Institute of Electronics
National Yang Ming Chiao Tung University
Hsinchu, Taiwan

AI Training Course Series

Image Semantic Segmentation and Its Applications on Medical AI

Lecture 9

Presenter: Yi-Zeng Fang

Advisor: Juinn-Dar Huang, Ph.D.

August 12, 2024

Outline

- Introduction of Image Segmentation
- Terminology
- Semantic Segmentation
- Image Segmentation Model
- Paper
  - Segment Anything
- Competition
  - Xilinx Adaptive Computing Challenge 2021
  - IEEE ISBI Challenge – PAIP 2023
- References

Introduction of Image Segmentation

Common Computer Vision Tasks

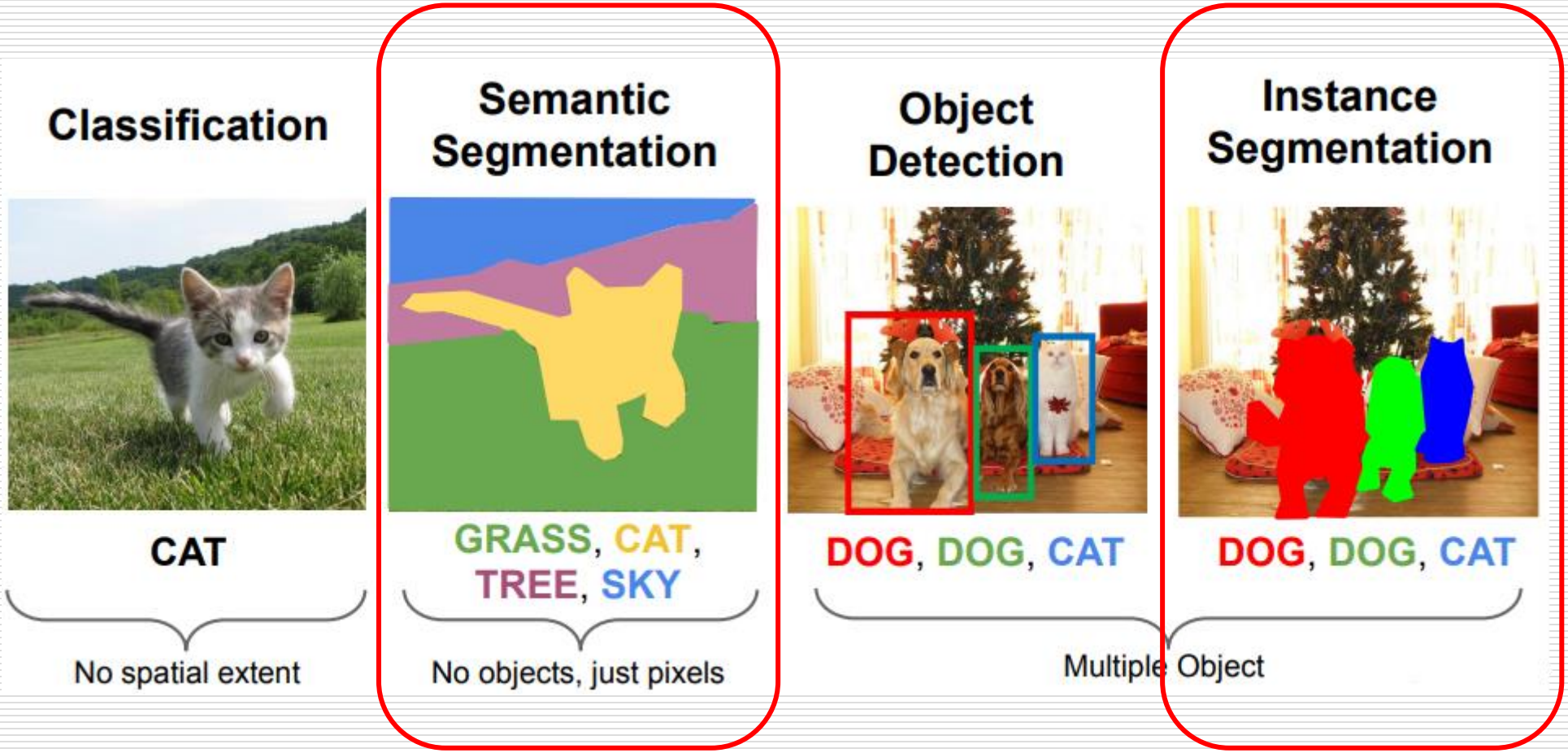
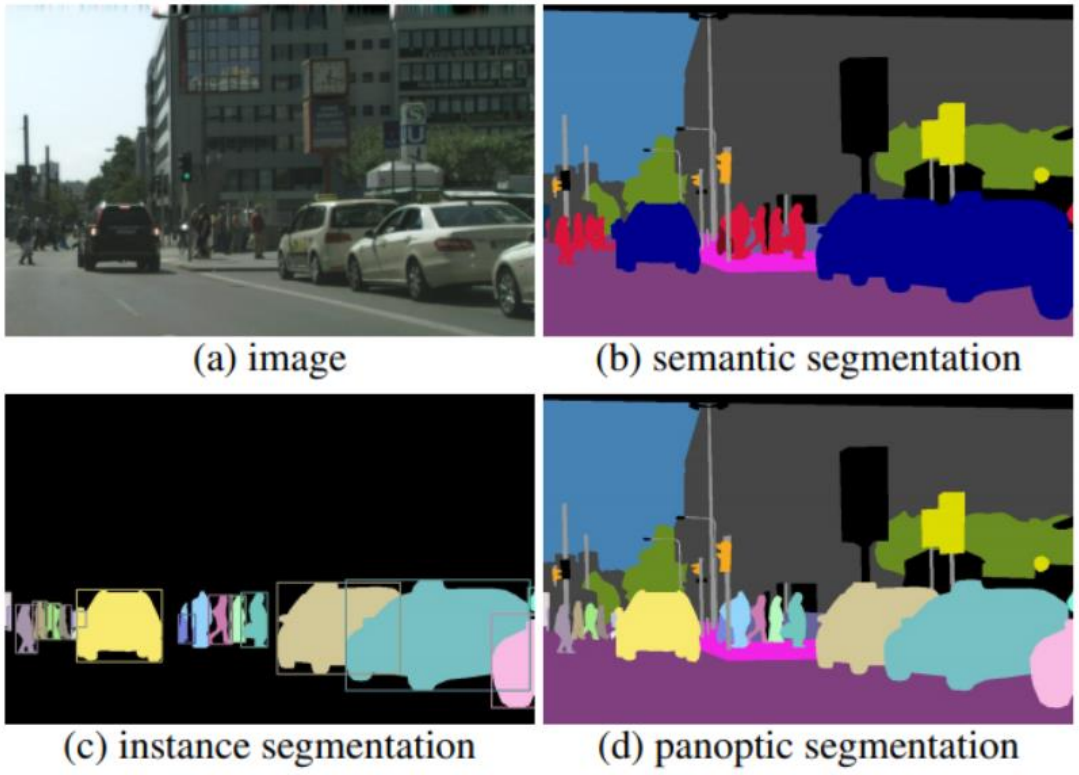


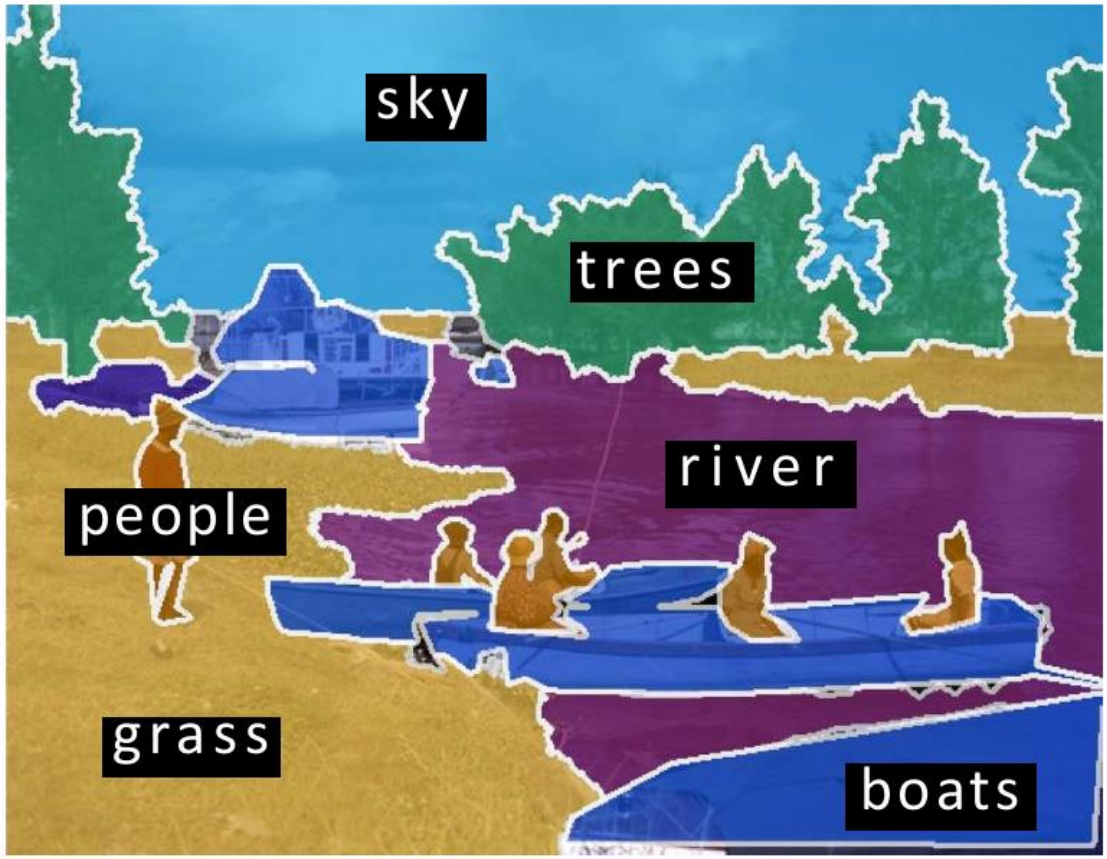
Image Segmentation

1. Semantic Segmentation
2. Instance Segmentation
3. Panoptic Segmentation



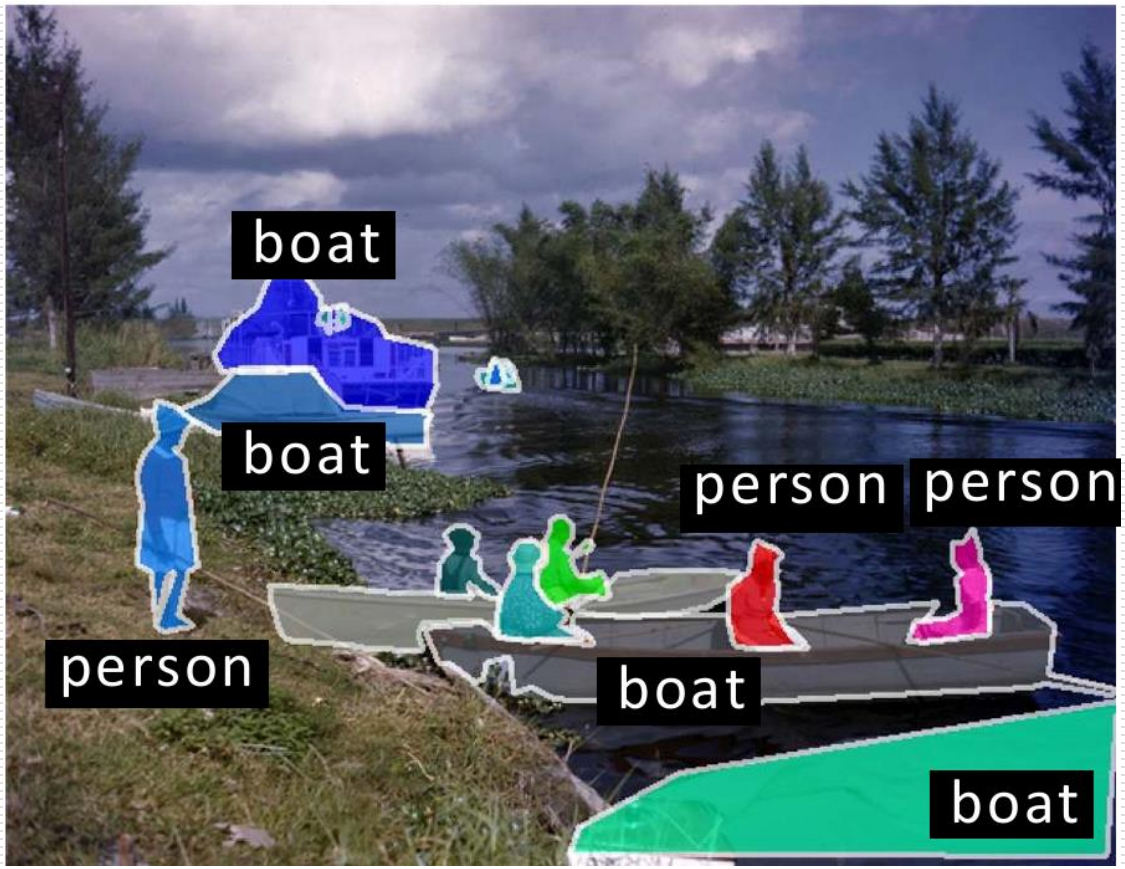
Semantic Segmentation

- Assign a semantic label to each pixel
- Pixel-wise Classification

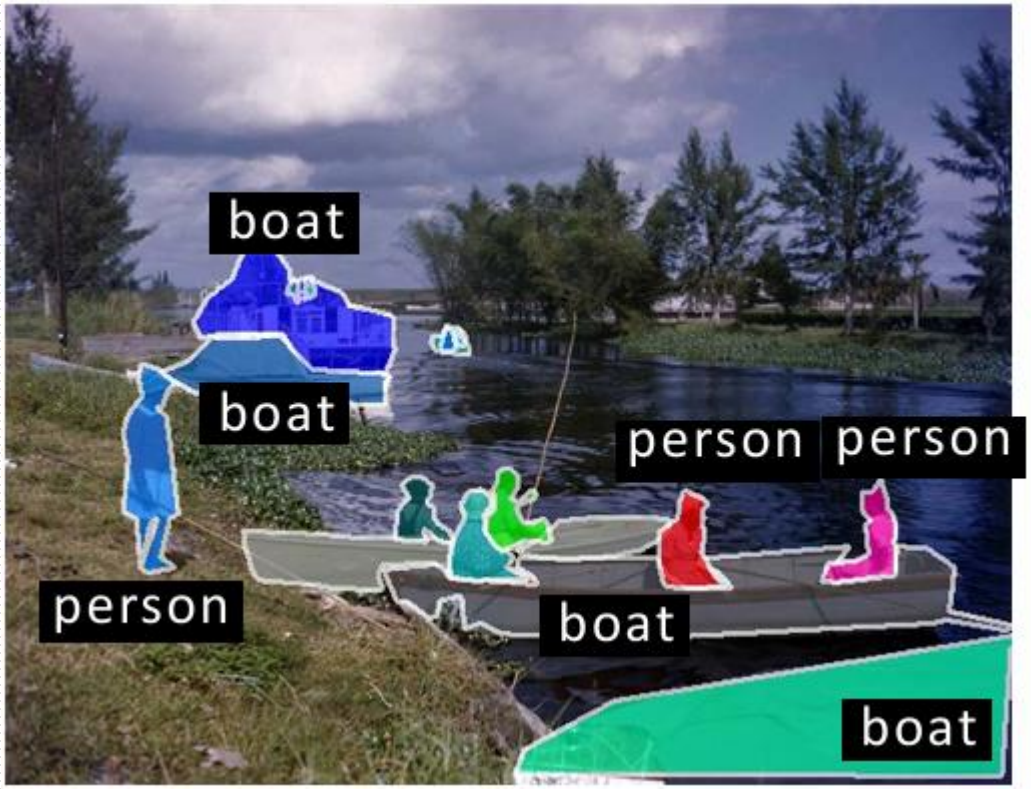


Instance Segmentation

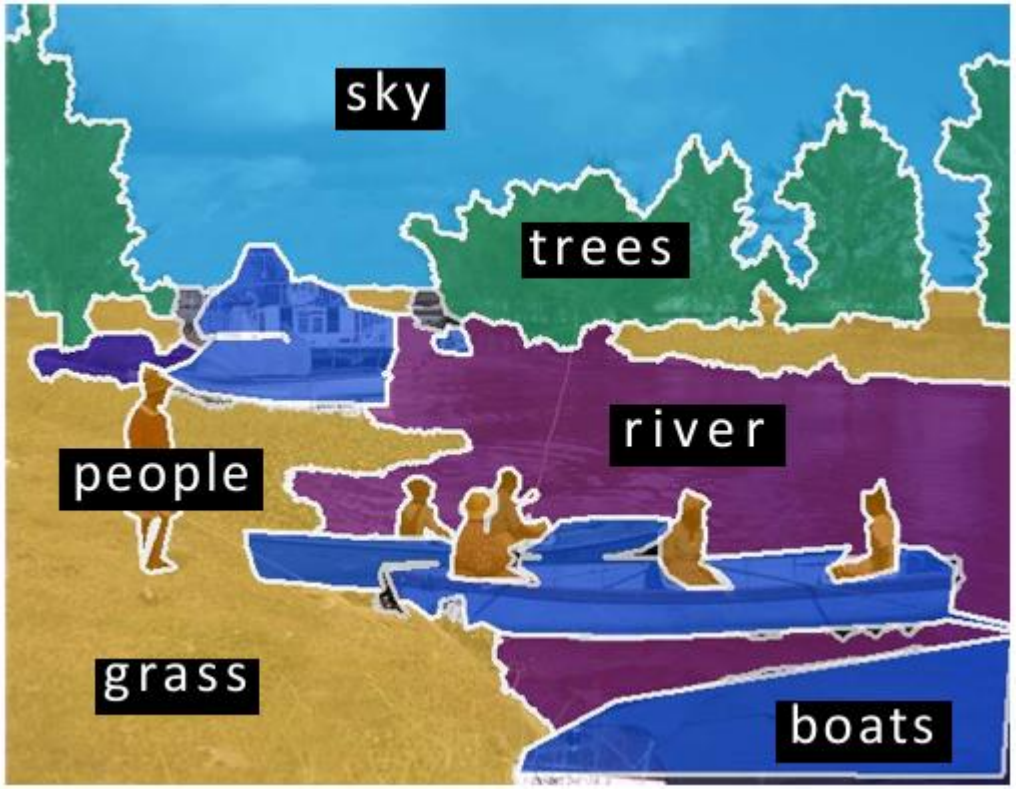
- Delineate each object with a mask
- Semantic + Localization



Instance + Semantic Segmentation



Instance Segmentation

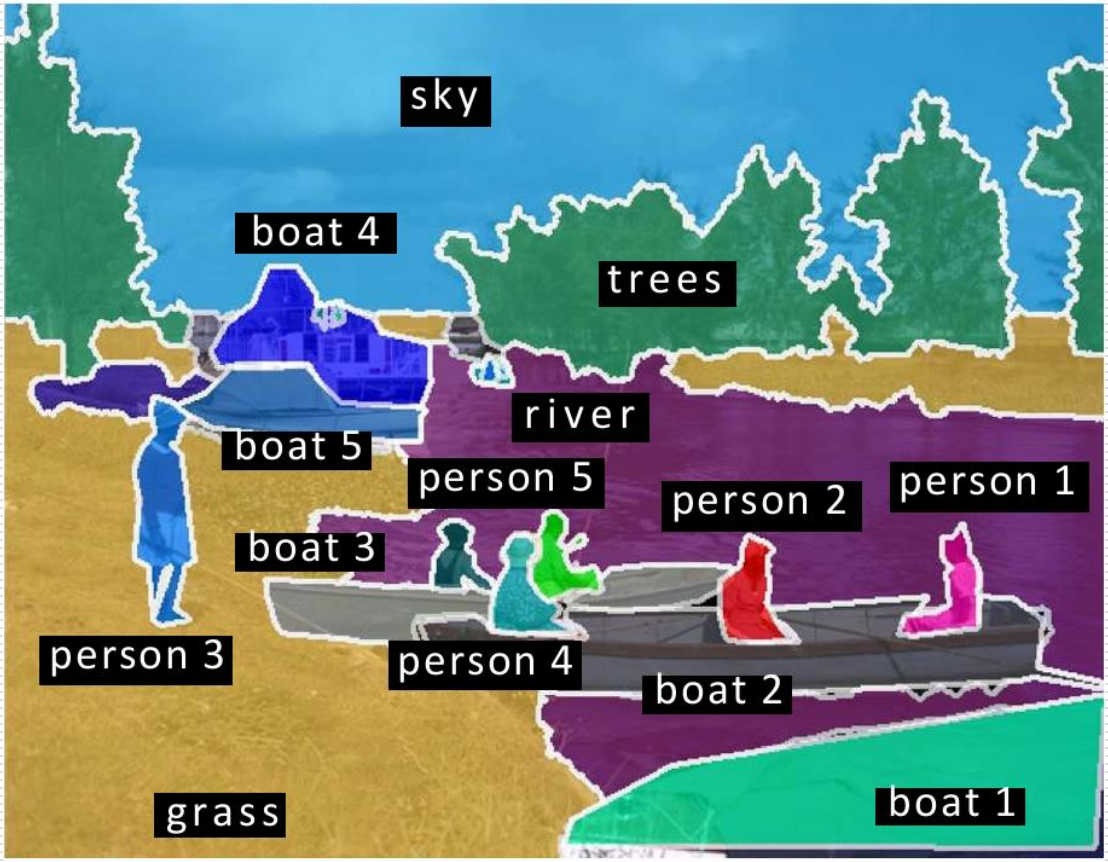


Semantic Segmentation

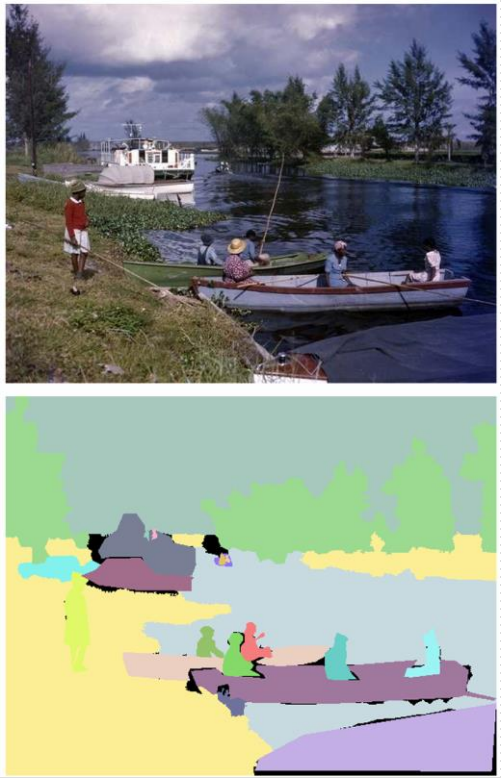
Real-world application likely requires both modalities !!

