Reproduction Study: Stepwise Feature Fusion: Local Guides Global by J. Wang et al.

Rahul Singhal
rahuls11@illinois.edu

Group ID: 151
Paper ID: Stepwise Feature Fusion: Local Guides Global by J. Wang et al.
Difficulty: []
Presentation link: YouTube
Code, Data, Weights link: drive.google.com

1 Introduction

Vision Transformers are showing comparative results to CNN based architectures at reduced computational requirements[2]. The authors of this paper use a Pyramid Vision Transformer (PVT) backbone[3], which is a coarse-to-fine multi-headed self-attention architecture for the encoder.

Their central contribution is a Progressive Locality Decoder module, implemented using CNNs. Additionally they introduce the PVT approach to the Polyp Segmentation task.

The Segmentation task would seem to benefit from a combination of Transformers (low-pass filter) and CNNs (high-pass filter), which benefits from pixel perfect accuracy.

2 Scope of reproducibility

This paper introduces a CNN based Decoder architecture that outperforms a MLP Decoder for the Polyp Segmentation task.

Their Progressive Locality Decoder module is implemented using dual CNN layers. The PLD module shows a higher mean Dice or mean IoU compared to an all MLP Decoder (as found in SegFormer).

2.1 Addressed claims from the original paper

The following claims are at the heart of the paper's main contribution.

- PLD-Cat shows greater mDice or mIoU than SeD on Kvasir
- PLD-Cat shows greater mDice or mIoU than SeD on CVC-ClinicDB

3 Methodology

At the onset of this study, the paper was still under review and the code repository was unavailable. However, a PyTorch code repository of various SOTA semantic segmentation methods is available. This was utilized to train the Polyp Datasets and extended to re-implement the Progressive Locality Decoder from the description. The paper included details with respect to experimental design, hyperparameters and GPU requirements.

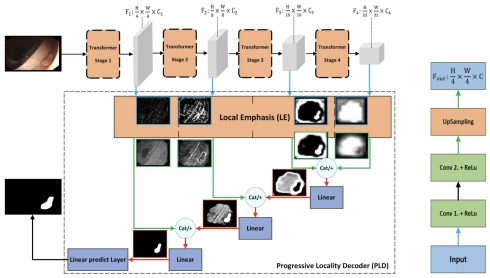
The paper's code repository was found to be available on 5/4/2022 and adjustments were possible to provide a more fair representation of the results. There was one method that could not be replicated in time. Namely, the paper's authors appear to have done their own pre-training on Image, possibly using the papers specific combined loss function. This study also utilized ImageNet pre-trained weights for the SegFormer backbone, though the loss function may have differed.

3.1 Model descriptions

The paper references 3 different Pyramid Vision Transformer backbones: CVT, PVTv2 and MiT. PVTv2 is their primary backbone with the others used in the ablation study.

All 3 backbones share a similar approach to using multiple Transformer blocks in a Course to Fine manner. They all also replace the positional encoding, use an overlapping patch method and employ a different method to reduce the self-attention computational complexity.

The Progressive Locality Decoder (PLD) takes each Transformer feature output as input into a standard CNN with Relu activation. This is done twice for each feature representation and then concatenated with the prior block and passed through a linear layer before being concatenated with the next prior feature representation. See Fig. 1 from the paper.



(a) Overview of SSFormer (b) LE

Fig. 1. (a) is the Overview of SSFormer; (b) is the structure of Local Emphasis module. In this figure, blue indicates unemphasized features, green indicates emphasized features, and red indicates fused features. The remainder of the PLC in Figure (a), excluding the Local Emphasis(LE), is the Stepwise Feature Aggregation(SFA). Feature fusion units can use concatenation(Cat) or addition(+) operations.

The loss is a combination of Dice and Binary Cross Entropy.

