kaggle

Kaggle Winner Presentation - Heng's part (7t

RSNA 2024 Lumbar Spine Degenerative Classification

https://www.kaggle.com/competitions/rsna-2024-lumbar-spine-degenerative-classification)

HLIP/HENG CHER KENG

https://www.kaggle.com/hengck23

Agenda

1. Background
2. Summary
3. Feature selection & engineering
4. Training methods
5. Important findings
6. Simple model
7. Code review

Background

- Contract computer vision and deep learning algorithm engineer.
  - identify fracture in x-ray images
  - implement visual slam for robotic navigation
  - finetune LLM models

- Familiar with deep learning and build deep models in my work.
- Experiences previous Kaggle competitions. Computer vision and medicine applications:
  - RSNA 2023 Abdominal Trauma Detection
  - RSNA 2022 Cervical Spine Fracture Detection

In this competition, our task is to predict lumbar spine degenerative conditions from given 3d volume MRI scans. Since my teammate @lihaoweicvch already has very good two-stage model, my task is to design one-stage model to improve his results.

While two-stage model uses "crop and classify" method, one-stage models uses "point-masking and pool" (see Figure.2 and 3 in next slides)

1. Predict the 5 lumbar level point pixel-wise heatmap for the L1/L2 to L5/S1. For learning the net parameter, we use the Jensen-Shannon divergence loss function.

2. Convert point pixel-wise heatmap to target xyz coordinates using the differentiable spatial-to-numerical transform (DSNT) from in paper [1].

3. Predict the 3 spine degeneration grade pixel-wise probability for the classes Normal/Mild, Moderate, or Severe.

4. Use the point pixel-wise heatmap to pool the grade pixel-wise probability using multiplication and summing over volume or area. The pooled results is the target level-wise grade conditions for submission,

[1] "Numerical Coordinate Regression with Convolutional Neural Networks" - Aiden Nibali, Arvix 2018, https://arxiv.org/abs/1801.07372

