Practices in visual computing 1

Lab11: Fine-Tuning SAM2 and Detectron2

Simon Fraser University
Fall 2024

Outline

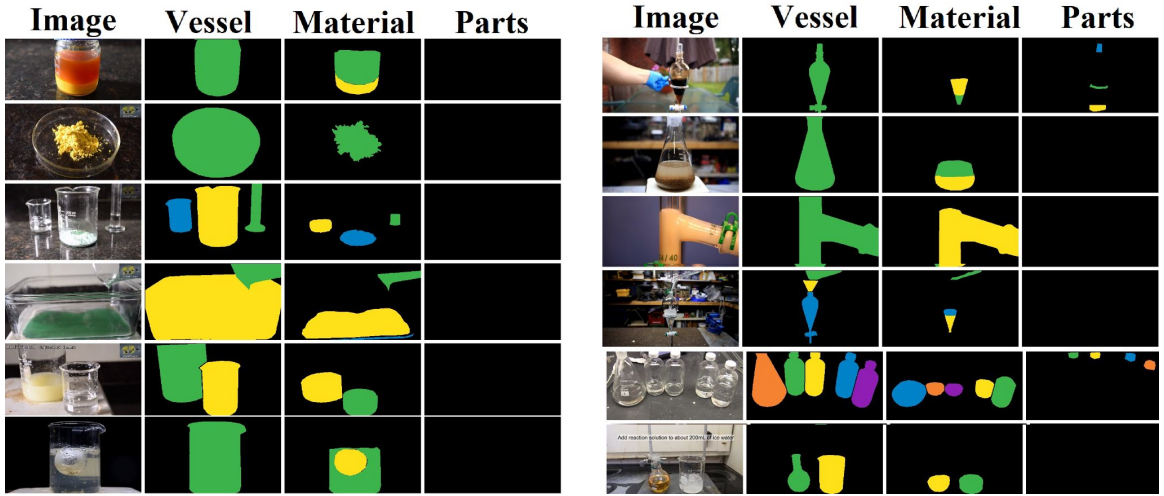
- SAM2 Fine-Tuning
- Mixed Precision Training
- Detectron2 Overview
- Fine-Tuning Detectron2

LabPicsV1 Dataset

A specialized dataset with annotated images of lab equipment, reagents, containers, and other lab objects.

Applications:

- Object detection and segmentation in laboratory settings
- Automated inventory management and safety monitoring



SAM2 Fine-Tuning

Steps:

- Prepare Data
  - Load LabPicsV1 in proper foramt
- Model Configuration
  - Load pre-trained SAM2 weights.
  - Adjust model layers if necessary (e.g., task-specific heads).
- Fine-Tuning

Optimizer: AdamW - Learning Rate: 1e-5 - Weight Decay: 4e-5

Mixed Precision Training

Combines FP16 and FP32 for faster computation and reduced memory usage.

Benefits:

- Faster training
- Lower GPU memory usage
- Maintains model accuracy using dynamic scaling

Key PyTorch tools:

- torch.cuda.amp
- GradScaler

Float16 vs Float32

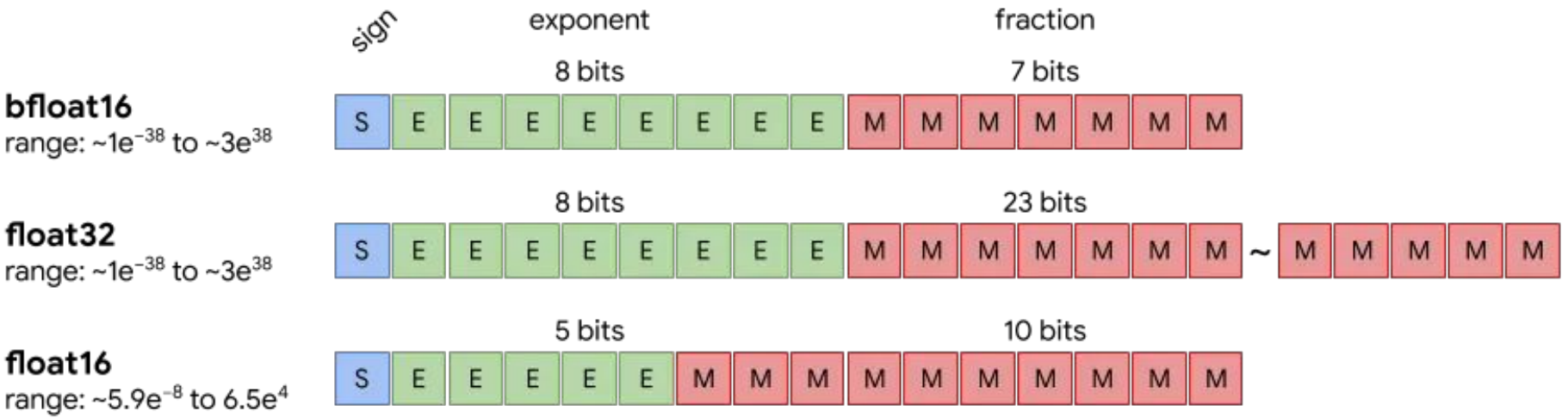
FP16 (Half-Precision):

- 16 bits per number, lower memory usage
- Faster computation on compatible GPUs
- Limited range and precision, prone to overflow/underflow

FP32 (Single-Precision):

- 32 bits per number, higher memory usage
- Stable gradients and wider precision range
- Slower computation compared to FP16

Float16 vs Float32



What is Mixed Precision?

Uses multiple numerical precision levels (e.g., FP32, FP16, BF16).

Balances computational efficiency and accuracy.

Benefits:

- Faster Computation: FP16/BF16 computations are faster.
- Reduced Memory Usage: FP16 uses half the memory of FP32.

Challenges of Mixed Precision

Numerical Instability:

- FP16 struggles with very small or very large values, leading to underflow or overflow.

Implementation Complexity:

- Managing precision requires careful handling.

Challenges of Mixed Precision

When to Use FP16:

- Matrix multiplications and other operations with large intermediate results.
- Suitable for most neural network layers to speed up computations.

When to Use FP32:

- Critical operations like loss calculations and gradient updates.
- Ensures numerical stability for small or sensitive values.

torch.cuda.amp

Automates mixed precision (AMP) training

- torch.cuda.amp.autocast(): Chooses appropriate precision automatically
- GradScaler: Dynamically scales gradients to prevent underflow

Simplifies implementation

Improves performance and memory usage

GradScaler

Scales gradients during backpropagation

Checks for inf/nan values in gradients

Dynamically adjusts scaling factor

GradScaler.step(optimizer)

Performs the optimizer's parameter update step, potentially skipping updates if gradients contain non-finite values (e.g., NaN or Inf).

- GradScaler checks the scaled gradients for finiteness.
- If gradients are valid, it unscales them (divides by the scale factor) and passes them to the optimizer for the weight update.
- If gradients are not finite, the update step is skipped, avoiding unstable parameter updates.

GradScaler.update()

Adjusts the scale factor used for scaling gradients in the next iteration.

- If gradients were finite during the previous iteration, the scale factor may increase to enhance training efficiency.
- If gradients were non-finite, the scale factor is reduced to improve stability.
- This dynamic adjustment ensures that the model can maximize the precision benefits of FP16 while avoiding instability.

Detectron2

Detectron2 is an open-source framework, developed by Facebook AI Research is the improved successor to Detectron, offering a more flexible and user-friendly approach for developers and researchers.

