Deep Learning for Vision (DLV)

Segmentation

Denis Coquenet

2024-2025

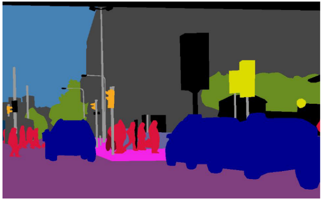
Université de Rennes
istic Informatique Électronique

Table of contents

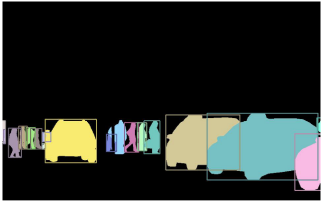
Segmentation tasks

- Per-pixel classification

Input image



Semantic segmentation



Instance segmentation

Panoptic segmentation

Why?

- Autonomous driving
- Medical image segmentation (tumor detection)
- Background removal (videoconference), filters

Challenges

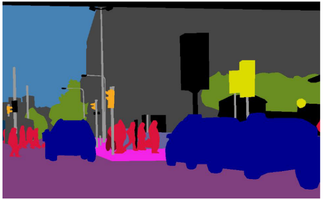
- Unknown number of items to recognize
- Items can overlap
- Must preserve the input shape

Goal

Each pixel is classified, all instances of same class are merged

Formulation

Input: $\boldsymbol{x} \in \mathbb{R}^{H \times W \times C}$, a set of N_c classes $\mathcal{C}$
Output: $\boldsymbol{y} \in [1..N_c]^{H \times W}$



- Adjacent objects of same class merged together
- No distinction of instances

Goal

Each instance is segmented, whether it is fully visible or not

➤ The same pixel can be associated to multiple classes or to multiple instances of the same class

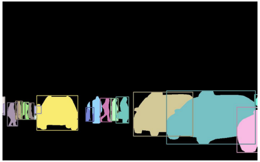
Formulation

Input: $\boldsymbol{x} \in \mathbb{R}^{H \times W \times C}$, a set of N_c classes $\mathcal{C}$

Output: $\boldsymbol{y} = \{(c_k, m_k) \in [1..N_c] \times \{0,1\}^{H \times W}\}_k$ with:

c_k: the class of the instance k

m_k: the binary mask for the instance k



➤ Object detection + semantic segmentation

➤ Only detected objects are segmented

Goal

Each pixel is classified and associated to an instance of that class

Formulation

Input: $\boldsymbol{x} \in \mathbb{R}^{H \times W \times C}$, a set of N_c classes $\mathcal{C}$
Output: $\boldsymbol{y} \in \mathbb{N}^{H \times W \times 2}$ with :
$\boldsymbol{y}_{i,j,1} \in [1..N_c]$: the class of pixel (i,j)
$\boldsymbol{y}_{i,j,2} \in \mathbb{N}$: the instance identifier of pixel (i,j)

➤ Best of both worlds

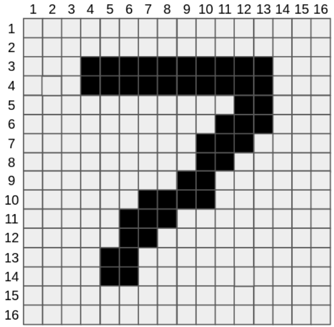
Pixel-level

- Accuracy
- Precision
- Recall
- F1
- IoU
- mAP

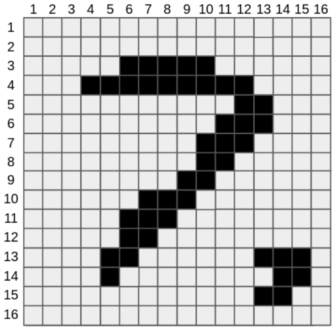
➤ Can also be computed at object level (as for object detection)

Compute the accuracy, precision, recall, F1 and IoU at pixel level for both predictions

