MICCAI 2024

Marrakesh

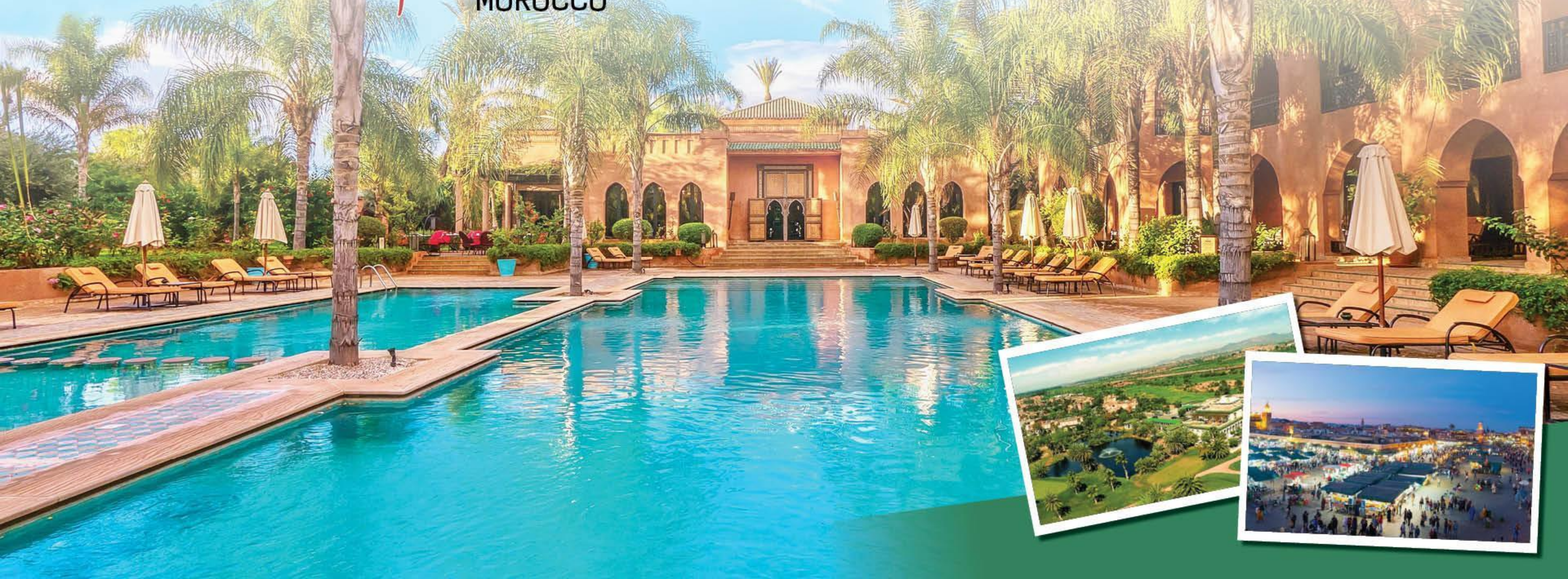
MOROCCO

27TH INTERNATIONAL CONFERENCE ON MEDICAL IMAGE COMPUTING AND COMPUTER ASSISTED INTERVENTION

6-10 OCTOBER 2024

PALMERAIE ROTANA RESORT

MARRAKESH / MOROCCO



A Self-Training Pipeline for Semi-Supervised 2D Teeth Instance Segmentation

Team: camerart2024

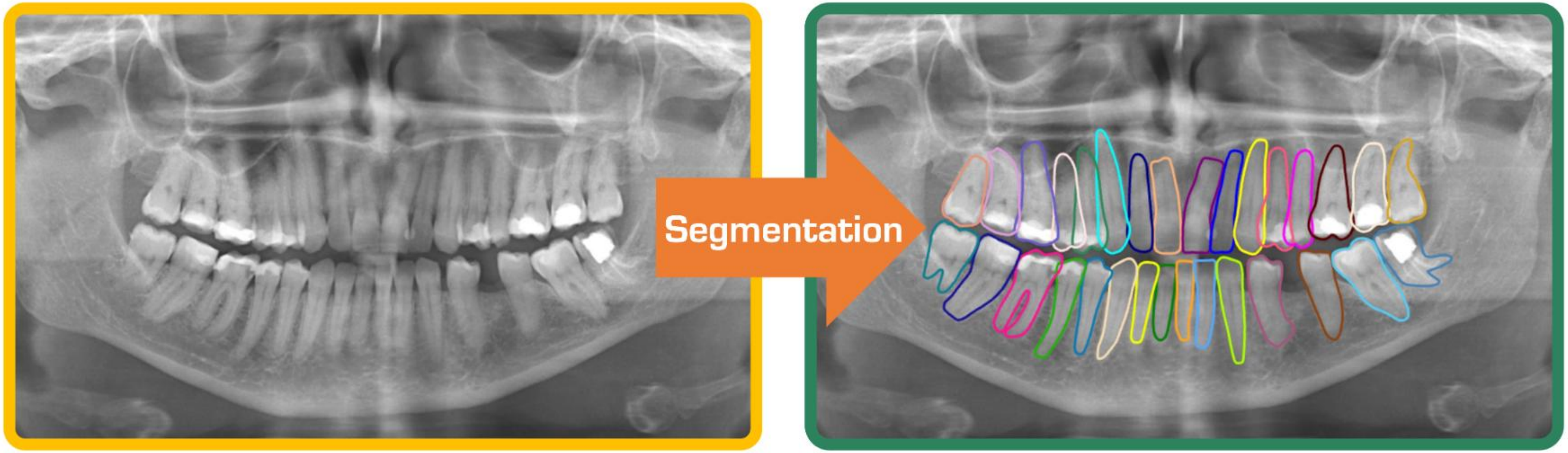
Members: Kaiwen Fu, Chengyuan Chang, Qinjie Hu, Jiahui Chen

Supervisor: Fei Qi

Affiliation: Xidian University

2024/10/06

Background

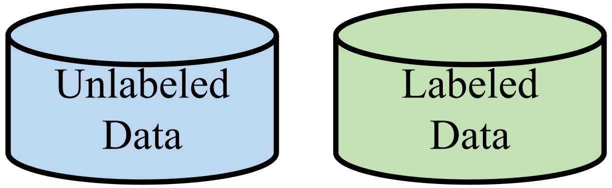


- The 2D panoramic X-ray image is an efficient ways for dentists to determine invisible caries, impacted teeth and supernumerary teeth among children.
- Identifying teeth in panoramic X-ray images and manual annotation is time-consuming and labor-intensive, resulting in a limited availability of labeled data.
- Semi-supervised learning provides a potential alternative for exploring useful information from unlabeled samples.

Problem analysis

The main challenges of the competition are as follows:

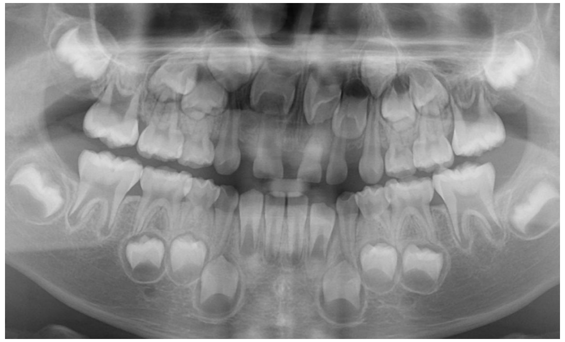
- Very limited labled data



×2,350 ×30

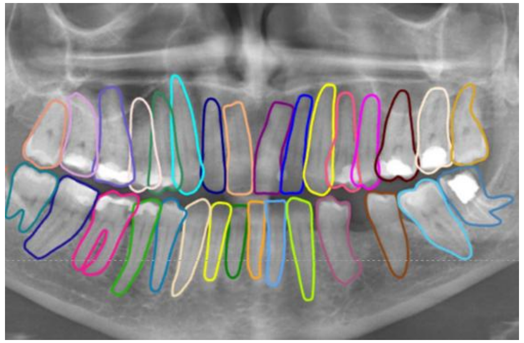
Only 30 labled data are available

- Excessive dental categories



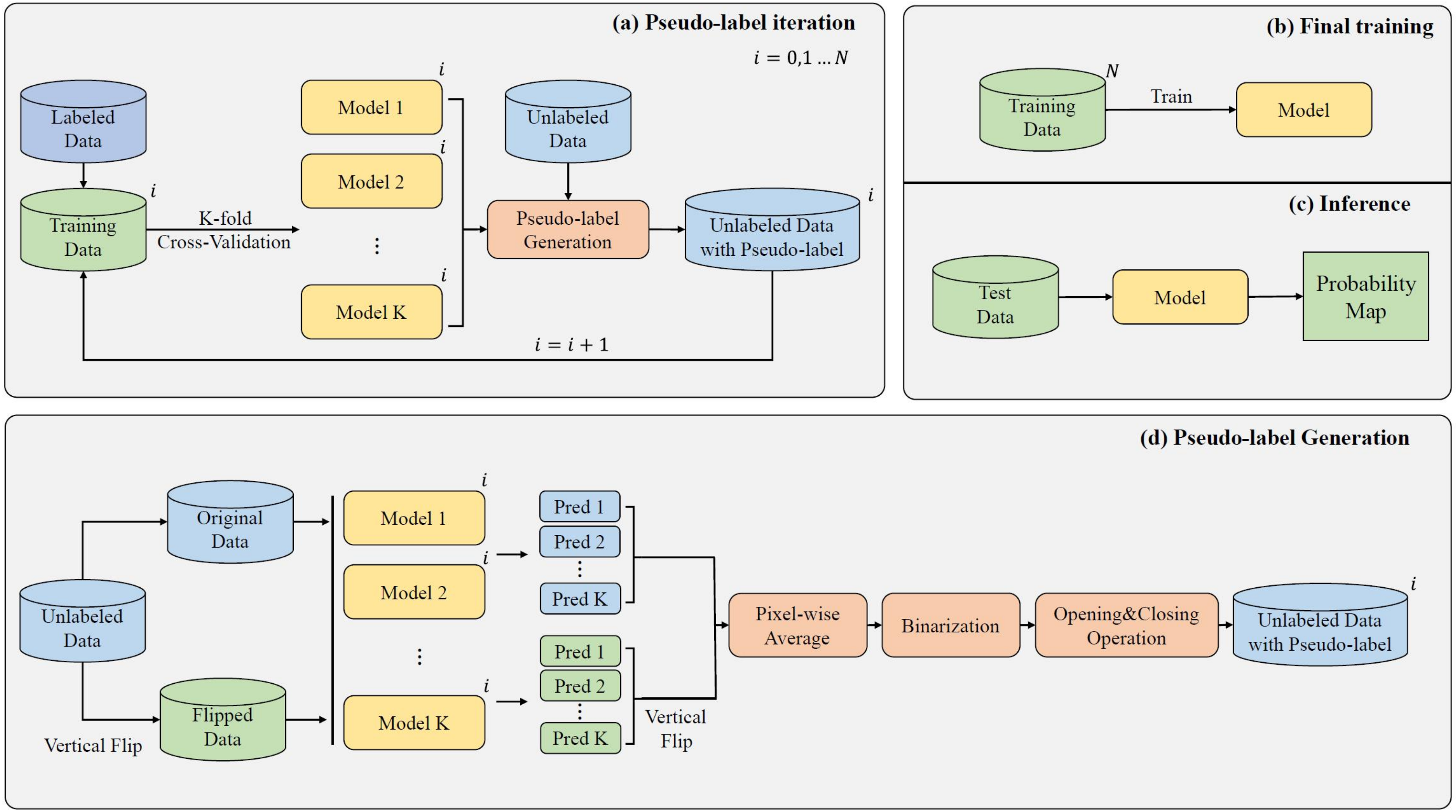
32 permanent teeth and 20 deciduous teeth

- Overlapping area



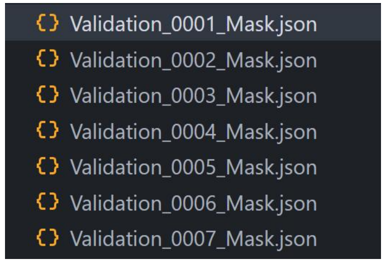
Many teeth overlap each other

Self-Training pipeline

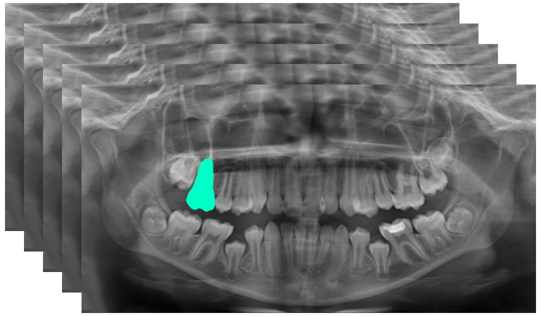


Preprocessing

- Label format conversion



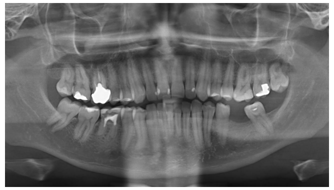
JSON



Ndarray(52 × H × W)

Convert JSON to Ndarray mask, where each channel represents a specific type of tooth

