Unsupervised Object Localization: Observing the Background to Discover Objects

Oriane Siméoni

Chloé Sekkat

Gilles Puy

Antonin Vobecky

Eloi Zablocki

Patrick Pérez

JUNE 18-22, 2023

CVPR

VANCOUVER, CANADA

CZECH INSTITUTE OF INFORMATICS ROBOTICS AND CYBERNETICS CTU IN PRAGUE

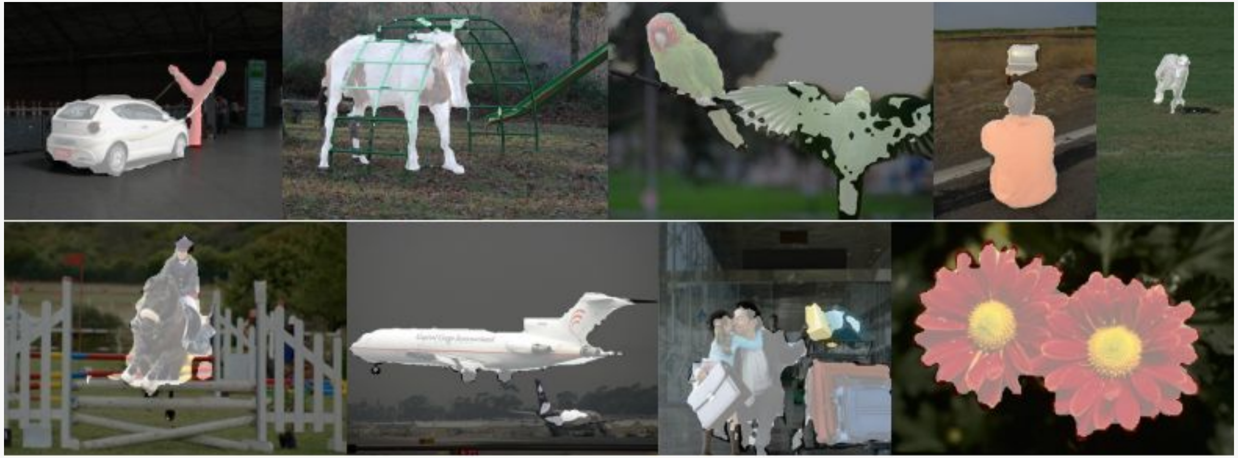
Valeo

Our method

FOUND

Key ideas

- **Unsupervised** object localization
- exploits DINO features with a single self-trained **conv1x1**
- **Quick 2h training** on **a single GPU** with **no annotation**
- Inference at **80 FPS** 🚀 on a V100



Our method

FOUND

- **Unsupervised** object localization
- exploits DINO features with a single self-trained **conv1x1**
- **Quick 2h training** on **a single GPU** with **no annotation**
- Inference at **80 FPS** 🚀 on a V100

How does it work?

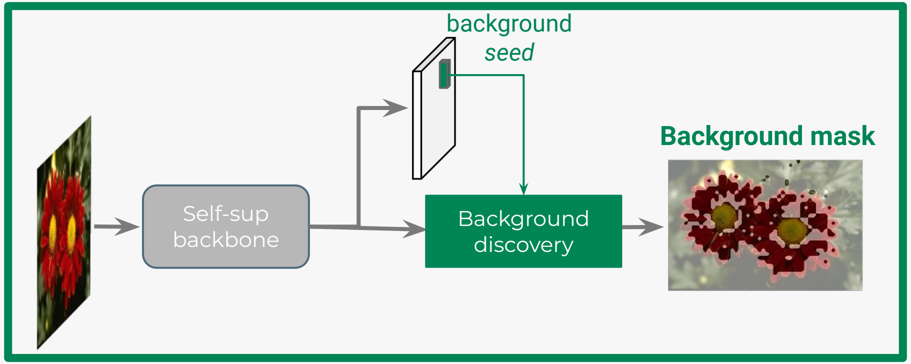
Our method

FOUND

- **Unsupervised** object localization
- exploits DINO features with a single self-trained **conv1x1**
- **Quick 2h training** on **a single GPU** with **no annotation**
- Inference at **80 FPS** 🚀 on a V100

How does it work?