Panoptic Segmentation

- Assign semantic labels to pixels
- Segment each instance separately



Panoptic Segmentation Datasets



COCO (2014)
+ COCO-stuff (2017)
~ 200k images
133 categories

Panoptic Segmentation Datasets

COCO (2014)
+ COCO-stuff (2017)
~ 200k images
133 categories

Mapillary Vistas (2017)
~ 25k images
66 categories

Panoptic Segmentation Datasets

COCO (2014)
+ COCO-stuff (2017)
~ 200k images
133 categories

Mapillary Vistas (2017)
~ 25k images
66 categories

Cityscapes (2017)
5k images
19 categories

Panoptic Segmentation Datasets

COCO (2014)
+ COCO-stuff (2017)
~ 200k images
133 categories

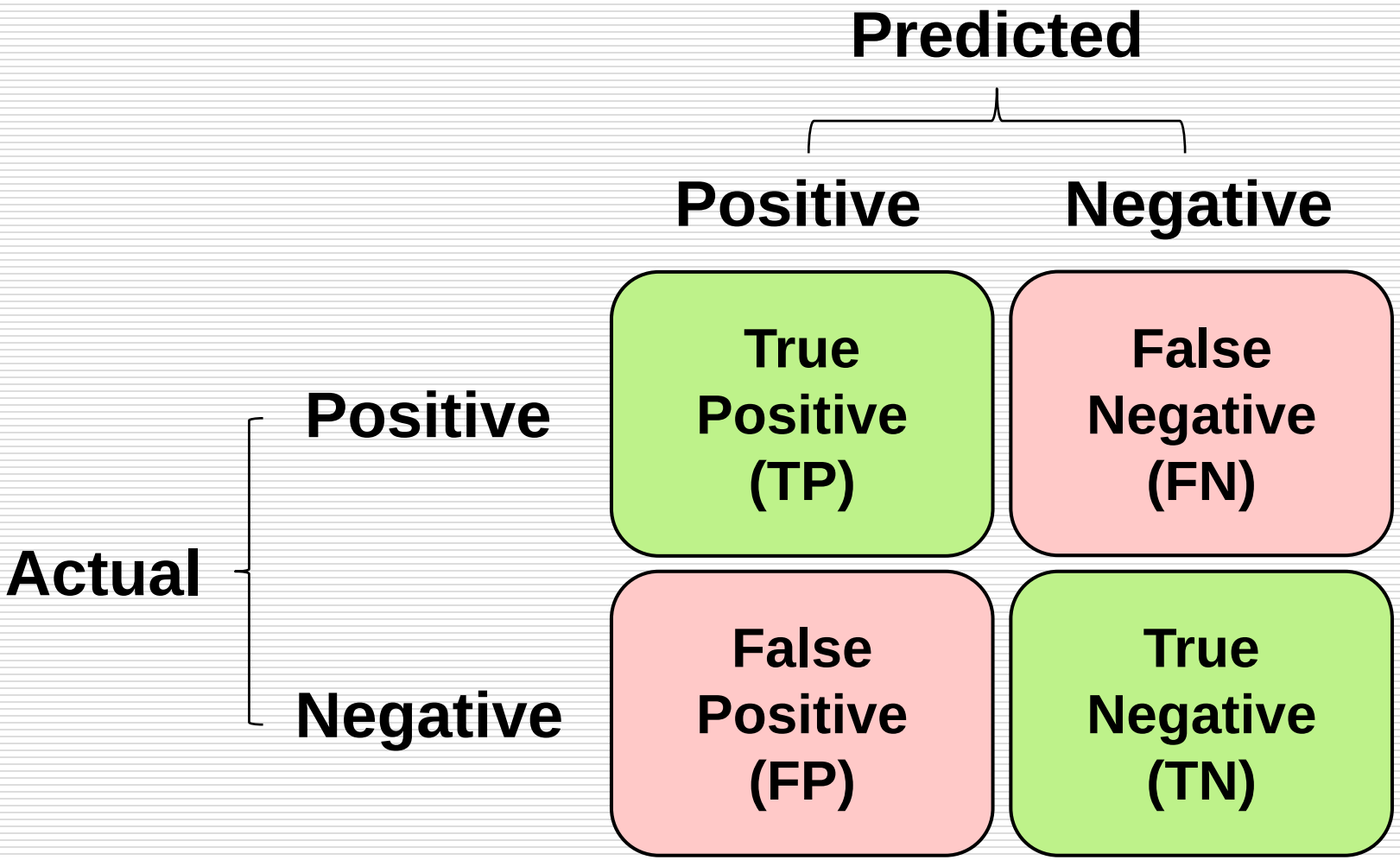
Mapillary Vistas (2017)
~ 25k images
66 categories

Cityscapes (2017)
5k images
19 categories

ADE20k (2016)
> 22k images
150 categories

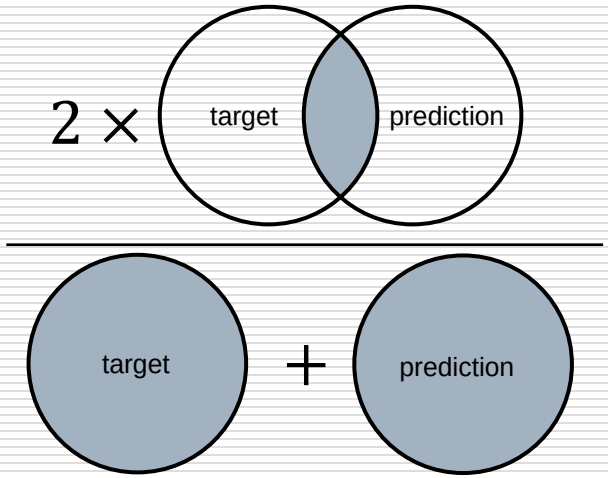
Terminology

Confusion Matrix

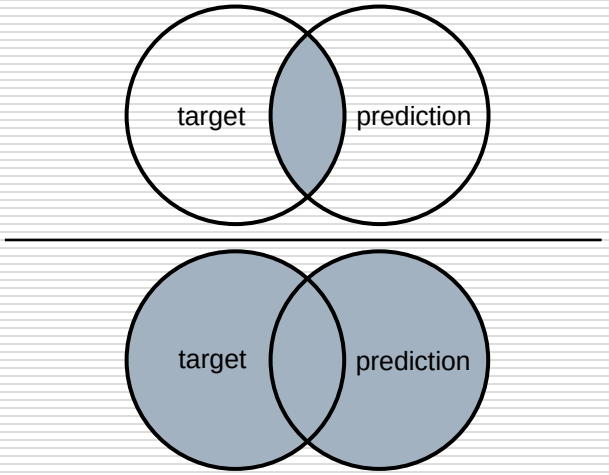


Common Performance Metrics (1/2)

- Dice coefficient (Dice):
  - $Dice = \frac{2 \times (target \cap prediction)}{target + prediction} = \frac{2 \times TP}{2 \times TP + FP + FN}$
- Intersection over Union (IoU):
  - $IoU = \frac{target \cap prediction}{target \cup prediction} = \frac{TP}{TP + FP + FN}$



Dice



IoU

Common Performance Metrics (2/2)

- True Positive Rate (TPR) $\frac{TP}{TP+FN}$
- Positive Predictive Value (PPV) $\frac{TP}{TP+FP}$
- F1 Score $2 \times \frac{PPV \times TPR}{PPV+TPR}$
- Dice = F1 Score

$$F_1 Score = 2\frac{PPV \times TPR}{PPV + TPR}$$
$$= 2\frac{\frac{TP}{TP+FP} \times \frac{TP}{TP+FN}}{\frac{TP}{TP+FP} + \frac{TP}{TP+FN}}$$
$$= 2\frac{\frac{TP \times TP}{(TP+FP) \times (TP+FN)}}{\frac{TP(TP+FN) + TP(TP+FP)}{(TP+FP) \times (TP+FN)}}$$
$$= 2\frac{TP \times TP}{TP(TP+FN) + TP(TP+FP)}$$
$$= \frac{2 \times TP}{2 \times TP + FP + FN}$$
$$= DC_{病}$$

Semantic Segmentation

Semantic Segmentation

- Label each pixel in the image with a category label

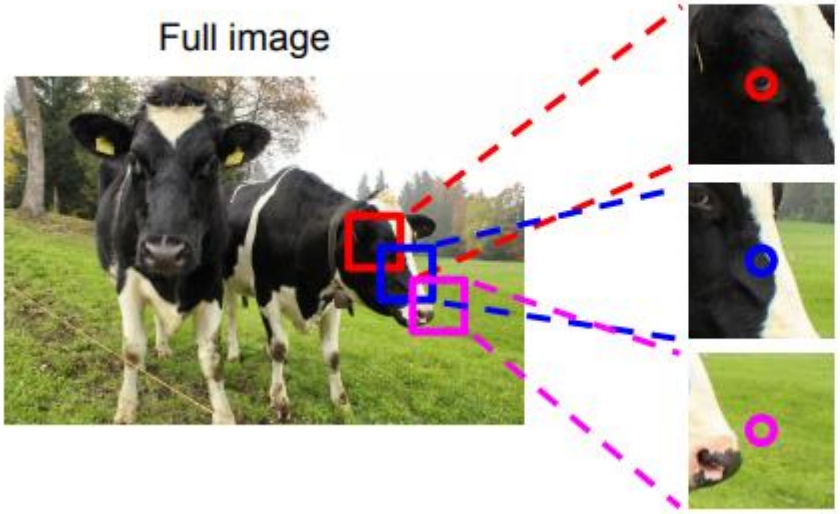
GRASS, CAT,
TREE, SKY, ...

Paired training data: for each training image, each pixel is labeled with a semantic category.

?

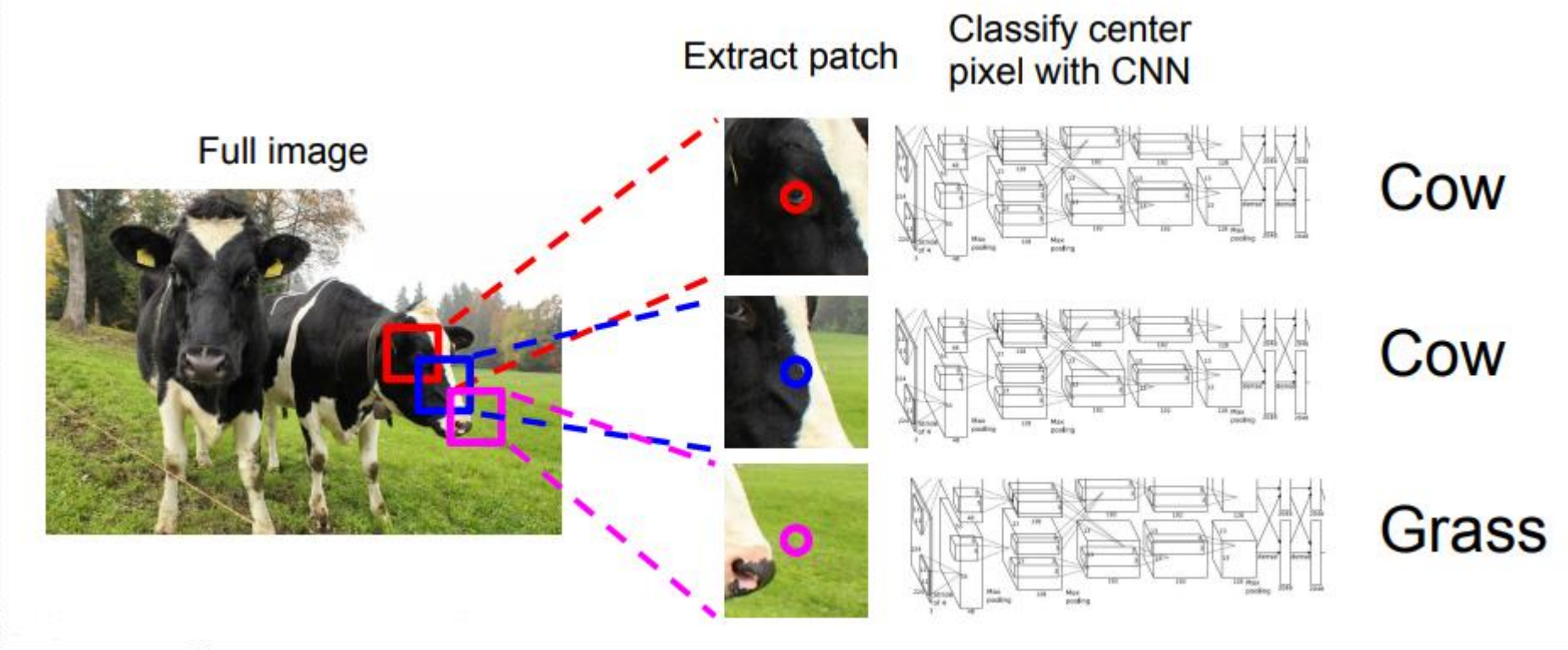
At test time, classify each pixel of a new image.

Idea: Sliding Window (1/2)



- How do we model this?

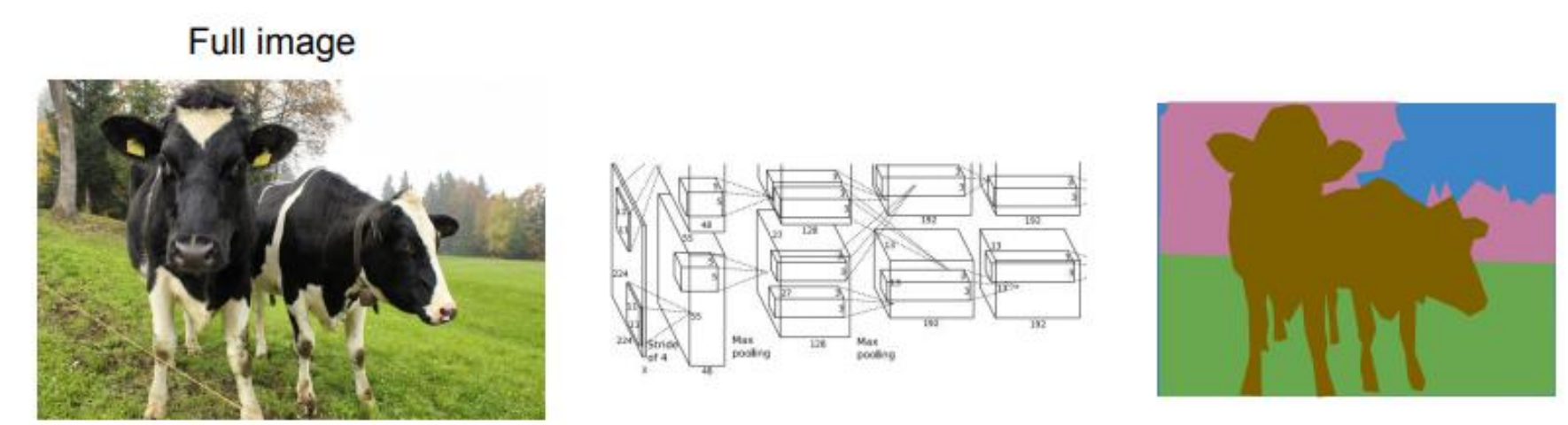
Idea: Sliding Window (2/2)



- Problem: Very inefficient!
- Not reusing shared features between overlapping patches

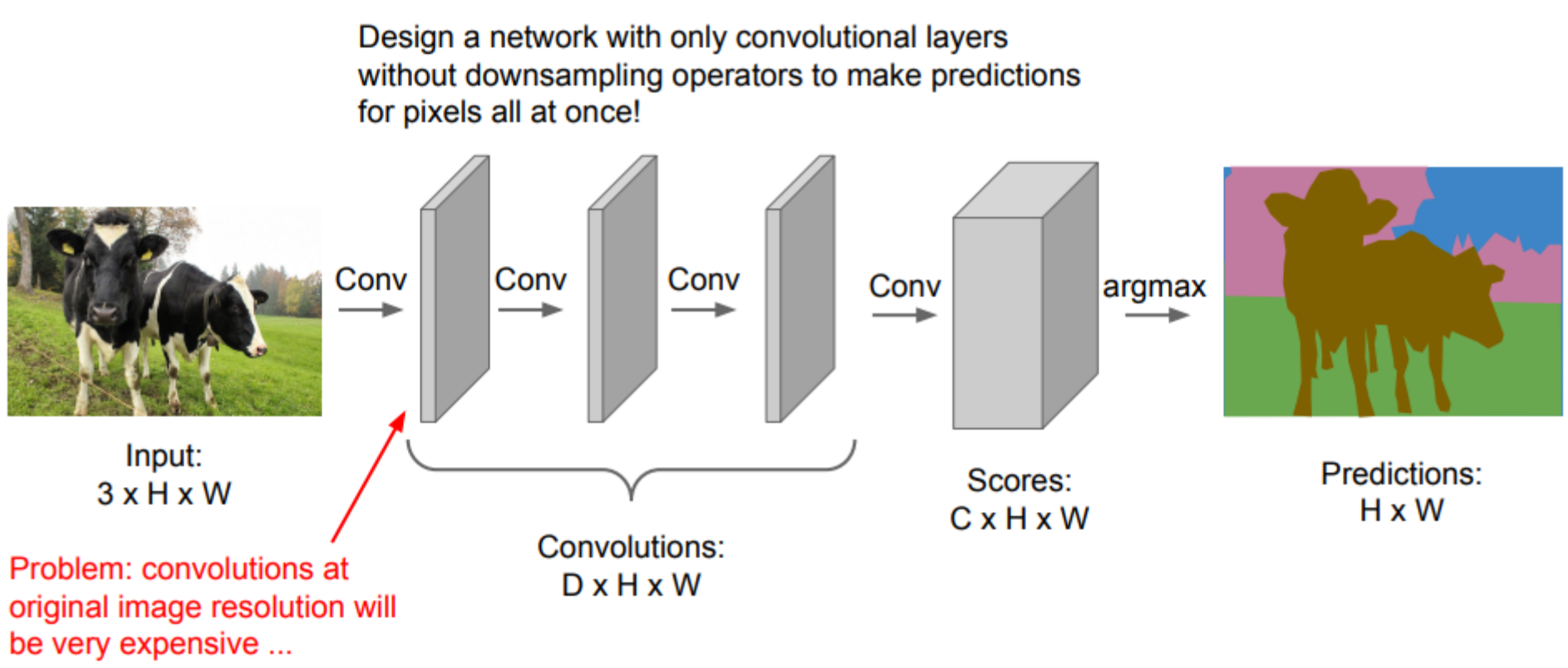
Idea: Convolution

- Encode the entire image with convolution net
- Do semantic segmentation on top



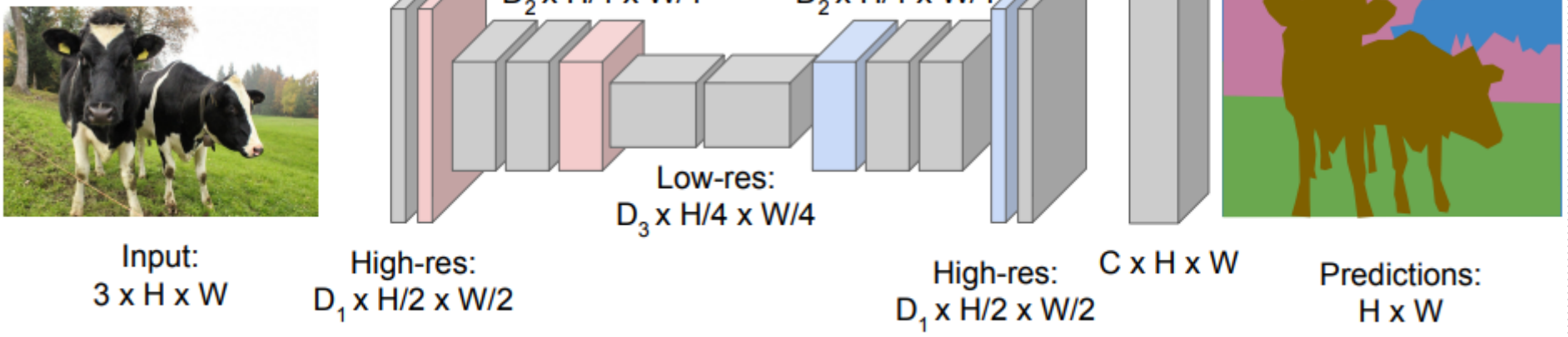
- Problem: Classification architectures often reduce feature spatial sizes to go deeper, but semantic segmentation requires the output size to be the same as input size

Idea: Fully Convolutional (1/4)

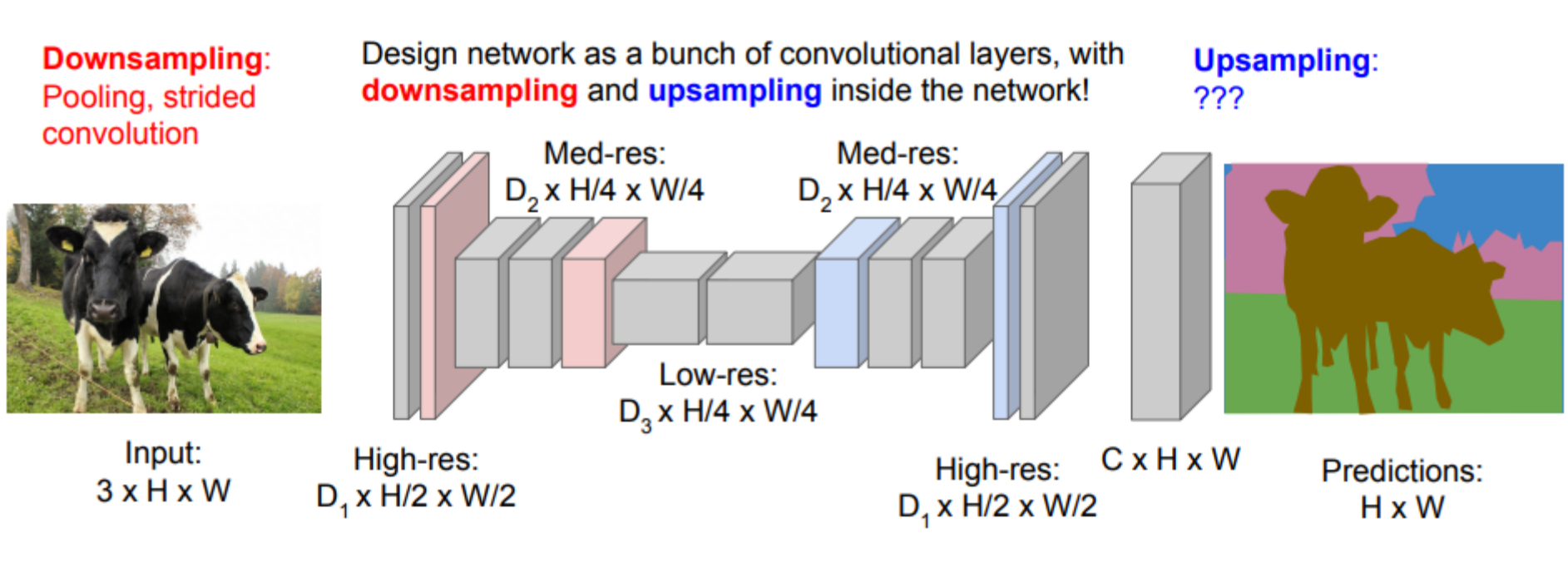


Idea: Fully Convolutional (2/4)

Design network as a bunch of convolutional layers, with **downsampling** and **upsampling** inside the network!

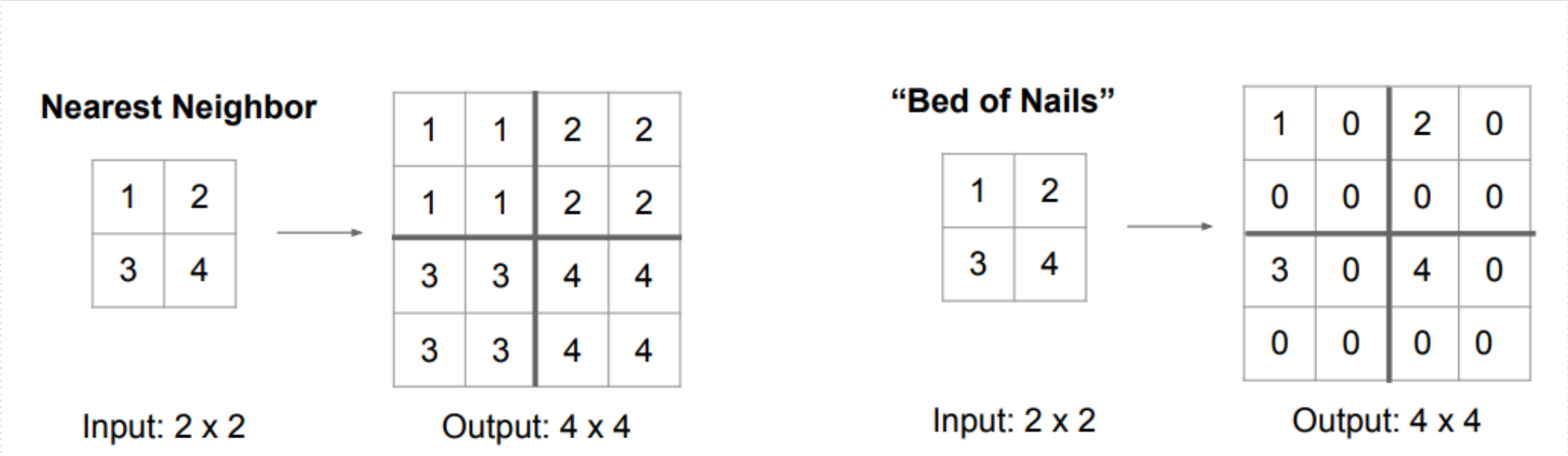


Idea: Fully Convolutional (3/4)



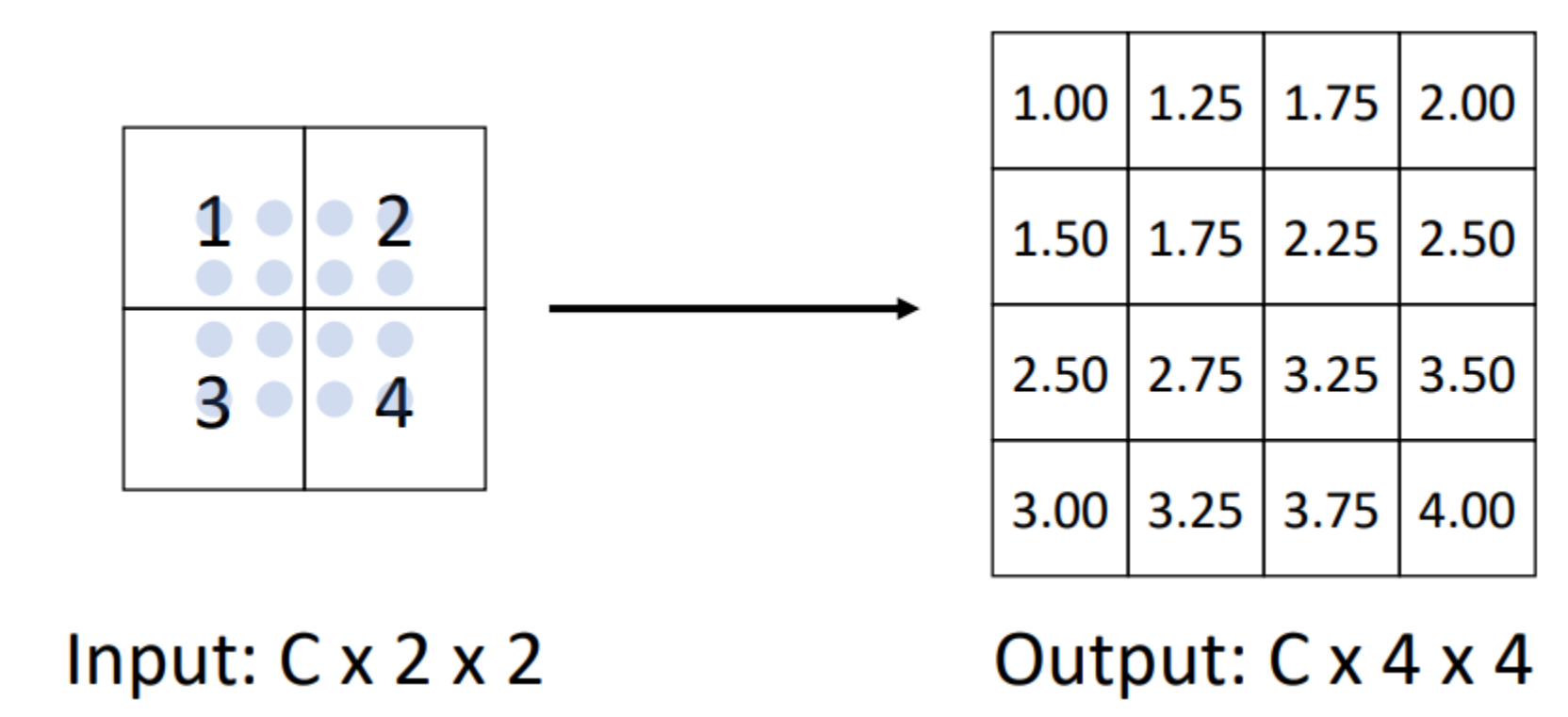
In-Network Upsampling (1/3)

- Unpooling



In-Network Upsampling (2/3)

