Classification

Semantic Segmentation

Object Detection

Instance Segmentation

No spatial extent

No objects, just pixels

Multiple Objects
Recap

“Slow” R-CNN: Run CNN independently for each region

Fast R-CNN: Apply differentiable cropping to shared image features

Faster R-CNN: Compute proposals with CNN

Single-Stage: Fully convolutional detector

With anchors: RetinaNet
Anchor-Free: FCOS
R-CNN: Region-Based CNN

Classify each region

Bounding box regression:
Predict “transform” to correct the RoI: 4 numbers \((t_x, t_y, t_h, t_w)\)

Forward each region through ConvNet

Warped image regions (224x224)

Input image

Fast R-CNN

Category and box transform per region

Per-Region Network

Crop + Resize features

Image features

Run whole image through ConvNet

Input image
Faster R-CNN: Learnable Region Proposals

Faster R-CNN is a **Two-stage object detector**

**First stage:** Run once per image
- Backbone network
- Region proposal network

**Second stage:** Run once per region
- Crop features: RoI pool / align
- Predict object class
- Prediction bbox offset
Faster R-CNN: Learnable Region Proposals
Single-Stage Detectors: RetinaNet

Run backbone CNN to get features aligned to input image

Input Image
(e.g. 3 x 640 x 480)

CNN

Each feature corresponds to a point in the input

Image features
(e.g. 512 x 5 x 6)

Conv

Similar to RPN – but rather than classify anchors as object/no object, directly predict object category (among C categories) or background

Anchor classification
2K*(C+1) x 5 x 6

Anchor transforms
4K x 5 x 6

Lin et al, "Focal Loss for Dense Object Detection", ICCV 2017
Single-Stage Detectors: RetinaNet

In practice, RetinaNet also uses Feature Pyramid Network to handle multiscale

Lin et al, “Focal Loss for Dense Object Detection”, ICCV 2017
Figure credit: Lin et al, ICCV 2017
Run backbone CNN to get features aligned to input image

Each feature corresponds to a point in the input

Finally, predict “centerness” for all positive points (using logistic regression loss)

Class scores
→ C x 5 x 6

Box edges
→ 4 x 5 x 6

Centerness
→ 1 x 5 x 6

Test-time: predicted “confidence” for the box from each point is product of its class score and centerness.

centerness = \frac{\min(L, R)}{\max(L, R)} \cdot \frac{\min(T, B)}{\max(T, B)}
How can we compare our prediction to the ground-truth box?

**Intersection over Union (IoU)**
(Also called “Jaccard similarity” or “Jaccard index”):

\[
\frac{\text{Area of Intersection}}{\text{Area of Union}}
\]
Precision & Recall

- True detection: high intersection over union
  - Choose IoU threshold
- Precision: #true detections / #detections
- Recall: #true detections / #true positives

Reject everything: no mistakes

Ideal!

Summarize by area under curve (avg. precision)

Accept everything: Miss nothing

Car AP = 0.65
Cat AP = 0.80
Dog AP = 0.86

mAP@0.5 = 0.77

mAP@0.5 = 0.77
mAP@0.55 = 0.71
mAP@0.60 = 0.65
...

mAP@0.95 = 0.2

COCO mAP = 0.4
Today’s class

**Classification**
- CAT
- No spatial extent

**Semantic Segmentation**
- GRASS, CAT, TREE, SKY
- No objects, just pixels

**Object Detection**
- DOG, DOG, CAT
- Multiple Objects

**Instance Segmentation**
- DOG, DOG, CAT
How to measure segmentation accuracy?

**IoU (Intersection over Union)**

**DICE Score (F score)**

Report mean IoU or DICE score over the test dataset

https://towardsdatascience.com/metrics-to-evaluate-your-semantic-segmentation-model-6bcb99639aa2
Semantic Segmentation vs. Instance Segmentation vs. Panoptic Segmentation

(a) Image

(b) Semantic Segmentation

(c) Instance Segmentation

(d) Panoptic Segmentation
Most popular Semantic Segmentation dataset: ADE20k

- 20K scene-centric images exhaustively annotated with pixel-level objects and object parts labels.
- 150 semantic categories, which include stuffs like sky, road, grass, and discrete objects like person, car, bed.
Most popular Instance & Panoptic Segmentation dataset: MS COCO

Instance Segmentation

Panoptic Segmentation
Semantic Segmentation

Label each pixel in the image with a category label.