Architecture of Detectron2

Backbone: Feature extractor (Feature Pyramid Network)

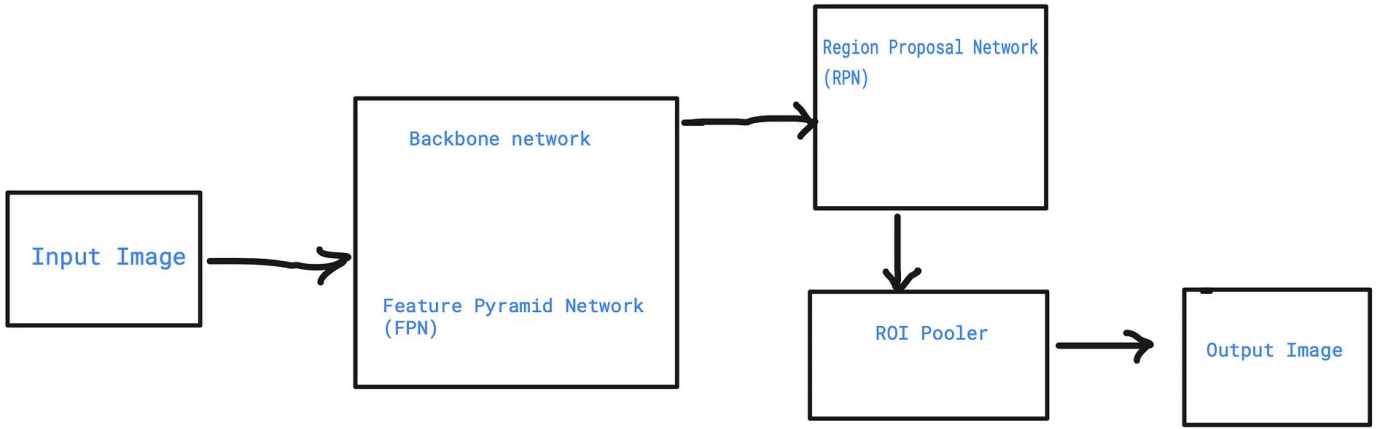
RPN (Region Proposal Network): Proposes candidate regions for detection.

Heads:

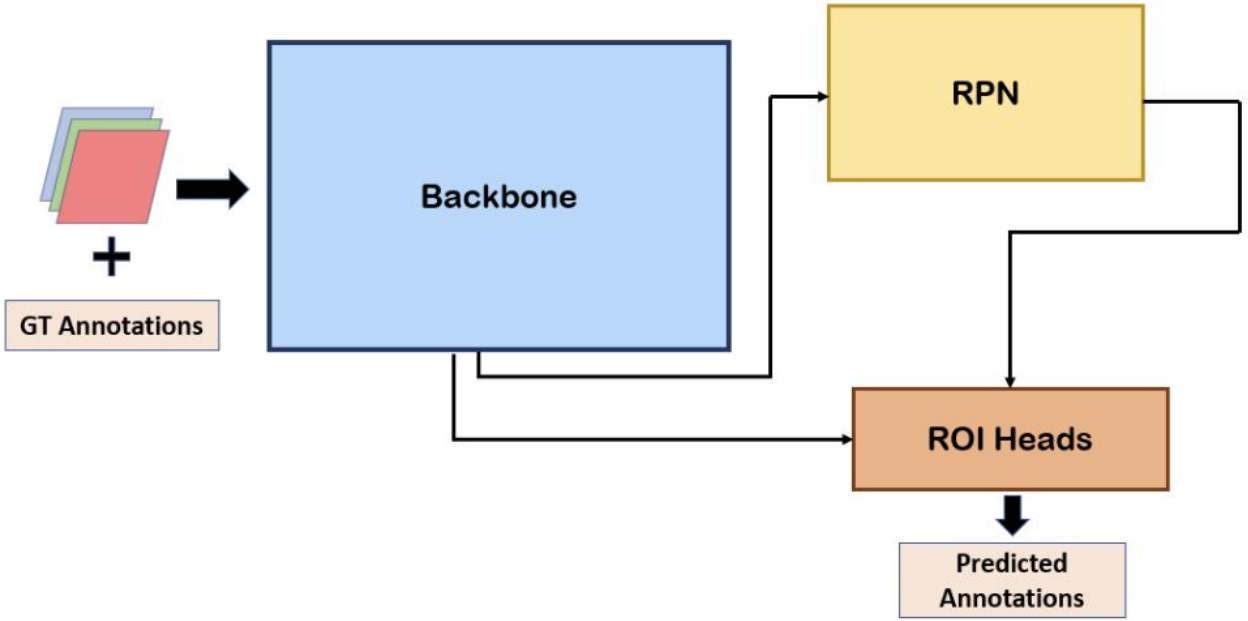
- Bounding Box Head: Refines region proposals into final detections.
- Mask Head: Predicts pixel-level masks for instances.
- Keypoint Head: Estimates keypoints for human pose estimation.

