to 26	fold2aa-Pvt2b3MeanPoolDaformer-00009159.model.pth


stage1 small crop
stage2 large crop (no freeze)
stage2 large crop (freeze)

3. Ensemble

- CNN and pVIT are complimentary.
- For final submission, we choose higher threshold (0.55) than local validation (0.45).

Since only 20% of the test data are used for the public leaderboard, the public score is not relieable. At threshold 0.45, local validation fbeta score is 0.64, but the public score is 0.75. This means that private score is likely to be lower than. Finally we think the optimal threshold for private should NOT be the optimal for public.

local validation		**threshold**	**fbeta**
r050_resnet34-unet-mean32-pool-05	fold 1	0.45	0.62
	fold 2aa	0.40	0.65
r091_pvt_v2_b3-daformer-mean32-aug2-00	fold 1	0.35	0.63
	fold 2aa	0.60	0.68
average		**0.45**	**0.64**

(public **0.75**)

		Private Score	Public Score
9th place : final-ensemble v1 - Version 4 Succeeded (after deadline) · 12h ago · Notebook 9th place : final-ensemble v1 | Version 4	threshold **0.55**	**0.654354**	**0.738625**

Reproduce the Submission

1. Training code: Github repository

   https://github.com/hengck23/solution-vesuvius-challenge-ink-detection

2. Submmision code: Kaggle notebook

   https://www.kaggle.com/code/hengck23/9th-place-final-ensemble-v1?scriptVersionId=136499959

3. For contact, please use

   email: hengcherkeg235@gmail.com

Training code: Github repository

https://github.com/hengck23/solution-vesuvius-challenge-ink-detection

README.md

Vesuvius Challenge - Ink Detection (9th place solution)

https://www.kaggle.com/competitions/vesuvius-challenge-ink-detection

For solution discussion, refer to https://www.kaggle.com/competitions/vesuvius-challenge-ink-detection/discussion/417361

1. Hardware

- GPU: 2x Nvidia Quadro RTX 8000, each with VRAM 48 GB
- CPU: Intel® Xeon(R) Gold 6240 CPU @ 2.60GHz, 72 cores
- Memory: 376 GB RAM

2. OS

- ubuntu 18.04.5 LTS

3. Set Up Environment

- Install Python >=3.8.10
- Install requirements.txt in the python environment
- Set up the directory structure as shown below.

```
└── solution
    ├── src
    ├── results
    ├── data
    |   ├── vesuvius-challenge-ink-detection
    |   |   ├── test
    |   |   ├── train
    |   |   ├── sample_submission.csv
    |   |
    |   ├── pretrained
    |       ├── pvt_v2_b3.pth
    |
    ├── r050_resnet34-unet-mean32-pool-05.sh
    ├── r090-pvtv2-daformer-pool-02a.sh
    ├── hyper-parameters.pdf
    ├── LICENSE
    ├── README.md
```

- The dataset "vesuvius-challenge-ink-detection" can be downloaded from Kaggle: https://www.kaggle.com/competitions/vesuvius-challenge-ink-detection/data
- Pretrained model can be download from PVT (Pyramid Vision Transformer) repository: https://github.com/whai362/PVT/releases/download/v2/pvt_v2_b3.pth

4. Training the model

Warning !!! training output will be overwritten to the "solution/results" folder

- Use the 2 commands to train the weights of the deep neural nets. The bash script will call "run_train.py" with appropriate configure file.

```
usage: python run_train.py <configure>

#pwd = /solution
>> bash ./r050_resnet34-unet-mean32-pool-05.sh
>> bash ./r090-pvtv2-daformer-pool-02a.sh
```

solution-vesuvius-challenge-ink-detection / r050_resnet34-unet-mean32-pool-05.sh

hengck23 Add files via upload

Code Blame 10 lines (9 loc) · 583 Bytes

```
#fold1
python src/r050_resnet34-unet-mean32-pool-05/run_train.py config_fold1_stage1_0
python src/r050_resnet34-unet-mean32-pool-05/run_train.py config_fold1_stage1_1
python src/r050_resnet34-unet-mean32-pool-05/run_train.py config_fold1_stage1_2
python src/r050_resnet34-unet-mean32-pool-05/run_train.py config_fold1_stage2_0

#fold2aa
python src/r050_resnet34-unet-mean32-pool-05/run_train.py config_fold2aa_stage1_0
python src/r050_resnet34-unet-mean32-pool-05/run_train.py config_fold2aa_stage1_2
python src/r050_resnet34-unet-mean32-pool-05/run_train.py config_fold2aa_stage2_0
```