Don’t differentiate instances, only care about pixels.
Problem: Very inefficient! Not reusing shared features between overlapping patches
Problem #1: Effective receptive field size is linear in number of conv layers: With $L$ 3x3 conv layers, receptive field is $1+2L$

Problem #2: Convolution on high res images is expensive! Recall ResNet stem aggressively downsamples
Semantic Segmentation: Fully Convolutional Network

Downsampling:
Pooling, strided convolution

Upsampling:

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

Input: 3 x H x W

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

Low-res: D_3 x H/4 x W/4

Med-res: D_2 x H/4 x W/4

Med-res: D_2 x H/4 x W/4

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

Predictions: H x W

---


In-Network Upsampling: “Unpooling”

**Bed of Nails**

```
1 2
3 4
```

```
1 0 2 0
0 0 0 0
3 0 4 0
0 0 0 0
```

**Nearest Neighbor**

```
1 2
3 4
```

```
1 1 2 2
1 1 2 2
3 3 4 4
3 3 4 4
```
In-Network Upsampling: Bilinear Interpolation

Input: $C \times 2 \times 2$

Output: $C \times 4 \times 4$

$$f_{x,y} = \sum_{i,j} f_{i,j} \max(0, 1 - |x - i|) \max(0, 1 - |y - j|) \quad i \in \{\lfloor x \rfloor - 1, \ldots, \lfloor x \rfloor + 1\}$$

$$\quad j \in \{\lfloor y \rfloor - 1, \ldots, \lfloor y \rfloor + 1\}$$

Use two closest neighbors in $x$ and $y$ to construct linear approximations
In-Network Upsampling: Bicubic Interpolation

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>0.68</th>
<th>1.02</th>
<th>1.56</th>
<th>1.89</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.35</td>
<td>1.68</td>
<td>2.23</td>
<td>2.56</td>
<td></td>
</tr>
<tr>
<td>2.44</td>
<td>2.77</td>
<td>3.32</td>
<td>3.65</td>
<td></td>
</tr>
<tr>
<td>3.11</td>
<td>3.44</td>
<td>3.98</td>
<td>4.32</td>
<td></td>
</tr>
</tbody>
</table>

Input: C x 2 x 2

Output: C x 4 x 4

Use **three** closest neighbors in x and y to construct cubic approximations
(This is how we normally resize images!)
In-Network Upsampling: “Max Unpooling”

**Max Pooling:** Remember which position had the max

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>6</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>5</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>3</td>
<td>4</td>
<td>8</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>8</td>
</tr>
</tbody>
</table>

**Max Unpooling:** Place into remembered positions

<table>
<thead>
<tr>
<th>0</th>
<th>0</th>
<th>2</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>4</td>
</tr>
</tbody>
</table>

Pair each downsampling layer with an upsampling layer

Regular vs Transposed Convolution

Filter size is 3x3

Regular Convolution reduces feature size

Transposed convolution increases feature size

Strided transpose convolution
Semantic Segmentation: Fully Convolutional Network

Downsampling: Pooling, strided convolution

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

**Input:** $3 \times H \times W$

**Med-res:** $D_2 \times H/4 \times W/4$

**Low-res:** $D_1 \times H/2 \times W/2$

**High-res:** $D_3 \times H/4 \times W/4$

Upsampling: ???

Predictions: $H \times W$

**Upsampling:**
- Bilinear Upsampling (non learnable)
- Strided Transpose Convolution (learnable)

U-Net for segmentation
SOTA Detection + Segmentation

Swin Transformer

- Ranked #13 Object Detection on COCO test-dev
- Ranked #6 Instance Segmentation on COCO test-dev (using additional training data)
- Ranked #10 Semantic Segmentation on ADE20K (using additional training data)
- Ranked #24 Action Classification on Kinetics-400 (using additional training data)

This repo is the official implementation of "Swin Transformer: Hierarchical Vision Transformer using Shifted Windows" as well as the follow-ups. It currently includes code and models for the following tasks:

- **Image Classification:** Included in this repo. See `get_started.md` for a quick start.
- **Object Detection and Instance Segmentation:** See Swin Transformer for Object Detection.
- **Semantic Segmentation:** See Swin Transformer for Semantic Segmentation.
- **Video Action Recognition:** See Video Swin Transformer.
- **Semi-Supervised Object Detection:** See Soft Teacher.
- **SSL:** Contrastive Learning: See Transformer-SSL.
- **SSL:** Masked Image Modeling: See `get_started.md#simmim-support`.
- **Mixture-of-Experts:** See `get_started` for more instructions.
- **Feature-Distillation:** See Feature-Distillation.
**Computer Vision Tasks: Instance Segmentation**

**Instance Segmentation:** Detect all objects in the image, and identify the pixels that belong to each object (Only things!)

**Approach:** Perform object detection, then predict a segmentation mask for each object!
Instance segmentation with Mask R-CNN
Mask R-CNN

CNN + RPN

RoI Align

256 x 14 x 14

Conv

256 x 14 x 14

Conv

Classification Scores: C
Box coordinates (per class):
4 * C

Predict a mask for each of C classes:
C x 28 x 28

He et al., “Mask R-CNN”, ICCV 2017
Mask R-CNN: Example Training Targets
Mask R-CNN: Very Good Results!
Can we generalize Faster R-CNN architecture to other Vision tasks?