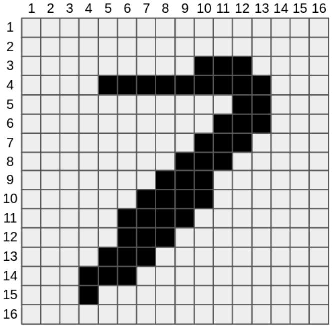
Ground truth

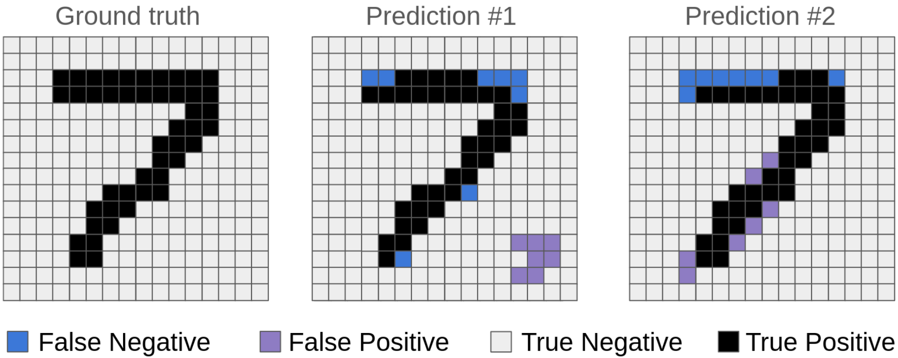


Prediction #1



Prediction #2





For both predictions: FN: 8, FP: 7, TP: 37, TN: 204

Accuracy: (37+204)/256 = 94.14%
Precision: 37/(37+7) = 84.09%
Recall: 37/(37+8) = 82.22%
IoU: 37/(37+8+7) = 71.15%
F1: $(2 \times 84.09 \times 82.22)/(84.09 + 82.22) = 83.14\%$

➤ Exactly the same values but two different error cases (would be different at object level)

Table of contents

➤ Specific architectures for each segmentation task

Semantic segmentation

- FCN, U-Net

Instance segmentation

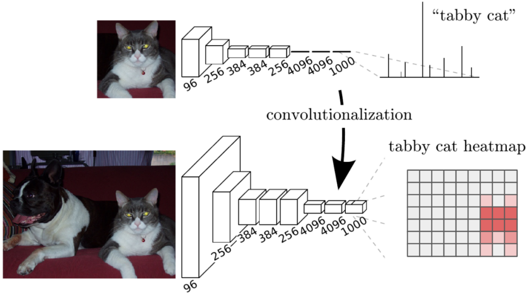
- Mask R-CNN

Panoptic segmentation

- UPSNet

How to go from classification to semantic segmentation?

➤ Fully Convolutional Network (FCN)



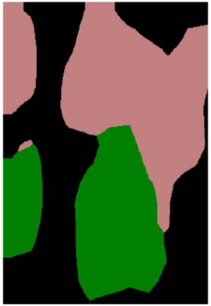
- Main constraint: output must be of same size than input
- Classification: fixed-size input because of fully-connected layers

➤ Idea: convert dense $4096 \to 1000$ by conv with 1000 kernels $1 \times 1 \times 4096$

➤ "Convolutionalization" of well-known classification architectures evaluated on Pascal VOC 2011 (validation set):

	FCN-AlexNet	FCN-VGG16	FCN-GoogLeNet[4]
mean IU	39.8	**56.0**	42.5
forward time	50 ms	210 ms	59 ms
conv. layers	8	16	22
parameters	57M	134M	6M
rf size	355	404	907
max stride	32	32	32

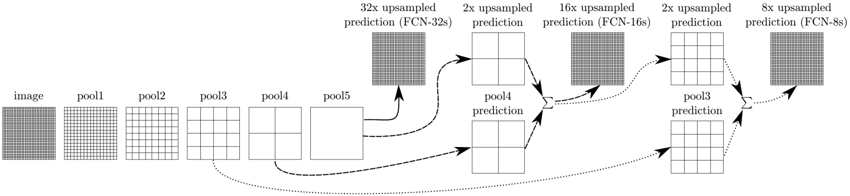
FCN-32s

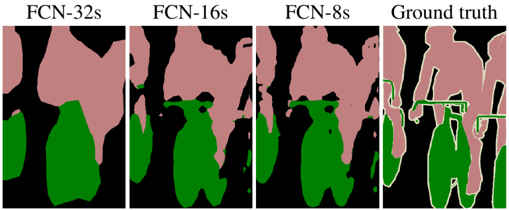


➤ Downsampling (max stride=32) = information loss
How to improve the results?