- Bilinear Interpolation

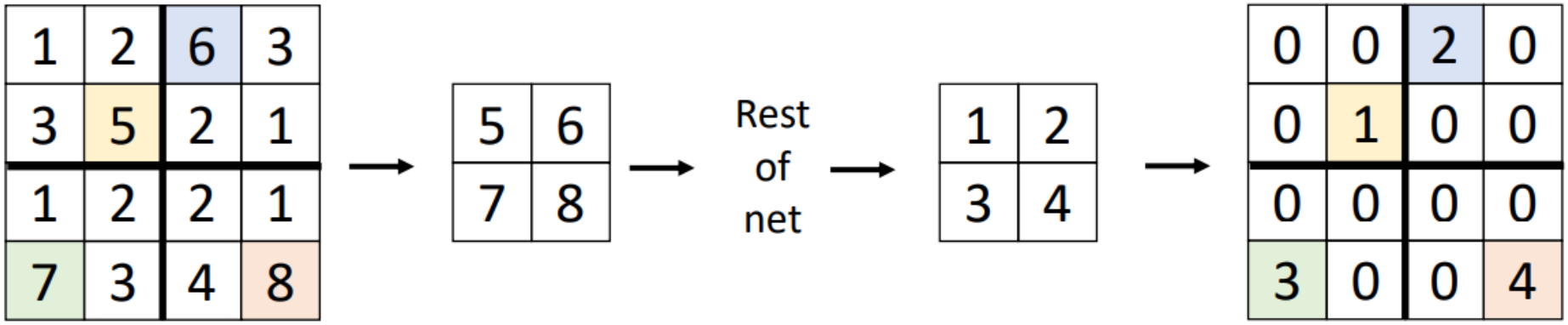


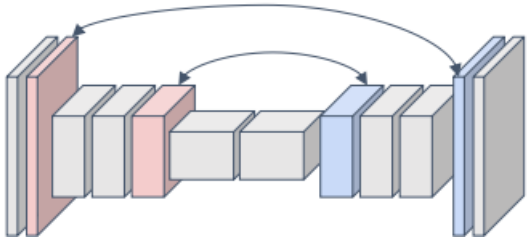
In-Network Upsampling (2/2)

- Max Unpooling

Max Pooling: Remember which position had the max

Max Unpooling: Place into remembered positions

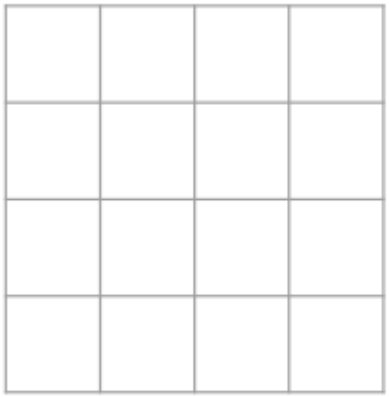




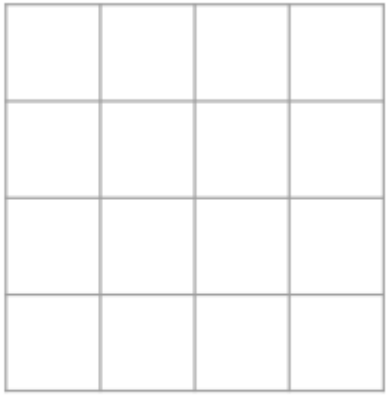
Pair each downsampling layer with an upsampling layer

Learnable Upsampling (1/3)

- Example 1
  - Normal 3×3 convolution
  - Stride 1
  - Pad 1



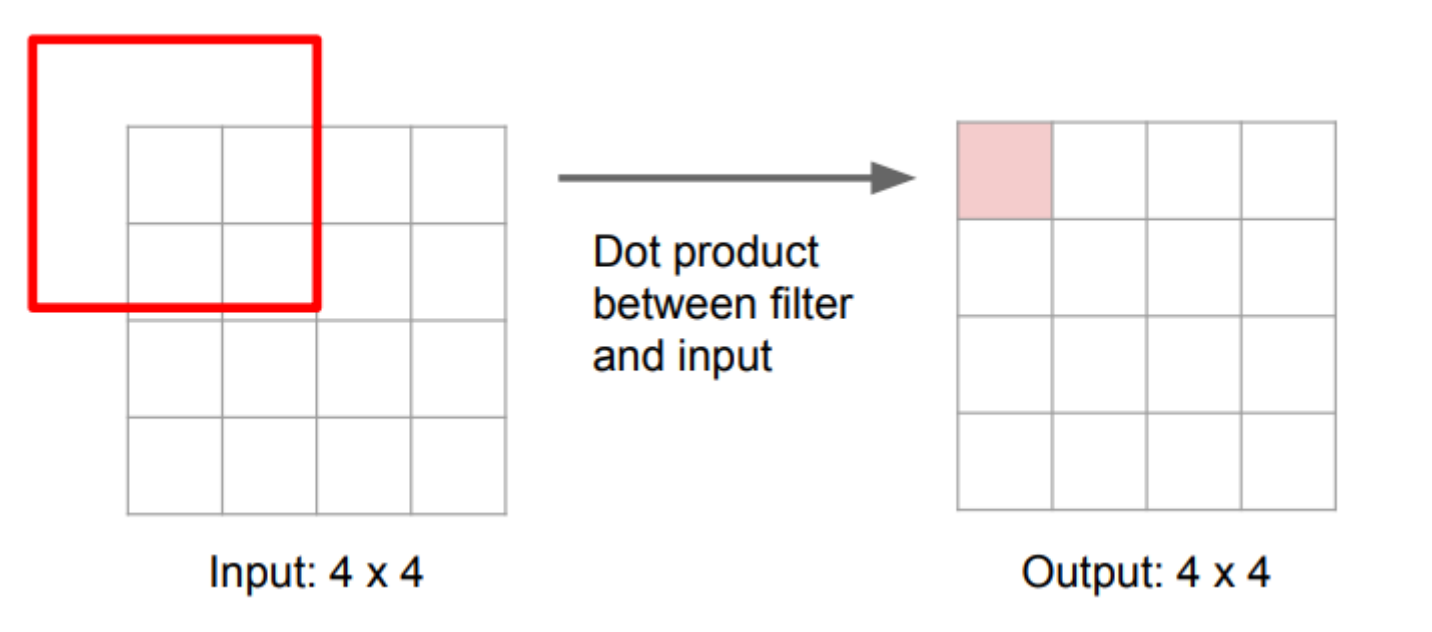
Input: 4 x 4



Output: 4 x 4

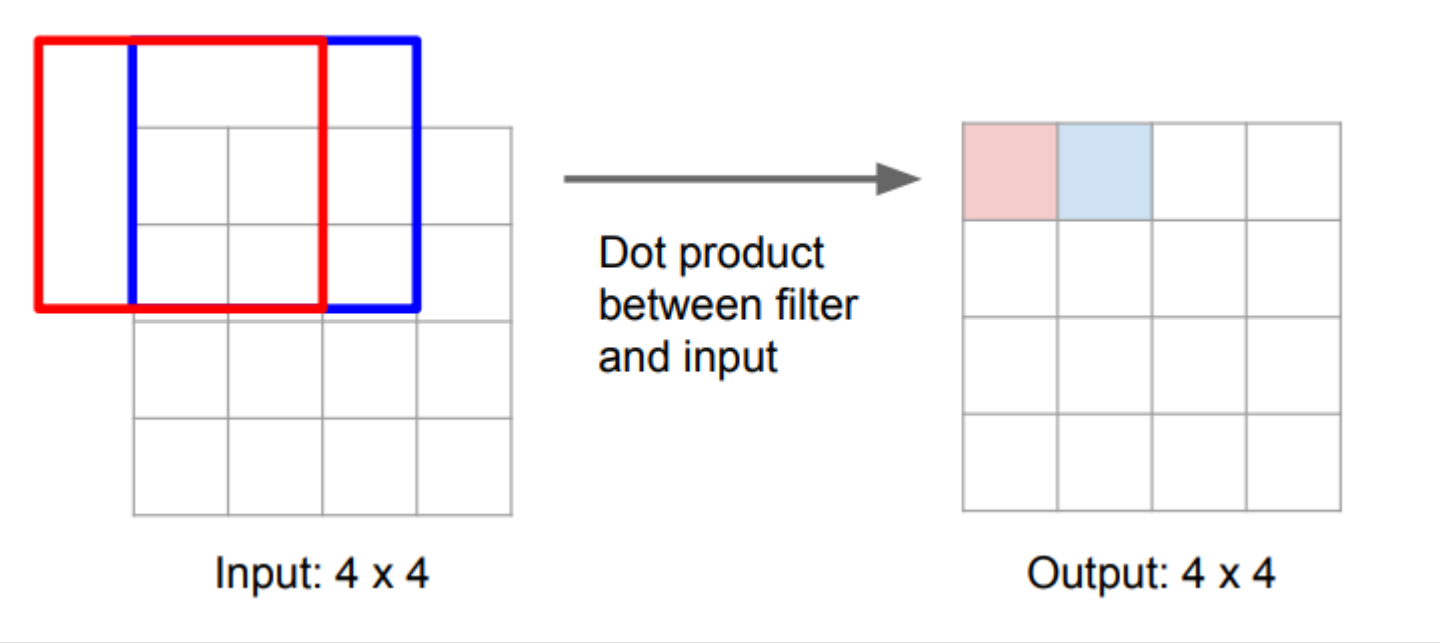
Learnable Upsampling (1/3)

- Example 1
  - Normal 3×3 convolution
  - Stride 1
  - Pad 1



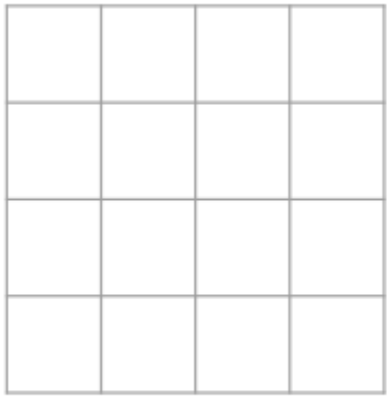
Learnable Upsampling (1/3)

- Example 1
  - Normal 3×3 convolution
  - Stride 1
  - Pad 1

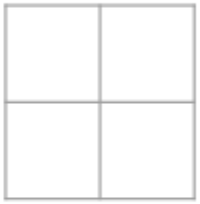


Learnable Upsampling (2/3)

- Example 2
  - Normal 3×3 convolution
  - Stride **2**
  - Pad 1



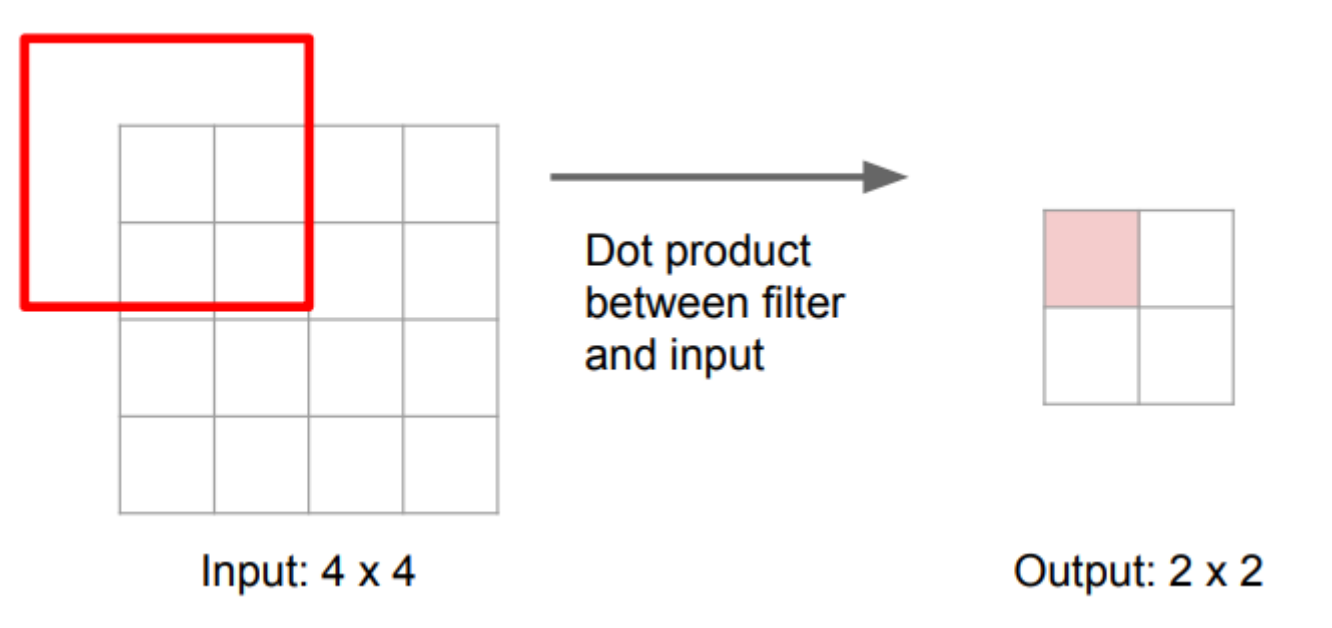
Input: 4 x 4



Output: 2 x 2

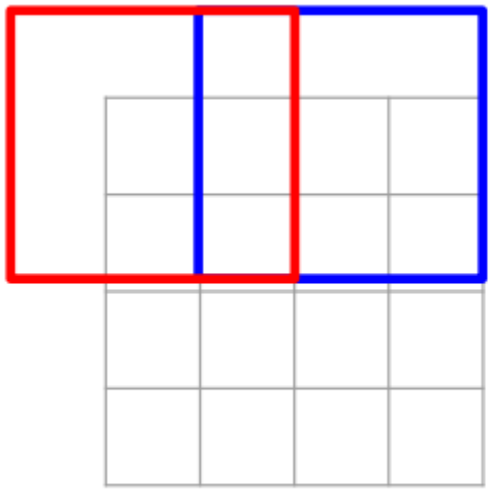
Learnable Upsampling (2/3)

- Example 2
  - Normal 3×3 convolution
  - Stride **2**
  - Pad 1



Learnable Upsampling (2/3)

- Example 2
  - Normal 3×3 convolution
  - Stride **2**
  - Pad 1



Input: 4 x 4

Dot product between filter and input

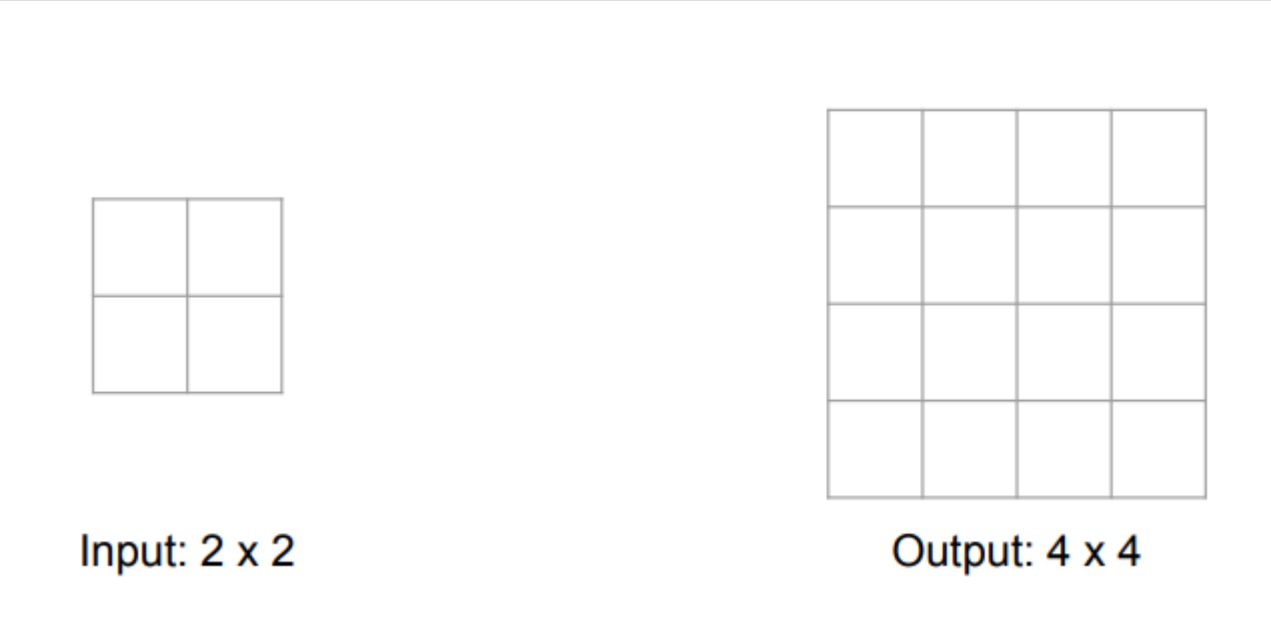
Output: 2 x 2

Filter moves 2 pixels in the input for every one pixel in the output

Stride gives ratio between movement in input and output

Learnable Upsampling (3/3)

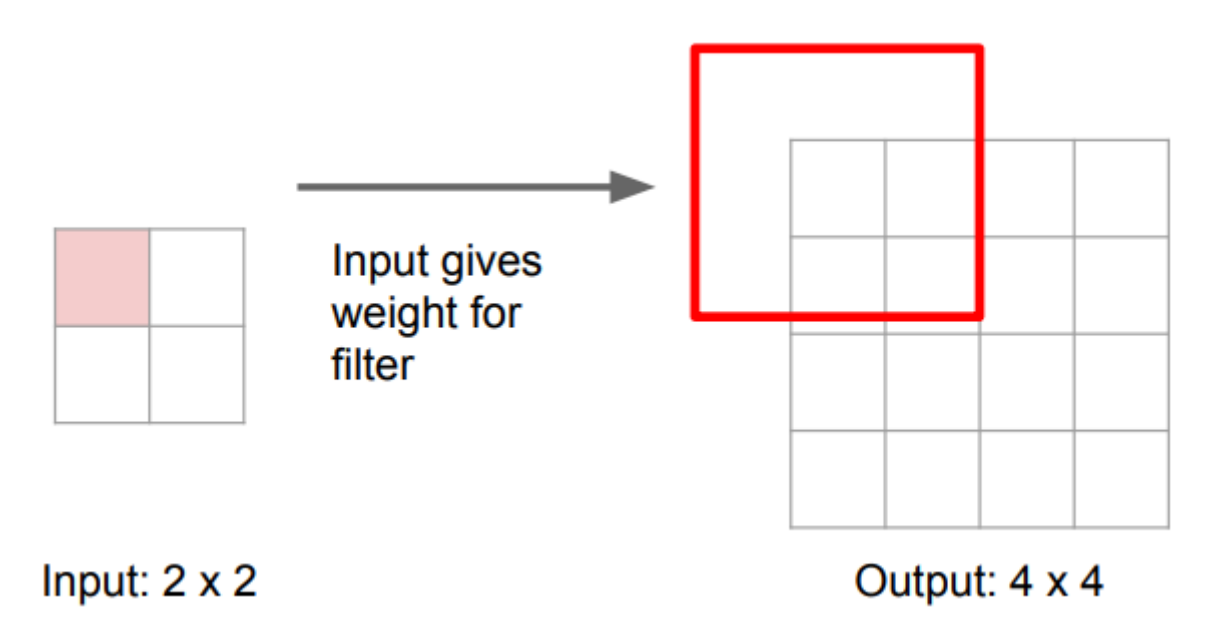
- Example 3:
  - 3×3 **transposed** convolution
  - Stride **2**
  - Pad 1



Q: Why is it called transposed convolution?

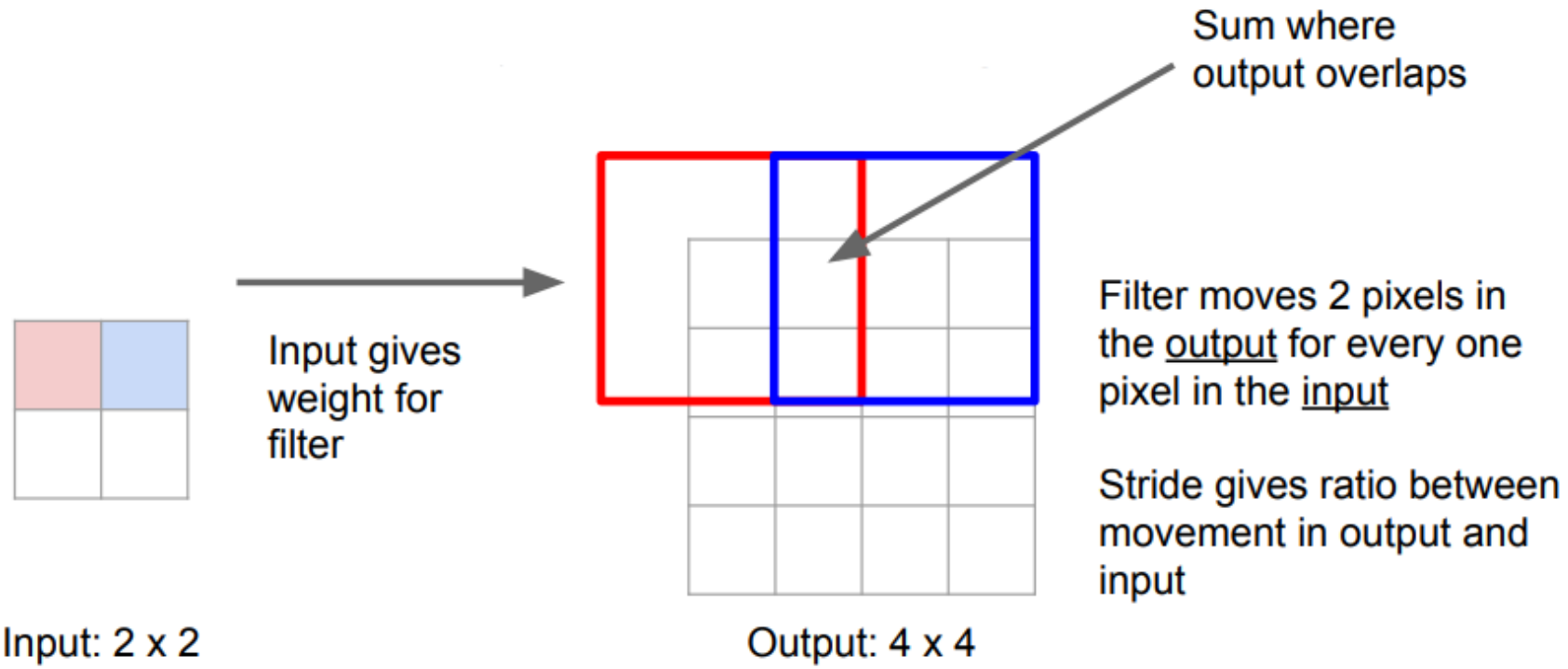
Learnable Upsampling (3/3)

- Example 3:
  - 3×3 **transposed** convolution
  - Stride **2**
  - Pad 1



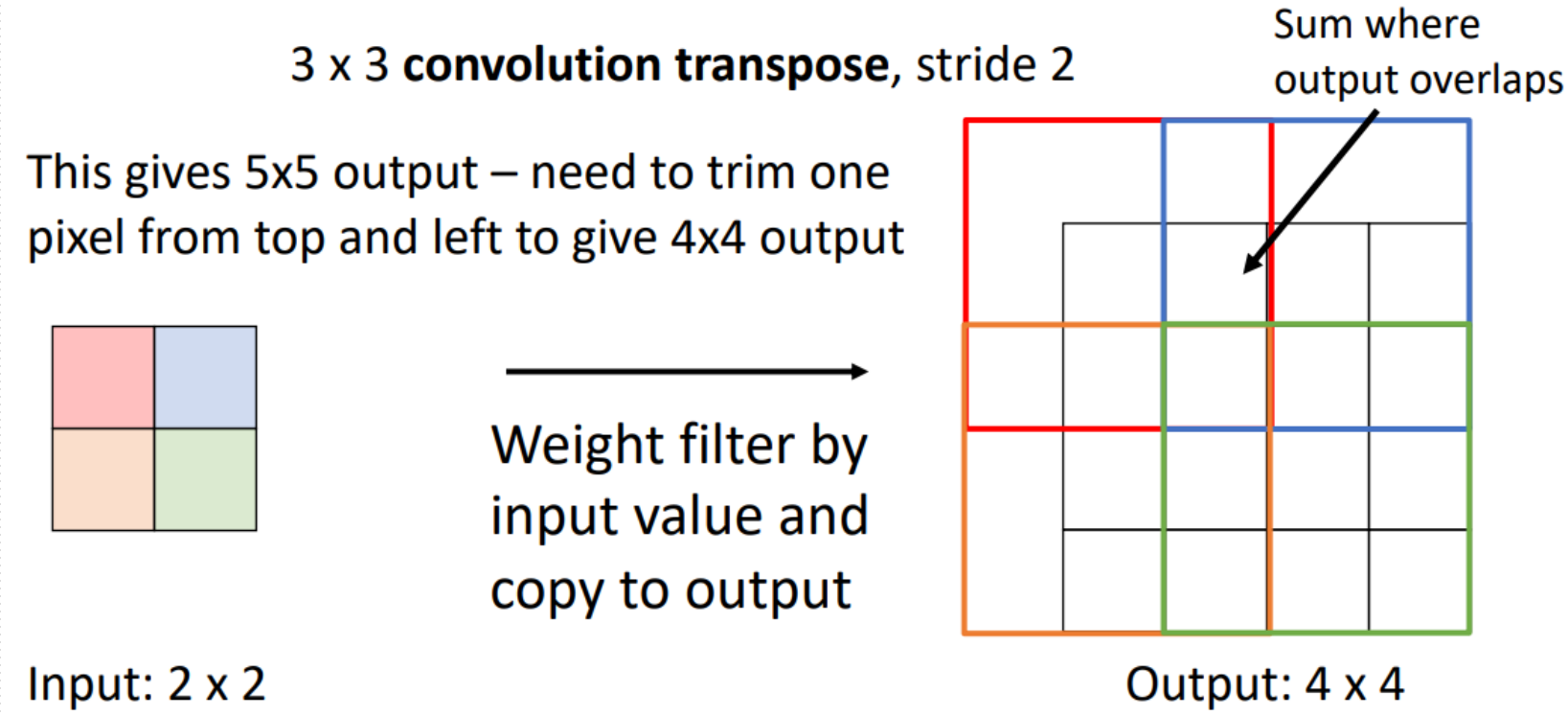
Learnable Upsampling (3/3)