- STEP 1: **Background Discovery**



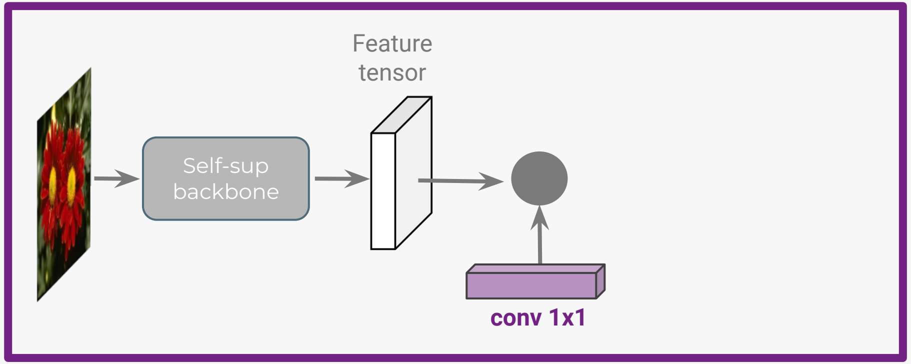
Our method

FOUND

- **Unsupervised** object localization
- exploits DINO features with a single self-trained **conv1x1**
- **Quick 2h training** on **a single GPU** with **no annotation**
- Inference at **80 FPS** 🚀 on a V100

How does it work?

- STEP 1: **Background Discovery**
- STEP 2: **Self-supervised refinement**



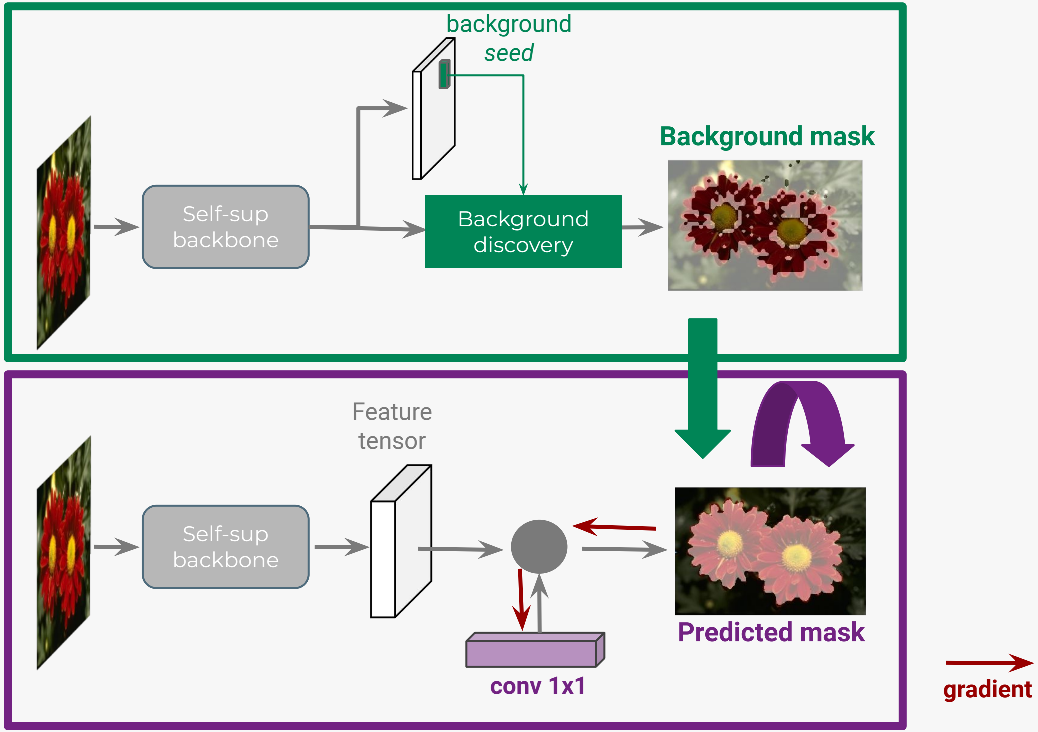
Our method

FOUND

- **Unsupervised** object localization
- exploits DINO features with a single self-trained **conv1x1**
- **Quick 2h training** on **a single GPU** with **no annotation**
- Inference at **80 FPS** 🚀 on a V100

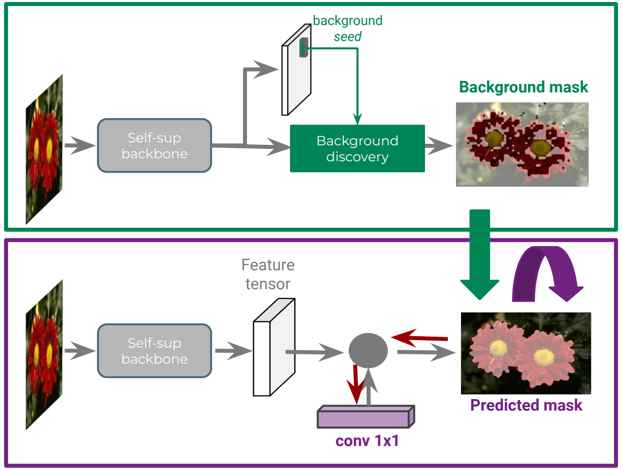
How does it work?