FCN (2015) [2]

- Combine multi-scale predictions to refine

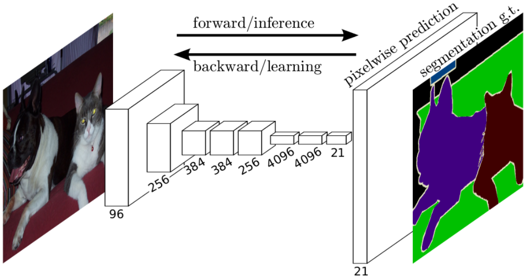




	Accuracy (%)	IoU (%)
FCN-32s	89.1	59.4
FCN-16s	90.0	62.4
FCN-8s	**90.3**	**62.7**

- *e.g.*, upsampling as bilinear interpolation

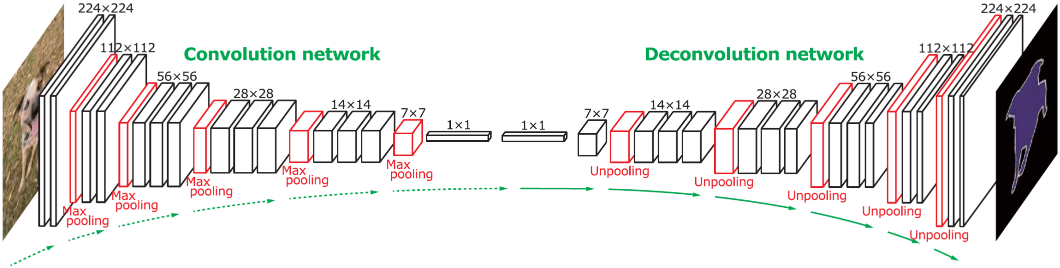
FCN (2015) [2]



Training

A pixel-level supervision using Cross-Entropy (CE):

$$\mathcal{L} = \sum_{i=1}^{W} \sum_{j=1}^{H} \mathcal{L}_{\text{CE}}(\hat{\boldsymbol{Y}}_{i,j}, \boldsymbol{Y}_{i,j})$$

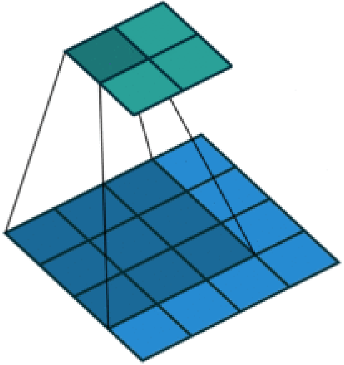


➤ Extend FCN with specific layers instead of bi-linear interpolations:

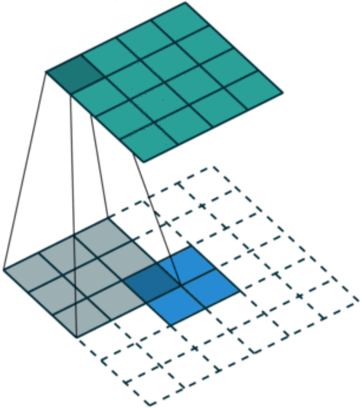
- Deconvolution
- Unpooling

Deconvolution (or transposed convolution)

Convolution 3×3



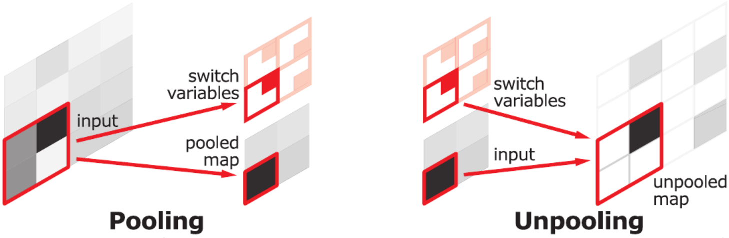
Deconvolution 3×3



- Use trainable kernels as convolutions

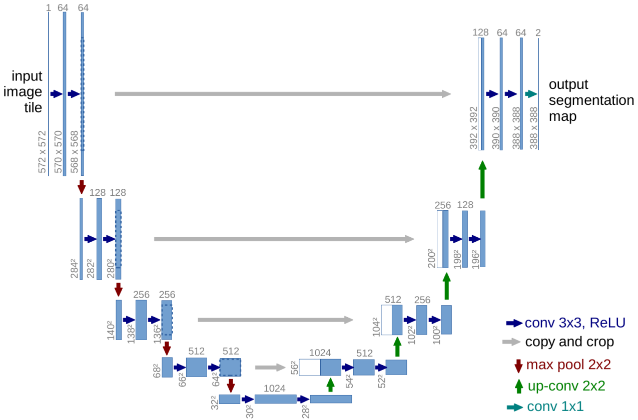
Source: `https://github.com/vdumoulin/conv_arithmetic` [4]

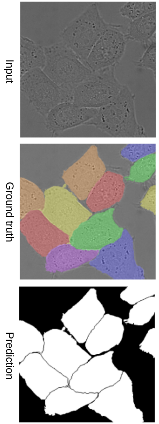
Unpooling



- No parameters involved
- Requires to remember pooled locations
- Sparse representation