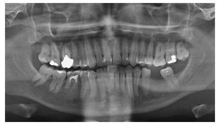
- Resize



1127 × 1991

942 × 2000

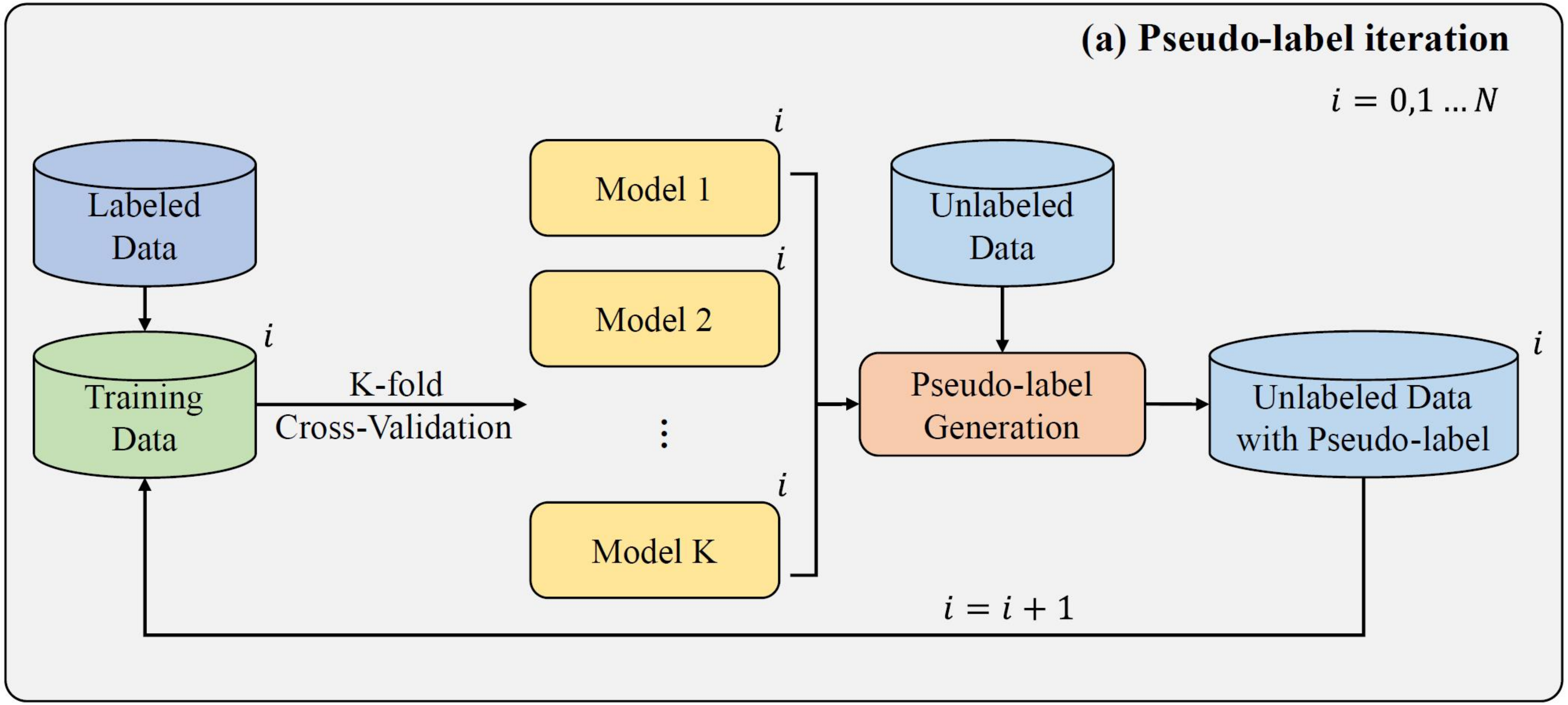
Resize



320 × 640

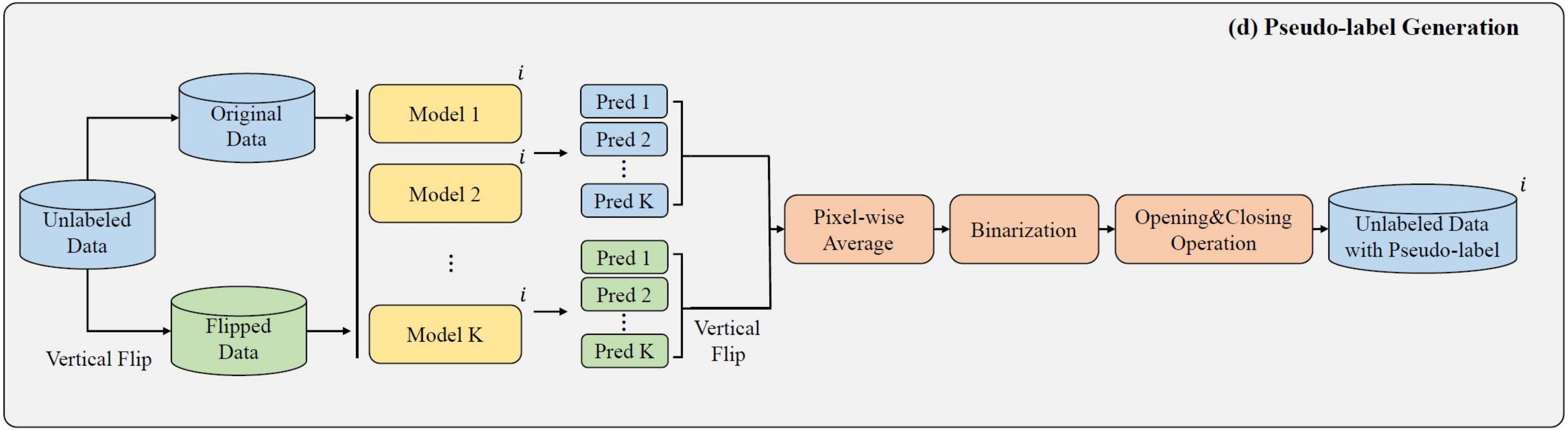
Resize both the image and mask to 320x640 to reduce computational load and facilitate subsequent processing

Pseudo-label generation and iteration



- ➢ Train the initial models using only labeled data
