5/2023

SKIN ABNORMAL DETECTION

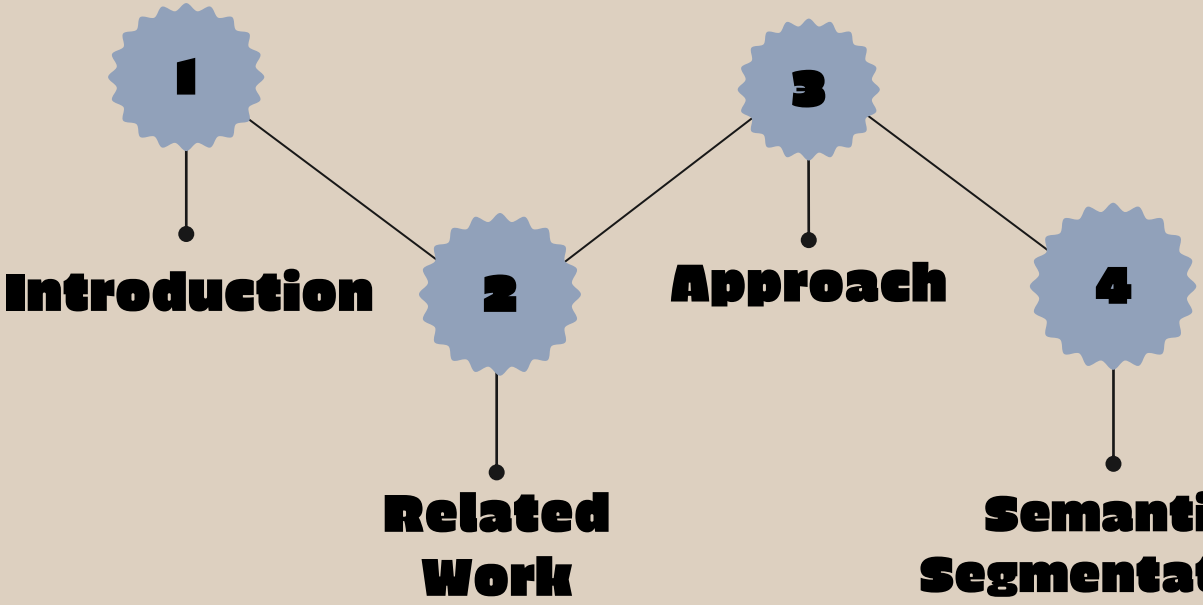
With MMsegmentation

Sinh viên thực hiện:

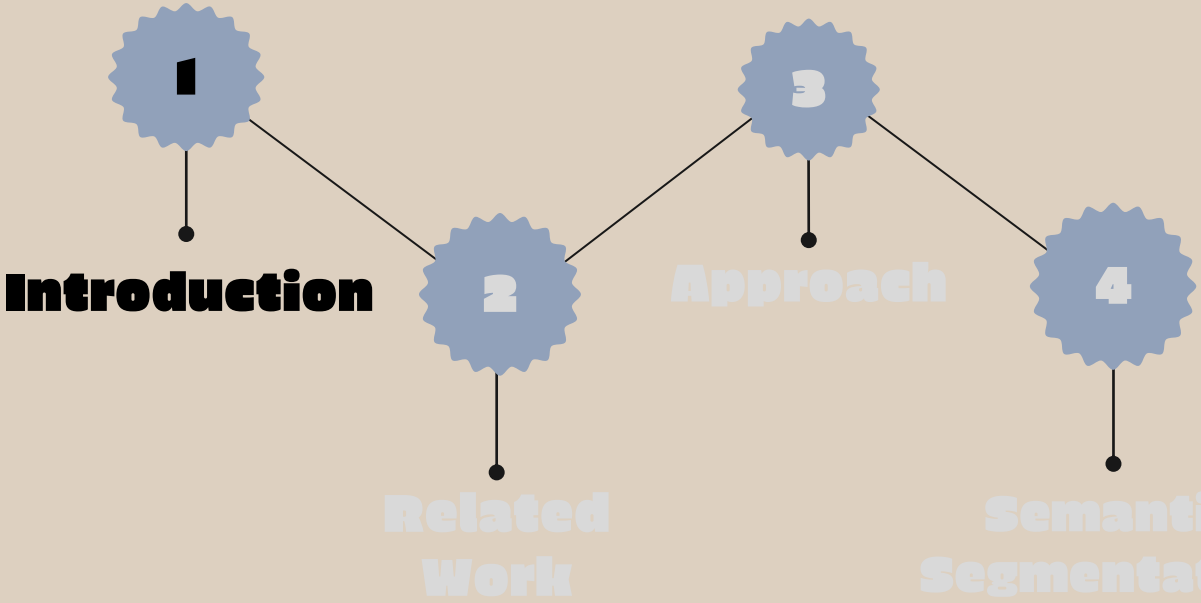
Võ Văn Khánh	20145322
Trần Minh Khôi	20139079
Trần Quốc Huy	20139076
Nguyễn Đình Đức Thọ	20139092
Nguyễn An Minh Triết	20139093
Nguyễn Viết Lương	20139081
Vũ Quang Huy	20139077

Giảng viên hướng dẫn:
TS Nguyễn Mạnh Hùng

Object-Contextual Representations (OCR)



Object-Contextual Representations

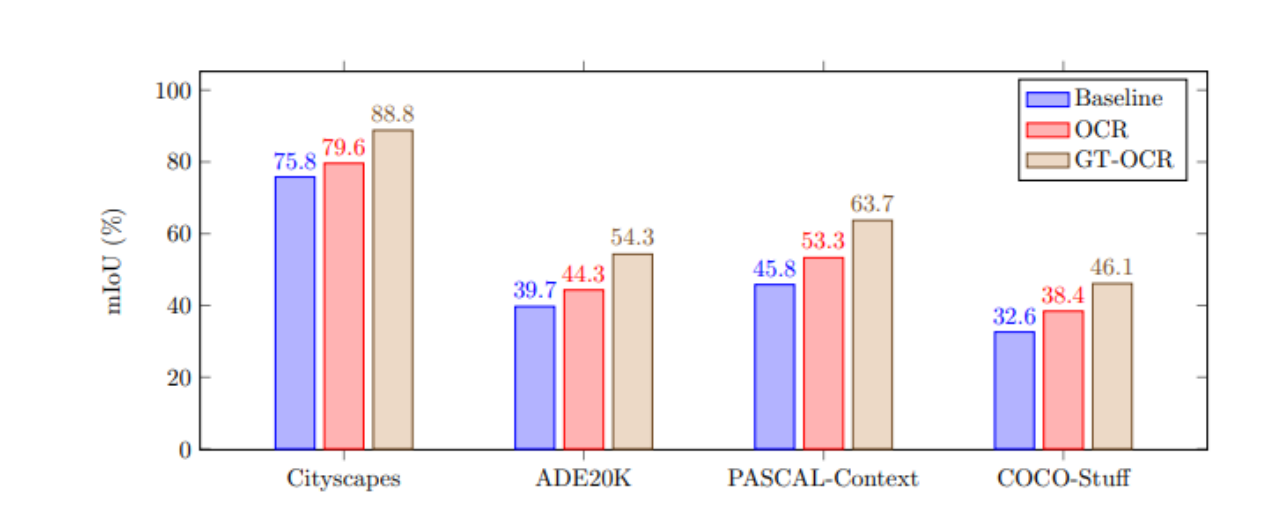


1 Introduction

Semantic segmentation (Thực hiện segment với từng lớp khác nhau) là vấn đề gán nhãn lớp cho từng pixel cho một tấm ảnh. Đây là một chủ đề cơ bản trong thị giác máy tính và rất quan trọng đối với các nhiệm vụ thực tế khác nhau như lái xe tự hành, chẩn đoán y học, nông nghiệp và cảm biến địa lý.