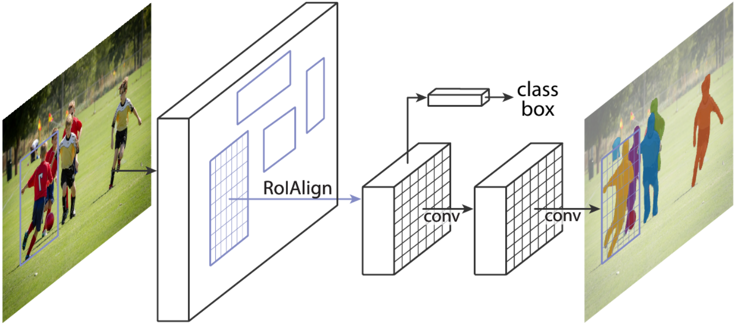
➤ Generalization of conv/deconv architectures with skip connections





HeLa cell segmentation

Goal: instance segmentation

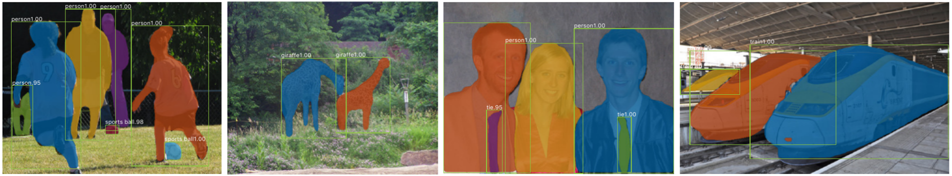


- Extend Faster R-CNN with mask branch (CNN module)
- Output binary masks (foreground/background) for each Region of Interest (RoI)

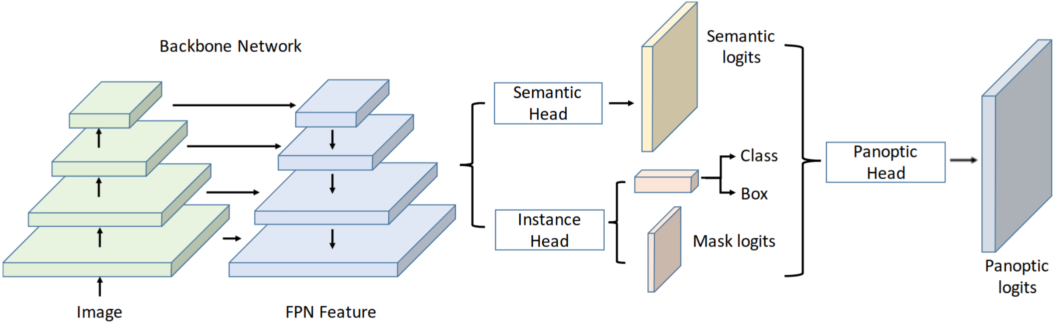
Training

$$\mathcal{L} = \mathcal{L}_{\text{cls}} + \mathcal{L}_{\text{reg}} + \mathcal{L}_{\text{mask}}$$

$\mathcal{L}_{\text{cls}}$: cross-entropy for RoI classification
$\mathcal{L}_{\text{reg}}$: smooth L1 for RoI regression
$\mathcal{L}_{\text{mask}}$: binary cross-entropy for RoI segmentation (binary masks).

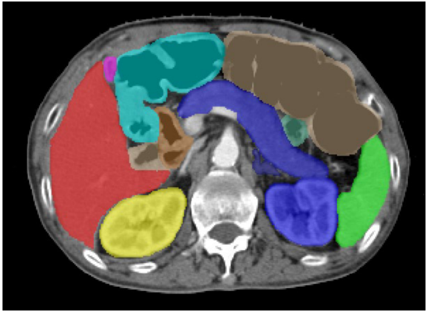


➤ Unified Panoptic Segmentation Network



- Instance head: Mask R-CNN
- Semantic head: CNN
- Panoptic head: parameter-free aggregation

Table of contents



Data

Inputs: high-resolution CT-scan volumes

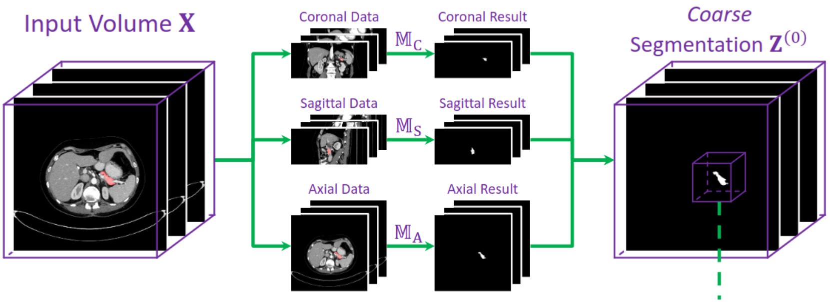
- ➤ tens/hundreds of slices 512×512
- ➤ How to process such 3D data?