Loss Functions: Combines classification, regression, and segmentation losses.

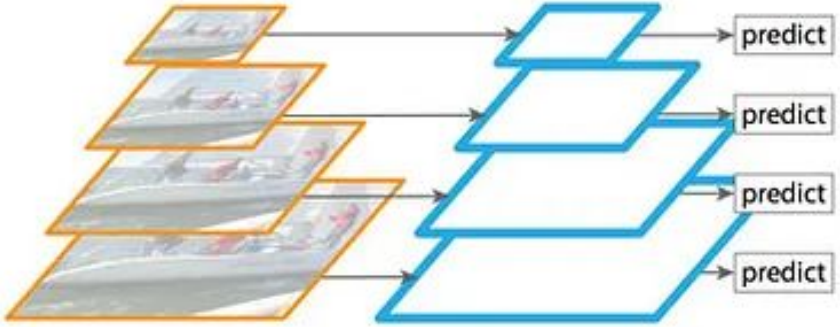
Architecture of Detectron2



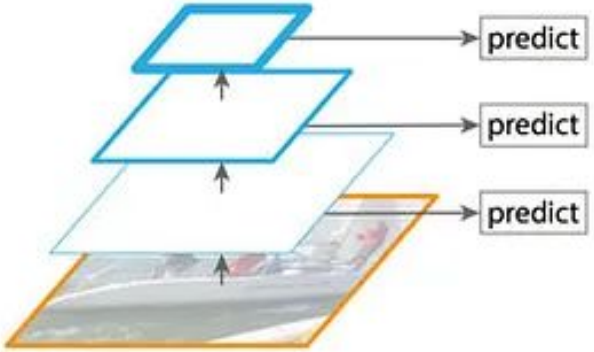
Architecture of Detectron2



Feature Pyramid Network



Pyramid of images