Feature Emphasis (Blue Box):

$$F_{le,i} = ReLU(Conv_i(C,C)(ReLU(Conv_i(C_i,C)(F_i)))), \quad (1)$$

Feature Fusion unit (Green Box):

$$F_{i-1,i} = \begin{cases} Liner(2C,C)(Concat(F_{i-1},F_i)), \\ OR, \\ Linear(C,C)(Add(F_{i-1},F_i)), \end{cases} \quad (2)$$

Parameters:

Typical Transformer Encoder parameter count includes: m blocks x [multi-headed attention + layer-norm + feed-forward-network].

PLD Decoder parameter count includes: 4 x [kernel size + 3 x feed-forward-network] + Upsample

Actual model sizes computed via PyTorch

Backbone	Head	Size
MiT-B0	PLD	15.462749
MiT-B0	SeD	14.926081
MiT-B1	PLD	55.315293
MiT-B1	SeD	54.778625
MiT-B2	PLD	114.261413
MiT-B2	SeD	109.518217
MiT-B3	PLD	193.848261
MiT-B3	SeD	189.104937
MiT-B4	PLD	261.020605
MiT-B4	SeD	256.277345
MiT-B5	PLD	343.498715
MiT-B5	SeD	338.755437

3.2 Data descriptions

Two public datasets were used for training in the paper Kavsir-SEG[2] and CVC-ClinicDB[1]. The paper and this reproduction study followed an 80/10/10 split similar to the one described in MSRF-Net. During training images are resized to 352 x 352. Data augmentation includes: random flipping, scaling, rotation, random dilation and erosion.

3.3 Hyperparameters

The paper employs the AdamW optimizer with an initial LR of 1E-4, decay of 0.1 and decaying over a period of 40 epochs with a minimum LR of 1E-4. All experiments use a training period of 200 epochs. Specific experimentation was conducted to confirm the approach of 40 period decay by reviewing the paper's repo code and log files.

3.4 Implementation

A public semantic-segmentation framework `https://github.com/sithu31296/semantic-segmentation` was used to recreate the baseline SegFormer model. The paper's repo `https://github.com/Qiming-Huang/scformer` was used to drop-in specific key components.

The following components appear to be unique implementations are utilized from the paper's public repo and incorporated into the custom code of the semantic-segmentation framework.

- (HEAD) Progressive Locality Decoder
- (Loss Function) DiceLoss
- (Scheduler) PolynomialLRDecay
- (Augmentation) RandomImageEnhance
- (Augmentation) RandomDilationErosion

3.5 Computational requirements

Originally a relative wall time estimate was made using a Google Colab Pro+ subscription, which provides V100 GPU. A benchmark comparison here suggests the V100 45% as fast as the A100 on CNN tasks, so I can expect to be slower, but still fairly reasonable.

More than 57 hrs of compute wall time (12hrs CPU) was dedicated to experimental runs on a V100 GPU. A straight run of 10 models on the CVC-ClinicDB dataset took 8.75hrs to complete.

4 Results

In the first half of the project most time (21hrs) has been spent in identifying a feasible paper to reproduce, re-reading(5x) through this paper, learning about Vision Transformers, as well as reading through the references listed in the paper. Unfortunately coding has been minimal so far.

In the second half of the project time (42hrs) was split between reading code, implementing components and experimentation.

The Progressive Locality Decoder was able to show improved results compared to the SegFormer MLP head in many of the runs, but not all.

4.1 Results

Kvasir-SEG and CVC-ClinicDB datasets were used to reproduce and compare baseline performance with SegFormer backbone (MLP Decoder) with the paper's PLD Decoder. Contact with the paper authors was critical to achieving comparative results.