- STEP 1: **Background Discovery**
- STEP 2: **Self-supervised refinement**



Our method
FOUND

- **Unsupervised** object localization
- exploits DINO features with a single self-trained **conv1x1**
- **Quick 2h training** on **a single GPU** with **no annotation**
- Inference at **80 FPS** 🚀 on a V100



Inference (no post-processing)

Let’s get into details

Related works

Construction of initial masks

Leveraging **self-supervised** features

- Object *seed* = patch with **least connection** & select connected similar patches [LOST, BMVC21]
- Use a **normalized graph-cut** & separate an object from the highly connected patches [TokenCut, CVPR22; DSM, CVPR22]
- Use **multiple** self-supervised features to perform spectral clustering [SelfMask, CVPRW22]
- Generates correlation maps with different queries and rank + filter them [FreeSolo, CVPR22]

Regularization through training with pseudo-labels (initial masks)

Train a classic **detector** (+CAD) on top of coarse bounding boxes

Train an instance segmenter [CutLER, CVPR23]

Train an encoder/decoder architecture with learnable queries

Train an instance segmenter

In all cases, we observe **a large boost**

Our approach: **FOUND**

Our **coarse masks**

- Look for the **background** instead of objects
- **No hypotheses** about objects
- Quick computation

Our **model**

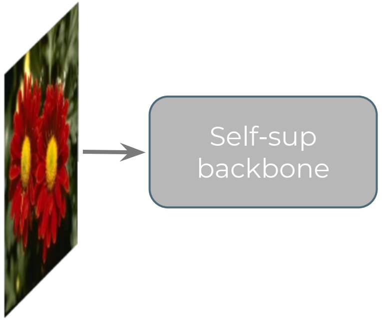
- No large detector/segmentation model
- A single **conv1x1** layer
- Trained in 2h on a single GPU

Our **inference**

- Runs at **80 FPS**
- Reaches SoTA results

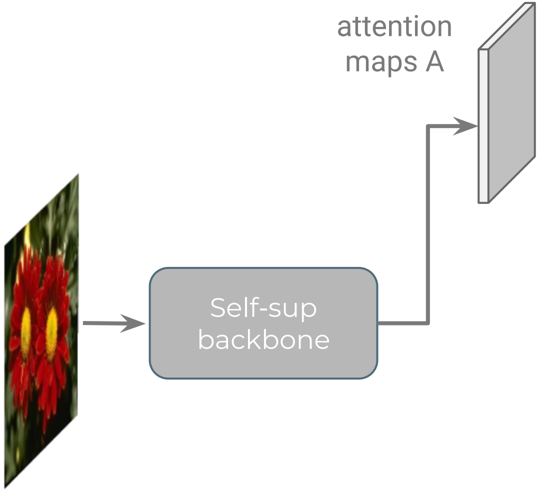
Background discovery

Background discovery



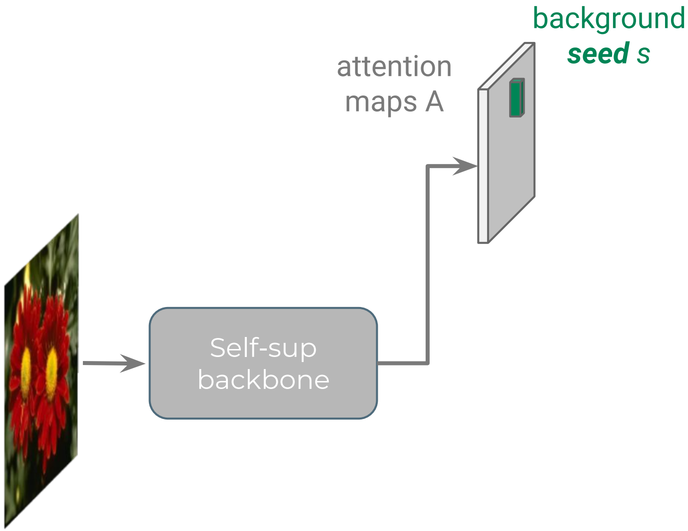
- backbone = **ViT model** trained in a self-sup fashion
- N patches

Background discovery



- backbone = **ViT model** trained in a self-sup fashion
- N patches
- $\mathbf{A} \in \mathbb{R}^{N \times h}$ the self-attention maps between the **CLS** token