- This will produce the 4 model files used in submission:

```
└── solution
    ├── ...
    ├── results
        ├── r050_resnet34-unet-mean32-pool-05
        |   ├── fold-1
        |   |   ├── stage2_0/checkpoint/00018924.model.pth
        |   |
        |   ├── fold-2aa
        |       ├── stage2_0/checkpoint/00014850.model.pth
        |
        ├── r090-pvtv2-daformer-pool-02a
            ├── fold-1
            |   ├── stage2_0/checkpoint/00029376.model.pth
            |
            ├── fold-2aa
                ├── stage2_0/checkpoint/00009159.model.pth
```

5. Submission notebook

- The public submission notebook (clean version) is at: https://www.kaggle.com/code/hengck23/9th-place-final-ensemble-v1?scriptVersionId=136499959
- The same learned weights from this repository are used in the submission notebook via the public dataset https://www.kaggle.com/datasets/hengck23/ink-weight-05a

  [private / public score] 0.654354 / 0.738625

6. Local validation

- Run the "run_infer_ms.py" python sript for local validation.

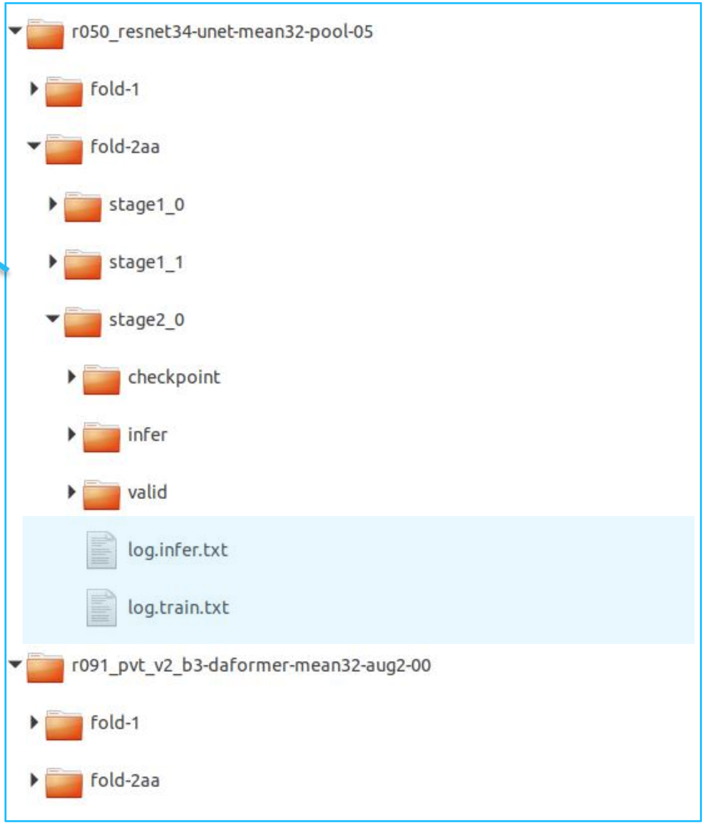
```
usage: python run_infer_ms.py <configure> <model_file>

#pwd = /solution
>> python src/r050_resnet34-unet-mean32-pool-05/run_infer_ms.py config_fold2aa_stage2_0 results/r050
```

7. Reference train and validation results

- Reference results can be found at the google share drive. It includes the weight files, train/validation logs and visualisation images. You can use this to check your results.

  [google drive link (1.8 GB)]



Submission code: Kaggle notebook

https://www.kaggle.com/code/hengck23/9th-place-final-ensemble-v1?scriptVersionId=136499959

9th place : final-ensemble v1

Notebook Input Output Logs Comments (0) Settings

```
else:

    configure = [

        dotdict(
            Net = Pvt2b3MeanPoolDaformer,
            checkpoint = '/kaggle/input/ink-weight-05a/fold1-Pvt2b3MeanPoolDaformer-00029376.model.pth',
            batch_size = 4,
            z0 = 8,
            z1 = 24,
            crop_size = 384,
            crop_fade = 32,
            stride = 192,
            amp=False,
            tta_scale = [1, 1.20, 0.80],
            tta_rot = True,
            enabled = True,
        ),
        dotdict(
            Net = Pvt2b3MeanPoolDaformer,
            checkpoint = '/kaggle/input/ink-weight-05a/fold2aa-Pvt2b3MeanPoolDaformer-00009159.model.pth',
            batch_size = 4,
            z0 = 8,
            z1 = 24,
            crop_size = 384,
            crop_fade = 32,
            stride = 192,
            amp=False,
            tta_scale = [1, 1.20, 0.80]
```

kaggle

kaggle