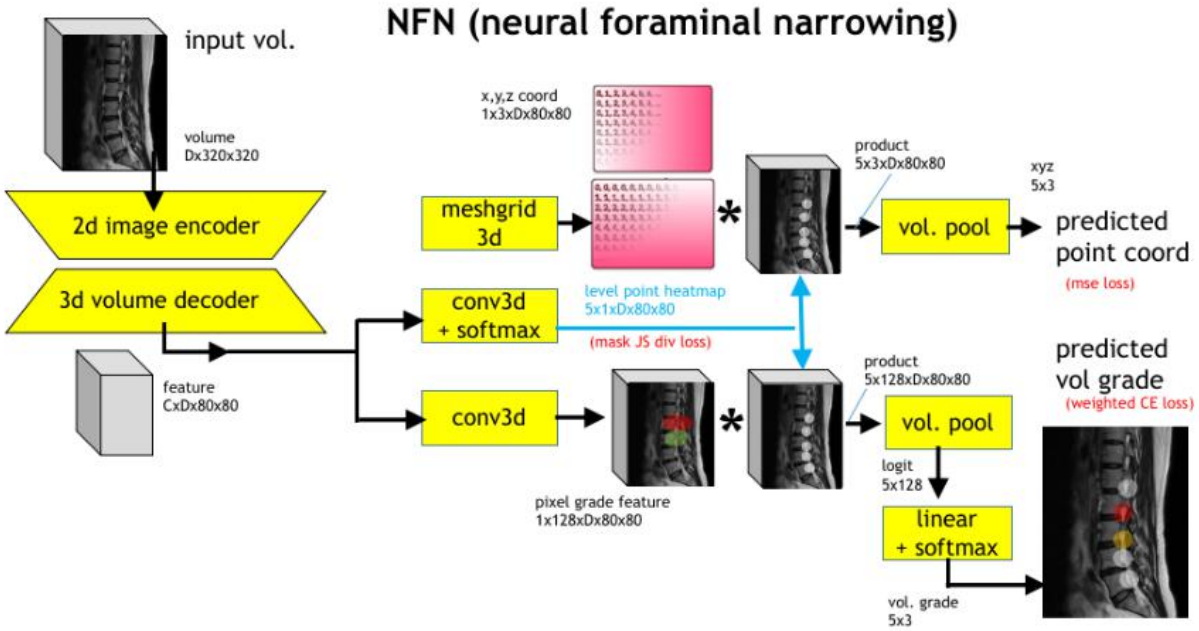

Figure.1 Deep net architecture for NFN (neural foraminal narrowing) detection

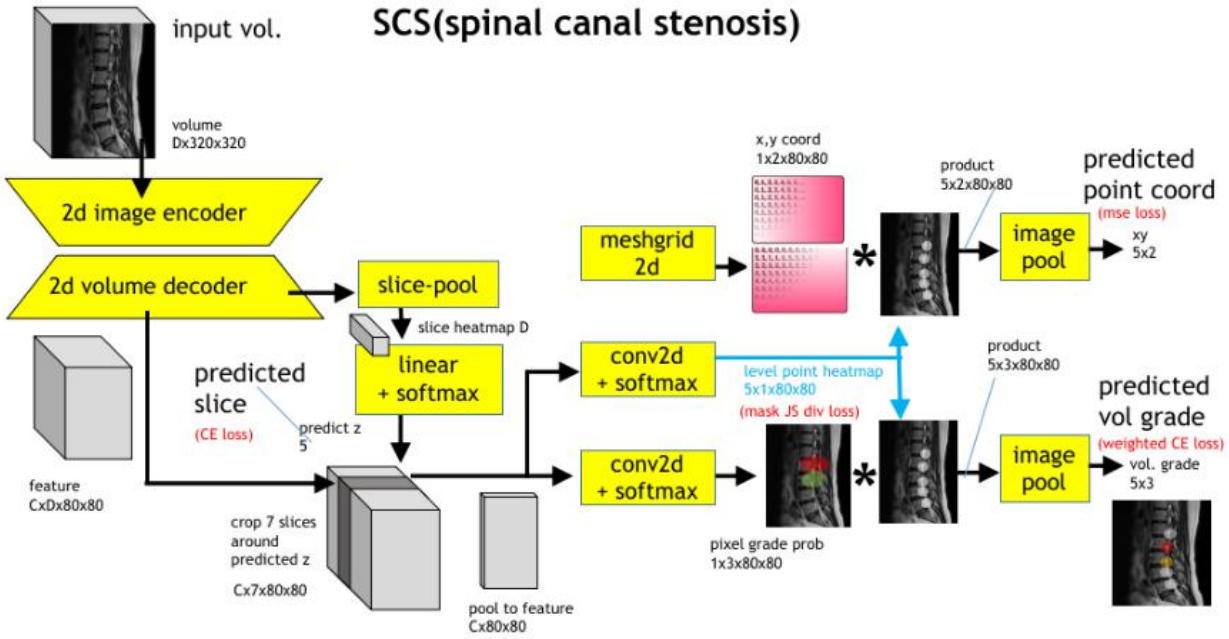

Figure.2 Deep net architecture for SCS (spinal canal stenosis) detection

Features Selection/ Engineering

1. Image encoder

We are using pyramid transformer PVTv2 [2] as image encoder backbone for submission. They have very large context which we think are important to get good results. For example, both left and right level are likely to have the same spine degeneration conditions.

2. Point-masking and pooling

The level point heatmap can be seen as learned attention map to select with pixel we should look at for spine degeneration conditions.

[1] "PVT v2: Improved Baselines with Pyramid Vision Transformer" - Wenhai Wanghttps://arxiv.org/abs/2106.13797

Training Methods

Training is straight forward with multi-task loss setup for each net in Figure.1 and 2:

- pixel-wise level point : Jensen-Shannon divergence loss
- pixel-wise grade probability : Cross entropy loss
- volume-wise level grade probability : weighted cross entropy loss

We perform back propagation with ADAM optimizer. Training hyper-parameters
  - learning rate is from 1e-4 to 1e-5
  - training epochs ranges from 50 to 100.
  - batch sizes is from 3 to 8 volumes.