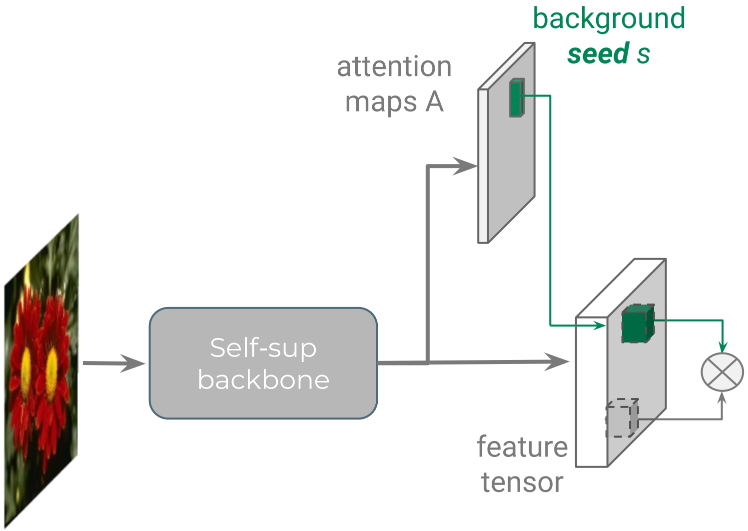
Background discovery



- backbone = **ViT model** trained in a self-sup fashion
- N patches
- $\mathbf{A} \in \mathbb{R}^{N \times h}$ the self-attention maps between the **CLS** token
- Background ***seed*** *s*

$$s = \operatorname{argmin}_{p \,\in\, \{1,\ldots,N\}} \sum_{i=1}^{h} \mathbf{A}_{pi}$$

Background discovery



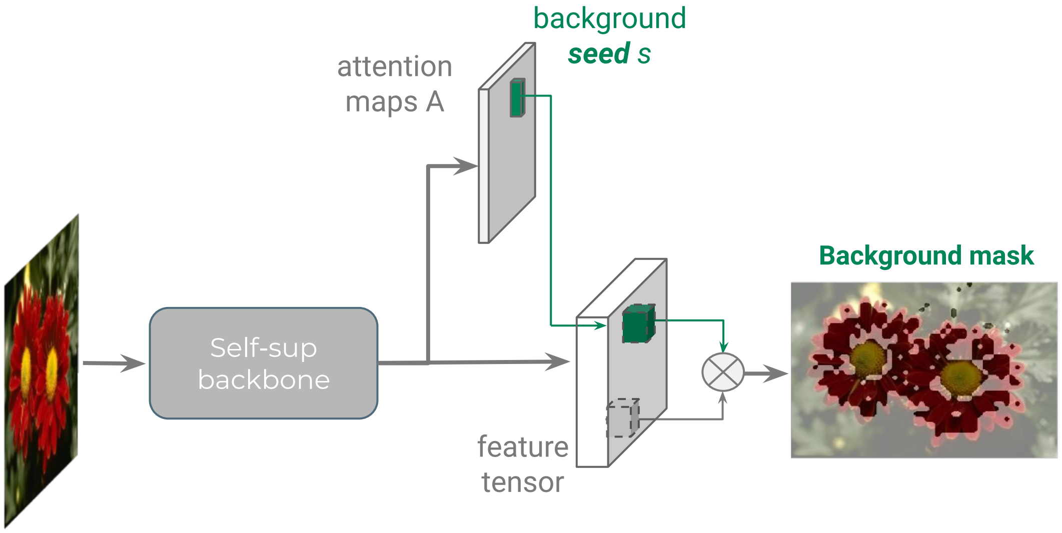
- backbone = **ViT model** trained in a self-sup fashion
- N patches
- $\mathbf{A} \in \mathbb{R}^{N \times h}$ the self-attention maps between the **CLS** token
- Background ***seed*** *s*

$$s = \operatorname{argmin}_{p \,\in\, \{1,\dots,N\}} \sum_{i=1}^{h} \mathbf{A}_{pi}$$

- Find all **correlated** patches to build **binary** coarse mask

$$\mathbf{M}_p^b = \begin{cases} 1 & \text{if } \operatorname{sim}(\tilde{f}_p, \tilde{f}_s) \geq \tau, \\ 0 & \text{otherwise,} \end{cases} \quad p = 1, \dots, N,$$

Background discovery



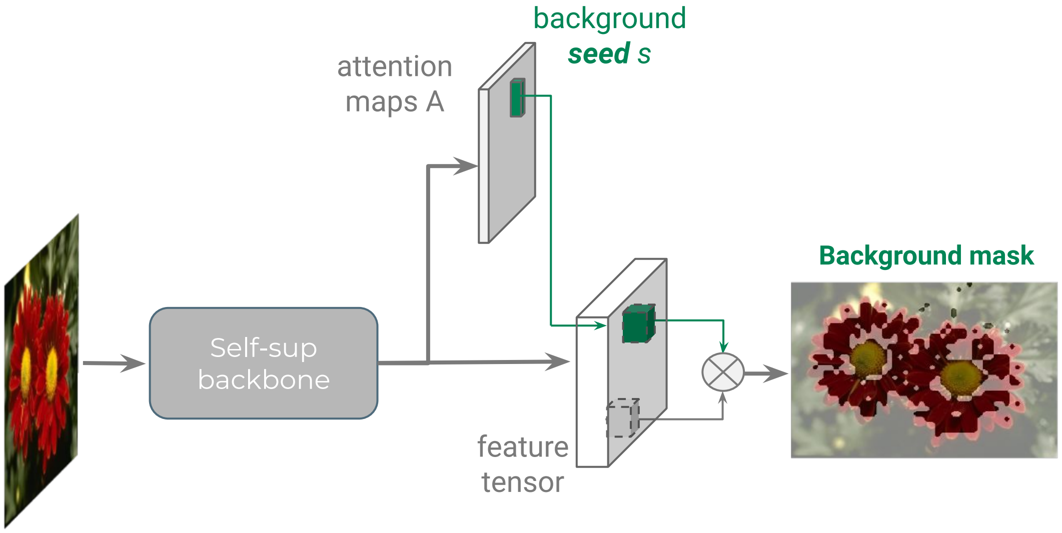
- backbone = **ViT model** trained in a self-sup fashion
- N patches
- $\mathbf{A} \in \mathbb{R}^{N \times h}$ the self-attention maps between the **CLS** token
- Background ***seed*** *s*