- Example 3:
  - 3×3 **transposed** convolution
  - Stride **2**
  - Pad 1

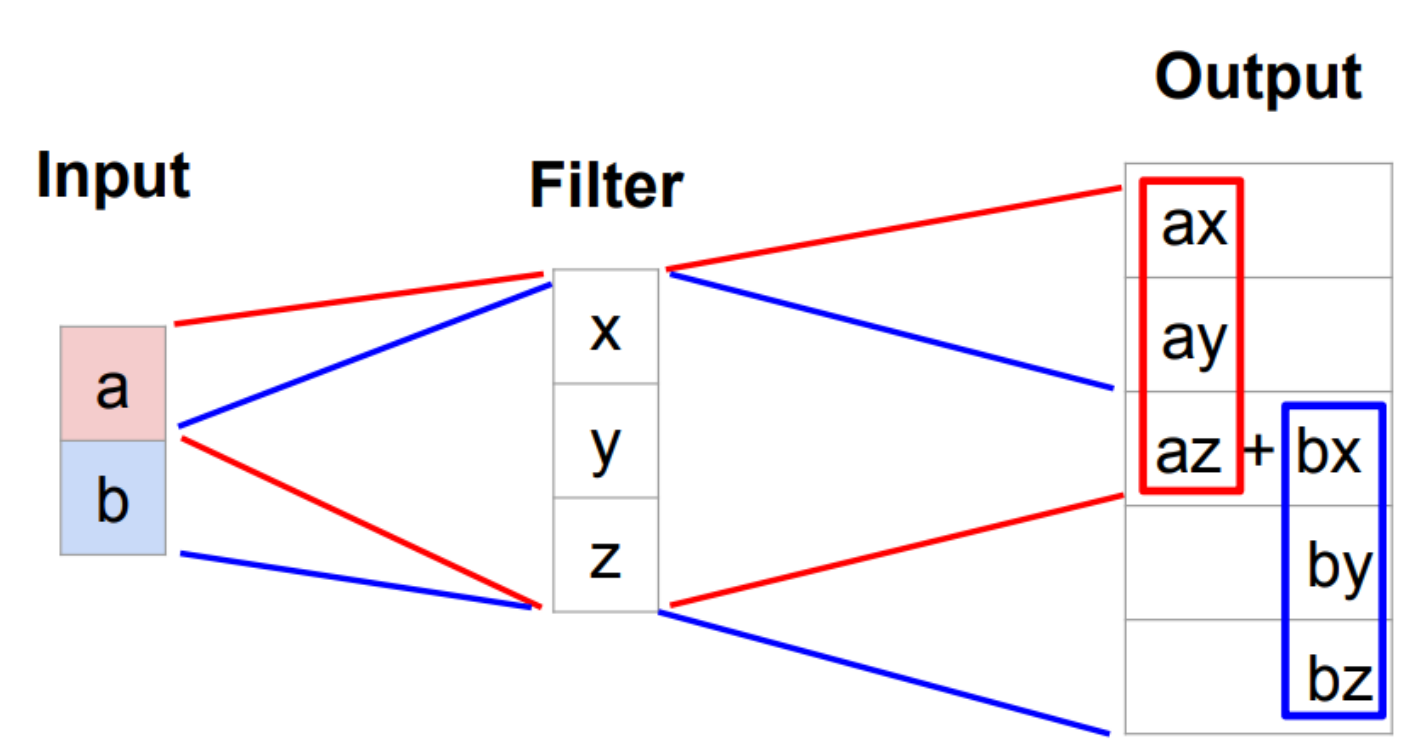


Learnable Upsampling (3/3)

- Example 3:
  - 3×3 **transposed** convolution
  - Stride **2**
  - Pad 1



Transpose Convolution: 1D Example (1/3)



- Output contains copies of the filter weighted by the input, summing at where at overlaps in the output
- Need to crop one pixel from output to make output exactly 2x input

Transpose Convolution: 1D Example (2/3)

We can express convolution in terms of a matrix multiplication

$$\vec{x} * \vec{a} = X\vec{a}$$

$$\begin{bmatrix} x & y & x & 0 & 0 & 0 \\ 0 & x & y & x & 0 & 0 \\ 0 & 0 & x & y & x & 0 \\ 0 & 0 & 0 & x & y & x \end{bmatrix} \begin{bmatrix} 0 \\ a \\ b \\ c \\ d \\ 0 \end{bmatrix} = \begin{bmatrix} ay + bz \\ ax + by + cz \\ bx + cy + dz \\ cx + dy \end{bmatrix}$$

Example: 1D conv, kernel size=3, stride=1, padding=1

Transposed convolution multiplies by the transpose of the same matrix:

$$\vec{x} *^T \vec{a} = X^T \vec{a}$$

$$\begin{bmatrix} x & 0 & 0 & 0 \\ y & x & 0 & 0 \\ z & y & x & 0 \\ 0 & z & y & x \\ 0 & 0 & z & y \\ 0 & 0 & 0 & z \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} ax \\ ay + bx \\ az + by + cx \\ bz + cy + dx \\ cz + dy \\ dz \end{bmatrix}$$

When stride=1, transposed conv is just a regular conv (with different padding rules)

Transpose Convolution: 1D Example (3/3)

We can express convolution in terms of a matrix multiplication

$$\vec{x} * \vec{a} = X\vec{a}$$

$$\begin{bmatrix} x & y & x & 0 & 0 & 0 \\ 0 & 0 & x & y & x & 0 \end{bmatrix} \begin{bmatrix} 0 \\ a \\ b \\ c \\ d \\ 0 \end{bmatrix} = \begin{bmatrix} ay + bz \\ bx + cy + dz \end{bmatrix}$$

Example: 1D conv, kernel size=3, stride=2, padding=1

Transposed convolution multiplies by the transpose of the same matrix:

$$\vec{x} *^T \vec{a} = X^T \vec{a}$$

$$\begin{bmatrix} x & 0 \\ y & 0 \\ z & x \\ 0 & y \\ 0 & z \\ 0 & 0 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} ax \\ ay \\ az + bx \\ by \\ bz \\ 0 \end{bmatrix}$$

When stride>1, transposed convolution cannot be expressed as normal conv

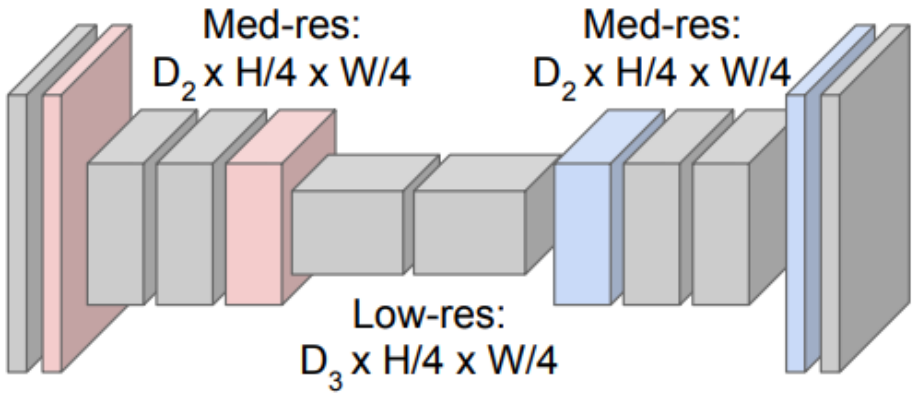
Idea: Fully Convolutional (4/4)

Downsampling:
Pooling, strided convolution

Design network as a bunch of convolutional layers, with **downsampling** and **upsampling** inside the network!

Upsampling:
Unpooling or strided transpose convolution

Input:
3 x H x W



High-res:
D_1 x H/2 x W/2

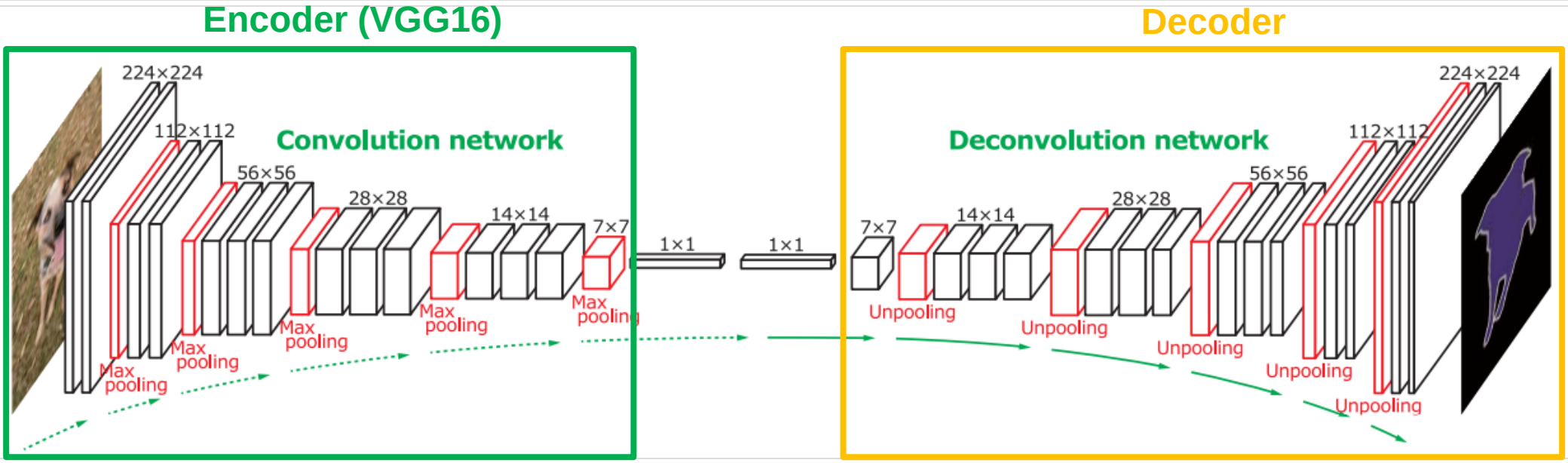
High-res:
D_1 x H/2 x W/2

Predictions:
H x W

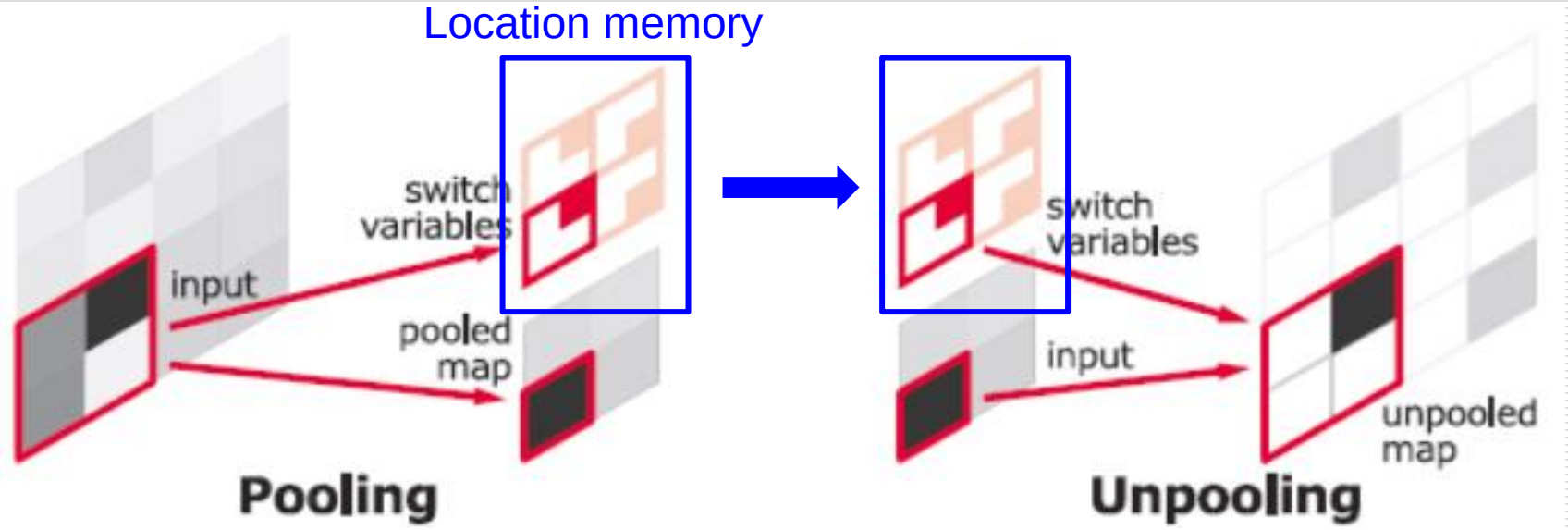
Image Segmentation Model

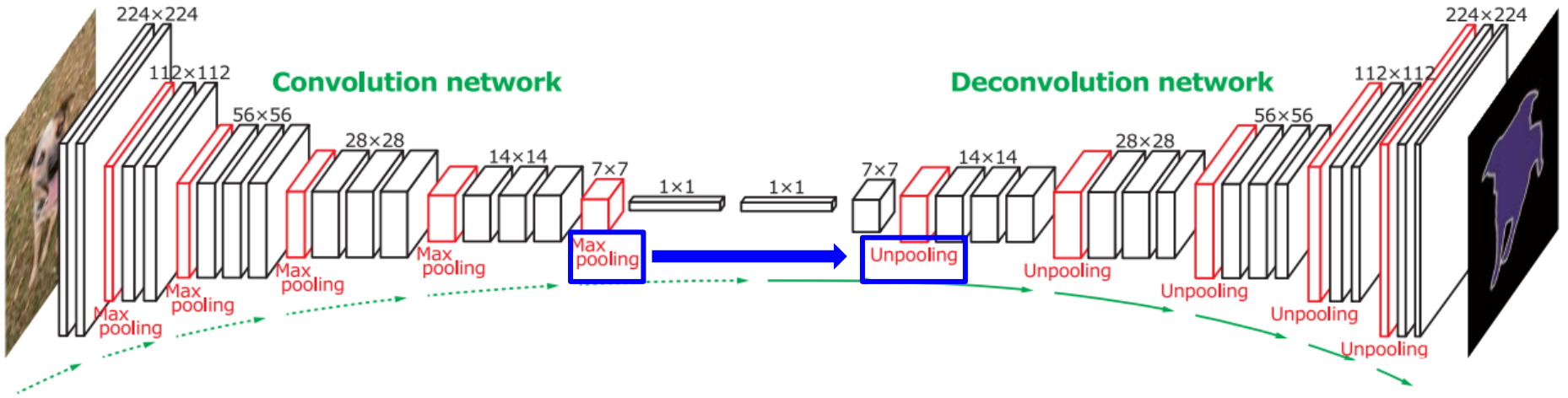
DeconvNet

- **Title: Learning Deconvolution Network for Semantic Segmentation**
- Maxpooling for downsampling
- Deconvolution & Unpooling for upsampling