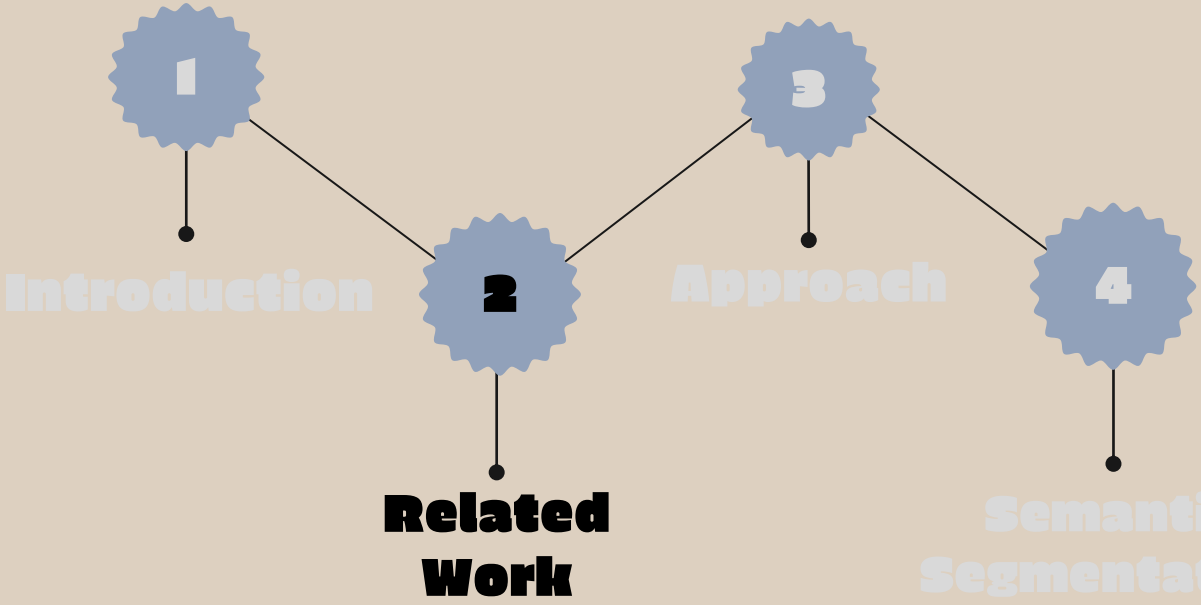
Các nghiên cứu ban đầu chủ yếu tập trung vào quy mô không gian của ngữ cảnh. Các công trình đáng chú ý như ASPP và PPM khai thác ngữ cảnh đa quy mô. Gần đây, một số công trình như DANet, CFNet và OCNet xem xét mối quan hệ giữa một vị trí và các vị trí ngữ cảnh của nó, và tập hợp các biểu diễn của các vị trí ngữ cảnh với trọng số cao cho các biểu diễn tương tự nhau.

1 Introduction



Minh họa hiệu quả của chương trình OCR

Object-Contextual Representations



2 Related Work

Phương pháp ngữ cảnh đa quy mô (Multi-scale context):được thực hiện bằng cách sử dụng các phép tích chập đặc biệt để bắt các ngữ cảnh đa quy mô.

Phương pháp ngữ cảnh quan hệ (Relational context): được thực hiện bằng cách tăng cường biểu diễn cho mỗi điểm ảnh bằng cách tổng hợp các biểu diễn của các điểm ảnh ngữ cảnh, trong đó ngữ cảnh bao gồm tất cả các điểm ảnh.

Phương pháp phân đoạn từ thô đến tinh vi: đã được phát triển để dần dần cải thiện các bản đồ phân đoạn từ đồ thô đến tinh vi. Ví dụ, một nghiên cứu trước đây đã coi bản đồ phân đoạn thô như một biểu diễn bổ sung và kết hợp nó với hình ảnh gốc hoặc các biểu diễn khác để tính toán một bản đồ phân đoạn tinh vi.

Phương pháp phân đoạn theo vùng: tức là tổ chức các điểm ảnh thành một tập hợp các vùng (thường là siêu điểm ảnh), sau đó phân loại từng vùng để có kết quả phân đoạn hình ảnh.

2 Related Work

Minh họa về ngữ cảnh đa quy mô bằng ví dụ ASPP và ngữ cảnh OCR cho điểm ảnh được đánh dấu

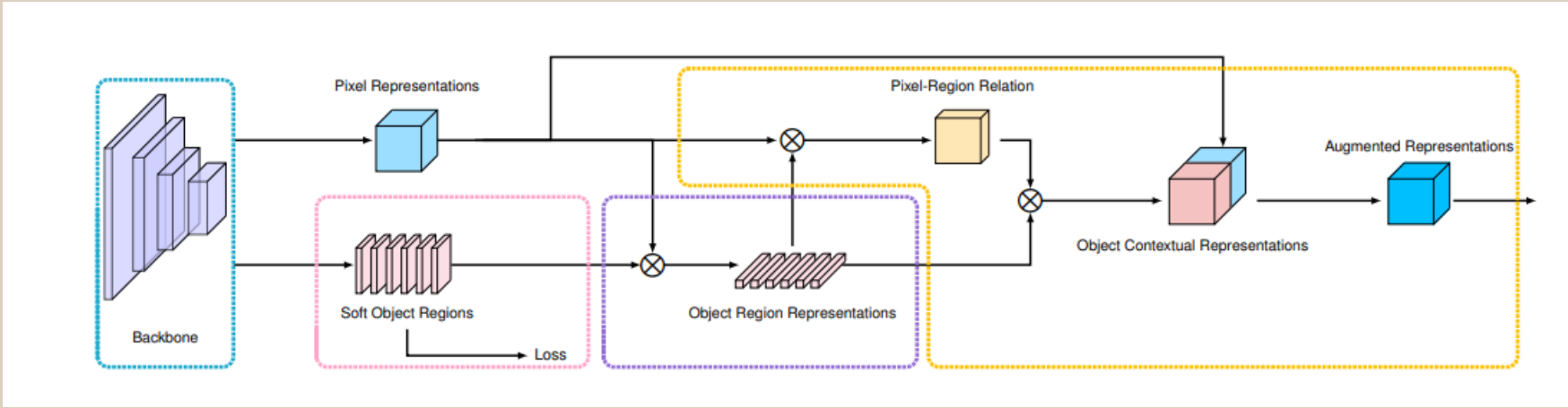
(a) ASPP (b) OCR

ASPP: Ngữ cảnh là một tập hợp các điểm ảnh được lấy mẫu một cách thưa thớtđược đánh dấu bằng

OCR : Ngữ cảnh dự kiến là một tập hợp các điểm ảnh nằm trong vùng đối tượng được đánh dấu bằng màu xanh lam.