Beyond Instance Segmentation: Human Keypoints

Represent the pose of a human by locating a set of keypoints

e.g. 17 keypoints:
- Nose
- Left / Right eye
- Left / Right ear
- Left / Right shoulder
- Left / Right elbow
- Left / Right wrist
- Left / Right hip
- Left / Right knee
- Left / Right ankle

Person image is CCO public domain
Instance segmentation with Mask R-CNN

Keypoint prediction with Mask R-CNN
Mask R-CNN: Keypoints

- Classification Scores: C
- Box coordinates (per class): 4 * C
- Segmentation mask: C x 28 x 28

One mask for each of the K different keypoints

Keypoint masks: K x 56 x 56

Ground-truth has one “pixel” turned on per keypoint. Train with softmax loss

He et al., “Mask R-CNN”, ICCV 2017
Interactive Segmentation with Scribbles
(Red = foreground, blue= background)

Earlier works of segmentation used Graph Cut techniques to solve this problem.
Deep Extreme Cut (DEXTR): From Extreme Points to Object Segmentation, CVPR 2018

Interactive Segmentation with few points

Efficient Annotation of Segmentation Datasets with Polygon-RNN++, CVPR 2018
Interactive Segmentation with 2 points

(Green = foreground, red = background)

(a) Input  (b) Ground truth  (c) Graph cuts  (d) DOS  (e) Our approach

Interactive Image Segmentation with Latent Diversity, CVPR 2018
Interactive Segmentation with points

SimpleClick: Interactive Image Segmentation with Simple Vision Transformers, Liu et al. 2022.
Foundation Models and Promptable Segmentation
What is a foundation model?

- A foundation model is a large-scale pretrained model (e.g., BERT, DALL-E, GPT-3) that can be adapted to a wide range of downstream applications.
- This term was first popularized by researchers in Stanford University in this review: [On the Opportunities and Risks of Foundation Models](https://paperswithcode.com/paper/on-the-opportunities-and-risks-of-foundation-models).
- Checkout more papers on foundation models: [Awesome-Foundation-Models](https://github.com/awslabs/foundation-models).

![Diagram](slide_image.png)
A foundation model can centralize the information from various modalities.
Segment Anything Model (SAM): the first foundation model for promptable segmentation.

Prompt it with interactive points and boxes

Automatically segment everything in an image

Generate multiple valid masks for ambiguous prompts

Try the demo: [https://segment-anything.com/demo](https://segment-anything.com/demo)

Slide Credits: Qin Liu
SAM is built with three interconnected components: A task, an model, and a data engine.
Task: Promptable Segmentation

- SAM considers two sets of prompts: **sparse** (clicks, boxes, text) and **dense** (masks).
- SAM’s promptable design enables flexible integration with other systems (i.e., used as **component** in larger systems).
Model: Segment Anything Model (SAM)

- A heavyweight **image encoder** outputs an image embedding.
- A lightweight **prompt encoder** efficiently queries the image embedding.
- A lightweight **mask decoder** produces object masks and confidence scores.
Ambiguity-Aware Segmentation

- SAM is designed to predict multiple masks (i.e., 3 masks: whole, part, subpart) for a single prompt.
- During training, the model only backprops the minimum loss over masks.
- To rank masks, the model predicts a confidence score (i.e., estimated IoU) for each mask.
Dataset: SA-1B

- Built with a SAM model in the loop
- 11M images with 1.1B segmentation masks
- 400x more masks than any prior segmentation dataset

Slide Credits: Qin Liu
Task 1: Zero-Shot Single Point Valid Mask Evaluation

- Training dataset: the whole SA-1B dataset
- Test datasets: 23 diverse segmentation datasets (only validation/test sets??)
Task 1: Zero-Shot Single Point Valid Mask Evaluation

SAM significantly outperforms baselines with 1 point and is on par with more points.
Limitations

- SAM may miss fine structures, hallucinate small disconnected components at times, and produce wrong boundaries.
- SAM is expected to be outperformed by dedicated interactive segmentation methods (e.g., SimpleClick) when many points are provided.
- SAM is expected to be outperformed by domain-specific tools (e.g., ilastik).
- SAM’s performance on the text-to-mask task is not entirely robust.
- While SAM is initialized with a self-supervised technique (i.e., MAE), the vast majority of its capabilities come from large-scale supervised training.
Grounded-Segment-Anything

Text Prompt: Bench

Grounded-SAM Output

Stable-Diffusion Inpainting
A Sofa, high quality, detailed
BLIP + Grounded-SAM: Automatic Label System!
Using BLIP to generate caption, extract tags and using Grounded-SAM for box and mask generating. Here's the demo output:

there are two dogs playing with a stick on the beach
Unsupervised: the user does not interact with the algorithm to obtain the segmentation masks. Methods should provide a set of object candidates.