Medical application

- Very few annotated data
  - Few experts
  - Data privacy
  - Datasets $\simeq$ tens of examples
- Accuracy is crucial
  - A matter of life and death
  - Specific metrics

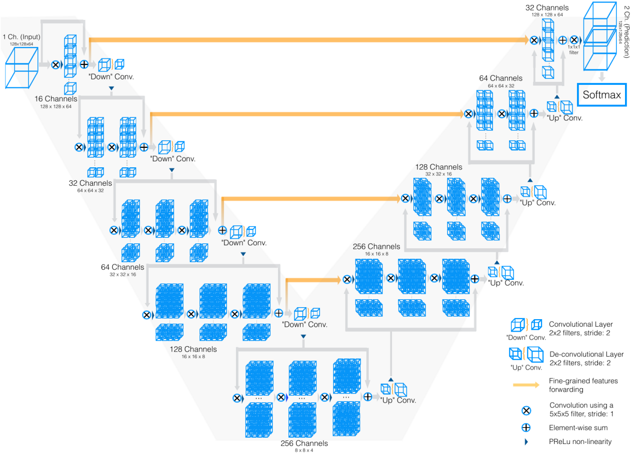
Slice-by-slice process (2017) [8]

➤ Reduce complexity through 2D processing

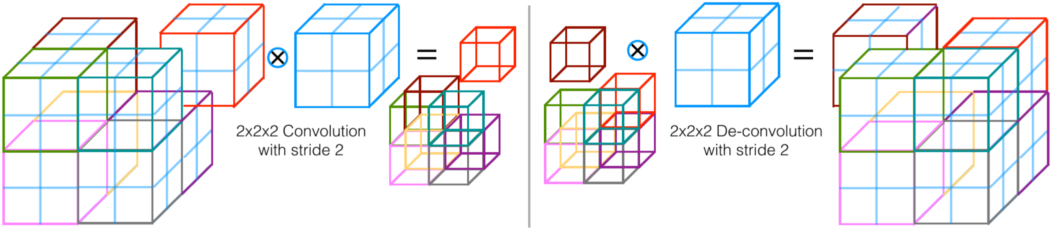


- 1 FCN / axis
- Slices processed independently across the same axis
  ➤ Context loss

➤ Reduce complexity by downsampling the input



- Extend U-Net with 3D convolutions
- Inputs downsampled to fixed size: $128 \times 128 \times 64$
  ➤ Information loss from pre-processing



Comparison

Input (256, 256, 512) $\rightarrow$ conv 2D with 1024 kernels 2×2, stride 2×2
➤ output (128, 128, 1024), 2.1 M parameters, 68.7 GFLOPs

Input (256, 256, 64, 512) $\rightarrow$ conv 3D with 1024 kernels $2 \times 2 \times 2$, stride $2 \times 2 \times 2$
➤ output (128, 128, 32, 1024), 4.2 M parameters, 4.4 TFLOPs

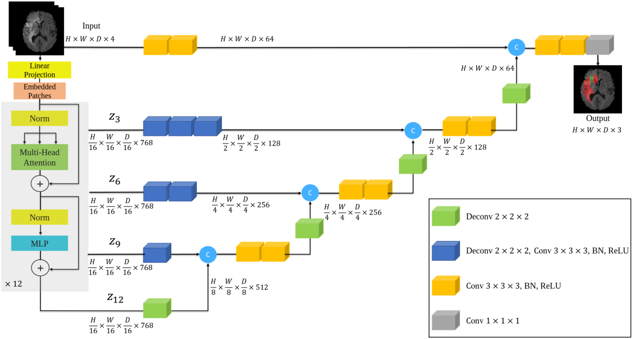
Approaches

- Process 2D slices independently: context loss in third axis
- Downsampling volume: compression $\rightarrow$ information loss

➤ Trade-off: process sub-volumes

- Preserve original resolution
- Preserve 3D nature of the input
- Long-term context loss