2 Related Work

Miêu tả quy trình của OCR.

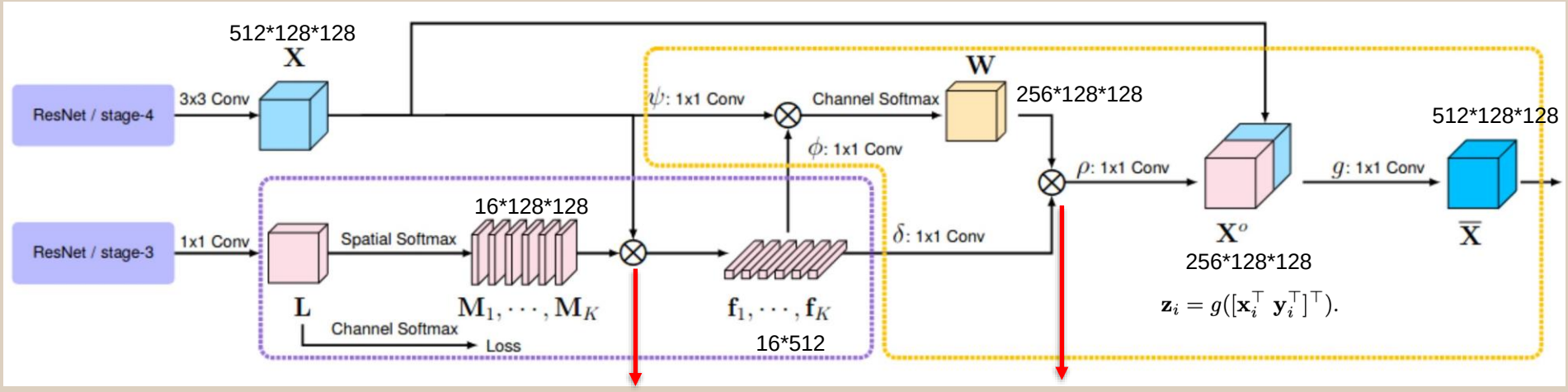


(i) Tạo ra các vùng đối tượng mềm trong hộp đường kẻ màu hồng.

(ii) Ước lượng các biểu diễn của vùng đối tượng trong hộp đường kẻ màu tím.

(iii) Tính toán các biểu diễn ngữ cảnh đối tượng và các biểu diễn được tăng cường trong hộp đường kẻ màu cam.

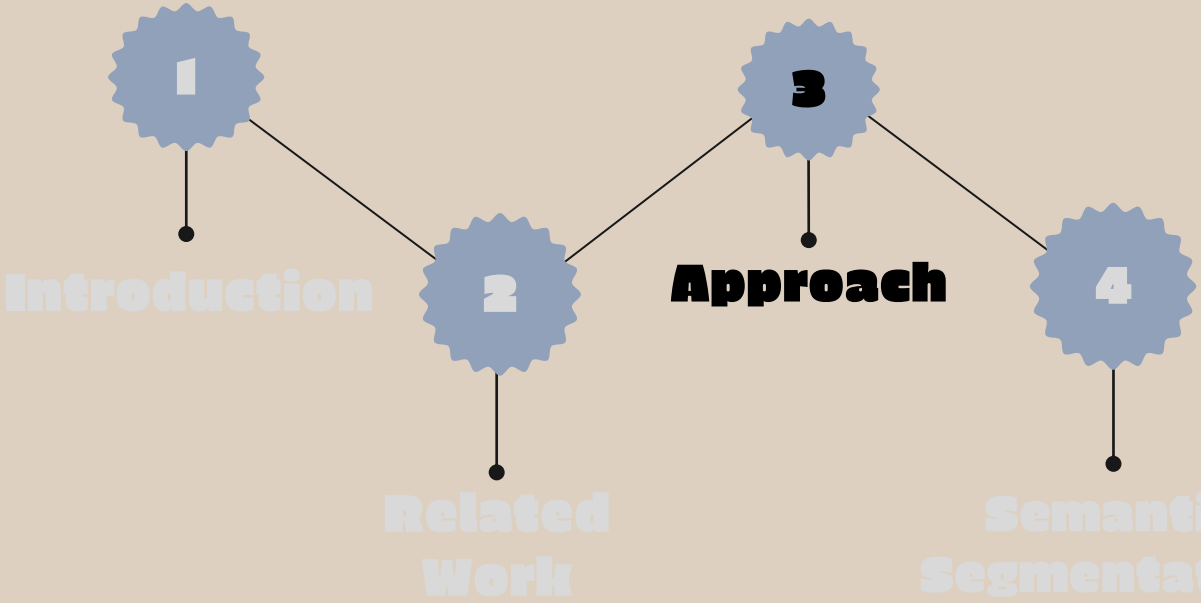
2 Related Work



$$\mathbf{f}_k = \sum_{i \in \mathcal{I}} \tilde{m}_{ki} \mathbf{x}_i.$$

$$\mathbf{y}_i = \rho(\sum_{k=1}^{K} w_{ik} \delta(\mathbf{f}_k)),$$

Object-Contextual Representations



3 Approach

Background

- **Multi-scale context (Ngữ cảnh đa quy mô):**

Mô-đun ASPP (Atrous Spatial Pyramid Pooling): Sử dụng dilated convolutions để bắt lấy thông tin ngữ cảnh đa quy mô.

$$\mathbf{y}_i^d = \sum_{\mathbf{p}_s = \mathbf{p}_i + d\Delta_t} \mathbf{K}_t^d \mathbf{x}_s.$$

Lợi ích: Mô-đun ASPP giúp bắt lấy thông tin ngữ cảnh đa quy mô mà không làm mất độ phân giải.

3 Approach

Background

- **Relational context (Ngữ cảnh tương quan)**

Cơ chế ngữ cảnh tương quan sử dụng mối quan hệ giữa các pixel để tính toán ngữ cảnh cho mỗi pixel.

$$\mathbf{y}_i = \rho\Big(\sum_{s \in \mathcal{I}} w_{is}\delta(\mathbf{x}_s)\Big),$$

Giải thuật ngữ cảnh tương quan là một phiên bản tổng quát hơn của ngữ cảnh toàn cục, cho phép tính toán ngữ cảnh dựa trên mối quan hệ giữa các pixel.