$$s = \operatorname{argmin}_{p \in \{1,\ldots,N\}} \sum_{i=1}^{h} \mathbf{A}_{pi}$$

- Find all **correlated** patches to build **binary** coarse mask

$$\mathbf{M}_p^b = \begin{cases} 1 & \text{if } \operatorname{sim}(\tilde{f}_p, \tilde{f}_s) \geq \tau, \\ 0 & \text{otherwise}, \end{cases} \quad p = 1, \ldots, N,$$

Background discovery



- backbone = **ViT model** trained in a self-sup fashion
- N patches
- $\mathbf{A} \in \mathbb{R}^{N \times h}$ the self-attention maps between the **CLS** token
- Background ***seed*** *s*

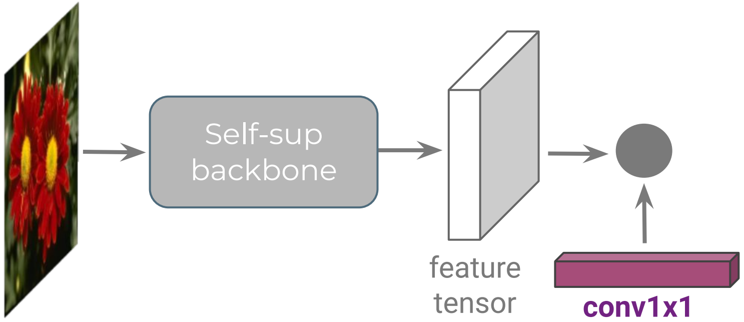
$$s = \mathrm{argmin}_{p \,\in\, \{1,\ldots,N\}} \sum_{i=1}^{h} \mathbf{A}_{pi}$$

- Find all **correlated** patches to build **binary** coarse mask

$$\mathbf{M}_p^b = \begin{cases} 1 & \text{if } \mathrm{sim}(\tilde{f}_p, \tilde{f}_s) \geq \tau, \\ 0 & \text{otherwise,} \end{cases} \quad p = 1, \ldots, N,$$