Semi-supervised: user inputs full mask of the object of interest in the first frame only.

Interactive: user gives iterative refinement inputs to the algorithm, in the form of a scribble, to segment the objects of interest.

DAVIS: Densely Annotated Video Segmentation
Beyond Segmentation: Alpha Matting

Image → Alpha matte → Composed Image

Under-constrained!

\[ I = \alpha \cdot F + (1 - \alpha) \cdot B \]
Solving Matting with user annotation!

User Annotation during inference
Image Matting with Trimap

<table>
<thead>
<tr>
<th>Image</th>
<th>Trimap</th>
<th>Alpha Matte</th>
<th>Composed Image</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Image" /></td>
<td><img src="trimap1.png" alt="Trimap" /></td>
<td><img src="alpha_matt1.png" alt="Alpha Matte" /></td>
<td><img src="composed1.png" alt="Composed Image" /></td>
</tr>
<tr>
<td><img src="image2.png" alt="Image" /></td>
<td><img src="trimap2.png" alt="Trimap" /></td>
<td><img src="alpha_matt2.png" alt="Alpha Matte" /></td>
<td><img src="composed2.png" alt="Composed Image" /></td>
</tr>
<tr>
<td><img src="image3.png" alt="Image" /></td>
<td><img src="trimap3.png" alt="Trimap" /></td>
<td><img src="alpha_matt3.png" alt="Alpha Matte" /></td>
<td><img src="composed3.png" alt="Composed Image" /></td>
</tr>
<tr>
<td><img src="image4.png" alt="Image" /></td>
<td><img src="trimap4.png" alt="Trimap" /></td>
<td><img src="alpha_matt4.png" alt="Alpha Matte" /></td>
<td><img src="composed4.png" alt="Composed Image" /></td>
</tr>
<tr>
<td><img src="image5.png" alt="Image" /></td>
<td><img src="trimap5.png" alt="Trimap" /></td>
<td><img src="alpha_matt5.png" alt="Alpha Matte" /></td>
<td><img src="composed5.png" alt="Composed Image" /></td>
</tr>
</tbody>
</table>

Information Flow Matting
Background Subtraction                Segmentation                Vs                Matting

• Binary Mask
• No color separation
Solving Matting with user annotation!

Solution: Simply capture an additional background image before/after the shoot.

User Annotation during inference

Expensive – especially for a video
Background Matting: The World is Your Green Screen

Soumyadip Sengupta
Vivek Jayaram
Brian Curless
Steve Seitz
Ira Kemelmacher-Shlizerman

University of Washington

Original Video

Replaced Background
Speed and Resolution

- **60fps**
  - BMv2
  - Real-time High-Resolution Background Matting, CVPR 2021
  - (Best student paper hon’ble mention)

- **30fps**
  - BMv2
  - Background Matting: The World is Your Green Screen, CVPR 2020

- **8fps**
  - BMv1
  - Low resolution
  - Not real-time

Resolutions:
- **512 x 512**
- **1920 x 1080 (HD)**
- **3840 x 2160 (4K)**

Measured on Nvidia RTX 2080 Ti with batch size 1
Drawbacks of Background Matting:

✗ requires explicit background capture.
✗ can’t handle large camera motion.

Idea: Use RNN to aggregate temporal motion information to separate foreground and background
Robust Video Matting, WACV 2022.
Alpha Matting in presence of foreground-background interaction

Original | Input mask | Omnimatte (alpha) | Omnimatte (RGBA) | Background
---|---|---|---|---

Original | Input mask | Omnimatte (alpha) | Omnimatte (RGBA) | Background
---|---|---|---|---

Original | Input mask 1 | Omnimatte 1 (alpha) | Omnimatte 1 (RGBA) | Background
---|---|---|---|---

Input mask 2 | Omnimatte 2 (alpha) | Omnimatte 2 (RGBA) | --- | ---

Omnimatte: Associating Objects and Their Effects in Video, Siggraph 2020
Recap

• Semantic segmentation: detect masks of specific object classes
• Instance segmentation: detect masks of each instances of specific object class
• Panoptic segmentation: Semantic + Instance combined.

Evolution of Neural network architectures to solve segmentation:
• Fully convolutional networks (VGGlike)
• Mask R-CNN (instance segmentation)
• U-Net
• Transformer based architecture (Swin V2)

Interactive Segmentation (scribbles, points, text prompts etc):
• Segment Anything (SAM)
• Way more robust than semantic/instance segmentation

Beyond Image segmentation:
• Video object segmentation (unsupervised, weak-supervised, interactive)
• Alpha Matting
Slide Credits

• EECS 442/498 Computer Vision, by Justin Johnson & David Fouhey, U Michigan.