3 Approach

Formulation

The proposed object-contextual representation scheme structurizes all the pixels in image I into K soft object regions, represents each object region as fk by aggregating the representations of all the pixels in the kth object region, and augments the representation for each pixel by aggregating the K object region representations with consideration of its relations with all the object regions:

$$\mathbf{y}_i = \rho(\sum_{k=1}^{K} w_{ik}\delta(\mathbf{f}_k)),$$

where fk is the representation of the kth object region, wik is the relation between the ith pixel and the kth object region. δ(·) and ρ(·) are transformation functions.

Soft object regions. We partition the image I into K soft object regions {M1, M2, . . . , MK}. Each object region Mk corresponds to the class k, and is represented by a 2D map (or coarse segmentation map), where each entry indicates the degree that the corresponding pixel belongs to the class k.

3 Approach

Formulation

Object region representations. We aggregate the representations of all the pixels weighted by their degrees belonging to the kth object region, forming the kth object region representation:

$$\mathbf{f}_k = \sum_{i \in \mathcal{I}} \tilde{m}_{ki} \mathbf{x}_i .$$

Here, xi is the representation of pixel pi. m_{ki} is the normalized degree for pixel pi belonging to the kth object region. We use spatial softmax to normalize each object region Mk.

3 Approach

Formulation

Object contextual representations. We compute the relation between each pixel and each object region as below:

$$w_{ik} = \frac{e^{\kappa(\mathbf{x}_i, \mathbf{f}_k)}}{\sum_{j=1}^{K} e^{\kappa(\mathbf{x}_i, \mathbf{f}_j)}}.$$

Here, $\kappa(x, f) = \varphi(x)^T \psi(f)$ is the unnormalized relation function, $\varphi(\cdot)$ and $\psi(\cdot)$ are two transformation functions implemented by 1×1 conv $\rightarrow$ BN $\rightarrow$ ReLU. This is inspired by self-attention [61] for a better relation estimation.

The object contextual representation yi for pixel pi is computed according to Equation 3. In this equation, $\delta(\cdot)$ and $\rho(\cdot)$ are both transformation functions implemented by 1×1 conv $\rightarrow$ BN $\rightarrow$ ReLU, and this follows non-local networks

3 Approach

Formulation

Augmented representations. The final representation for pixel p_i is updated as the aggregation of two parts, the original representation xi, and the object contextual representation yi :

$$\mathbf{z}_i = g([\mathbf{x}_i^\top \ \mathbf{y}_i^\top]^\top).$$

where g(·) is a transform function used to fuse the original representation and the ob ject contextual representation, implemented by 1 × 1 conv → BN → ReLU.

3 Approach

Segmentation Transformer: Rephrasing the OCR Method

Quy trình OCR

soft object region extraction

object region representation computation

object-contextual representation computation

The attention weight aij :

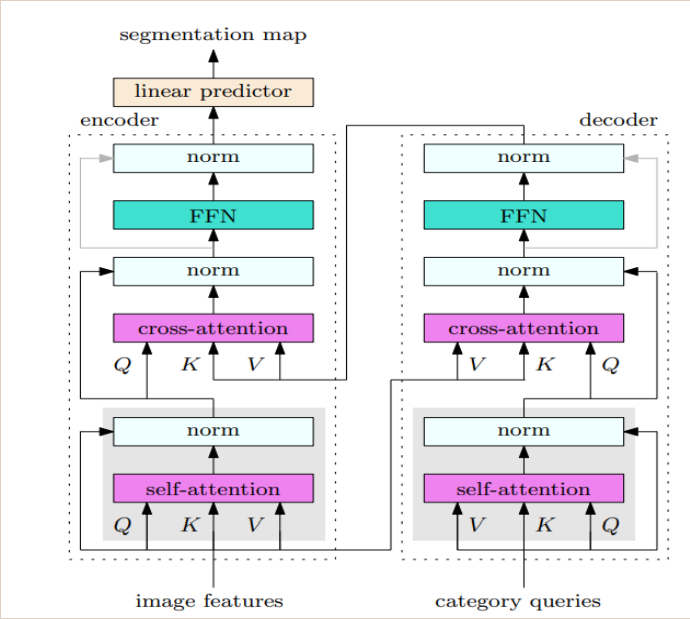
$$a_{ij} = \frac{e^{\frac{1}{\sqrt{d}}\mathbf{q}_i^\top \mathbf{k}_j}}{Z_i} \quad \text{where} \quad Z_i = \sum_{j=1}^{N_{kv}} e^{\frac{1}{\sqrt{d}}\mathbf{q}_i^\top \mathbf{k}_j}.$$

Attn(qi , K, V):

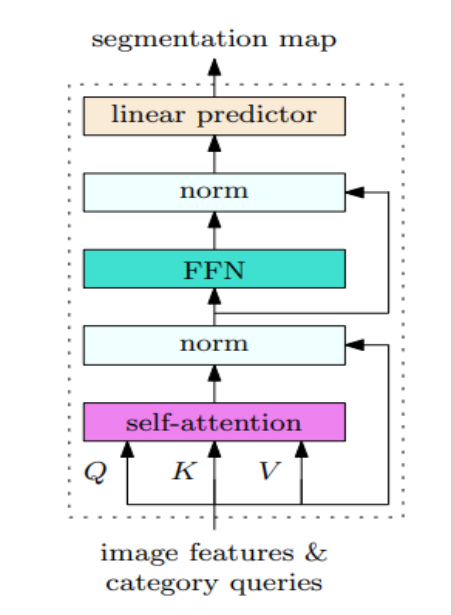
$$\text{Attn}(\mathbf{q}_i, \mathbf{K}, \mathbf{V}) = \sum_{j=1}^{N_{kv}} \alpha_{ij}\mathbf{v}_j.$$

3 Approach

Segmentation Transformer: Rephrasing the OCR Method



Kiến trúc bộ mã hóa-giải mã Transformer



Giải pháp khác cho tranformer

3 Approach

Architecture

Backbone

ResNet-101

- Có 2 đại diện cho đầu vào modun OCR
+ stage 3: dự đoán phân đoạn thô
+ stage 4: tích chập 3x3 (512 kênh đầu ra)

HRNetW48

Chỉ có một đại diện đầu vào cho modun OCR

OCR module

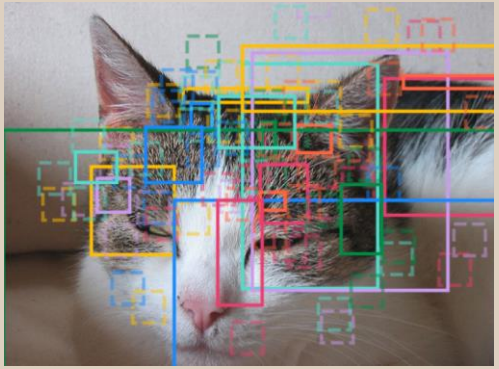
+ Sử dụng hàm tuyến tính (tích chập 1 × 1) để dự đoán phân đoạn thô (vùng đối tượng mềm) được giám sát bằng phép tính chéo pixel-wise cross-entropy loss
+ Tất cả các hàm biến đổi, $\psi(\cdot)$, $\varphi(\cdot)$, $\delta(\cdot)$, $\rho(\cdot)$ và $g(\cdot)$, được triển khai dưới dạng chuyển đổi 1 × 1 → BN → ReLU và ba kênh đầu ra 256 đầu tiên và hai kênh đầu ra 512 cuối cùng
+ Dự đoán phân đoạn cuối cùng từ biểu diễn cuối cùng bằng hàm tuyến tính và chúng tôi cũng áp dụng pixel-wise cross-entropy loss cho dự đoán phân đoạn cuối cùng.