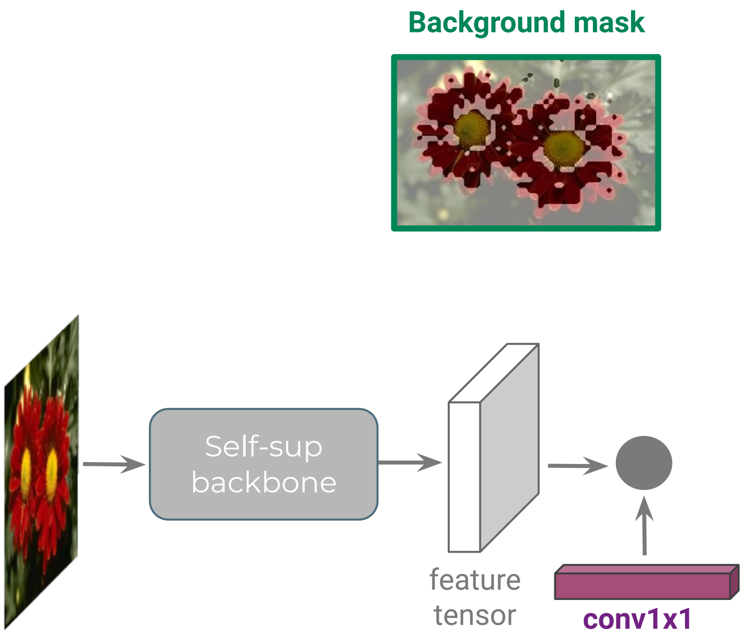
Self-Supervised Refinement

Self-supervised refinement



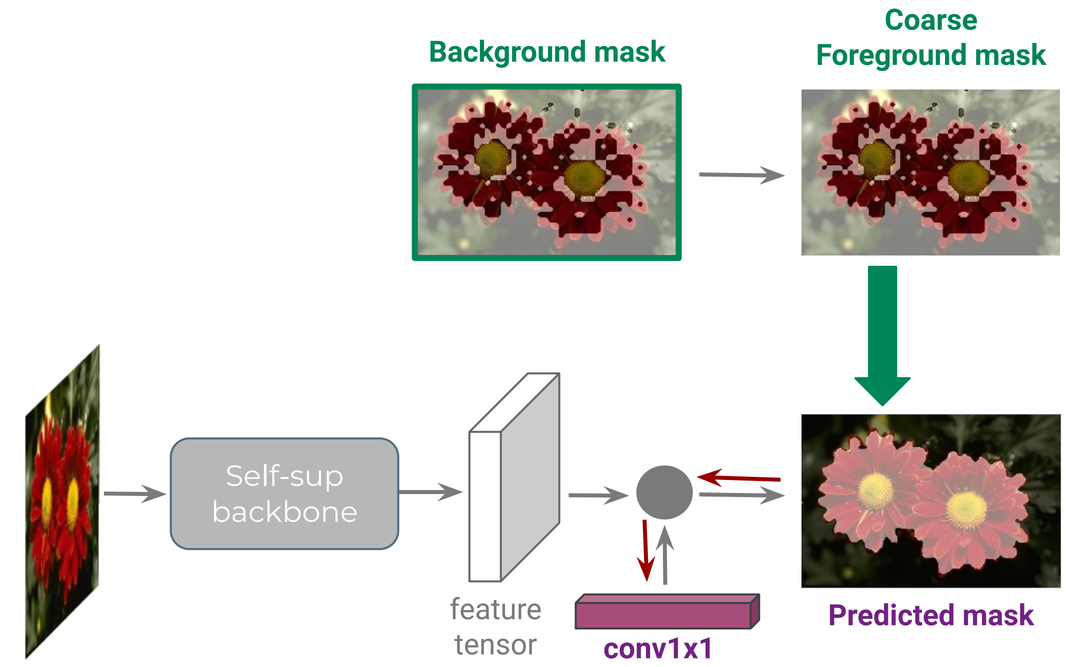
- Our segmentation head is a **single 1×1 convolution**

Self-supervised refinement



- Our segmentation head is a **single 1×1 convolution**

Self-supervised refinement

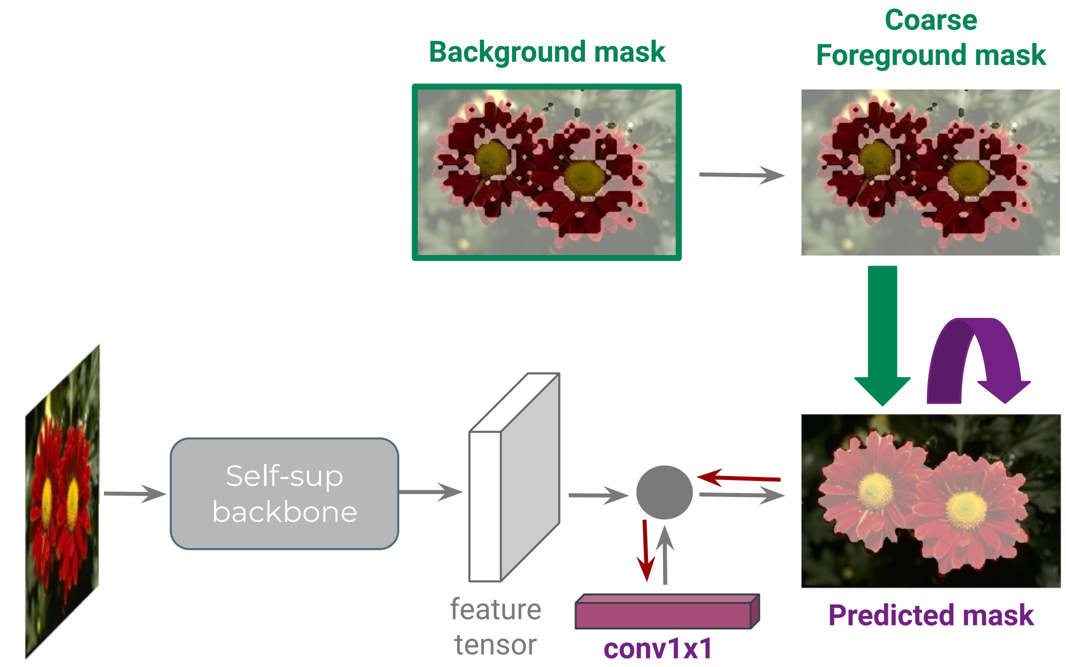


- Our segmentation head is a **single 1×1 convolution**
- **Trained** with 2 *binary cross-entropy losses* using as pseudo-gt:
  - the coarse foreground masks