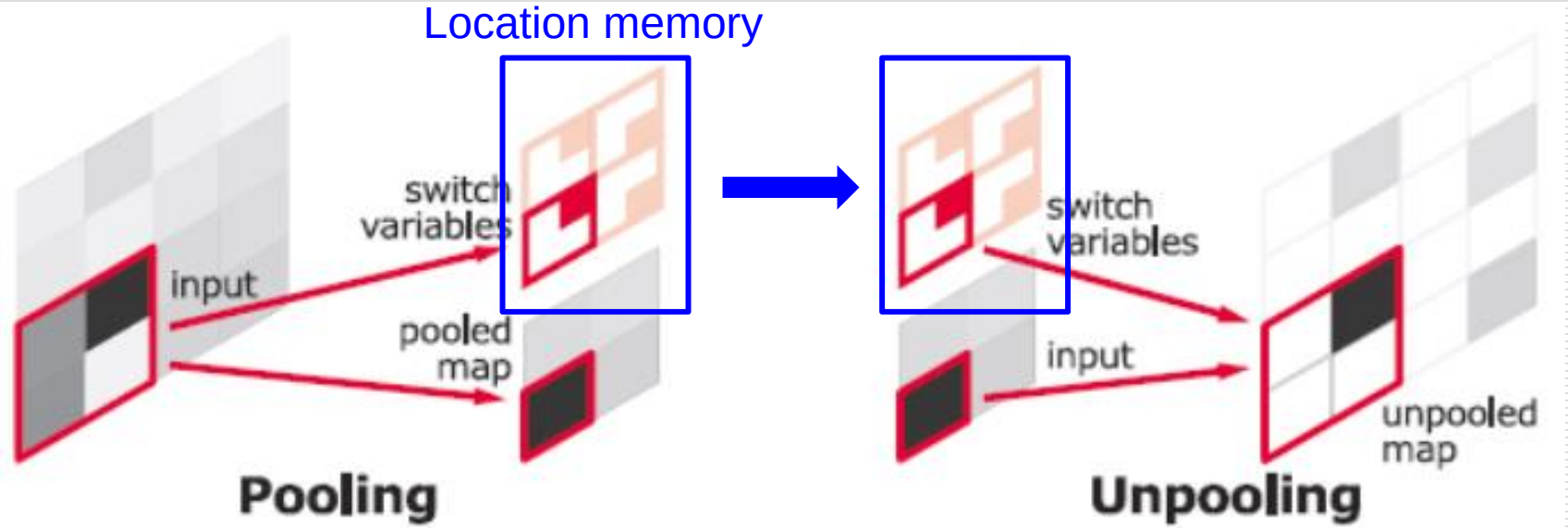


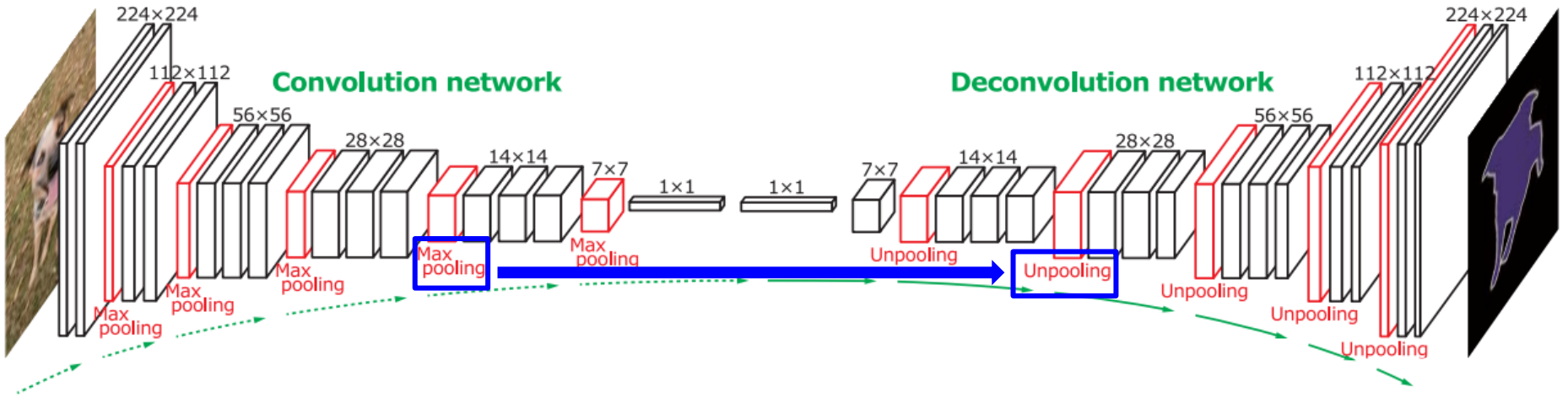
Unpooling



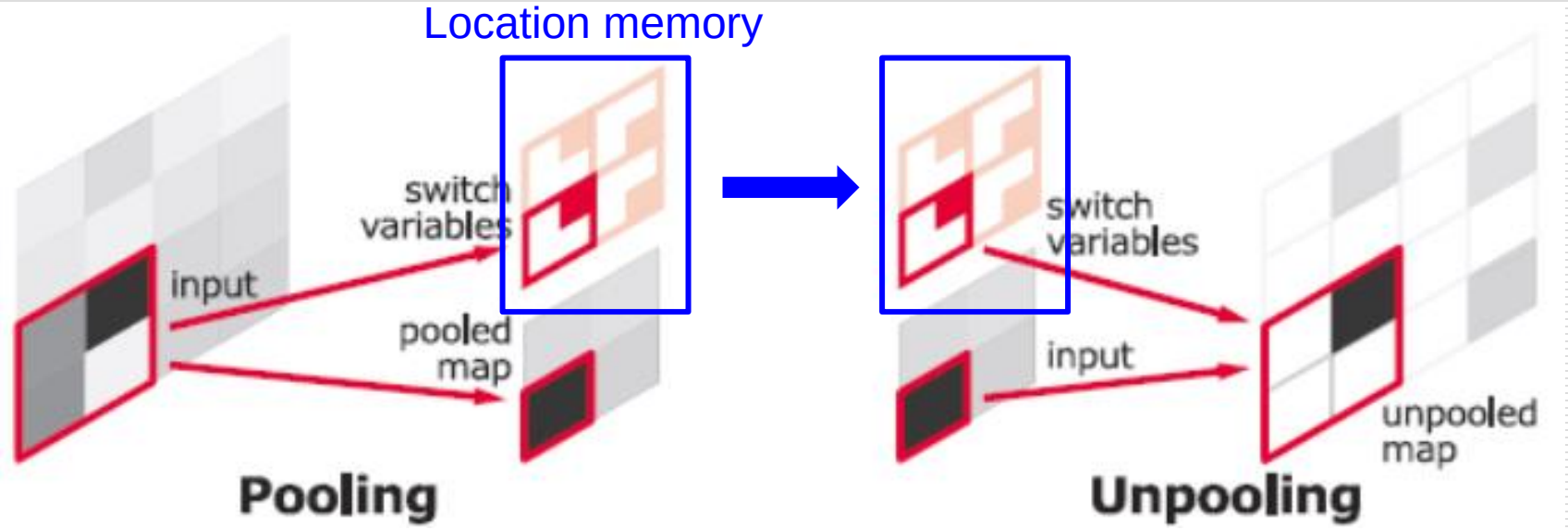


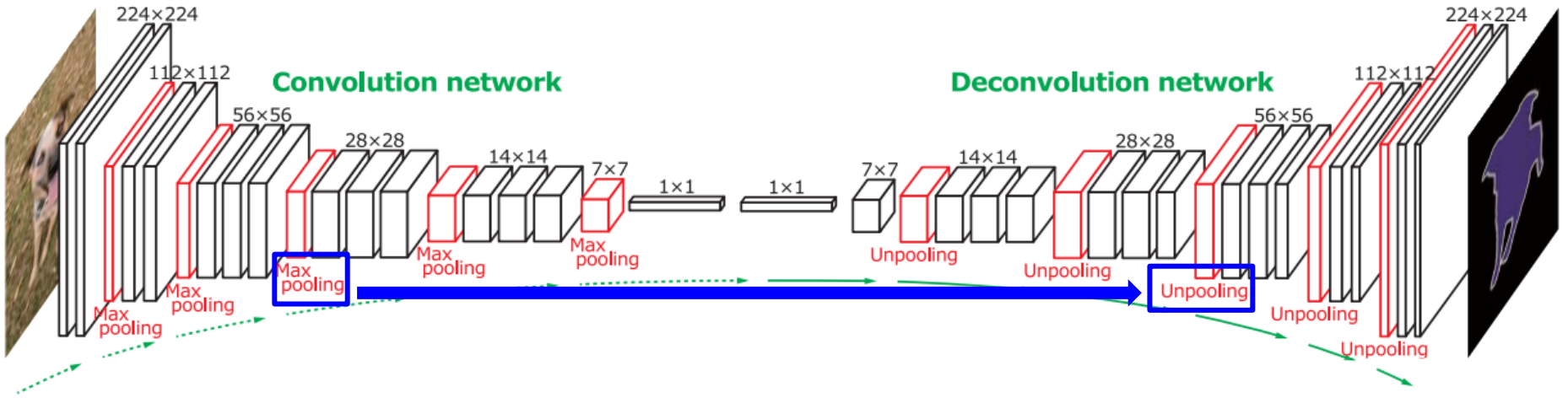
Unpooling



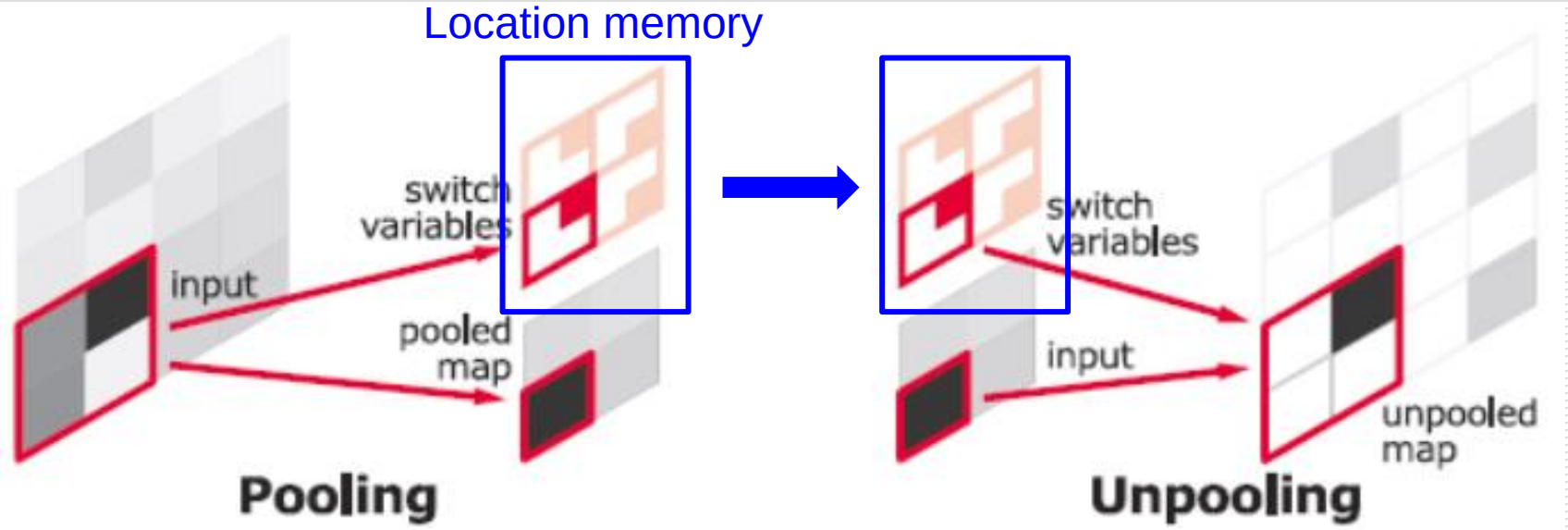


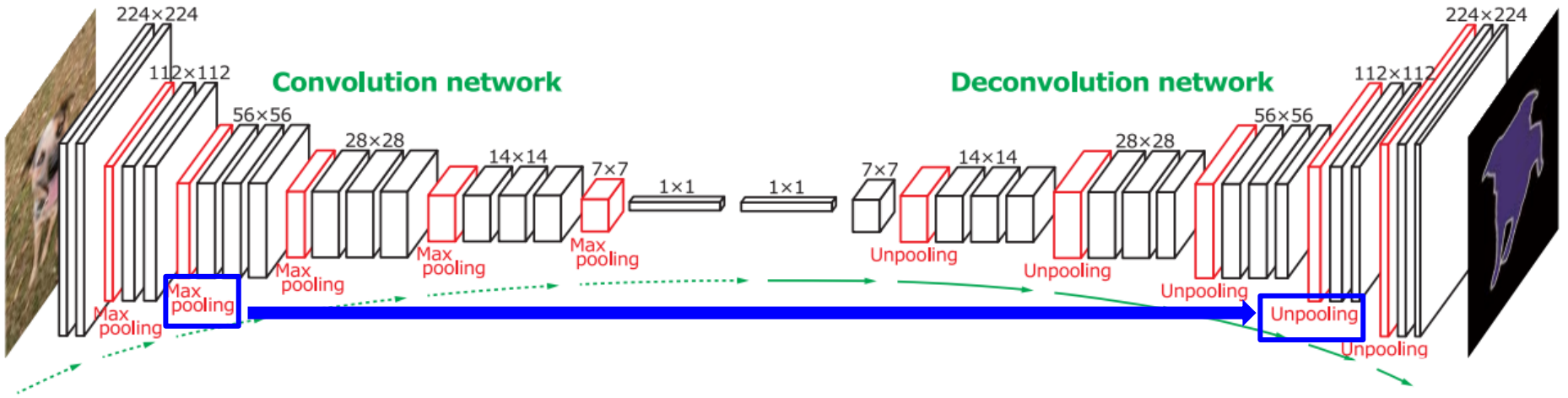
Unpooling



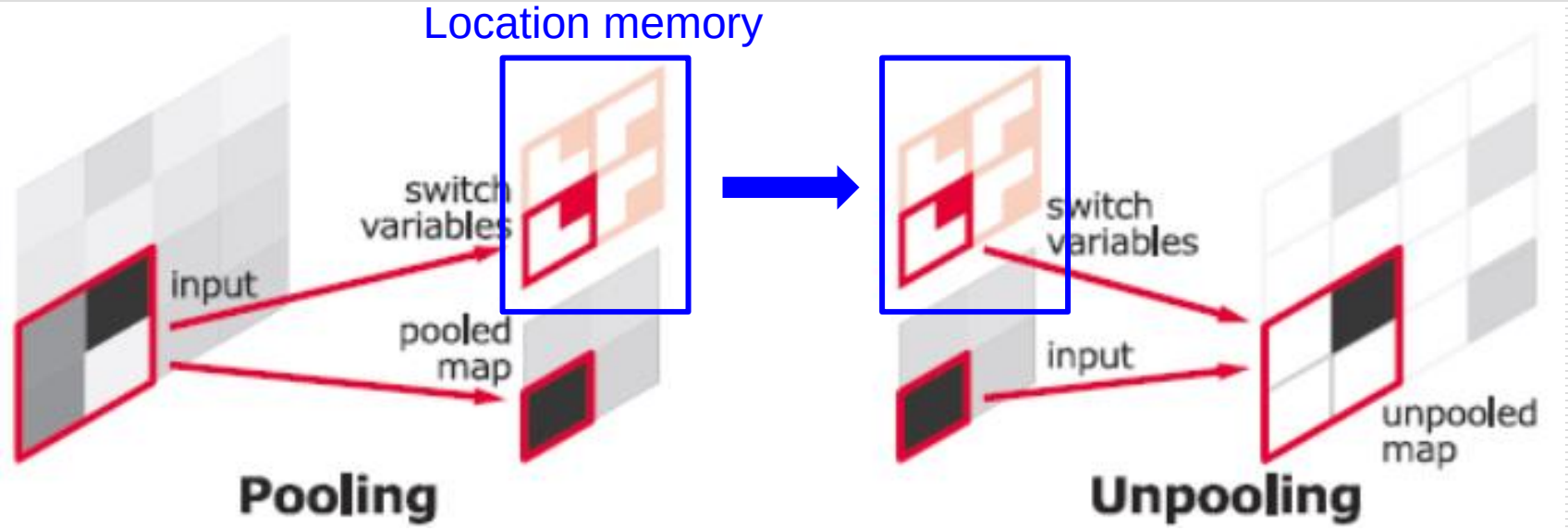


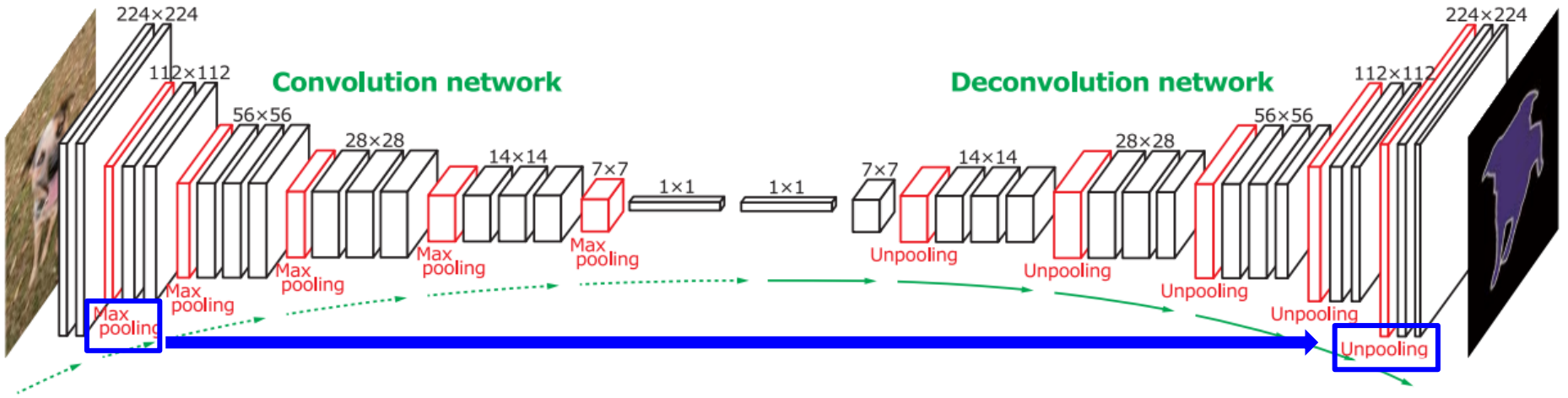
Unpooling





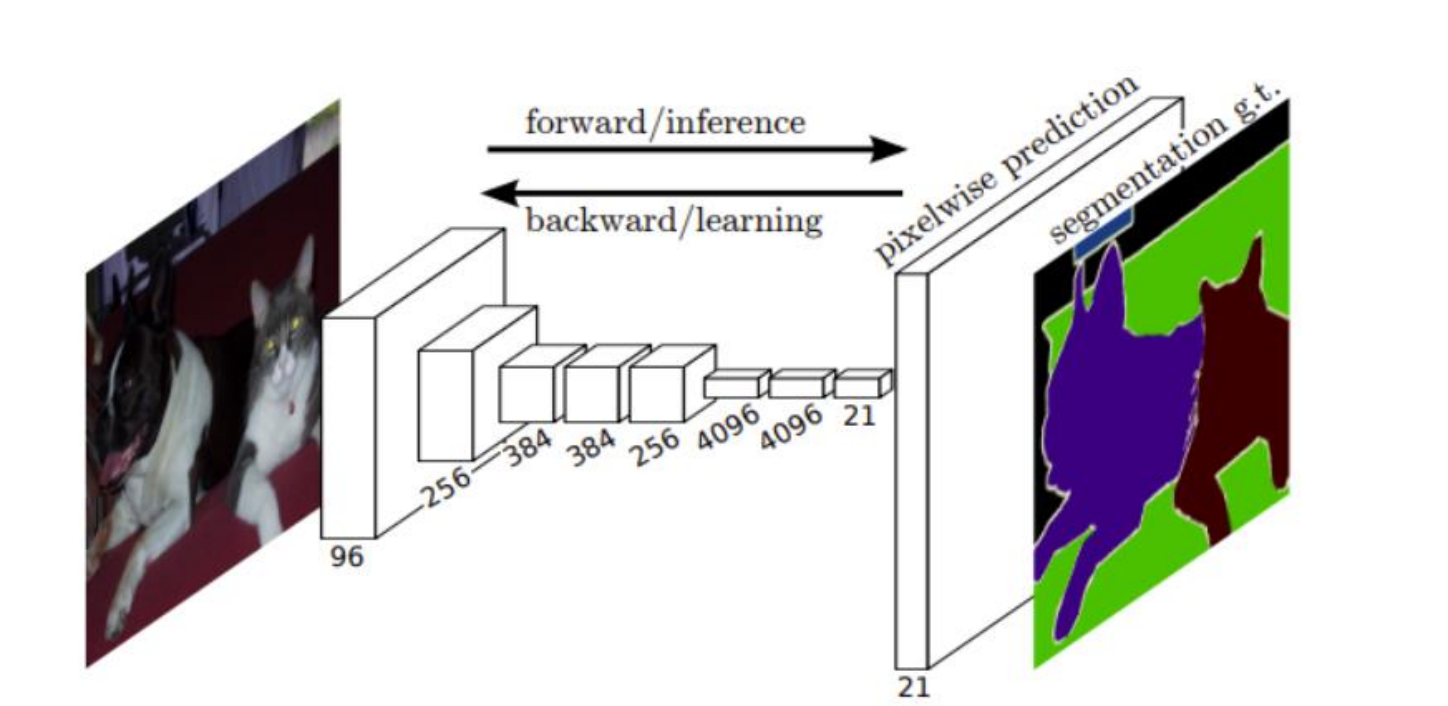
Unpooling





FCN

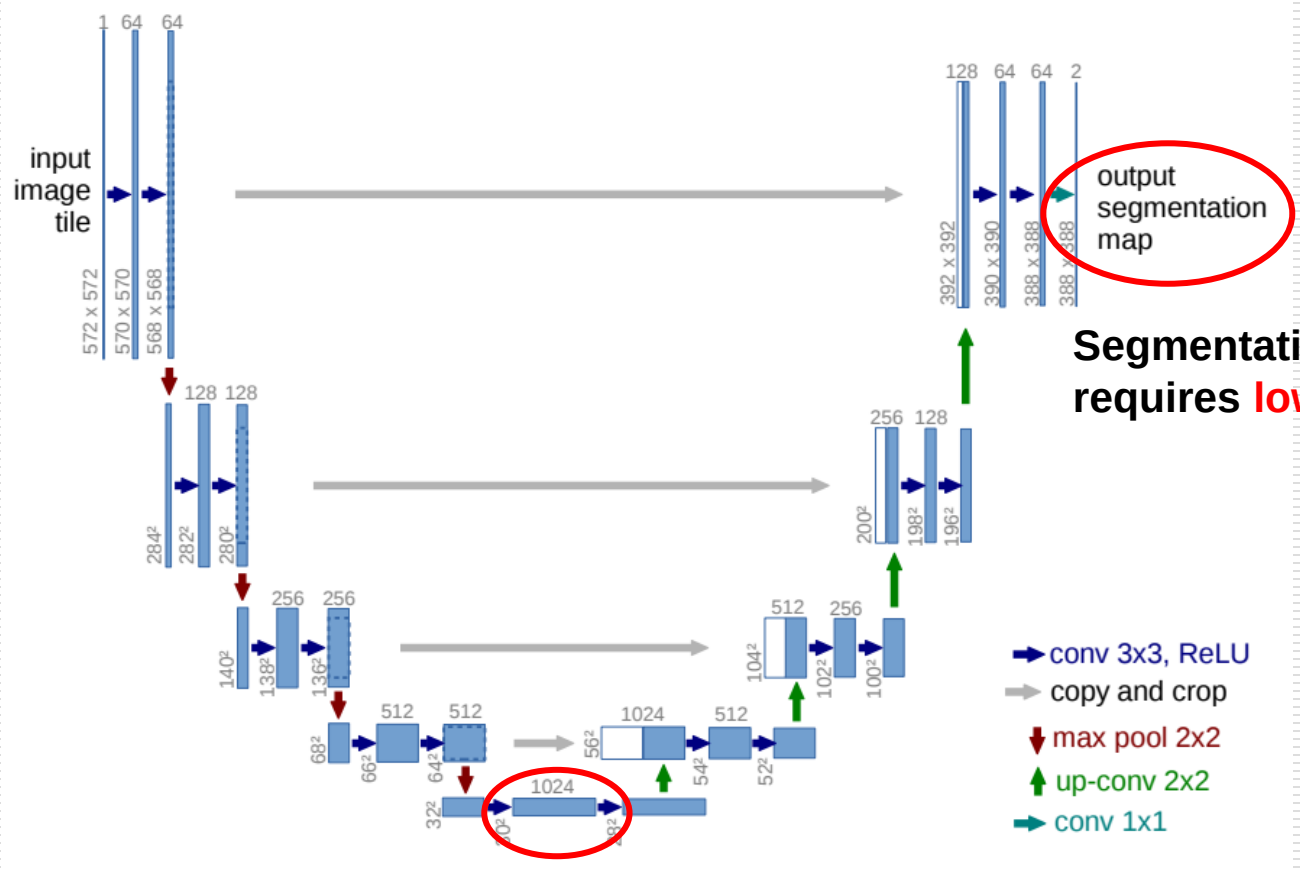
- Gradually generate higher-level feature maps as the network gets deeper
- Transposed Convolution to generate segmentation map



Low-level (ex. Edges, curves) Higher-level (ex. Context information)

U-Net

- **Title: U-Net: Convolutional Networks for Biomedical Image Segmentation**

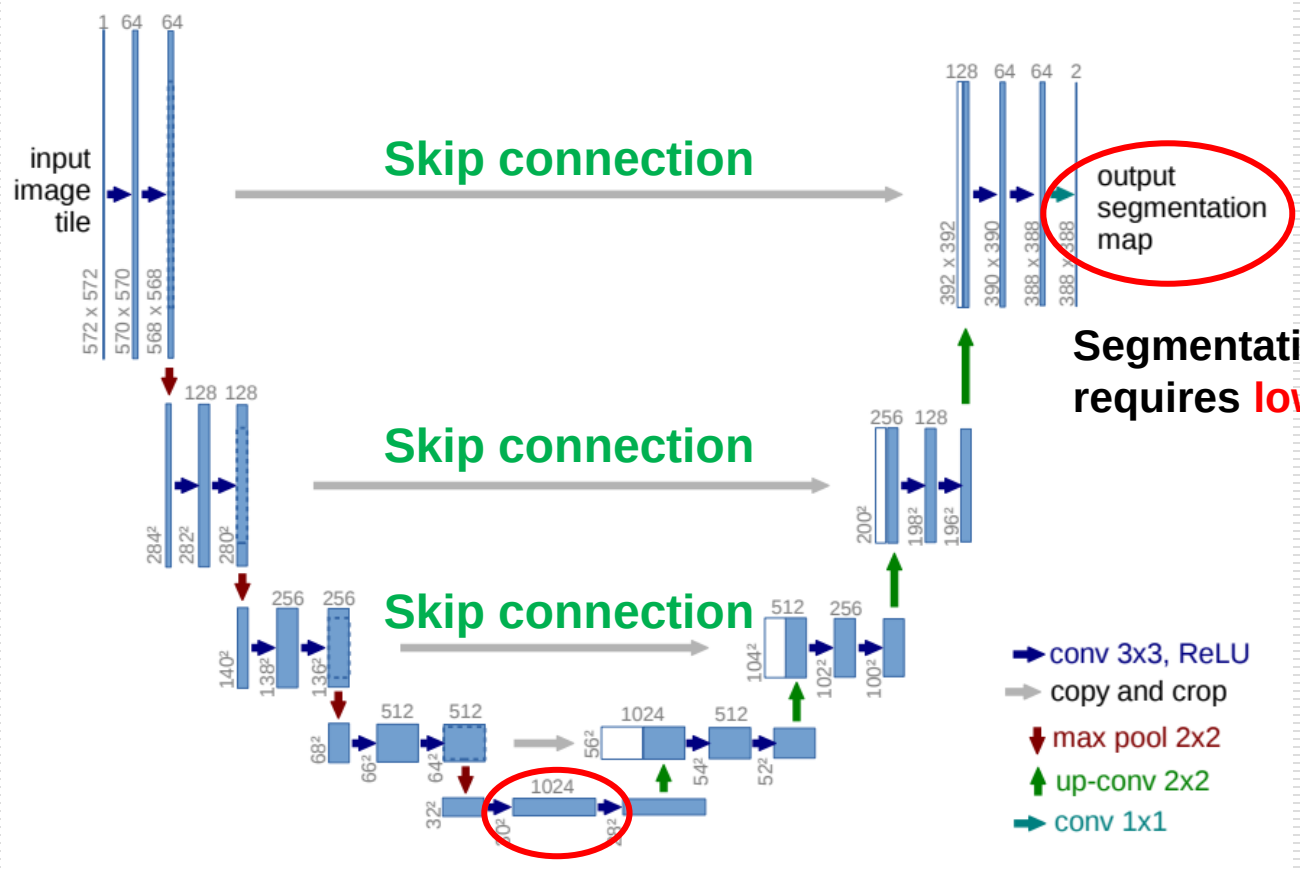


Segmentation map requires low-level

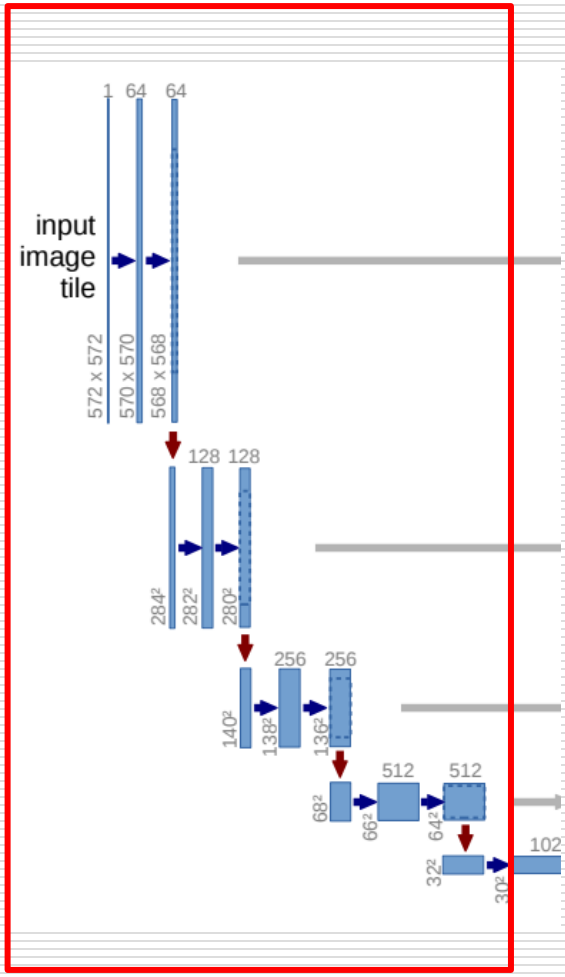
Contain high-level

U-Net

- **Title: U-Net: Convolutional Networks for Biomedical Image Segmentation**



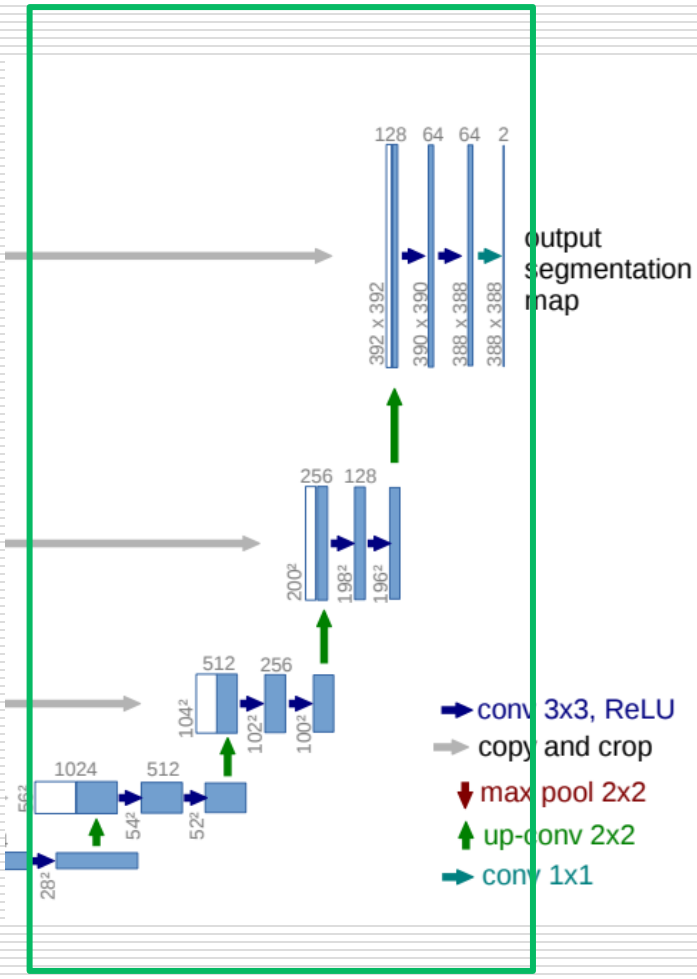
U-Net



Contracting path

- Contracting path
  - Channels increase (×2)
  - Resolution decreases (÷2)
  - Gradually generate higher-level features
  - Double Conv.
    - 3×3 filter
    - No padding
  - ReLU
  - 2×2 Max Pooling
    - Stride 2

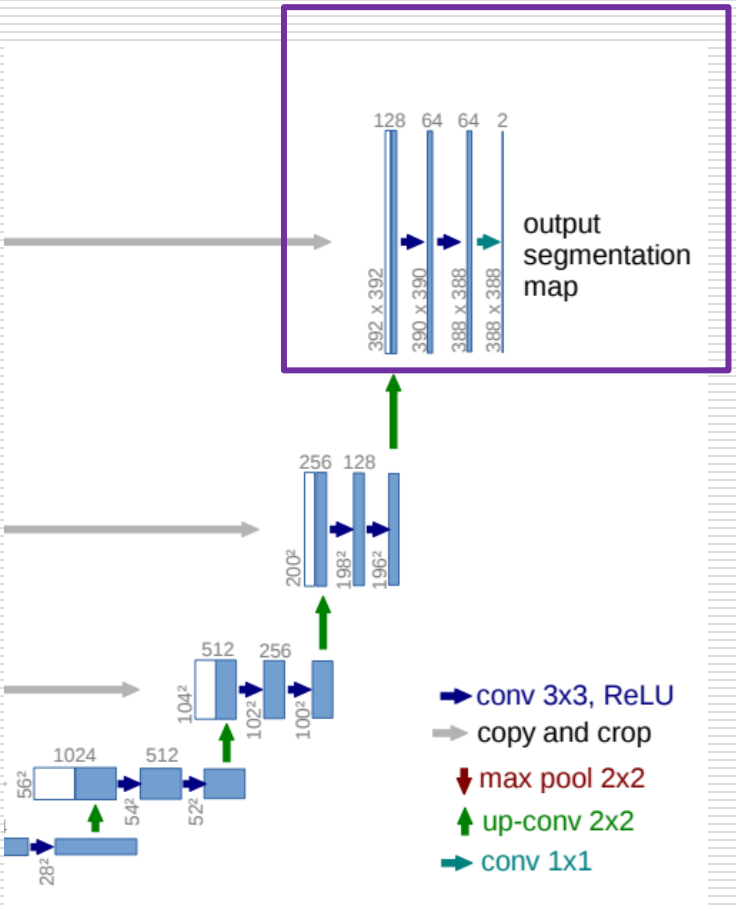
U-Net



Expansive path

- **Expansive path**
  - 2×2 Transpose convolution
  - Channels decrease (÷2)
  - Resolution increase (×2)
  - More precise localization
  - Double Conv.
  - ReLU
  - Skip Connection
    - Concatenation

U-Net

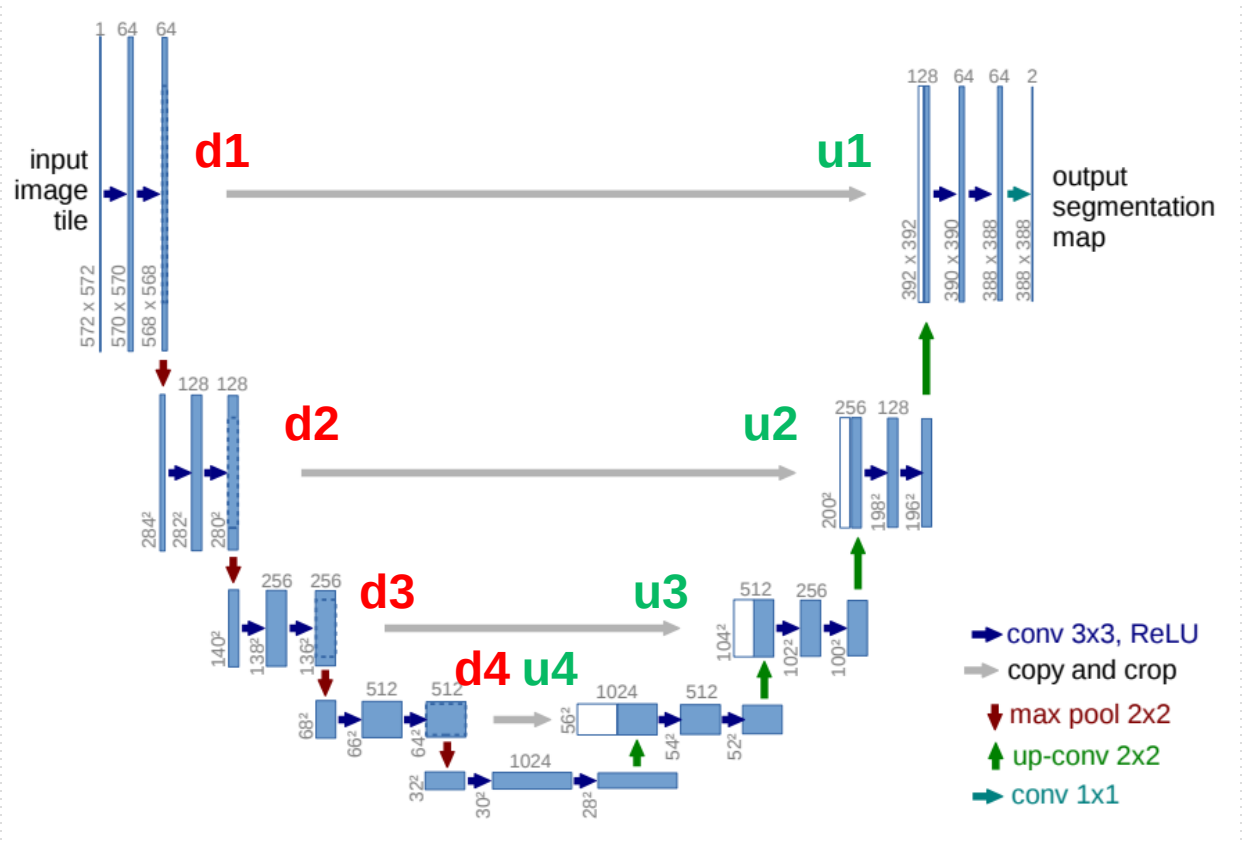


- **Final Layer**
  - Assign a class label to each pixel value
  - 1×1 conv layer (n filters)
    - n is the number of classes
  - Final output: (B, n, H, W)
  - Softmax to produce segmentation map