For hardware, we use one Nvidia Ada A6000/48GB Ampere GPU. Powerful GPU makes our training fast and large memory enables us to train with whole 3d volume.

Some observations from training are:

1. Upper bound for grade prediction performance.
   - if we use the ground truth level point heatmap and just learn grade prediction, this gives upper bound for grade prediction performance.

2. One-stage networks are challenging to train because the optimal parameters for coordinate prediction are not necessarily the best for grade prediction.
   - confirmed by observing the loss for level and grade during training iterations. They seem to be competing with each other.

We believe one reason for this is labeling errors, particularly the significant confusion between L5 and S1 level labels. Additionally, the point labels themselves are not very precise.

Important and Interesting Findings

sagittal-t1 (single stage joint grade and point prediction)

NFN	grade				
	bugged #1	fixed			
	320	320	320	384	
	pvtv2-b4	pvtv2-b4	convnext-	effnet-b5	ensemble
fold0	0.487	**0.473**	0.4777	0.4860	0.4664
fold1	0.524	**0.504**	0.5003	0.4960	0.4848
fold2	0.507	**0.500**	0.4950	0.4802	0.4802
fold3	0.521	**0.520**	0.5182	0.5116	0.5014
fold4	0.468	**0.465**	0.4640	0.4641	0.4591
avg	0.501	**0.4923**	**0.4910**	**0.4876**	**0.4784**

	ensemble							
	x<1	x<2	x<5	y<1	y<2	y<5	z<1	z<5
fold0	0.9673	0.9894	0.9998	0.8963	0.9560	0.9771	0.9747	1.0000
fold1	0.9598	0.9847	0.9992	0.8829	0.9537	0.9729	0.9698	0.9987
fold2	0.9693	0.9892	0.9992	0.9054	0.9644	0.9798	0.9706	0.9973
fold3	0.9702	0.9948	1.0000	0.9036	0.9687	0.9876	0.9715	0.9948
fold4	0.9676	0.9903	0.9998	0.9030	0.9656	0.9812	0.9718	0.9973
avg								

with tta (flip left-right condition)

	320	320	384	
	pvtv2-b4	convnext-	effnet-b5	ensemble
fold0	0.4710	0.4743	0.4835	0.4663
fold1	0.5041	0.4992	0.4910	0.4852
fold2	0.4988	0.4939	0.4822	0.4782
fold3	0.5159	0.5166	0.5095	0.5018
fold4	0.4648	0.4603	0.4591	0.4467
	0.4909	**0.4889**	**0.4851**	**0.4757**

Note:
- x,y err is normalised at feature size 80x80
- high threshold (x,y<5) error could be due to label error, e.g. L5/S1 confusion
- z corresponds to dicom slice instance numer

- #1 bug: we forget to reorder the point in left-right flipping augmentation at training. This is discover at last day before submission deadline.
- gray cell: used in submission to get final ranking in private LB

Figure.3 Local validation for NFN (neural foraminal narrowing) detection

We had a bug in flip augmentation at train. We forgot to reorder the left right level points (i.e. at training the left is mistakenly labeled as right and vice versus for 50% of the time). We discovered the bug just one day before submission and was unable to correct all models for submission. Hence the final submission consisted of both bugged and unbugged models.

We later make post submission after the competition deadline with unbugged models only and showed better results.

sagittal-t2 (single stage joint grade and point prediction)

SCS	grade					ensemble							
	320	320	384										
	pvtv2-b4	convnext_	effnet-b4	ensemble		x<1	x<2	x<4	y<1	y<2	y<4	z<1	z<3
fold0	0.2443	0.2498	0.2447	0.2376	fold0	0.9721	0.9980	0.9990	0.9477	0.9775	0.9858	0.9941	0.9995
fold1	0.2465	0.2520	0.2741	0.2483	fold1	0.9661	0.9929	1.0000	0.9256	0.9722	0.9833	0.9843	0.9965
fold2	0.2281	0.2233	0.2406	0.2209	fold2	0.9754	0.9936	0.9995	0.9406	0.9770	0.9888	0.9872	1.0000
fold3	0.3317	0.3244	0.3239	0.3160	fold3	0.9683	0.9934	1.0000	0.9371	0.9760	0.9887	0.9882	0.9995
fold4	0.2949	0.2866	0.2876	0.2807	fold4	0.9787	0.9985	0.9995	0.9485	0.9847	0.9921	0.9906	0.9990
avg	0.2691	0.2672	0.2742	0.2607	avg	0.9721	0.9953	0.9996	0.9399	0.9775	0.9877	0.9889	0.9989