- ➢ Use initial models for pseudo-label generation on unlabeled data and add to training set
- ➢ Re-train the models, update pseudo-labels, repeat N times

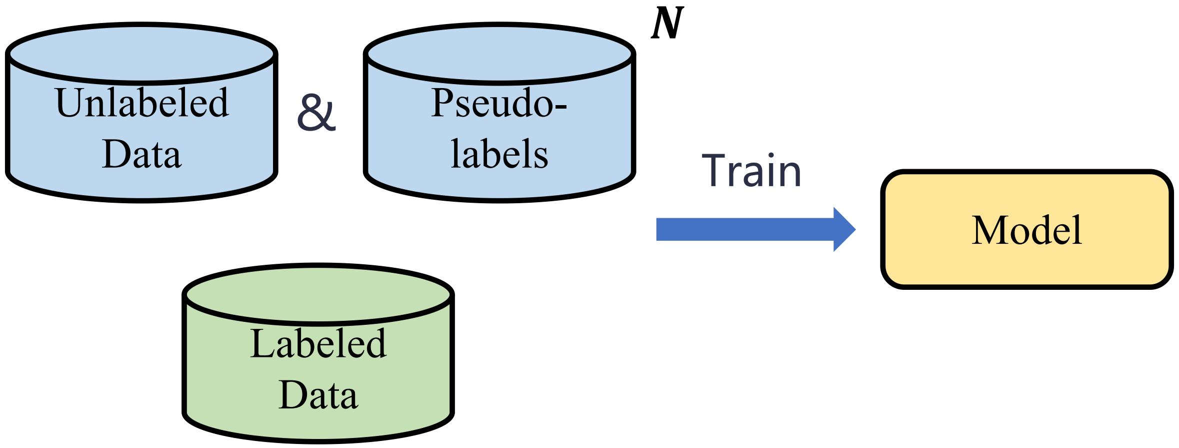
Pseudo-label generation strategy

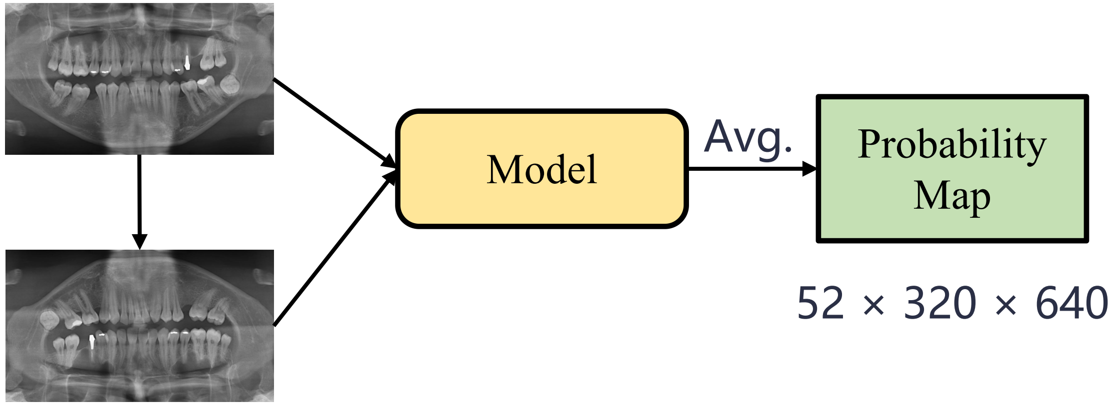


We employed the following strategies to generate more reliable pseudo-labels:

- Model ensemble: the inference result takes the mean of K model outputs (K=5)
- Test-time augmentation: an image is inferred twice, the original image and the vertically flipped image
- Morphological operation: remove small area noise using opening and closing operations

Final training & Inference



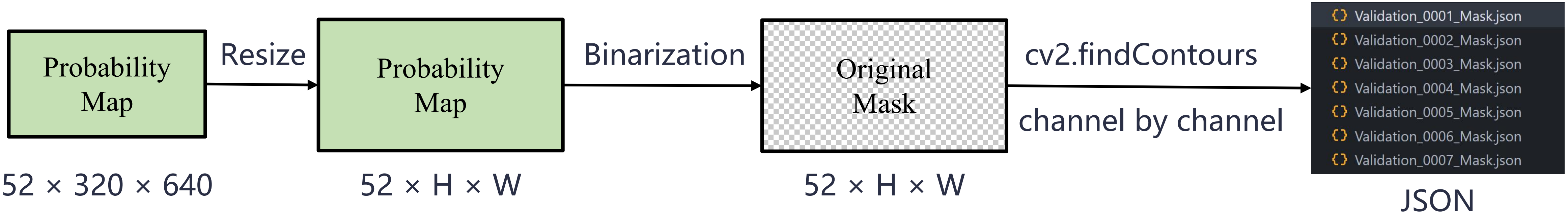


Vertical flip

Training a single model with labeled data, unlabeled data, and pseudo-labels iterated N times

The same **test-time augmentation** is used in inference as in pseudo-label generation

Post-processing



- Resize the image to its original size
- Binarization with a threshold of 0.5
- Find connected regions channel by channel using cv2.findContours
  (PS: We exclude regions with contour points fewer than 45 to eliminate small area noise)
- Remap channel numbering to tooth numbering

Implementation details

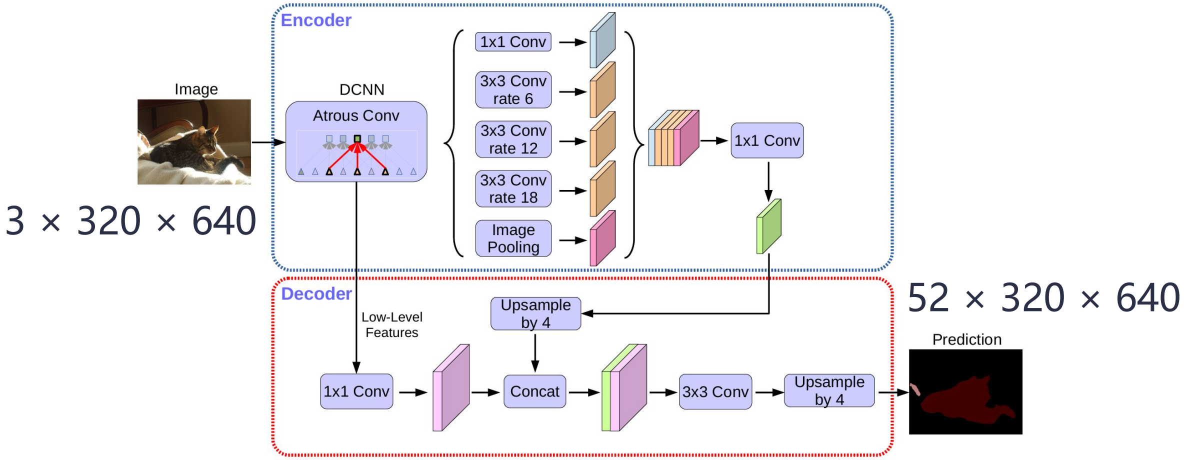
Table 1. Development environments and requirements.

System	Ubuntu 20.04.6 LTS
CPU	Intel(R) Xeon(R) Gold 6326 CPU@2.90GHz
RAM	8 × 32GB; 3200MT/s
GPU (number and type)	1 × NVIDIA RTX 4090 24G
CUDA version	11.8
Programming language	Python 3.8.11
Deep learning framework	Torch 1.13.1, Torchvision 0.14.1
Specific dependencies	Opencv-Python 4.6.0
Code	VSCode, XShell 7

Table 2. Training protocols and hyperparameter settings.