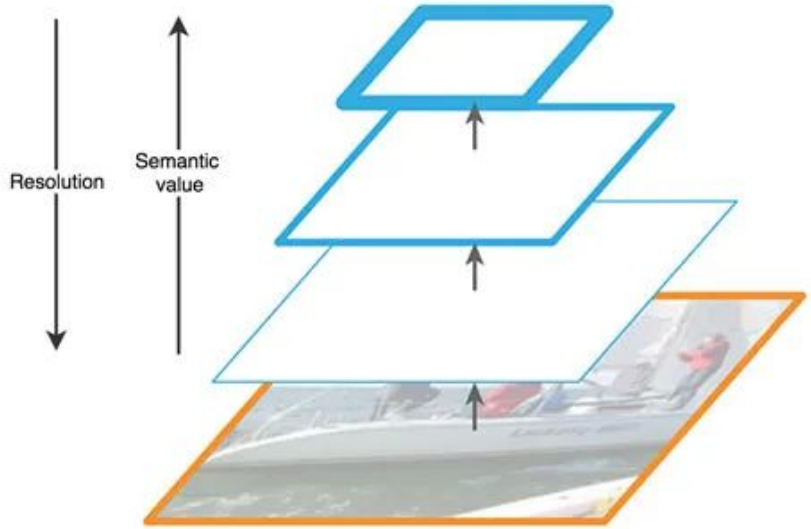


Pyramid of feature maps

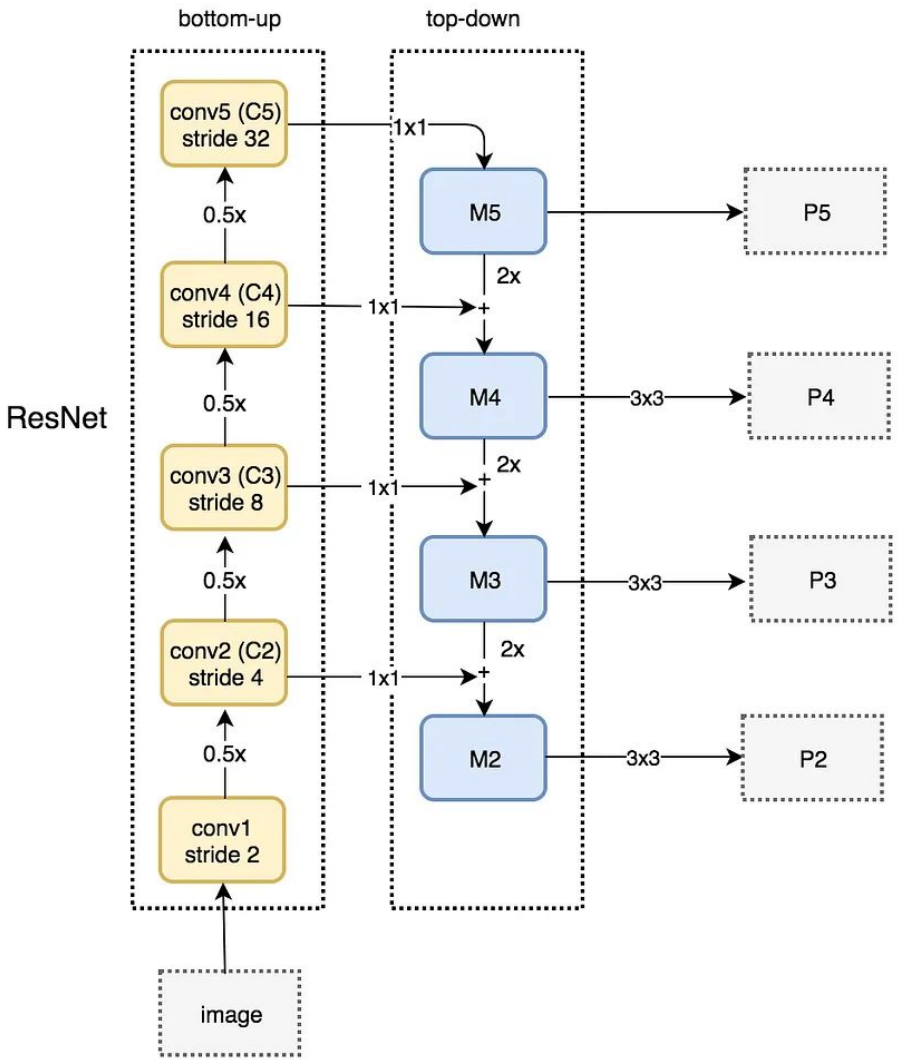
Bottom-up Pathway

Standard convolutional network for feature extraction.

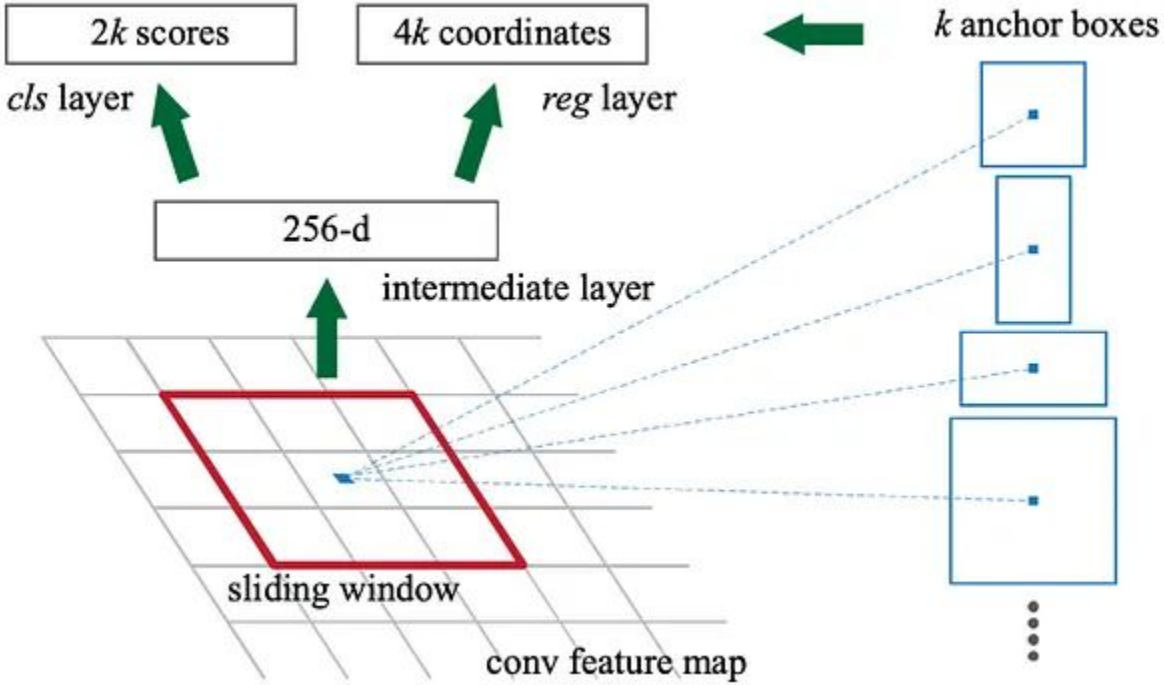
Processes input image to generate hierarchical feature maps.