	any severe			
	320	320	384	
	pvtv2-b4	convnext_	effnet-b4	ensemble
fold0	0.1877	0.1926	0.2001	0.1877
fold1	0.2162	0.2101	0.2392	0.2191
fold2	0.1788	0.1943	0.2001	0.1842
fold3	0.3182	0.2863	0.3078	0.3008
fold4	0.2971	0.3244	0.3088	0.3019
avg	0.2396	0.2416	0.2512	0.2387
		grade + any severe		0.2497

Figure.4 Local validation for SCS (spinal canal stenosis) detection

submission		
2stage : @lhw base model : 2 stage crop and classify		
	private	public lb
2stage	0.4046	0.3500
2stage + 1stage(nfn: pvtv2-v4-bugged)	0.4021	0.3481
2stage + 1stage(nfn: pvtv2-v4-bugged + two fixed)	0.4013	0.3467 (7-th rank)
post submission		
2stage + 1stage(nfn: pvtv2-v4-fixed)	0.4011	0.3478
2stage + 1stage(nfn: ensemble)	0.4000	0.3466

Figure.5 Public and private leader board submission scores

One-stage NFN model indeed improves results. Blending is: 0.75*2stage + 0.25*1stage

1. The baseline two stage has private /public score of 0.406/0.3500.

2. Even with bugged models only, the improvement is 0.402/0.3481.

3. For unbugged models, we have the best score of 0.400/0.3466.

4. One stage SCS models did not improve results and are not included for final submission. The reasons are unclear. When alone, one stage SCS model are have comparable accuracy as the two-stage counterparts. However after ensemble, the combined results falls a little.

Simple Model

From our report it is sufficient just to use only two stage mode.There will be drop of 5% accuracy though.

Code Review

https://github.com/hengck23/solution-rsna-2024-lumbar-spine

Kaggle Competition Solution (7th)

RSNA 2024 Lumbar Spine Degenerative Classification

- Classify lumbar spine degenerative conditions
  https://www.kaggle.com/competitions/rsna-2024-lumbar-spine-degenerative-classification
- For discussion, please refer to:
  https://www.kaggle.com/competitions/rsna-2024-lumbar-spine-degenerative-classification/discussion/539439

This repo contains the code to train one-stage models used in the team solution for the RSNA 2024 lumbar spine degenerative classification Kaggle competition.

1. Hardware

- GPU: 2x Nvidia Ada A6000 (Ampere), each with VRAM 48 GB
- CPU: Intel® Xeon(R) w7-3455 CPU @ 2.5GHz, 24 cores, 48 threads
- Memory: 256 GB RAM

2. OS

- ubuntu 22.04.4 LTS

3. Set Up Environment

- Install Python >=3.10.9
- Install requirements.txt in the python environment
- Set up the directory structure as shown below.

```
└── <repo_dir>
    ├── <DATA_KAGGLE_DIR>
    |         ├── rsna-2024-lumbar-spine-degenerative-classification
    |               ├── test_images
    |               ├── train_images
    |               ├── train.csv
    |               ├── train_label_coordinates.csv
    |               ├── train_series_descriptions.csv
    |               ├── ... other files ...
    ├── <DATA_PROCESSED_DIR>
    |         ├── train_label_coordinates.fix01b.csv
    |         ├── nfn_sag_t1_mean_shape.512.npy
    |         ├── scs_sag_t2_mean.512.npy
    |
    ├── <RESULT_DIR>
    ├── src
    ├── LICENSE
    ├── README.md
    ├── requirements.txt
```