3 Approach

Empirical Analysis

Tiến hành 2 Thí nghiệm trên Cityscapes val bằng cách Sử dụng kiến trúc ResNet-101 mở rộng làm backbone.

- Ảnh hưởng của giám sát vùng đối tượng: loại bỏ giám sát trên các vùng đối tượng mềm và thêm một tổn thất phụ khác trong giai đoạn 3 của ResNet-101



- Ước tính mối quan hệ vùng pixel: so sánh cách tiếp cận của với hai cơ chế khác không sử dụng biểu diễn vùng.

+ Double-Attention: sử dụng biểu diễn pixel để dự đoán mối quan hệ

+ ACFNet: sử dụng trực tiếp một bản đồ phân vùng trung gian để chỉ ra mối quan hệ

3 Approach

Empirical Analysis

Kết quả thí nghiệm

- Giám sát vùng đối tượng và lược đồ quan hệ vùng pixel đều quan trọng đối với hiệu suất, cho kết quả vượt trội.

- Biểu diễn vùng có thể mô tả đối tượng trong hình ảnh cụ thể → mối quan hệ chính xác hơn cho hình ảnh cụ thể so với chỉ sử dụng biểu diễn pixel.

Object region supervision		Pixel-region relations		
w/o supervision	w/ supervision	DA scheme	ACF scheme	Ours
77.31%	**79.58%**	79.01%	78.02%	**79.58%**

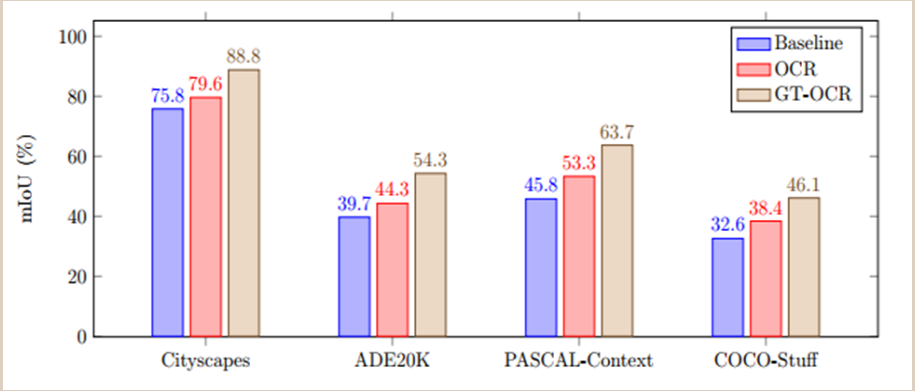
Bảng : Ảnh hưởng của giám sát vùng đối tượng và sơ đồ ước tính quan hệ vùng pixel.

3 Approach

Empirical Analysis

Chứng minh nhận xét: Ground-truth OCR

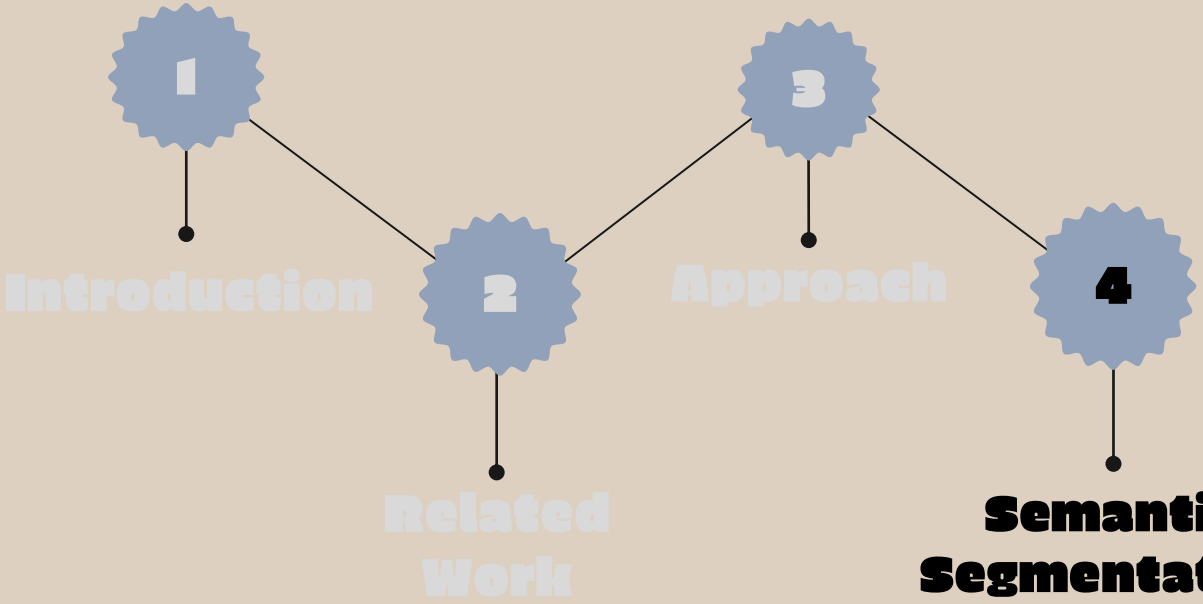
Đánh giá hiệu suất phân đoạn bằng cách sử dụng phân đoạn groundtruth để hình thành các vùng đối tượng và mối quan hệ vùng pixel



(i) Hình thành vùng đối tượng bằng ground-truth: đặt độ tin cậy của pixel i thuộc vùng đối tượng thứ k *mki* = 1 nếu nhãn ground-truth *li* ≡ *k* and *mki* = 0 nếu ngược lại.

(ii) Tính toán quan hệ vùng pixel bằng ground-truth: đặt quan hệ vùng pixel *wik* = 1 nếu nhãn ground-truth *li* ≡ *k* and *wik* = 0 nếu ngược lại.

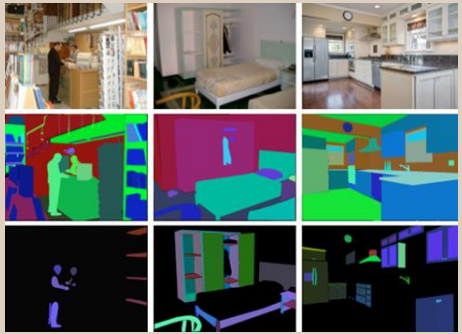
Object-Contextual Representations



4 Semantic Segmentation

Datasets

Cityscapes



ADE20K

PASCAL-Contex

COCO-Stuff

4 Semantic Segmentation

Implementation Details

TASK 1

Khởi tạo mô hình cơ bản bằng cách sử dụng mô hình được huấn luyện trước trên ImageNet và mô-đun OCR ngẫu nhiên.