Feature Pyramid Network



RPN (Region Proposal Network)



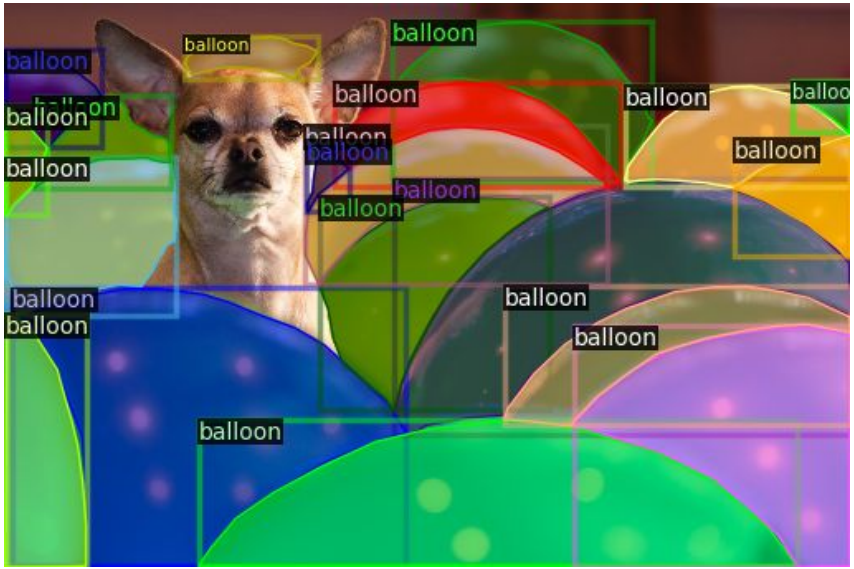
Detectron2


kite
kite
kite
kite
kite
person
person
person
person
person
person
person
person
person
person

Fine-Tuning Detectron2 on Balloon Dataset

Train a Detectron2 model to detect and segment balloons in images.

Balloons are not included in Detectron2's pre-built datasets (e.g., COCO), so a custom dataset is required.



Fine-Tuning Detectron2 on Balloon Dataset

https://detectron2.readthedocs.io/en/latest/tutorials/datasets.html

https://detectron2.readthedocs.io/en/latest/modules/utils.html?highlight=visualizer#module-detectron2.utils.visualizer

https://detectron2.readthedocs.io/en/latest/modules/solver.html

Reference

https://moocaholic.medium.com/fp64-fp32-fp16-bfloat16-tf32-and-other-members-of-the-zoo-a1ca7897d407

https://viso.ai/deep-learning/detectron2/

https://medium.com/@sumiteshn/a-gentle-introduction-to-detectron2-d5cb599cf001

Segment Anything, Kirillov et al - 2023