Network initialization	He initialization
Batch size	4
Total epochs	200
Optimizer	AdamW [6]
Betas	(0.9, 0.999)
Weight decay	0.01
Initial learning rate (lr)	0.001
Lr decay schedule	CosineAnnealingLR
Loss function	Dice and BCE
Number of model parameters	26.69M
Number of flops	29.11G

Segmentation model: DeepLabv3+ Backbone：ResNet50



Data augmentation used include:

- Vertical flipping
- Random gamma adjustment
- Brightness and contrast enhancement
- Blurring
- Optical distortion
- Elastic transformation
- Grid distortion
- Motion blur
- Hue saturation adjustments

For more details: https://github.com/Liaaaar/2024-MICCAI-STS-2D

Loss function selection

Table 4. The impact of the loss function when training solely with labeled data.

Loss function	image-level			instance-level				Avg.(%)
	Dice(%)	IoU(%)	NSD(%)	Dice(%)	IoU(%)	NSD(%)	IA(%)	
Dice	88.05	78.82	91.39	61.72	66.35	79.06	59.15	74.93
0.5Dice+0.5BCE	88.41	79.35	91.83	60.18	64.98	77.69	72.34	76.40
0.2Dice+0.8BCE	87.86	78.49	91.36	**69.09**	64.71	77.98	72.14	**77.38**
BCE	**89.22**	**80.69**	**92.76**	46.36	**67.69**	**80.07**	**73.80**	75.80

We first experimented using only labeled data to determine the loss function during self-training

0.2 × dice loss + 0.8 × BCE loss performs the best

Quantitative results on validation set

Table 3. Quantitative results on validation sets. Supervised denotes training conducted solely with labeled data, while Ours refers to our self-training pipeline, where N represents the number of iterations for pseudo-labels.

Method	image-level			instance-level				Avg.(%)
	Dice(%)	IoU(%)	NSD(%)	Dice(%)	IoU(%)	NSD(%)	IA(%)	
Supervised	87.86	78.49	91.36	69.09	64.71	77.98	72.14	77.38
Ours (N=0)	88.65	79.77	92.05	69.42	67.72	80.68	77.47	79.40
Ours (N=1)	88.67	79.80	92.03	73.77	67.65	80.53	76.72	79.88
Ours (N=2)	89.41	80.93	92.95	**77.80**	68.58	81.65	78.80	81.45
Ours (N=3)	**89.57**	**81.19**	**93.12**	77.54	**69.02**	**82.05**	**79.36**	**81.70**

Our self-training pipeline leverages information from unlabeled data to improve the model performance

As the number of pseudo-labels iteration increases, the performance of the model can be further improved