TASK 3

Thiết lập trọng số trên hàm mất mát cuối cùng là 1 và trọng số trên hàm mất mát được sử dụng để giám sát ước tính vùng đối tượng là 0.4.

TASK 2

Áp dụng chính sách tỷ lệ học tập đa thức với hệ số $(1 - (\frac{iter}{iter\ max})^{0.9})$.

TASK 4

Sử dụng InPlace-ABNsync để đồng bộ hóa giá trị trung bình và độ lệch chuẩn của Batch Normalization trên nhiều GPU.

4 Semantic Segmentation

Implementation Details

TASK 5

Áp dụng tăng cường dữ liệu bằng cách thực hiện lật ngang ngẫu nhiên, điều chỉnh tỷ lệ ngẫu nhiên trong khoảng [0.5, 2] và điều chỉnh độ sáng ngẫu nhiên trong khoảng [-10, 10].

TASK 6

Sử dụng cùng cài đặt huấn luyện cho các phương pháp tái tạo khác như PPM, ASPP, để đảm bảo tính công bằng.

TASK 7

Sử dụng cùng cài đặt huấn luyện cho các phương pháp tái tạo khác như PPM, ASPP, để đảm bảo tính công bằng.

4 Semantic Segmentation

Comparison with Existing Context Schemes

Method	Cityscapes (w/o coarse)	Cityscapes (w/ coarse)	ADE20K	LIP
PPM [80]	78.4%*	81.2%	43.29%	–
ASPP [6]	–	81.3%	–	–
PPM (Our impl.)	80.3%	81.6%	44.50%	54.76%
ASPP (Our impl.)	81.0%	81.7%	44.60%	55.01%
OCR	**81.8%**	**82.4%**	**45.28%**	**55.60%**

Trong phần so sánh với phương pháp ngữ cảnh đa tỷ lệ, đã sử dụng bộ dữ liệu Cityscapes val để huấn luyện OCR. Khi không sử dụng bộ dữ liệu này để huấn luyện, chúng tôi sử dụng ký hiệu "F" để chỉ kết quả. Kết quả cho thấy OCR liên tục vượt trội hơn cả PPM và ASPP trên các bộ dữ liệu khác nhau trong các cuộc so sánh công bằng.

4 Semantic Segmentation

Comparison with Existing Context Schemes

Method	Cityscapes (w/o coarse)	Cityscapes (w/ coarse)	ADE20K	LIP
CC-Attention [27]	81.4%	-	45.22%	-
DANet [18]	81.5%	-	-	-
Self Attention (Our impl.)	81.1%	82.0%	44.75%	55.15%
Double Attention (Our impl.)	81.2%	82.0%	44.81%	55.12%
OCR	**81.8%**	**82.4%**	**45.28%**	**55.60%**

Trong phần so sánh với phương pháp ngữ cảnh liên hệ, phương pháp OCR liên tục có hiệu suất tốt hơn trên các bộ dữ liệu khác nhau. Đáng chú ý, phương pháp Double Attention nhạy cảm đối với việc chọn số lượng vùng và chúng tôi đã điều chỉnh siêu tham số này thành 64 để đạt được hiệu suất tốt nhất.

4 Semantic Segmentation

Comparison with Existing Context Schemes

Method	Parameters▲	Memory▲	FLOPs ▲	Time▲
PPM (Our impl.)	23.1M	792M	619G	99ms
ASPP (Our impl.)	15.5M	284M	492G	97ms
DANet (Our impl.)	10.6M	2339M	1110G	121ms
CC-Attention (Our impl.)	10.6M	427M	804G	131ms
Self-Attention (Our impl.)	10.5M	2168M	619G	96ms
Double Attention (Our impl.)	**10.2**M	209M	**338**G	46ms
OCR	10.5M	**202**M	340G	**45**ms

Trong phần so sánh độ phức tạp, sử dụng bản đồ đặc trưng đầu vào có kích thước [1 x 2048 x 128 x 128] để đánh giá độ phức tạp trong quá trình suy luận. Các con số được thu được trên một GPU P40 duy nhất với CUDA 10.0. Tất cả các con số đều càng nhỏ càng tốt. OCR yêu cầu ít bộ nhớ GPU nhất và thời gian chạy ít nhất.

4 Semantic Segmentation

Comparison with Existing Context Schemes

Thực hiện các thí nghiệm sử dụng mạng lưới ResNet-101 với kỹ thuật dilated làm phần cơ bản và sử dụng cùng cài đặt huấn luyện/kiểm tra để đảm bảo tính công bằng.

So sánh với phương pháp ngữ cảnh liên hệ: Đã so sánh OCR với Self-Attention, Criss-Cross attention (CC-Attention), DANet và Double Attention trên cùng ba bộ dữ liệu và kết quả cho thấy OCR của chúng tôi vượt trội hơn.

So sánh với phương pháp ngữ cảnh đa tỷ lệ: Đã so sánh OCR của chúng tôi với PPM và ASPP trên ba bộ dữ liệu khác nhau (Cityscapes, ADE20K, LIP) và đạt kết quả tốt hơn so với những kết quả ban đầu của các phương pháp đó.

Độ phức tạp: OCR có hiệu suất tốt hơn với số lượng tham số, bộ nhớ GPU, FLOPs và thời gian chạy ít hơn so với các phương pháp ngữ cảnh đa tỷ lệ và ngữ cảnh liên hệ.

4 Semantic Segmentation

Comparison with Existing Context Schemes

Tham số: OCR yêu cầu ít tham số hơn so với PPM và ASPP.

Bộ nhớ: OCR và Double Attention yêu cầu ít bộ nhớ GPU hơn so với các phương pháp khác như DANet và PPM.

FLOPs: OCR yêu cầu ít FLOPs hơn so với PPM, ASPP, DANet, CC-Attention và Self-Attention.

Thời gian chạy: Thời gian chạy của OCR của rất nhỏ so với PPM, ASPP, DANet, CC-Attention và Self-Attention.