Final Layer

U-Net

Low level



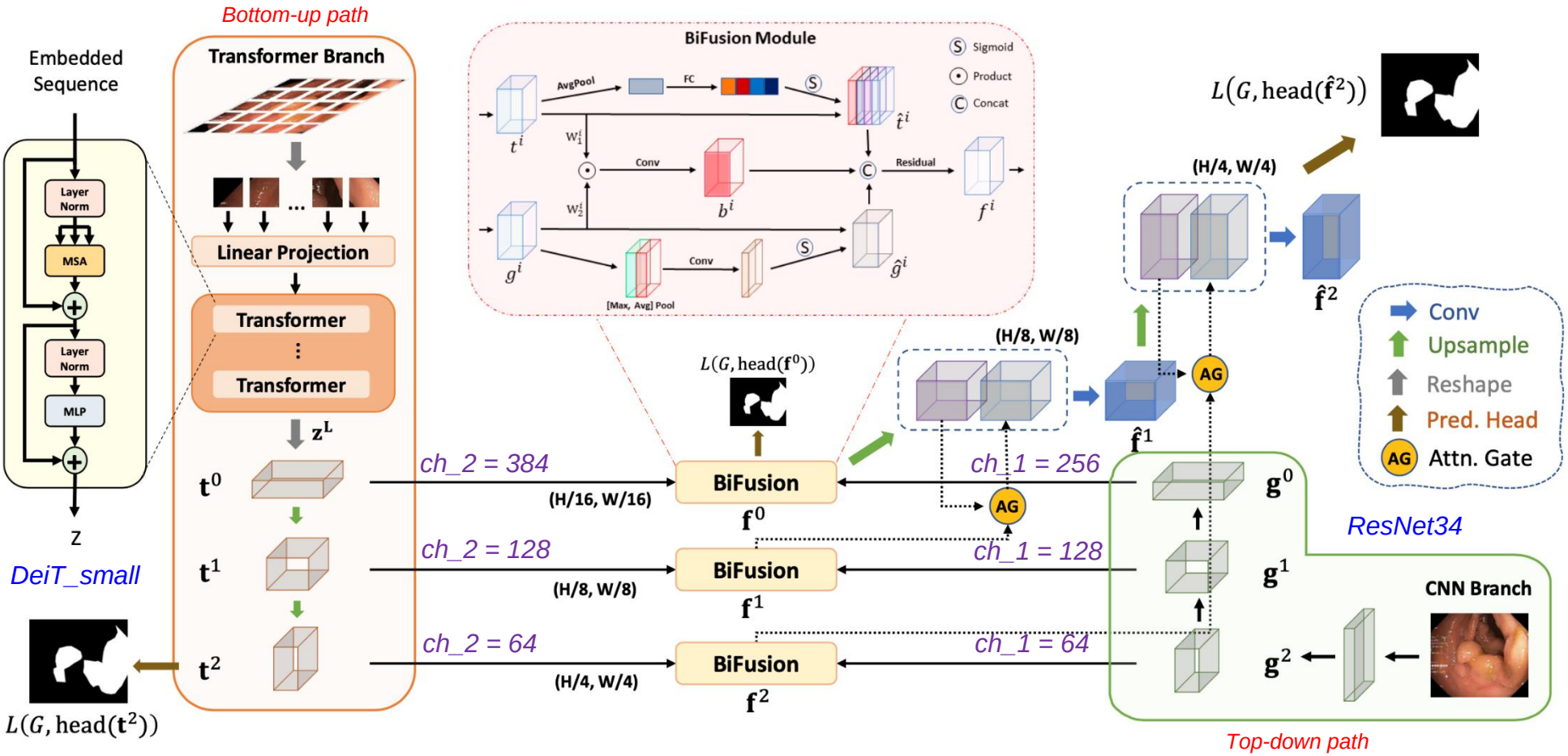
Higher level

Low level

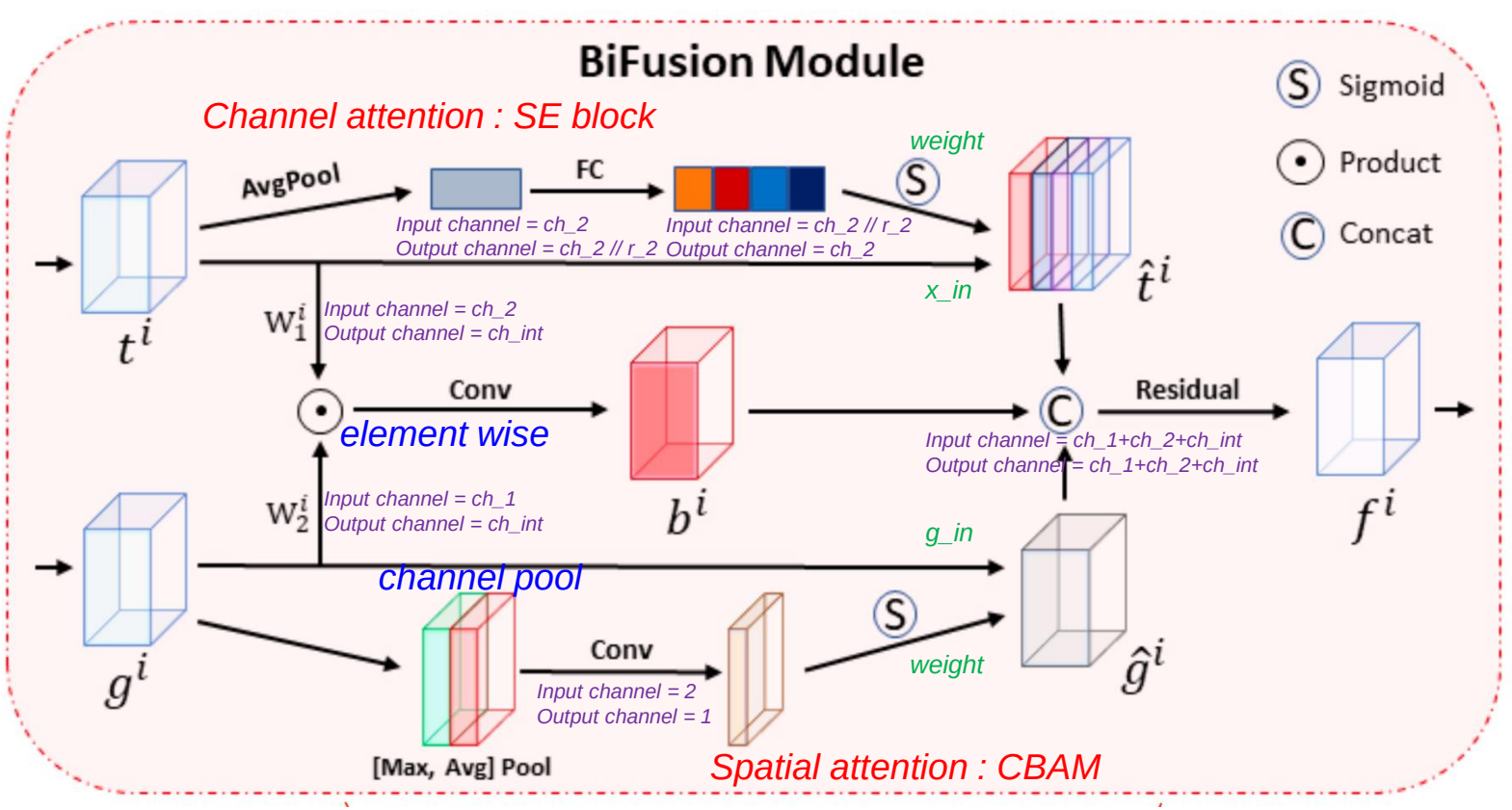
Higher level

TransFuse (1/2)

- **Title: TransFuse: Fusing Transformers and CNNs for Medical Image Segmentation**



TransFuse (2/2)



Paper: Segment Anything

Introduction

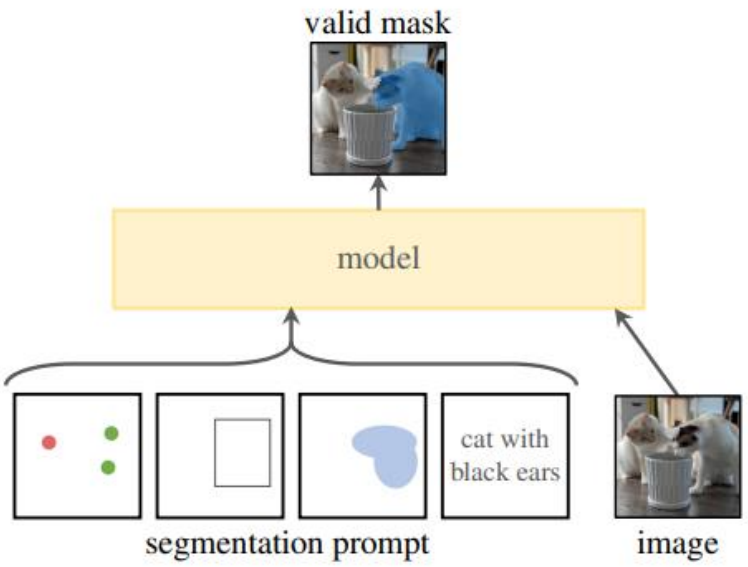
- Goal: build a **foundation model** for image segmentation
  - A promptable model
  - A broad dataset
  - Can solve downstream segmentation problems

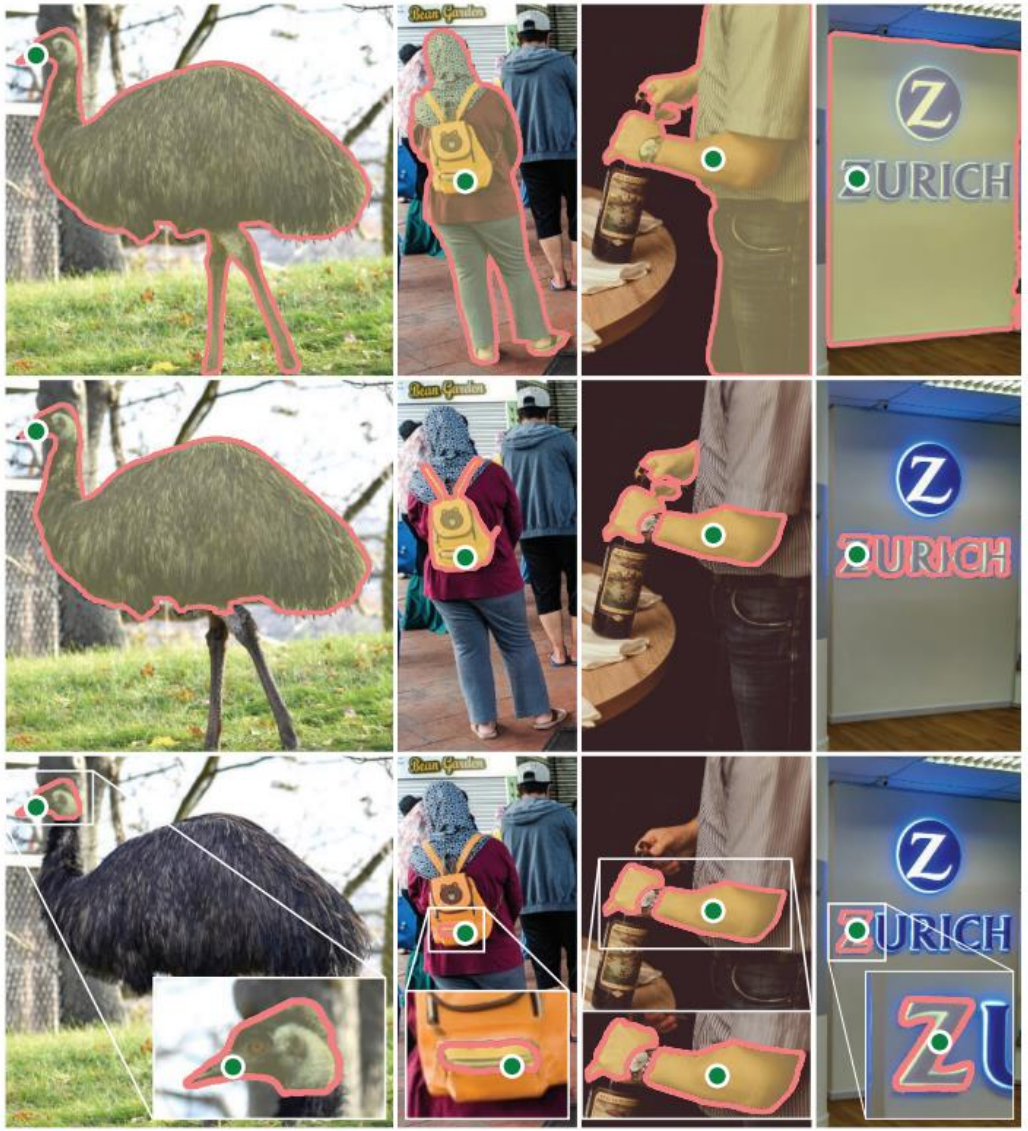
1. What **task** will enable zero-shot generalization?
2. What is the corresponding **model** architecture?
3. What **data** can power this task and model?

Segment Anything Task (1/2)

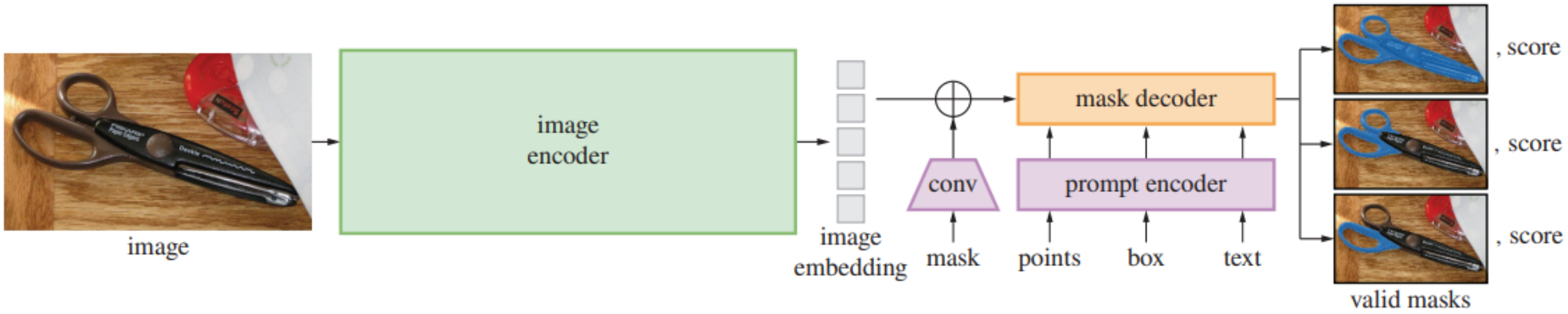
- Give a prompt
  - *point*, *box*, *mask,* text
- Return a valid segmentation mask
- Demo website: https://segment-anything.com/

Segment Anything Task (2/2)

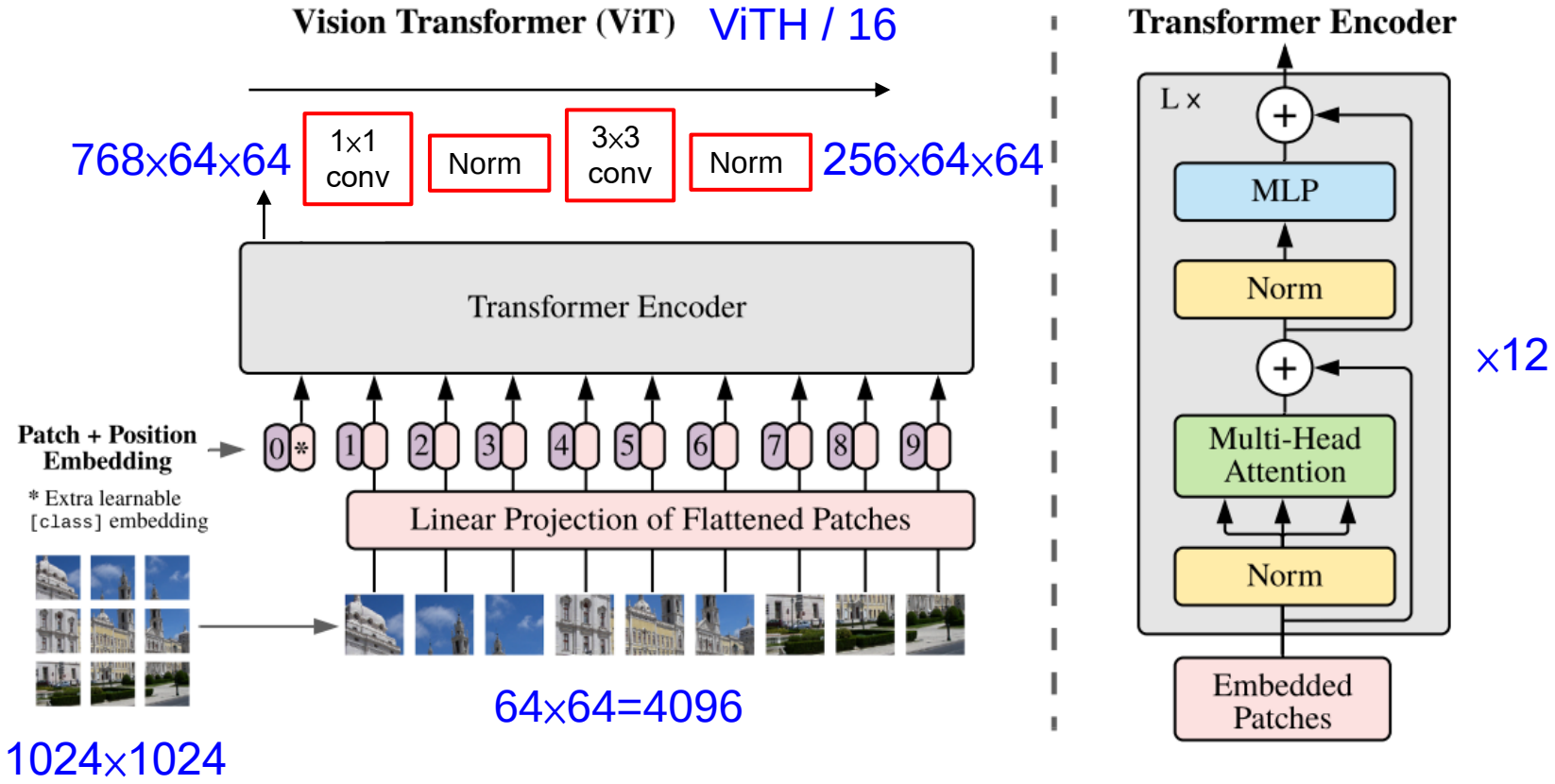




Segment Anything Model



Model - Image Encoder



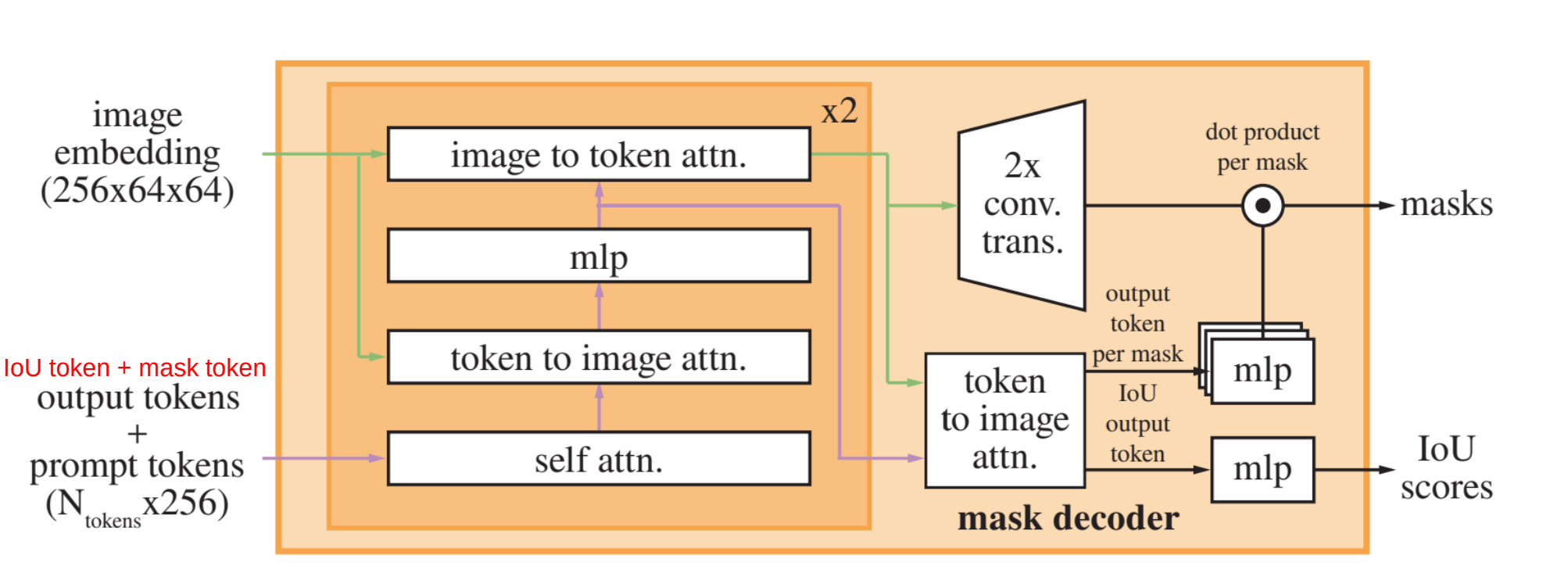
Run only once per image

Model - Prompt Encoder

- Sparse
  - points, boxes, text
  - 256-dimensional vectorial embeddings
- Dense: masks

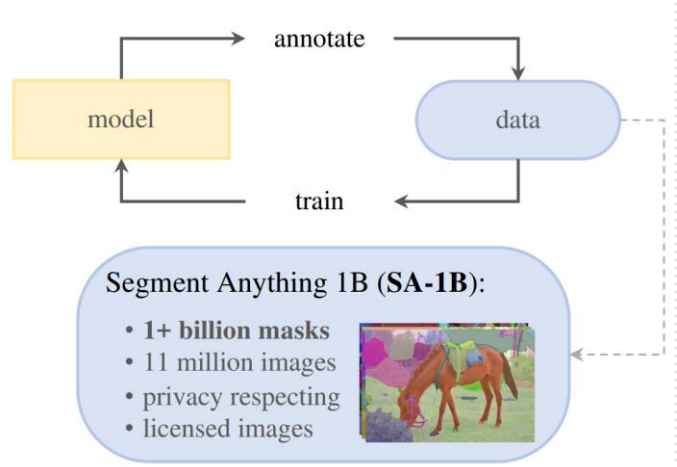
- Points -> sum of a positional encoding
- Boxes -> an embedding pair
  - (1) top-left corner (2) bottom-right corner
- Text -> off-the-shelf text encoder from **CLIP**
- Masks -> image embedding

Model - Lightweight Mask Decoder

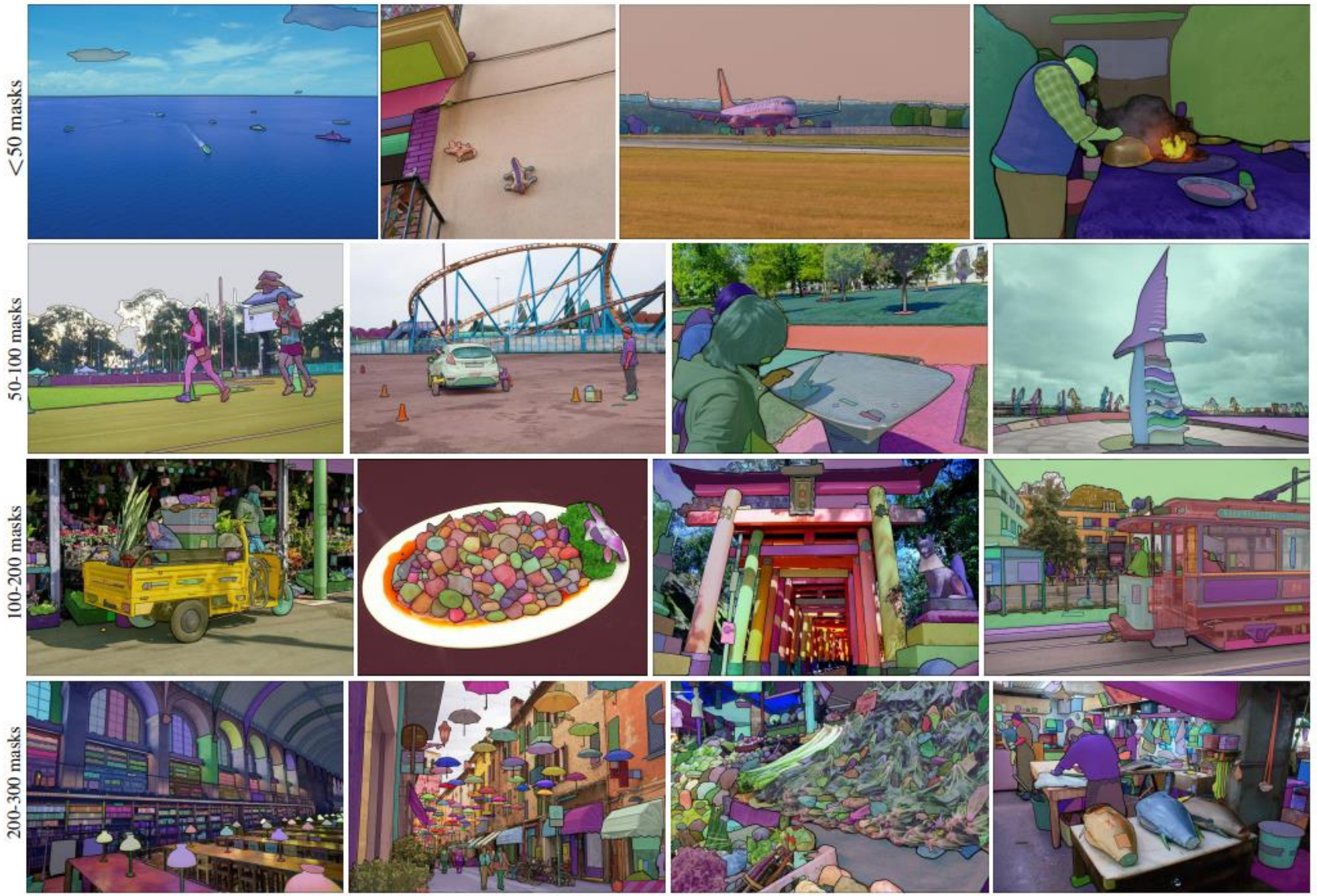


Segment Anything Data Engine

- Assisted-manual stage
  - Aimed to collect enough masks
  - 4.3M masks from 120k images
- Semi-automatic stage
  - Aimed to increase the diversity of masks
  - 5.9M masks in 180k images
- Fully automatic stage
  - 1.1B masks from 11M images
  - ~100 masks per image on average