			Head		
Training Data	*Backbone*	*Values*	PLD	SeD	
CVC-ClinicDB	MiT-B0	mIoU	90.59	90.01	PLD
		mDice	95.06	94.74	PLD
	MiT-B1	mIoU	91.41	90.71	PLD
		mDice	95.51	95.13	PLD
	MiT-B2	mIoU	92.11	91.61	PLD
		mDice	95.89	95.62	PLD
	MiT-B3	mIoU	91.98	92.11	
		mDice	95.82	95.90	
	MiT-B4	mIoU	91.82	92.03	
		mDice	95.74	95.85	
	MiT-B5	mIoU	91.91	92.35	
		mDice	95.79	96.02	
CVC-ClinicDB Total		mIoU	91.64	91.47	PLD
		mDice	95.64	95.54	PLD
Kvasir	MiT-B0	mIoU	84.02	84.77	
		mDice	91.31	91.76	
	MiT-B1	mIoU	84.37	84.91	
		mDice	91.52	91.84	
	MiT-B2	mIoU	85.68	85.38	PLD
		mDice	92.29	92.12	PLD
	MiT-B3	mIoU	86.41	86.30	PLD
		mDice	92.71	92.65	PLD
	MiT-B4	mIoU	85.34	86.04	
		mDice	92.09	92.50	
	MiT-B5	mIoU	86.62	86.59	PLD
		mDice	92.83	92.81	PLD
Kvasir Total		mIoU	85.41	85.67	
		mDice	92.13	92.28	

4.2 Analysis

The Kvasir-SEG dataset is only 42MB with 1000 images+masks with resolution varying from 332x487 to 1920x1072. Multiple computation runs were feasible. CVC-ClinicDB (56MB with 614 images+masks at a resolution of 384x288) was also added for an additional comparison.

4.3 Plan

Train baseline model (MiT backbone and SegFormer head) with Kvasir-SEG dataset. After metrics are confirmed for both datasets, then implement a custom head for PLD and compare metrics. With time permitting, additional datasets and training regiments can be added for comparison.

This plan was executed.

4.4 Additional results not present in the original paper

A multi-size Backbone comparison was conducted.

5 Discussion

The results table shows that the PLD Decoder outperforms for smaller backbones on the CVC-ClinicDB dataset and larger backbones for the Kvasir dataset. This may be function of differences in the dataset more so than the model.

The PLD Head does show improvement over the MLP Decoder. I suspect my implementation missed some detail that prevented consistent improvement in every category. Results are also comparative at an absolute level with the original paper.

5.1 What was easy

Acquiring and downloading datasets as they were already public was straightforward. It was fortunate that a public framework for standing up semantic segmentation pipelines exists. Dropping in certain components from the paper's code repository was also straightforward.

5.2 What was difficult

Grokking half dozen papers and background knowledge on Vision Transformers, Pyramid Vision Transformers took considerable time. Reading through 2 code repositories was non-trivial. Adjusting the semantic segmentation framework for medical polyp segmentation took effort. Training a full size model due to GPU RAM constraints, but a workaround solution was found through PyTorch Automatic Mixed Precision allowed the largest models to be computed. Correctly identifying hyper-parameters and proper data augmentation sequence before having access to the code was a challenge.

5.3 Recommendations for reproducibility

For the paper authors, besides the public code repository, I would recommend straightforward guidance on usage of hyper-parameters and providing pre-trained weights (if not already public).

6 Communication with original authors

Initial contact was made with author Sifan Song, who confirmed that code was not yet public due to the paper still being under review. A follow

up communication was sent regarding scope of reproduction to which there was no immediate comment. Another communication was made to clarify data augmentation parameters and loss function to which Sifan Song replied that the first author would reply.

First author, Jinfeng Wang, replied back with clarification along with news that the code would be made public. The public repo was found on May 4th, 2022 which contains code, config files and log files, but no pre-trained weights.

References

[1] Bernal, J., Sánchez, F. J., Fernández-Esparrach, G., Gil, D., Rodríguez, C., Vilariño, F. (2015). WM-DOVA maps for accurate polyp highlighting in colonoscopy: Validation vs. saliency maps from physicians. Computerized Medical Imaging and Graphics, 43, 99-111.

[2] D. Jha, P. H. Smedsrud, M. A. Riegler, P. Halvorsen, T. d. Lange, D. Johansen, and H. D. Johansen, "Kvasir-seg: A segmented polyp dataset," in International Conference on Multimedia Modeling. Springer, 2020, pp. 451–462.