4 Semantic Segmentation

Comparison with the State-of-the-Art

Method	Baseline	Stride	Context schemes	Cityscapes (w/o coarse)	Cityscapes (w/ coarse)	ADE20K	LIP	PASCAL Context	COCO-Stuff
Simple baselines									
PSPNet [80]	ResNet-101	8×	M	78.4♭	81.2	43.29	-	47.8	-
DeepLabv3 [6]	ResNet-101	8×	M	-	81.3	-	-	-	-
PSANet [81]	ResNet-101	8×	R	80.1	81.4	43.77	-	-	-
SAC [79]	ResNet-101	8×	M	78.1	-	44.30	-	-	-
AAF [29]	ResNet-101	8×	R	79.1♭	-	-	-	-	-
DSSPN [41]	ResNet-101	8×	-	77.8	-	43.68	-	-	38.9
DepthSeg [32]	ResNet-101	8×	-	78.2	-	-	-	-	-
MMAN [48]	ResNet-101	8×	-	-	-	-	46.81	-	-
JPPNet [39]	ResNet-101	8×	M	-	-	-	51.37	-	-
EncNet [76]	ResNet-101	8×	-	-	-	44.65	-	51.7	-
GCU [38]	ResNet-101	8×	R	-	-	44.81	-	-	-
APCNet [24]	ResNet-101	8×	M,R	-	-	45.38	-	54.7	-
CFNet [77]	ResNet-101	8×	R	79.6	-	44.89	-	54.0	-
BFP [12]	ResNet-101	8×	R	81.4	-	-	-	53.6	-
CCNet [27]	ResNet-101	8×	R	81.4	-	45.22	-	-	-
ANNet [84]	ResNet-101	8×	M,R	81.3	-	45.24	-	52.8	-
OCR (Seg. transformer)	ResNet-101	8×	R	81.8	82.4	45.28	55.60	54.8	39.5

4 Semantic Segmentation

Comparison with the State-of-the-Art

DenseASPP [68]	DenseNet-161	8×	M	80.6	-	-	-	-	-
DANet [18]	ResNet-101 + MG	8×	R	81.5	-	45.22	-	52.6	39.7
DGCNet [78]	ResNet-101 + MG	8×	R	82.0	-	-	-	53.7	-
EMANet [36]	ResNet-101 + MG	8×	R	-	-	-	-	53.1	39.9
SeENet [51]	ResNet-101 + ASPP	8×	M	81.2	-	-	-	-	-
SGR [40]	ResNet-101 + ASPP	8×	R	-	-	44.32	-	52.5	39.1
OCNet [72]	ResNet-101 + ASPP	8×	M,R	81.7	-	45.45	54.72	-	-
ACFNet [75]	ResNet-101 + ASPP	8×	M,R	81.8	-	-	-	-	-
CNIF [63]	ResNet-101 + ASPP	8×	M	-	-	-	56.93	-	
GALD [37]	ResNet-101 + ASPP	8×	M,R	81.8	82.9	-	-	-	-
GALD† [37]	ResNet-101 + CGNL + MG	8×	M,R	-	83.3	-	-	-	-
Mapillary [52]	WideResNet-38 + ASPP	8×	M	-	82.0	-	-	-	-
GSCNN† [55]	WideResNet-38 + ASPP	8×	M	82.8	-	-	-	-	
SPGNet [10]	2× ResNet-50	4×	-	81.1	-	-	-	-	-
ZigZagNet [42]	ResNet-101	4×	M	-	-	-	-	52.1	-
SVCNet [13]	ResNet-101	4×	R	81.0	-	-	-	53.2	39.6
ACNet [19]	ResNet-101 + MG	4×	M,R	82.3	-	45.90	-	54.1	40.1
CE2P [45]	ResNet-101 + PPM	4×	M	-	-	-	53.10	-	
VPLR†‡ [83]	WideResNet-38 + ASPP	4×	M	-	83.5	-	-	-	
DeepLabv3+ [7]	Xception-71	4×	M	-	82.1	-	-	-	-
DPC [4]	Xception-71	4×	M	82.7	-	-	-	-	
DUpsampling [57]	Xception-71	4×	M	-	-	-	-	52.5	-
HRNet [54]	HRNetV2-W48	4×	-	81.6	-	-	55.90	54.0	
OCR (Seg. transformer)	HRNetV2-W48	4×	R	82.4	83.0	45.66	56.65	56.2	40.5
OCR† (Seg. transformer)	HRNetV2-W48	4×	R	83.6	84.2	-	-	-	-

AD20K. Từ Bảng có thể thấy rằng OCR của chúng tôi đạt được hiệu suất cạnh tranh (45,28% và 45,66%) so với hầu hết các phương pháp trước đây dựa trên cả đường cơ sở đơn giản và đường cơ sở nâng cao

LIP đạt được hiệu suất tốt nhất 55,60% trên LIP val dựa trên các đường cơ sở đơn giản. Áp dụng đường trục HRNetV2-W48

4 Semantic Segmentation

Comparison with the State-of-the-Art

Backbone	Method	AP	PQ^{Th}	mIoU	PQ^{St}	PQ
ResNet-50	Panoptic-FPN	40.0	48.3	42.9	31.2	41.5
	Panoptic-FPN + OCR	40.4 (+*0.4*)	48.6 (+*0.3*)	44.3 (+*1.4*)	33.9 (+*2.7*)	42.7 (+*1.2*)
ResNet-101	Panoptic-FPN	42.4	49.7	44.5	32.9	43.0
	Panoptic-FPN + OCR	42.7 (+*0.3*)	50.2 (+*0.5*)	45.5 (+*1.0*)	35.2 (+*2.3*)	44.2 (+*1.2*)

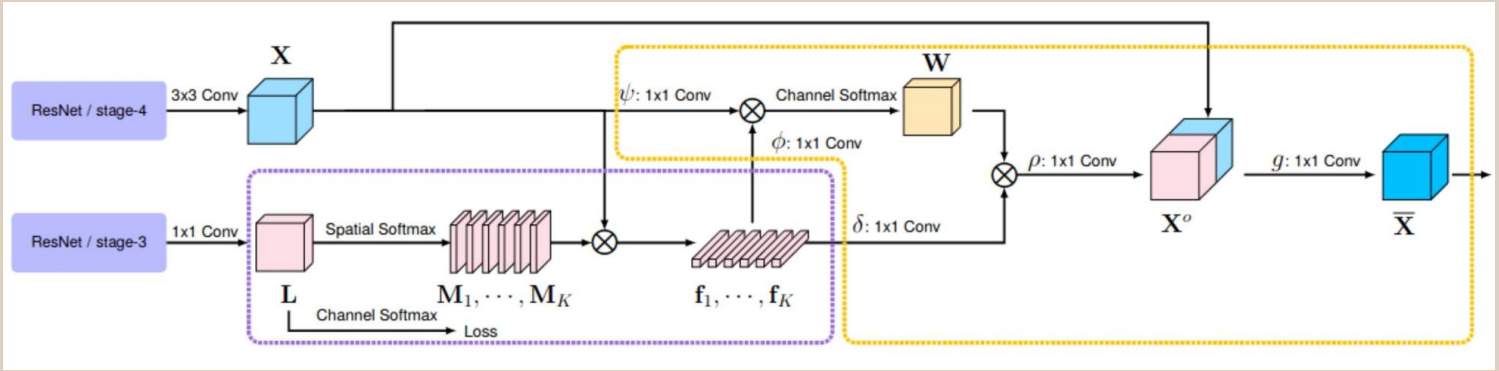
PASCAL-Context: cách tiếp cận của chúng tôi vượt trội hơn cả phương pháp tốt nhất trước đây dựa trên đường cơ sở đơn giản và phương pháp tốt nhất trước đó dựa trên đường cơ sở nâng cao. Phương pháp HRNet-W48 + OCR đạt hiệu suất tốt nhất 56,2%, vượt trội đáng kể so với phương pháp tốt thứ hai, ví dụ: ACPNet (54,7%).và ACNet (54,1%).