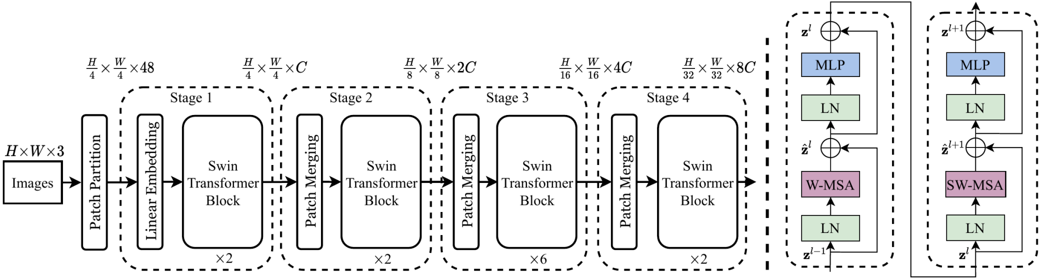
- Combine CNN U-Net with Vision Transformer



Size of input sub-volume: $128 \times 128 \times 128 \times 4$
Patch size: $16 \times 16 \times 16$
Embedding size: 768
Input dimension for ViT: $(8 \times 8 \times 8) \times 768 \rightarrow 512 \times 768$

Swin (2021) [11]

➤ Hierarchical Vision Transformer using Shifted windows (Swin)



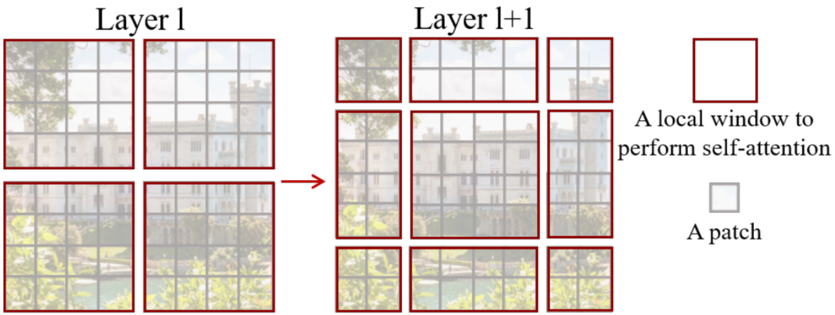
(a) Architecture (b) Two Successive Swin Transformer Blocks

Patch size: 4×4

Patch merging: 2×2 adjacent patches are merged together

(S)W-MSA: (Shifted) Window Multi Self-Attention

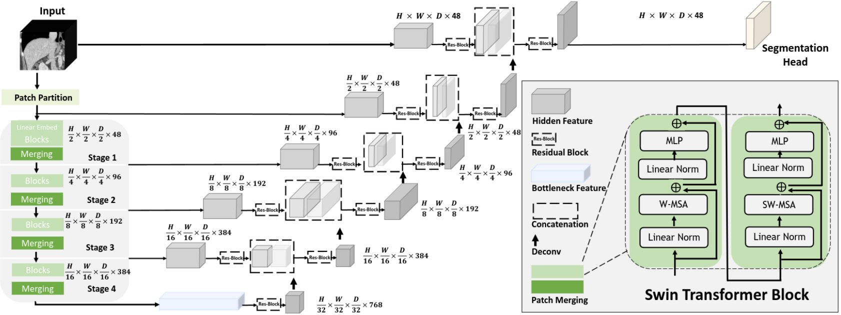
➤ Idea: perform attention on patch windows (locally), in parallel



Modeling global context with shifted windows

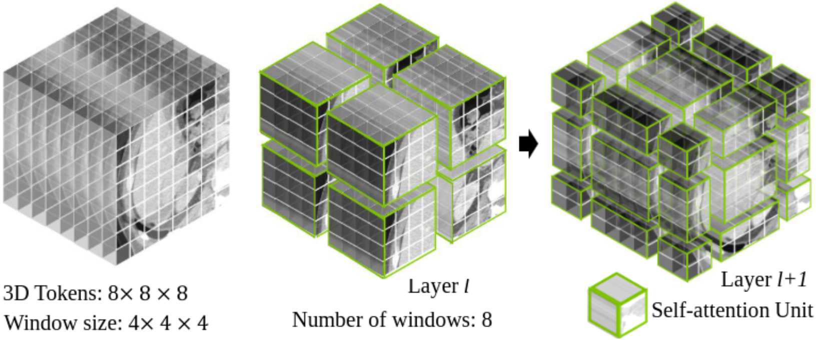
Windows are shifted from one layer to another to propagate the information from one window to its neighbours