gradient

Binary-cross entropy

Self-supervised refinement

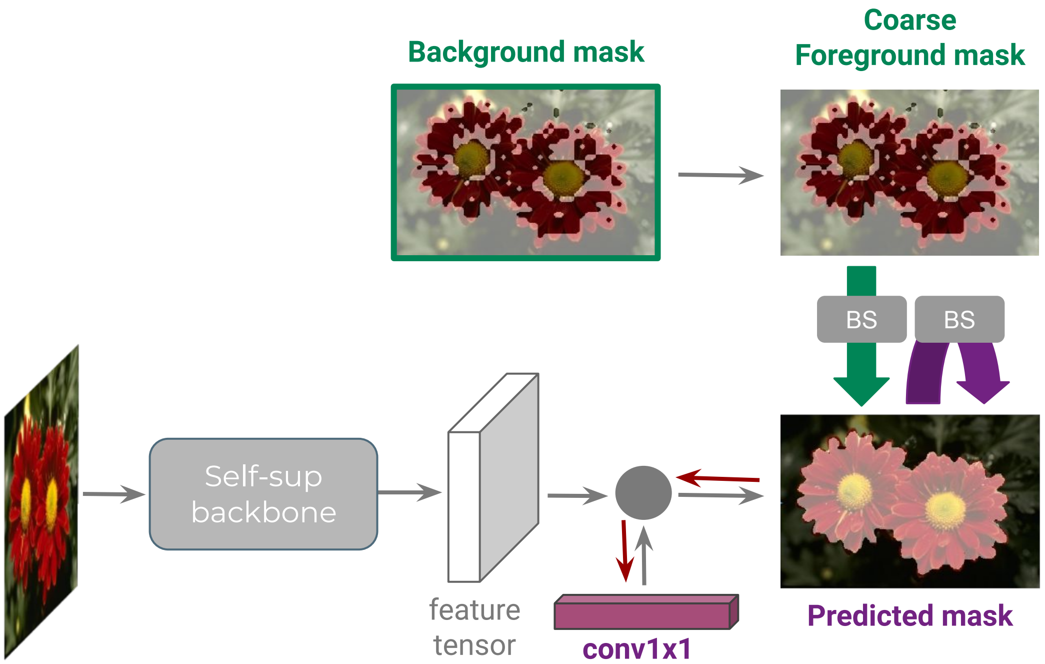


- Our segmentation head is a **single 1×1 convolution**
- **Trained** with 2 *binary cross-entropy losses* using as pseudo-gt:
  - the coarse foreground masks
  - a refined version of the layer output (using bilateral solver)

gradient

Binary-cross entropy

Self-supervised refinement



- Our segmentation head is a **single 1×1 convolution**
- **Trained** with 2 *binary cross-entropy losses* using as pseudo-gt:
  - the coarse foreground masks
  - a refined version of the layer output (using bilateral solver)
- Bilateral solver is used to refine masks along pixel edges