COCO-Stuff :Có thể thấy rằng phương pháp của chúng tôi đạt được hiệu suất tốt nhất, 39,5% dựa trên ResNet-101 và 40,5% dựa trên HRNetV2-48.

Method used: OCRNET

Config file:
ocnfigs/ocrnet/cornet_hr18s_4xb2-40k_cityscapes-512x1024.py

Checkpoint file:
https://download.openmmlab.com/mmsegmentation/v0.5/ocrnet/ocrnet_hr18s_4xb2-40k_cityscapes-512x1024/ocrnet_hr18s_4xb2-40k_cityscapes-512x1024_20230227_145026-6c052a14.pth



Config file modified

- Change data path, datatype
- Define number of batches and gpu
- Define default hook for logging and saving checkpoint
- Define dataset (type, root, prefix, pipeline)
- Define param_scheduler for auto modify learning rate
- Define the output num for two decoders
- Load check point pretrained
- Define epoch and iteration

```
#Define num of class
• cfg.model.decode_head[0].num_classes = 2
• cfg.model.decode_head[1].num_classes = 2
#Define batch norm
• cfg.norm_cfg = dict(type='BN', requires_grad=True)
• cfg.model.backbone.norm_cfg = cfg.norm_cfg
• cfg.model.decode_head[0].norm_cfg = cfg.norm_cfg
• cfg.model.decode_head[1].norm_cfg = cfg.norm_cfg
# Modify dataset type and path
• cfg.dataset_type = 'ISICDATASET_'
• cfg.data_root = '/content/dataset'
• cfg.train_dataloader.dataset.type = 'ISICDATASET_'
• cfg.train_dataloader.dataset.data_root = '/content/dataset'
• cfg.train_dataloader.dataset.data_prefix = dict(img_path='images/ISIC2018_Task1-2_Training_Input’,
  seg_map_path='groundTruth/ISIC2018_Task1_Training_GroundTruth')
• cfg.train_dataloader.dataset.pipeline = cfg.train_pipeline
• cfg.val_dataloader.dataset.type = 'ISICDATASET_'
• cfg.val_dataloader.dataset.data_root = '/content/dataset'
• cfg.val_dataloader.dataset.data_prefix = dict(img_path='images/ISIC2018_Task1-2_Validation_Input’,
  seg_map_path='groundTruth/ISIC2018_Task1_Validation_GroundTruth’)
• cfg.val_dataloader.dataset.pipeline = cfg.test_pipeline
```

```
• cfg.test_dataloader.num_workers = 2
• cfg.test_dataloader.batch_size = 1
• cfg.test_dataloader.dataset.data_root = '/content/dataset'
• cfg.test_dataloader.dataset.data_prefix =
    dict(img_path='images/ISIC2018_Task1-2_Test_Input’,
    seg_map_path='groundTruth/ISIC20T8_Task1_Test_GroundTruth')
• cfg.test_dataloader.dataset.pipeline = cfg.test_pipeline
• cfg.test_evaluator = dict(
type='IoUMetric',
iou_metrics=['mIoU'],
format_only=False,
output_dir='work_dirs/format_results'
)
# the number of samples and workers per GPU
• cfg.train_dataloader.batch_size = 4
• cfg.train_dataloader.num_workers = 1
• cfg.work_dir = './work_dirs/final'
• cfg.train_cfg = dict(
    type='EpochBasedTrainLoop', max_epochs=5, val_begin=T, val_interval=1)
```

- cfg.param_scheduler = [

```
dict(type='LinearLR', by_epoch=False, start_factor=0.1, begin=0, end=200),
dict(type='PolyLR', eta_min=0.0001, power=0.9, begin=0, end=160, by_epoch=False)
]
```

- cfg.default_hooks = dict(

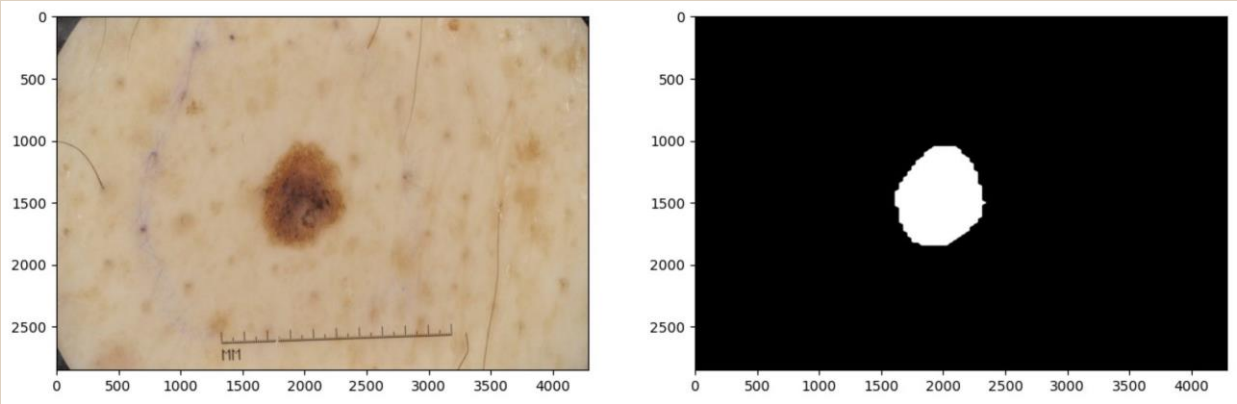
```
timer=dict(type='IterTimerHook'),
logger=dict(type='LoggerHook', interval=50, log_metric_by_epoch=False),
param_scheduler=dict(type='ParamSchedulerHook'),
checkpoint=dict(type='CheckpointHook', interval = 1000, by_epoch=False),
sampler_seed=dict(type= DistSamplerSeedHook')
)
```

- cfg.log_processor = dict(by_epoch=True)
- cfg['randomness'] = dict(seed=32)
- cfg.dump('/content/mmsegmentation/configs/ocrnet/ocrnet_khanh.py')

```
#Load pretrain model
cfg.load_from =
```

"https://download.openmmlab.com/mmsegmentation/v0.5/ocrnet/ocrnet_hr18s_4xb2-40k_cityscapes-512x1024/ocrnet_hr18s_4xb2-40k_cityscapes-512x1024_20230227_145026-6c052a14.pth"

Dataset



- Add custom palette for each label
- Modify file images suffix and seg_map_suffix

Train command

```
from mmengine.runner import Runner

runner = Runner.from_cfg(cfg)

runner.train()
```

Evaluation command

- `cfg = Config.fromfile('/content/mmsegmentation/configs/ocrnet/ocrnet_khanh.py')`
- `checkpoint_path = '/content/mmsegmentation/work_dirs/tutorial/iter_1000.pth'`
- `cfg.model.pretrained = checkpoint_path`
- `runner_1 = Runner.from_cfg(cfg)`
- `runner.test()`

Review

The model does extremely good at predict the abnormal region of for all the test images. At the early stage of training, model reach high accuracy, and then quickly to converge.