- ➤ Combining UNETR with Swin



Goal: reducing information compression when patching

Size of input sub-volume: $128 \times 128 \times 128 \times 4$
Patch size: $2 \times 2 \times 2$
Input dimension for Swin: $(64 \times 64 \times 64) \times 48 \rightarrow 262,144 \times 48$


3D Tokens: 8× 8 × 8
Window size: 4× 4 × 4
Layer l
Number of windows: 8
Layer $l+1$
Self-attention Unit

Exercise: comparison with global self-attention

Reminder: simplified self-attention

Number of FLOPs: $8d^2L + 4dL^2$
With d the number of dimensions and L the sequence length

Compute the number of FLOPs for an input volume of size (64, 64, 32, 256)

- for a traditional self-attention layer
- for the shifted windows approach (window size: $8 \times 8 \times 8$, first step only)

➤ Remark: the number of dimensions is preserved (from 256 to 256)

Reminder: simplified self-attention

Number of FLOPs: $8d^2L + 4dL^2$
With d the number of dimensions and L the sequence length

For a traditional self-attention layer

Flatten: $L = HWD = 64 \times 64 \times 32 = 131,072$
➤ $8 \times 256^2 \times 131,072 + 4 \times 256 \times 131,072^2 = 17.7$ TFLOPs

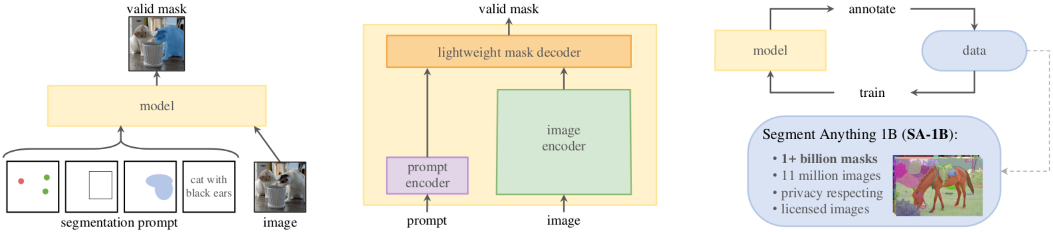
For a shifted-window self-attention layer

Window size: $8 \times 8 \times 8$
Flatten: $L = HWD = 8 \times 8 \times 8 = 512$
➤ $8 \times 256^2 \times 512 + 4 \times 256 \times 512^2 = 537$ MFLOPs

Number of windows: $\frac{64}{8} \times \frac{64}{8} \times \frac{32}{8} = 256$
➤ $256 \times 537 = 137$ GFLOPs

Table of contents

➤ Segment Anything Model (SAM)



Goal: segmentation task generalization

- Interactive segmentation (point, rectangle)
- Free-text prompt segmentation
- Instance segmentation
- Segmentation refinement from mask

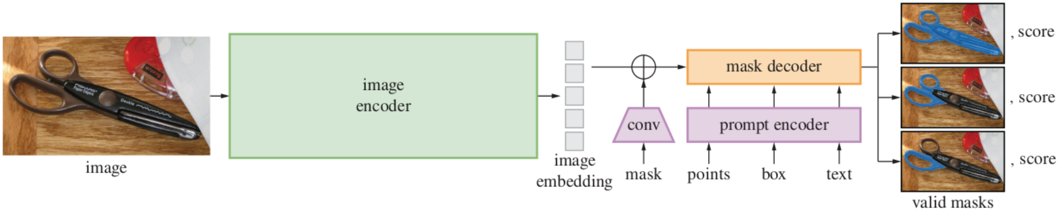


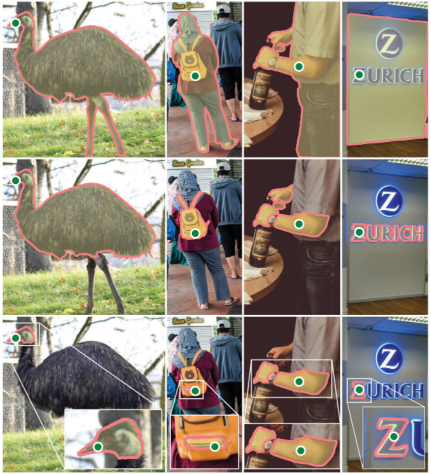
Image encoder

Vision Transformer (pre-trained with Masked Auto Encoder)