BS Bilateral Solver

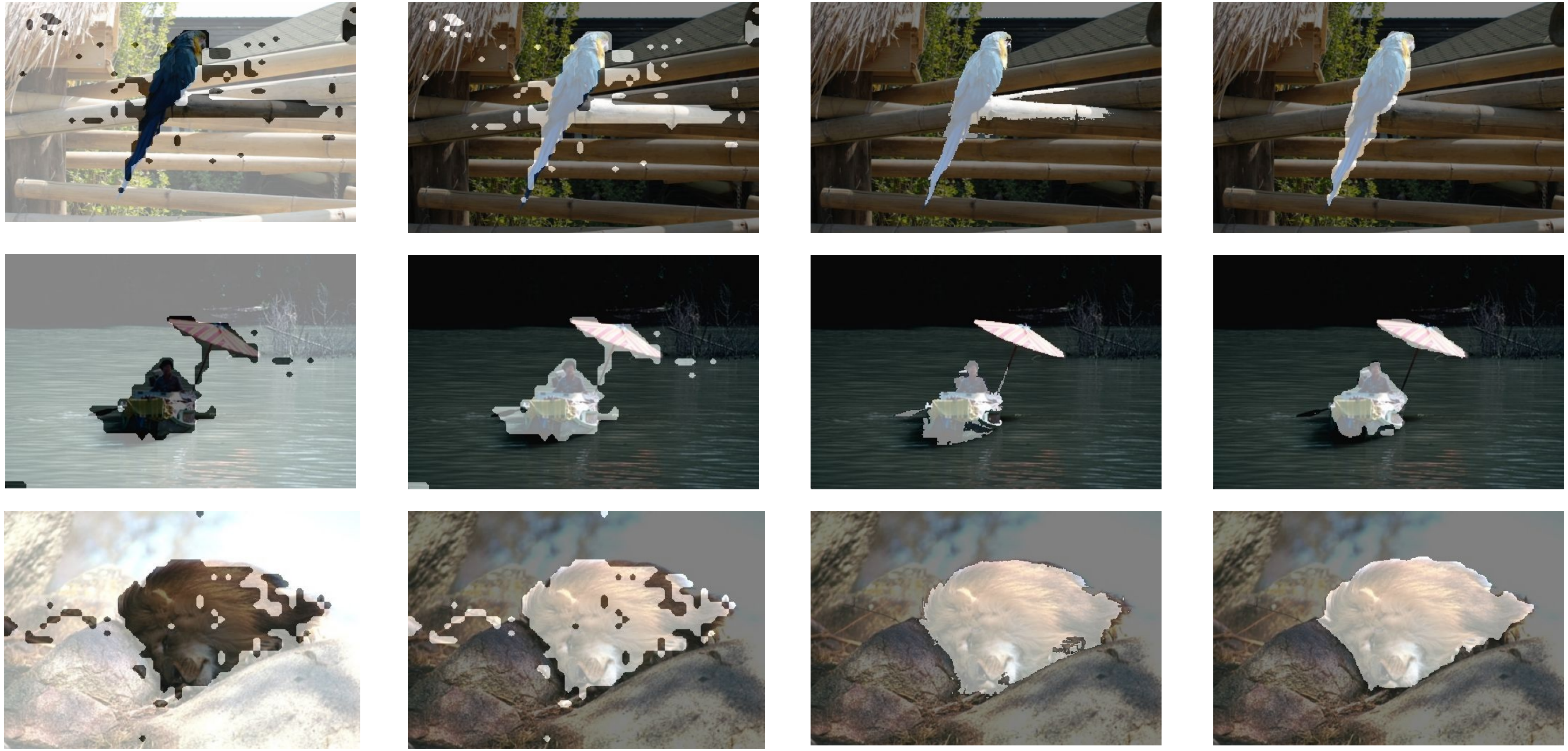
gradient

Binary-cross entropy

Overview results

FOUND

Background mask	Foreground mask	Refined foreground mask	



More details & Results

Some details

- **Backbone**: ViT-S/8
  Self-supervised features: DINO [Caron et al, NeurIPS20]

- **FOUND** trained for 500 iterations on **DUTS-TR** (10k images) [Wang et al, CVPR17] ~ **2 epochs**.

- We evaluate on diverse images from datasets like PascalVOC, COCO, DUT-OMRON and ECSSD

SoTA in unsupervised localization tasks

Unsupervised object discovery

Discover at least a correct **single object** in the image

Unsupervised saliency detection

Discover the **salient** objects

Unsup. Semantic Segmentation Retrieval

Compute feature per object mask and perform **retrieval**

→ We achieve **SoTA** results, with a **lighter** & **faster** model

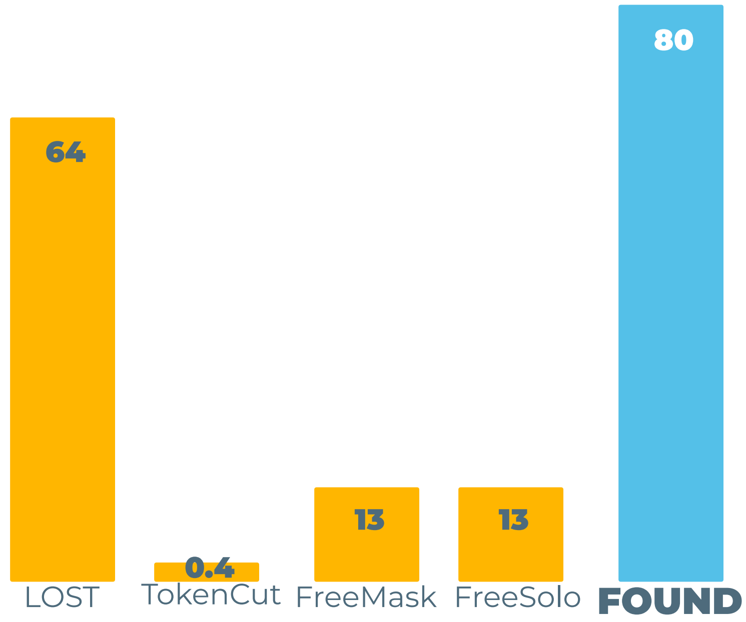
Lighter & Faster

FOUND is fast at **training** & **inference**

- Our model is a **single 1×1 convolution**
- Trained in **only 2h**

SelfMask	FreeSolo	DINOSAUR	**FOUND**
~ 36M	~ 66M	> 5M	**770**

 # learnable parameters



Inference FPS

Out-of-domain predictions (no post-processing)





A last video & come and talk to us

Thank you for your attention !