SA-1B Dataset



Competition
Xilinx Adaptive Computing Challenge 2021

Introduction (1/2)

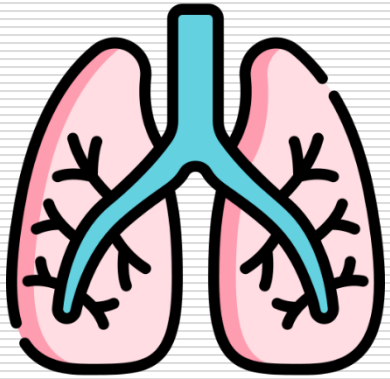
- Goal:
  - **Vitis AI development environment** to solve **real-world problems**!
- Categories:
  - Edge Computing: Xilinx Kria™ KV260 Vision AI Starter Kit
  - **Data Center AI: Xilinx VCK5000 development card**
  - Big Data Analytics: Xilinx Varium™ C1100 accelerator card

Introduction (2/2)

- FPGA-based adaptive computing is proving itself to be the most efficient and cost-effective solution for running complex AI workloads
- Xilinx VCK5000 development card is designed for designs requiring high throughput AI inference and signal processing compute performance

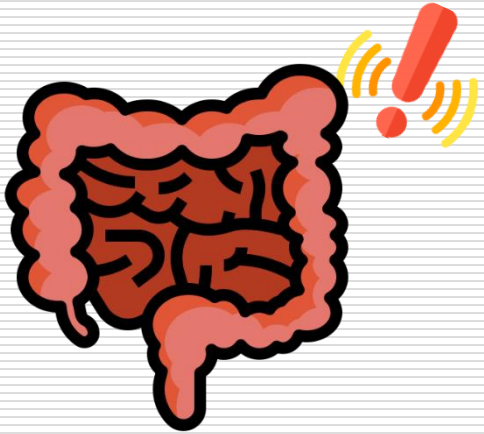
Motivation

- Top three cancers of death in 2020



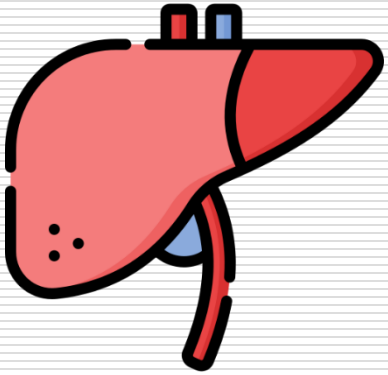
Lung

1.8 million



Colon

0.92 million

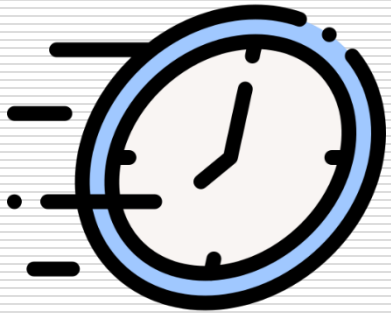


Liver

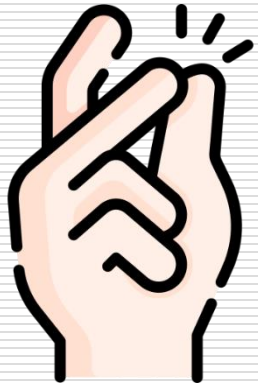
0.83 million

Goal

Accuracy



Speed

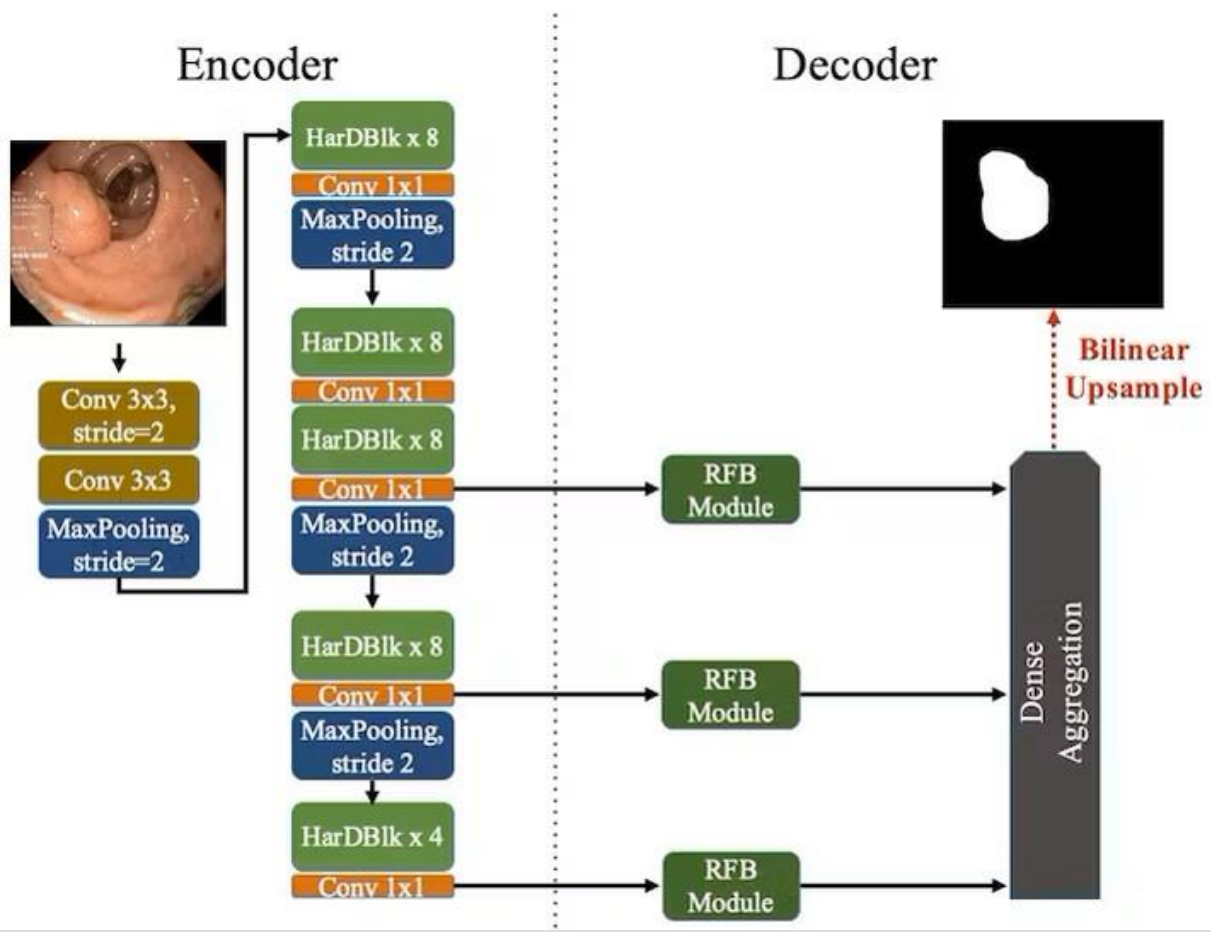


Simplicity

Flexibility

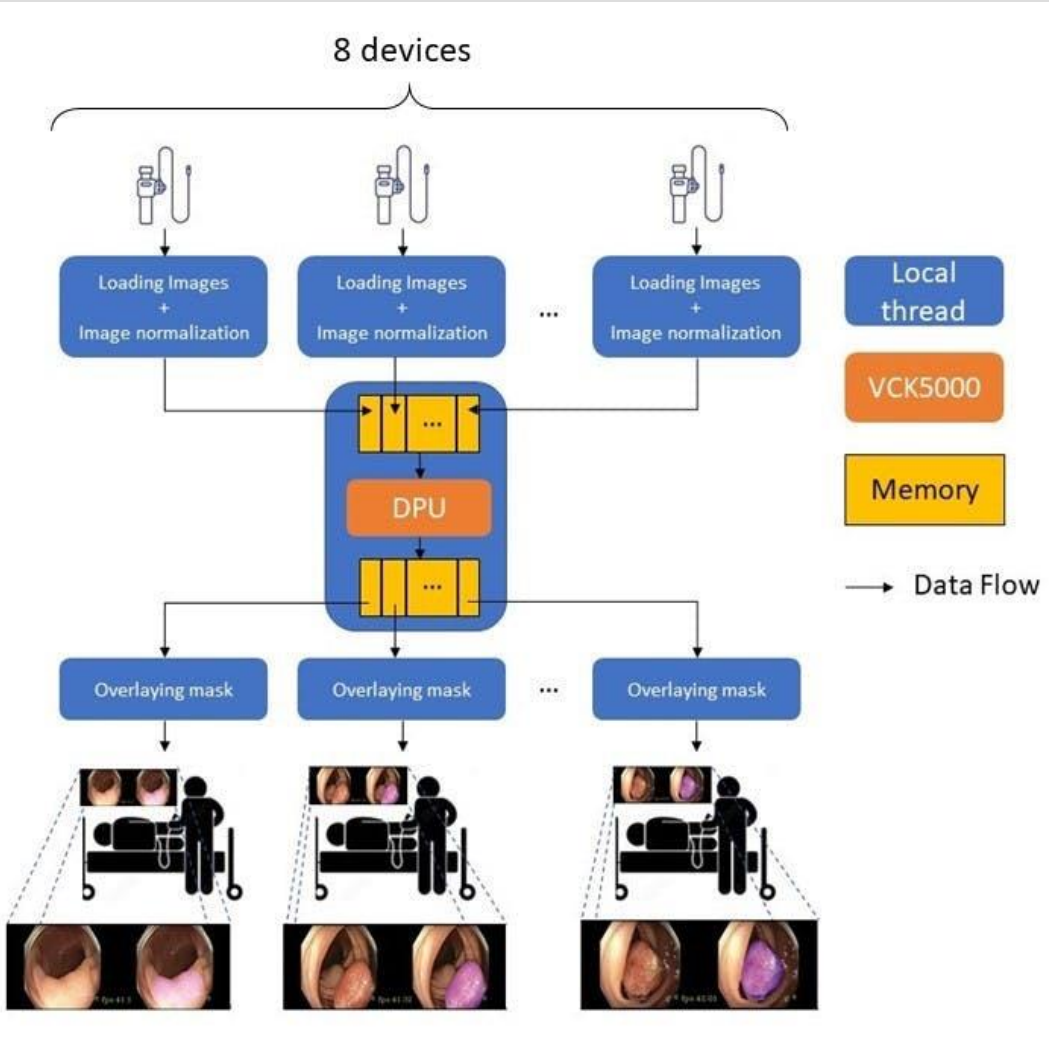
Base Model

- HarDNet-MSEG



Models	mDice	FPS
U-Net	0.597	11
ResUNet	0.69	15
FCN8	0.831	25
HRNet	0.845	12
U-Net [ResNet34]	0.876	35
HarDNet-MSEG	**0.904**	**86.7**

System Flow (Cloud Computing)

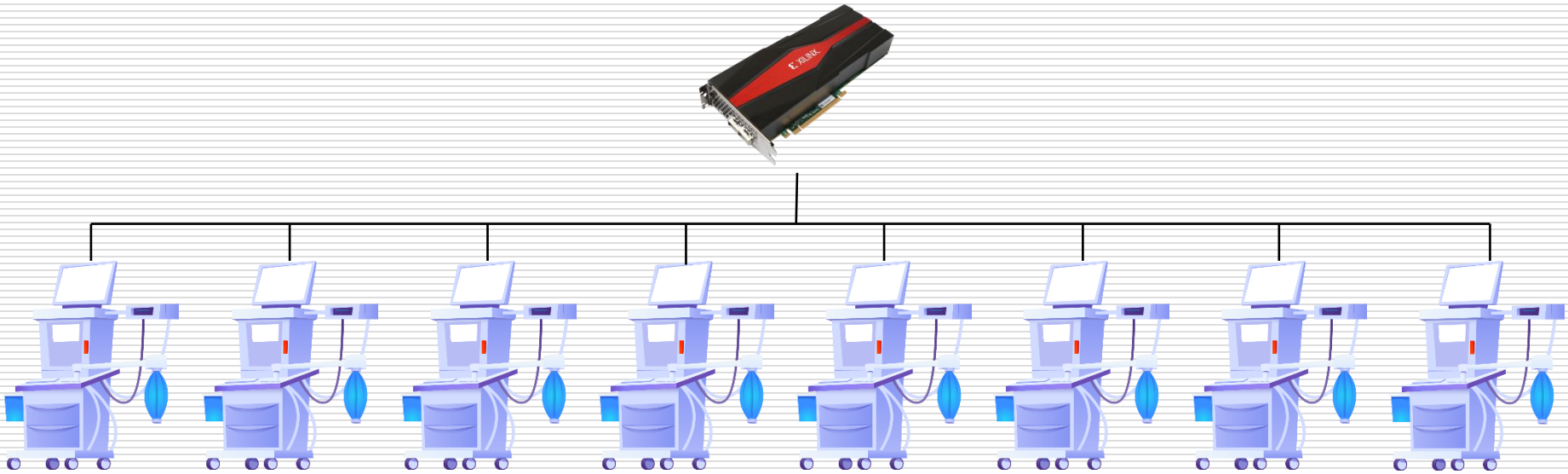


Pre-processing
video to image normalization

DPU

Post-processing
masked images to video

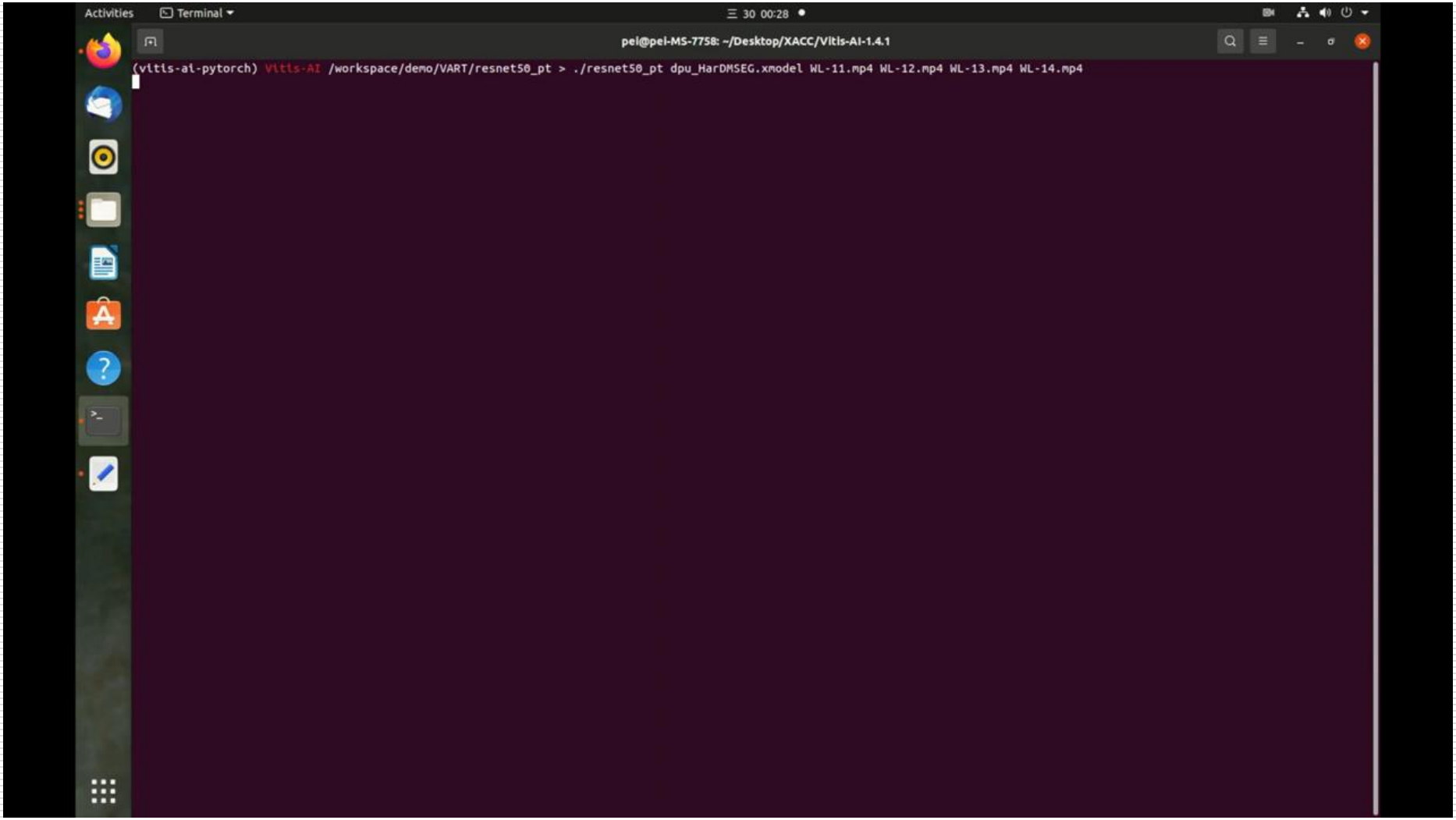
Performance Results



1 video	43.58 fps	5 videos	29.72 fps
2 videos	38.87 fps	6 videos	25.36 fps
3 videos	35.97 fps	7 videos	21.57 fps
4 videos	32.90 fps	8 videos	18.84 fps

Intel® Core™ i7-3770

Demonstration



Competition
IEEE ISBI Challenge - PAIP 2023

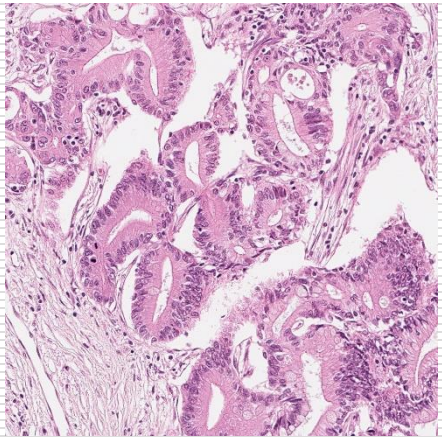
Introduction

- **Tumor cellularity prediction** in **pancreatic cancer** (supervised learning) and **colon cancer** (transfer learning)
- Promote the development of automatic and reliable **TC evaluation algorithms** that are flexibly applicable to multiple organs.

Dataset

Dataset	Pancreas	Colon	Total
Training	50	3	53
Validation	10	-	10
Testing	20	20	40
Total	80	23	103

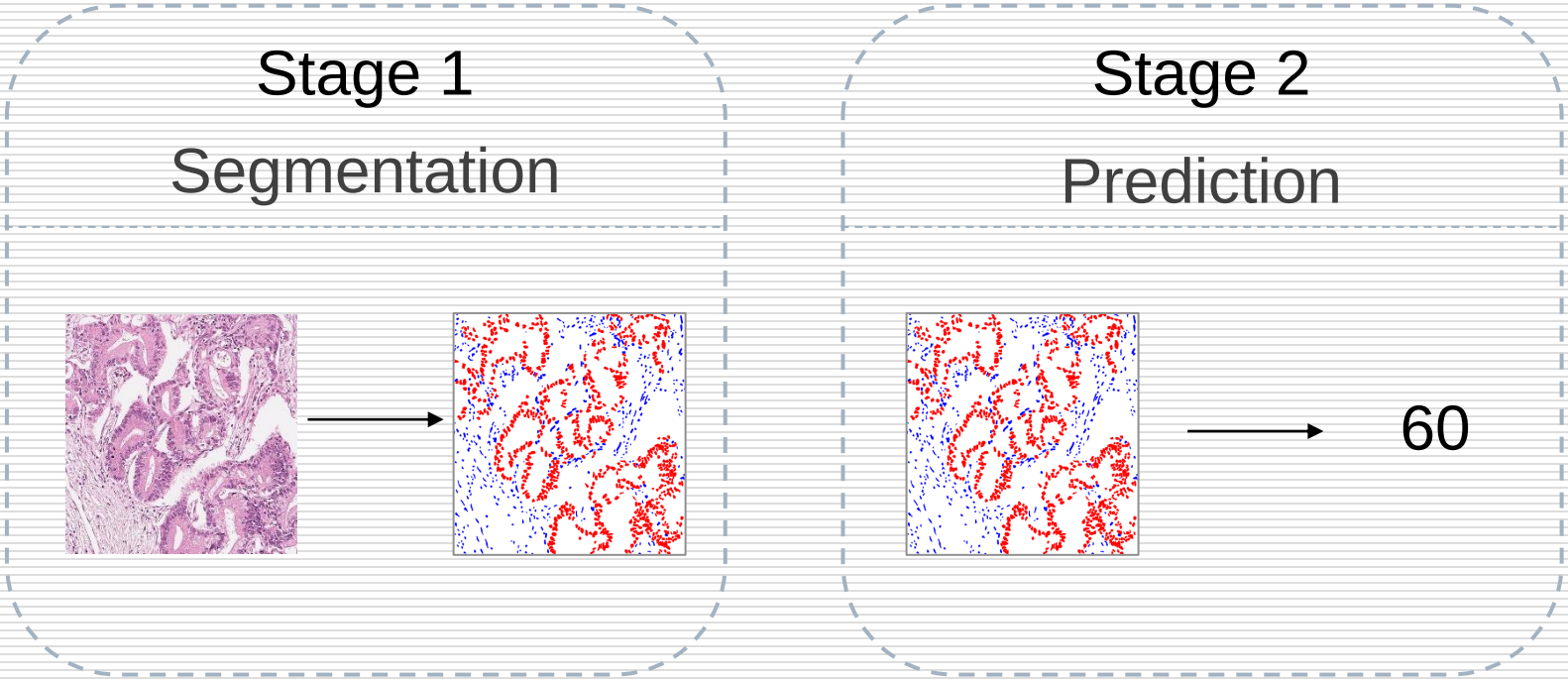
Image size: 1024 * 1024 pixels



Tumor Cellularity (TC)

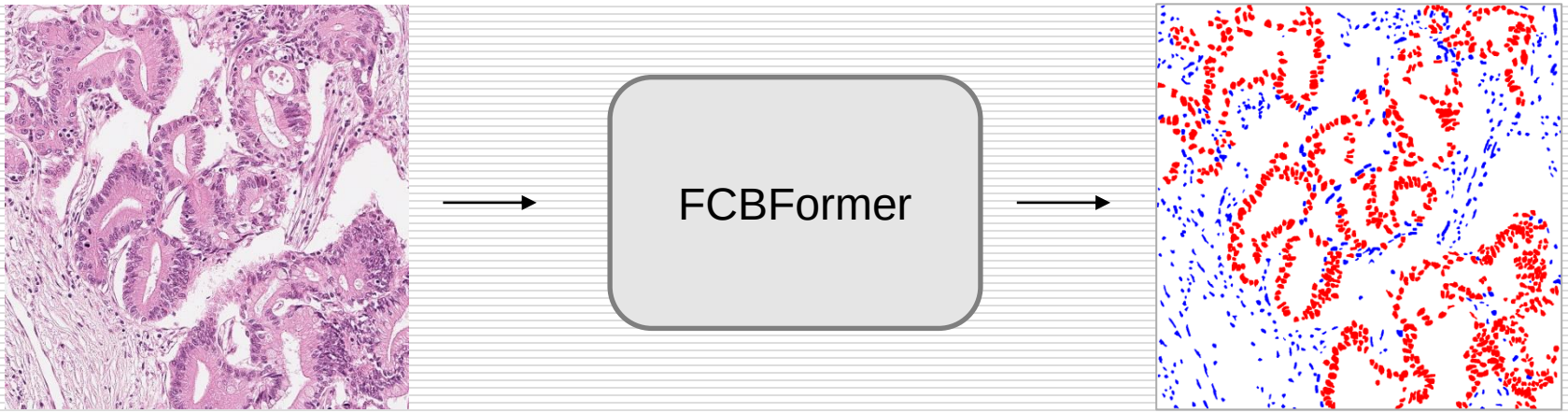
$$\left\{\begin{array}{l}\text{Tumor}\\ \\ \text{Non-Tumor}\end{array}\right. \quad \frac{|\text{Tumor}| * 100}{|\text{Tumor}|+|\text{Non}-\text{Tumor}|} \longrightarrow \quad 60$$

Method



Stage1

- **FCBFormer** is applied to label tumors and non-tumors

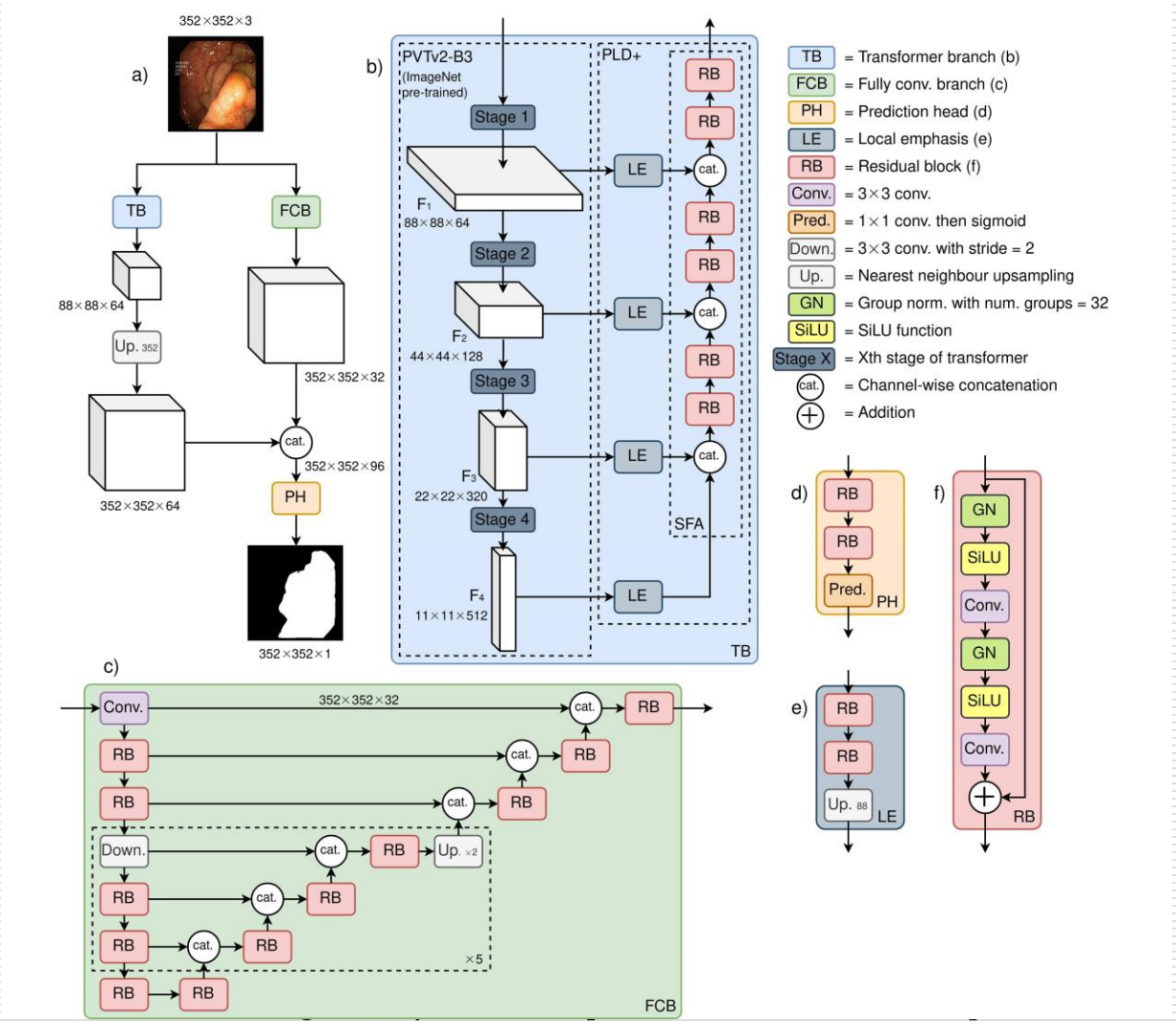


Why we choose FCBFormer?