- Modify the path setting by editing "/src/third_party/_dir_setting_.py"

```
# please use the full path
DATA_KAGGLE_DIR     = '... for downloaded and unzipped Kaggle data ... '
DATA_PROCESSED_DIR  = '... for intermediate processed data ...'
RESULT_DIR          = '... for training output like model weights, training logs, etc ...'
```

4. Set Up Dataset

- <DATA_KAGGLE_DIR> contains data from "rsna-2024-lumbar-spine-degenerative-classification.zip" at: https://www.kaggle.com/competitions/rsna-2024-lumbar-spine-degenerative-classification/data
- <DATA_PROCESSED_DIR> contains the 3 manually created files from the DATA_PROCESSED_DIR folder in this repo.
    - train_label_coordinates.fix01b.csv: correct annotation for spinal canal stenosis point
    - nfn_sag_t1_mean_shape.512.npy, scs_sag_t2_mean.512.npy: mean reference shape created from https://www.kaggle.com/code/hengck23/shape-alignment
    - Other processed data can be created by running the python script:

```
python "/src/process-data-01/run_make_data.py"
```

    - A backup copy of all processed data can be found in the google-share drive: https://drive.google.com/drive/folders/1jPPxAP6DHGQMHJPUGjPO7_Q5Asrj_LL3?usp=sharing

5. Training the model

Warning !!! training output will be overwritten to the "<RESULT_DIR>" folder

NFN (neural foraminal narrowing) models

- Bugged Models: Due to a bug in the flip augmentation (left-right points not reordered) in training, the submitted model weights are not correct. You can reproduce the bugged models by running the following script:

```
cd src/nfn_trainer_bugged
python run_train_nfn_pvtv2_b4_bugged.py

output model:
- <RESULT_DIR>/one-stage-nfn-bugged/pvt_v2_b4-decoder3d-01/
```

SCS (spinal canal stenosis) models

- Optional Models: These are not used in submission as one-stage SCS models did not improve the public lb score when added to the team solution.

```
cd src/scs_trainer
python run_train_scs_pvtv2_b4_fixed.py
python run_train_scs_covnext_base.py
python run_train_scs_effnet_b3.py

output model:
- <RESULT_DIR>/one-stage-scs/pvt_v2_b4-decoder2d-01/
- <RESULT_DIR>/one-stage-scs/convnext_base-decoder2d-01/
- <RESULT_DIR>/one-stage-scs/effnet_b4-decoder2d-01/
```

- Similarly, if you want to ensemble and perform local validation, run the script:

```
cd src/scs_trainer
python run_ensemble_and_local_validation.py
```

5. Submission csv

Team submission notebook can be found at:
https://www.kaggle.com/code/hengck23/lhw-v24-ensemble-add-heng

✓	[lhw] v24 ensemble + add heng - Version 6 Succeeded · hengck23 · 6d ago · Notebook [lhw] v24 ensemble + add he...	0.401322	0.346725

Team post-submission notebook can be found at:
https://www.kaggle.com/code/hengck23/post-lhw-v24-ensemble-add-heng

✓	post [lhw] v24 ensemble + add heng - Version 6 Succeeded (after deadline) · hengck23 · 1d ago · Notebook post [lhw] v24 ...	0.400018	0.346647

6. Demo

heng's part: https://www.kaggle.com/code/hengck23/clean-final-submit02-scs-nfn-ensemble

7. Reference trained models and validation results

- Reference results can also be found in the shared google drive at :
  /kaggle-submit-rsna-2024-spine/result
  https://drive.google.com/drive/folders/1jPPxAP6DHGQMHJPUGjPO7_Q5Asrj_LL3?usp=sharing
- It includes the weight files and train/validation logs.

Authors

- https://www.kaggle.com/hengck23

License

- This project is licensed under the MIT License - see the LICENSE file for details.

Acknowledgement

"We extend our thanks to HP for providing the Z8 Fury-G5 Data Science Workstation, which empowered our deep learning experiments. The high computational power and large GPU memory enabled us to design our models swiftly."

Question and Answer


kaggle