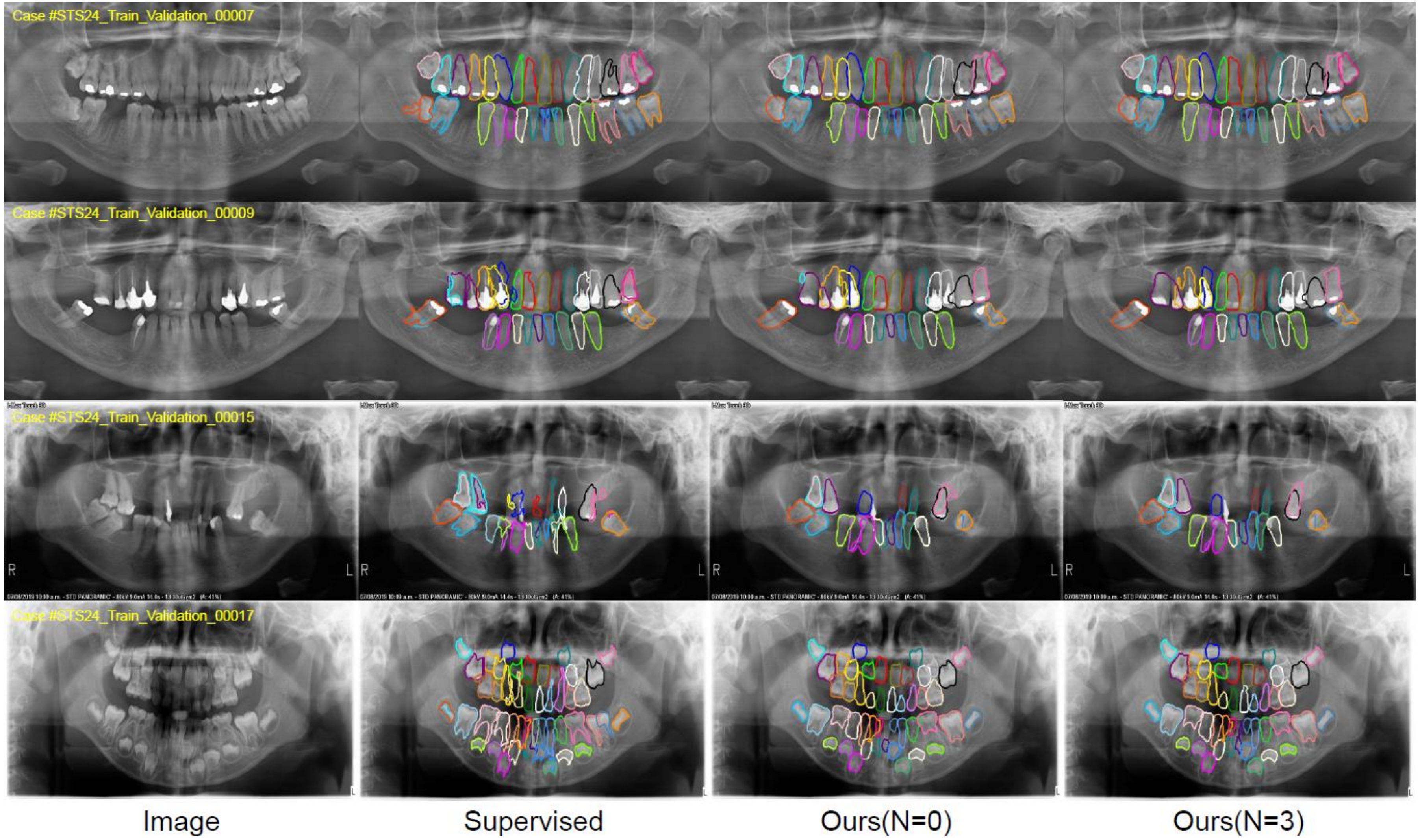
Qualitative results on validation set

Easy cases

- ➢ Straight teeth
- ➢ Many similar patterns in the training set

Hard cases

- ➢ Misaligned teeth
- ➢ Few similar patterns in the training set



Results on testing set & Rank

Final Rank

指标\队伍名称	ChohoTech	camerart2024	jichangkai	dew123	junqiangmler	isjinghao	lazyman	caiyichen	guo77777	cccc2024
Dice_instance	0.845	0.703	0.801	0.224	0.368	0.750	0.385	0.530	0.324	0.319
Rank	1	4	2	10	7	3	6	5	8	9
Dice_image	0.918	0.869	0.915	0.844	0.826	0.826	0.597	0.820	0.755	0.651
Rank	1	3	2	4	5	6	10	7	8	9
NSD_instance	0.872	0.740	0.817	0.699	0.678	0.676	0.725	0.566	0.386	0.218
Rank	1	3	2	5	6	7	4	8	9	10
NSD_image	0.956	0.907	0.944	0.886	0.867	0.863	0.870	0.855	0.803	0.679
Rank	1	3	2	4	6	7	5	8	9	10
mIoU_instance	0.765	0.613	0.734	0.574	0.545	0.587	0.347	0.492	0.285	0.192
Rank	1	3	2	5	6	4	8	7	9	10
mIoU_image	0.849	0.771	0.859	0.736	0.713	0.730	0.430	0.703	0.614	0.492
Rank	2	3	1	4	6	5	10	7	8	9
Identification Accuracy	0.883	0.734	0.832	0.658	0.552	0.698	0.086	0.574	0.279	0.244
Rank	1	3	2	5	7	4	10	6	8	9
Time	13.291	13.274	55.897	13.906	14.047	21.134	11.810	19.531	18.421	13.462
Rank	3	2	10	5	6	9	1	8	7	4
GPU_Consumption	7341.120	14250.980	25461.260	15088.500	12483.380	27987.900	13910.320	26666.571	19694.260	17730.480
Rank	1	4	8	5	2	10	3	9	7	6
AVG	1.333	3.111	3.444	5.222	5.667	6.111	6.333	7.222	8.111	8.444
Final Rank	1	2	3	4	5	6	7	8	9	10

Thanks for listening!

https://github.com/Liaaaar/2024-MICCAI-STS-2D