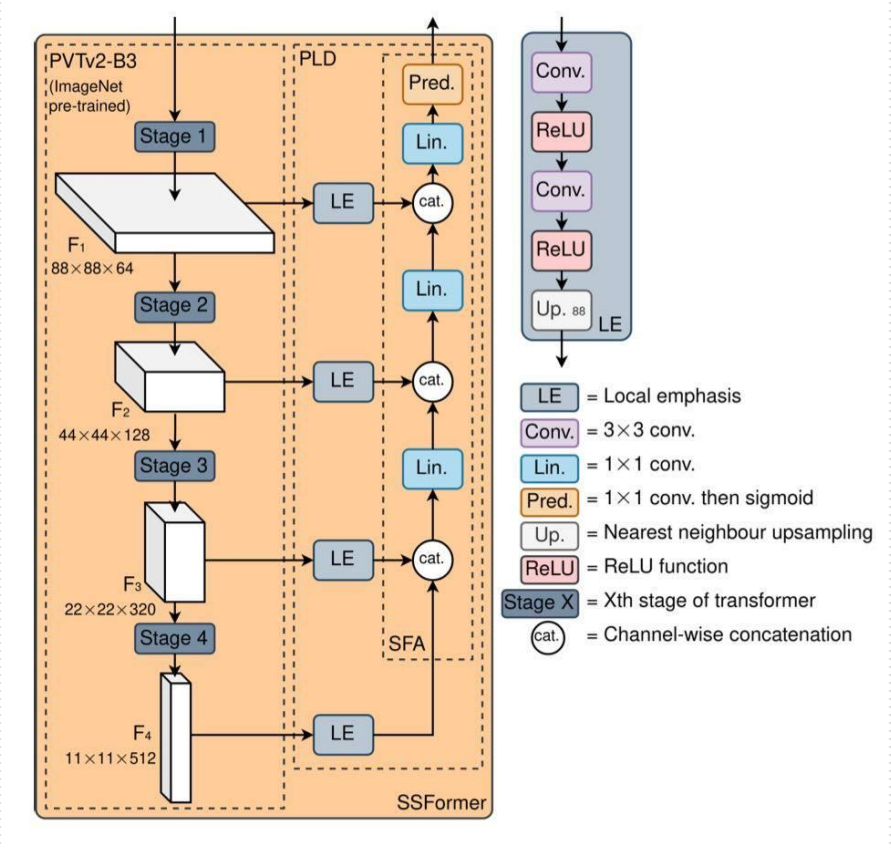
Medical Image Segmentation on Kvasir-SEG

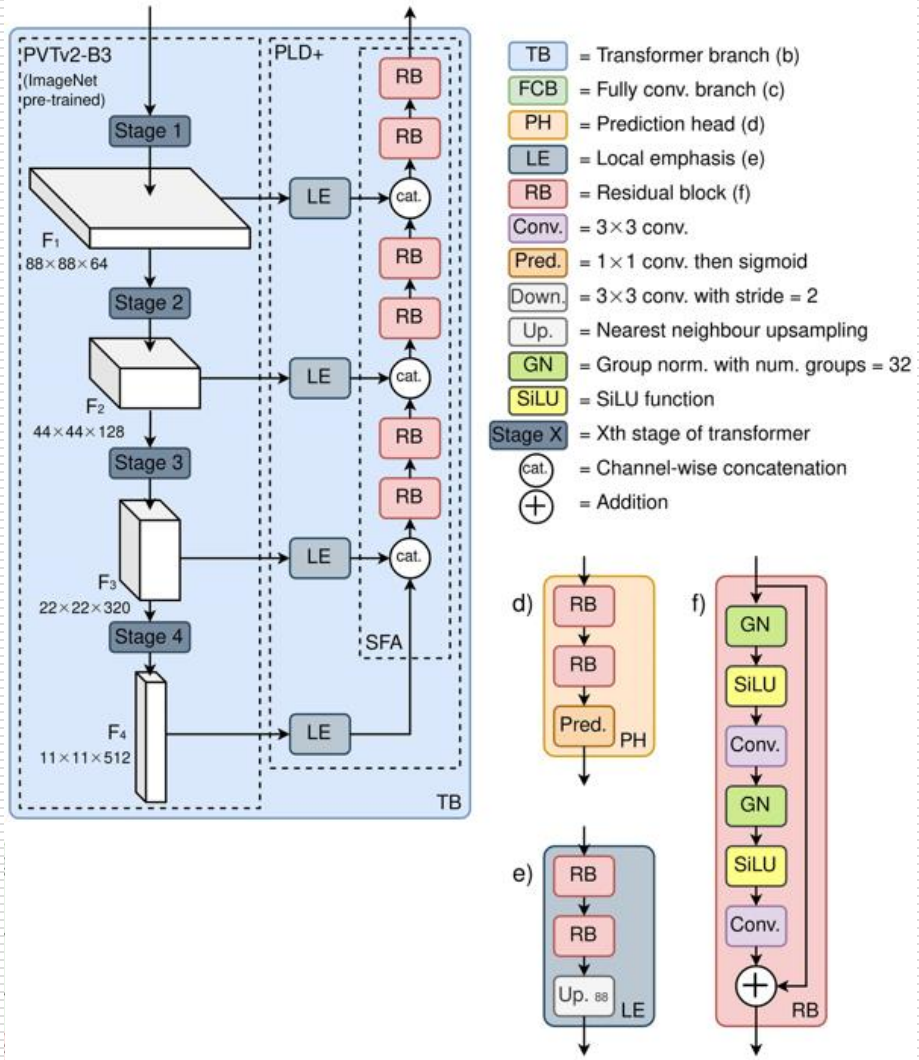
Rank	Model	mean Dice ↑	Average MAE	S-Measure	max E-Measure	mIoU	FPS	Paper	Code	Result	Year	Tags
1	**FCB-SwinV2 Transformer**	0.9420				0.8973		FCB-SwinV2 Transformer for Polyp Segmentation			2023	
2	**SEP**	0.9411				0.9002		Spatially Exclusive Pasting: A General Data Augmentation for the Polyp Segmentation			2022	
3	**FCBFormer**	0.9385				0.8903		FCN-Transformer Feature Fusion for Polyp Segmentation			2022	
4	**HarDNet-DFUS**	0.9363				0.8894		HarDNet-DFUS: An Enhanced Harmonically-Connected Network for Diabetic Foot Ulcer Image Segmentation and Colonoscopy Polyp Segmentation			2022	
5	**SSFormer-L**	0.9357				0.8905		Stepwise Feature Fusion: Local Guides Global			2022	

FCBFormer (1/2)

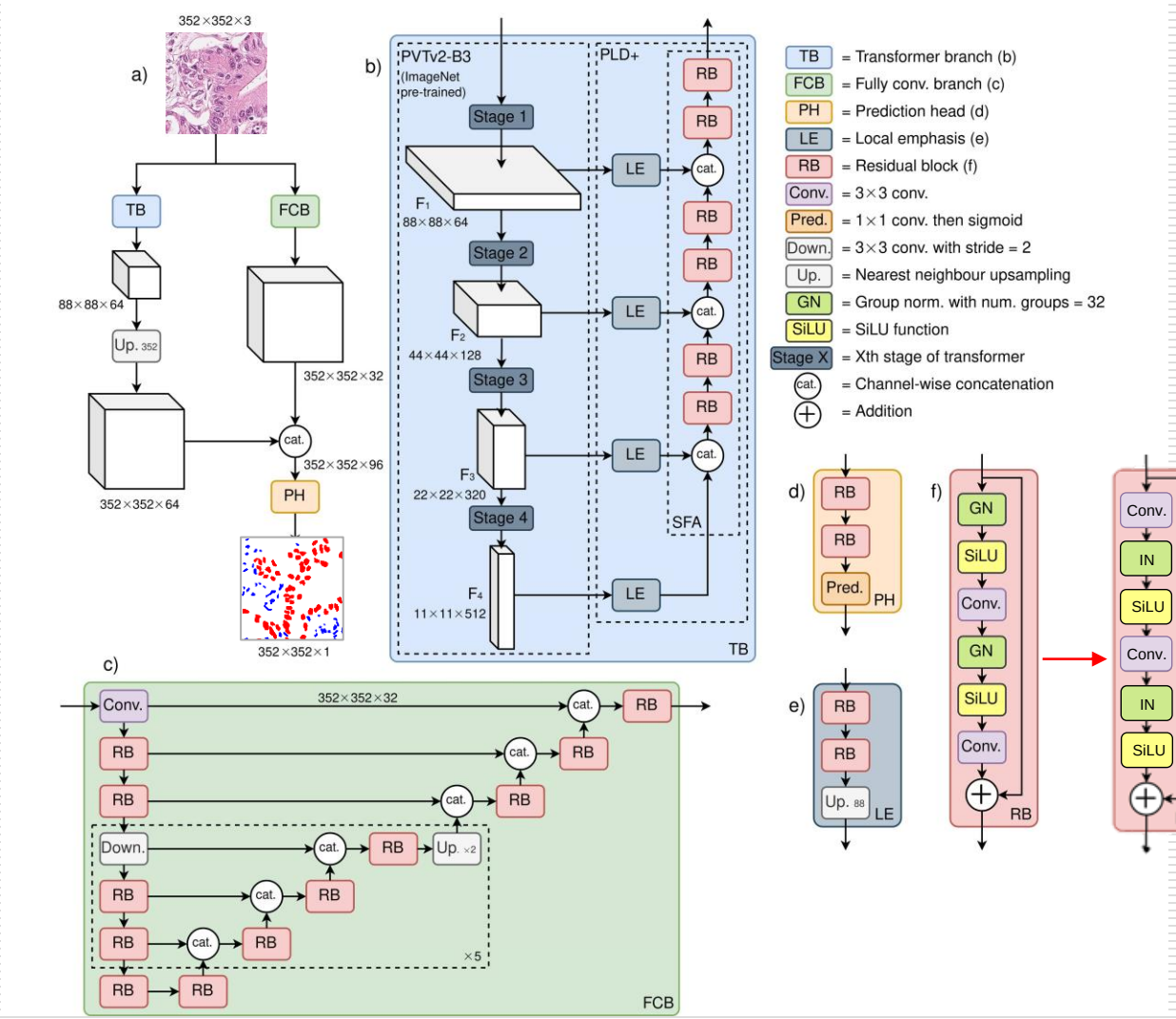


SSFormer V.S. TB





FCBFormer (2/2)



Group Norm. VS Instance Norm.

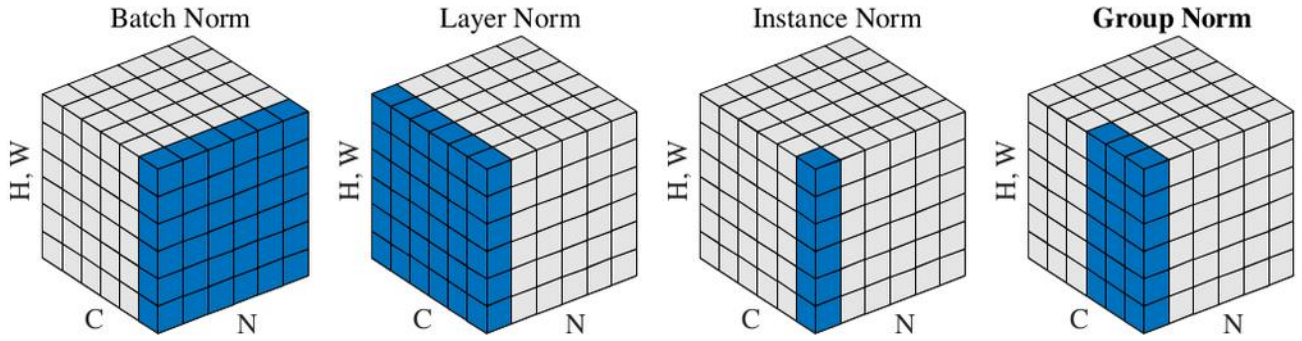
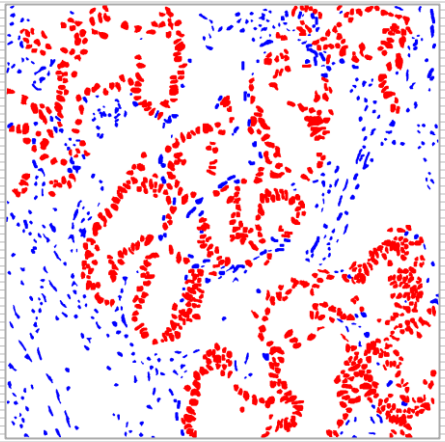


Figure 2. **Normalization methods**. Each subplot shows a feature map tensor, with N as the batch axis, C as the channel axis, and (H, W) as the spatial axes. The pixels in blue are normalized by the same mean and variance, computed by aggregating the values of these pixels.

Stage2

- The algorithm to find the connected component is applied on the mask
- TC value is computed based on the ratio of tumor cells to all cells



Connected Component

Tumor

Non-Tumor

$$\frac{|\text{Tumor}| * 100}{|\text{Tumor}| + |\text{Non-Tumor}|}$$

60

Connected Component

Algorithm 1: Prediction-by-Segmentation

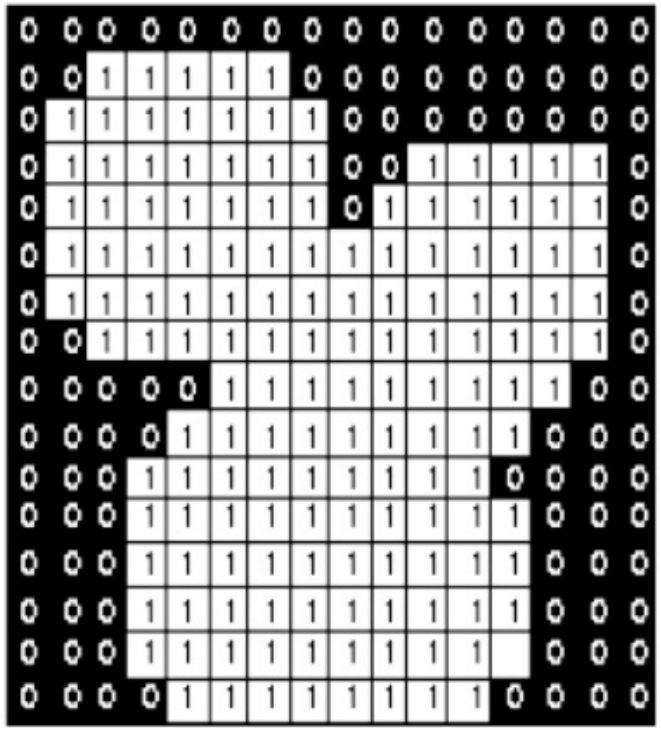
Input: Image X, FCBFormer F
Output: Tumor Cellularity T
M = F(X);
C_t, C_{non} = **FindComponent**(M);
T=ROUND($C_t \times 100/(C_t + C_{non})$);
Return T;

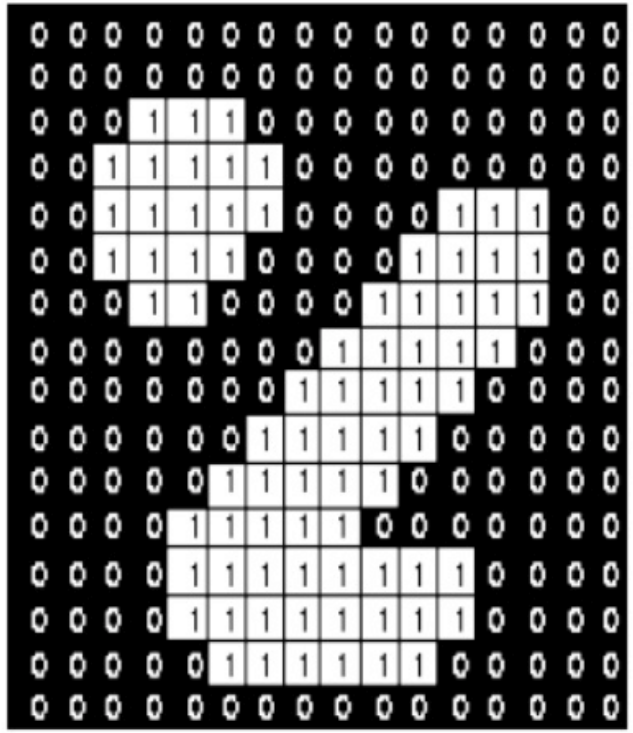
Algorithm 2: FindComponent

Input: Mask M
Output: C_t, C_{non}
Intialize: $C_t = 0$, $C_{non} = 0$
$E_M = M$;
for $j = 1;\ \ j \leq 2;$ **do**
    E_M = ERODE(E_M);
C_{cells}, A_{tumor}, A_{non} = ConnectedComponent(E_M);
for $j = 1;\ \ j \leq C_{cells};$ **do**
    if $A^i_{tumor} \geq A^i_{non}$ **then**
        $C_t = C_t + 1$;
    else
        $C_{non} = C_{non} + 1$;
Return C_t, C_{non};

C_t: Number of tumor cells
C_{non}: Number of non−tumor cells
$C_{component}$: Number of extracted components
A^i_{tumor}: Area of tumor cells in i^{th} component
$A^i_{non-tumor}$: Area of non−tumor cells in i^{th} component

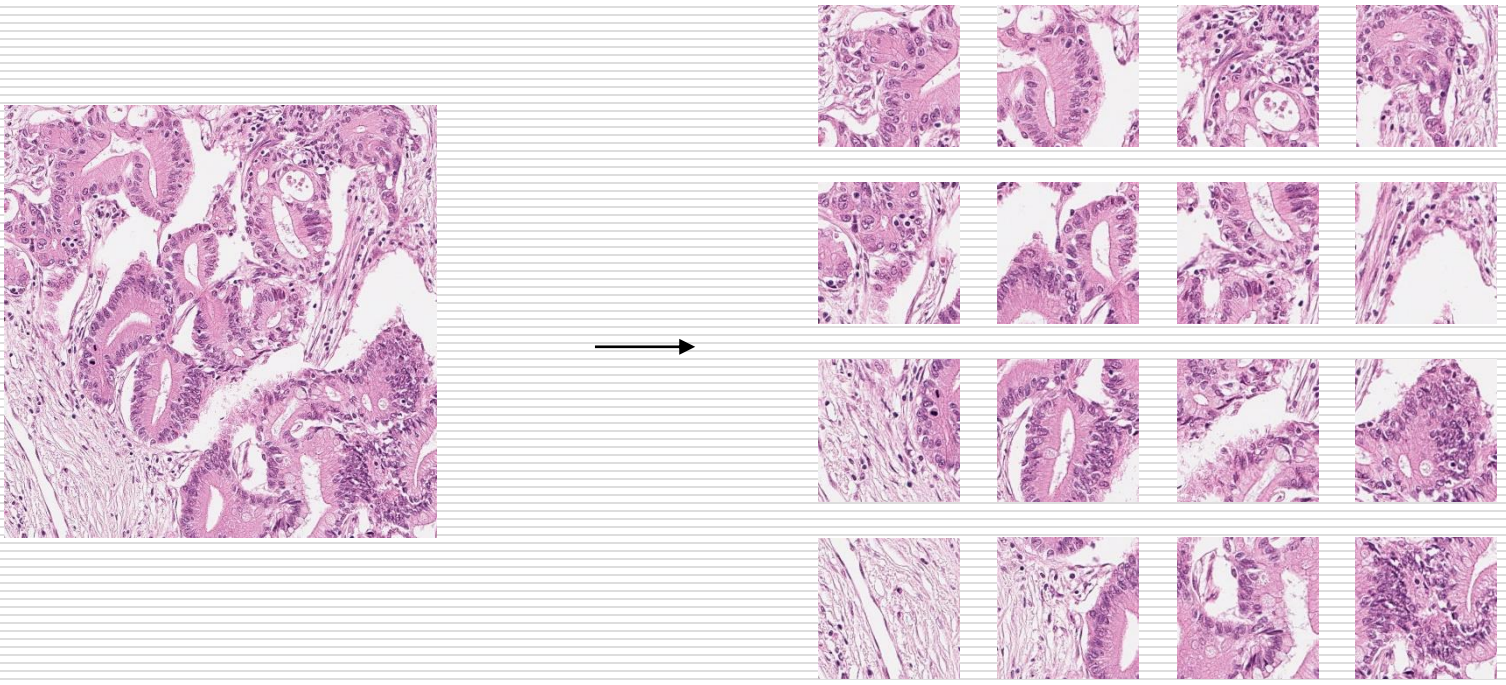
slightly modified





Data Scaling

- Due to the small size of the dataset, we divide each image into 64 small patches
- Each image patch is equally sized with 256X256



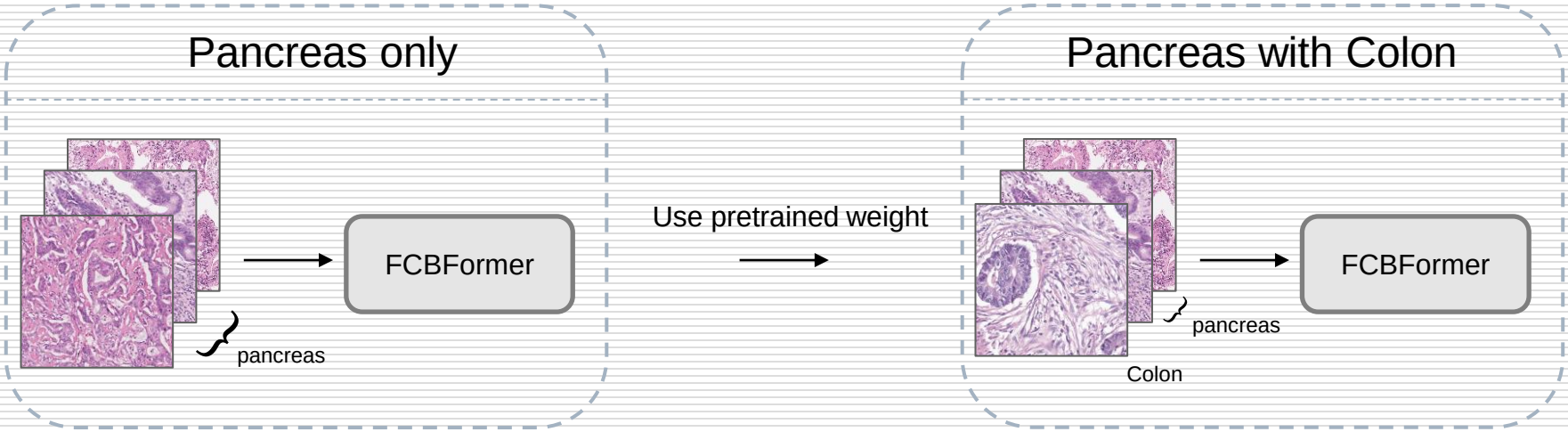
Data Augmentation

- To ensure the robustness of our model, different data augmentations are applied

Type	Augmentation
Combination	Mosaic
Spatial Transformation	Horizontal/Vertical Flip, Random Resized Crop
Distortion	Elastic Distortion, Grid Distortion, Optical Distortion
Color Transformation	Image Sharpen, Random Brightness Contrast, Random HSV, CLAHE

Transfer Learning

- We **transfer the knowledge** from pancreas to colon



References (1/2)

- 影像切割任務常用的指標-IoU和Dice coefficient
  - https://chih-sheng-huang821.medium.com/%E5%BD%B1%E5%83%8F%E5%88%87%E5%89%B2%E4%BB%BB%E5%8B%99%E5%B8%B8%E7%94%A8%E7%9A%84%E6%8C%87%E6%A8%99-iou%E5%92%8Cdice-coefficient-3fcc1a89cd1c
- Stanford CS231n Object Detection and Image Segmentation
  - http://cs231n.stanford.edu/slides/2023/lecture_11.pdf
- DeconvNet
  - https://medium.com/image-processing-and-ml-note/deconvnet-unpooling-layer-semantic-segmentation-840e16835e1e
- U-Net
  - https://www.slideshare.net/ZGoo/unet-1pptx

References (2/2)

- TransFuse: Fusing Transformers and CNNs for Medical Image Segmentation
  - https://arxiv.org/abs/2102.08005
- Segment Anything
  - https://arxiv.org/abs/2304.02643
- HarDNet-MSEG
  - https://arxiv.org/abs/2101.07172
- FCN-Transformer Feature Fusion for Polyp Segmentation
  - https://arxiv.org/abs/2208.08352

Thank you