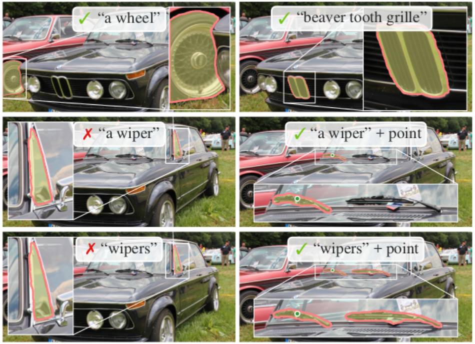
➤ Used only once for multiple prompts

Prompt encoders

- text: CLIP encoder
- point/box: positional encoding + learned embedding
- mask: CNN encoder



➤ Consider several predictions for the same inputs



➤ Combining segmentation strategies can help refine predictions

Want to try?

➤ https://segment-anything.com/demo

Summary

Challenges

- Output size (mostly same as input)
  ➤ U-Net-like models (compression then unpooling, deconvolution)
- Hardware constraints, notably for 3D inputs (CT scan, video)
  ➤ Trade-off between context modeling and information compression

Segmentation tasks are diverse

- Semantic, instance, panoptic
  ➤ Interactive segmentation (point, box, text prompting)

➤ Next time: handwritten text recognition!

[1] Alexander Kirillov, Kaiming He, Ross B. Girshick, Carsten Rother, and Piotr Dollár. "Panoptic Segmentation". In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 9404–9413.

[2] Jonathan Long, Evan Shelhamer, and Trevor Darrell. "Fully convolutional networks for semantic segmentation". In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 3431–3440.

[3] Hyeonwoo Noh, Seunghoon Hong, and Bohyung Han. "Learning Deconvolution Network for Semantic Segmentation". In: *2015 IEEE International Conference on Computer Vision*. 2015, pp. 1520–1528.

[4] Vincent Dumoulin and Francesco Visin. "A guide to convolution arithmetic for deep learning". In: abs/1603.07285 (2016). arXiv: 1603.07285. URL: http://arxiv.org/abs/1603.07285.

[5] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-Net: Convolutional Networks for Biomedical Image Segmentation". In: *Medical Image Computing and Computer-Assisted Intervention*. Ed. by Nassir Navab, Joachim Hornegger, William M. Wells III, and Alejandro F. Frangi. Vol. 9351. Lecture Notes in Computer Science. Springer, 2015, pp. 234–241.

[6] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. "Mask R-CNN". In: *IEEE International Conference on Computer Vision*. 2017, pp. 2980–2988.

[7] Yuwen Xiong, Renjie Liao, Hengshuang Zhao, Rui Hu, Min Bai, Ersin Yumer, and Raquel Urtasun. "UPSNet: A Unified Panoptic Segmentation Network". In: *IEEE Conference on Computer Vision and Pattern Recognition*. Computer Vision Foundation / IEEE, 2019, pp. 8818–8826.

[8] Yuyin Zhou, Lingxi Xie, Wei Shen, Yan Wang, Elliot K. Fishman, and Alan L. Yuille. "A Fixed-Point Model for Pancreas Segmentation in Abdominal CT Scans". In: *Medical Image Computing and Computer Assisted Intervention*. Vol. 10433. Lecture Notes in Computer Science. Springer, 2017, pp. 693–701.

[9] Fausto Milletari, Nassir Navab, and Seyed-Ahmad Ahmadi. "V-Net: Fully Convolutional Neural Networks for Volumetric Medical Image Segmentation". In: *Fourth International Conference on 3D Vision*. 2016, pp. 565–571.

[10] Ali Hatamizadeh, Yucheng Tang, Vishwesh Nath, Dong Yang, Andriy Myronenko, Bennett A. Landman, Holger R. Roth, and Daguang Xu. "UNETR: Transformers for 3D Medical Image Segmentation". In: *IEEE/CVF Winter Conference on Applications of Computer Vision*. 2022, pp. 1748–1758.

[11] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. "Swin Transformer: Hierarchical Vision Transformer using Shifted Windows". In: *2021 IEEE/CVF International Conference on Computer Vision1*. 2021, pp. 9992–10002.

[12] Yucheng Tang, Dong Yang, Wenqi Li, Holger R. Roth, Bennett A. Landman, Daguang Xu, Vishwesh Nath, and Ali Hatamizadeh. "Self-Supervised Pre-Training of Swin Transformers for 3D Medical Image Analysis". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 20698–20708.

[13] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloé Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, and Ross B. Girshick. "Segment Anything". In: abs/2304.02643 (2023). arXiv: 2304.02643. URL: https://doi.org/10.48550/arXiv.2304.02643.