

Learning to Understand Visual Data with Minimal Human Supervision

Yong Jae Lee

University of California, Davis



Success in modern visual recognition research

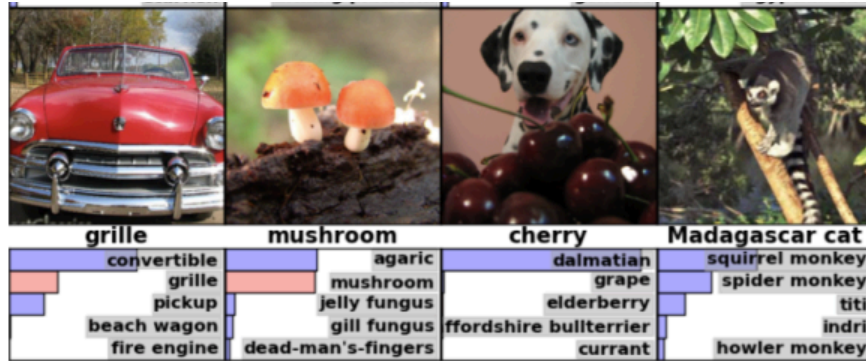
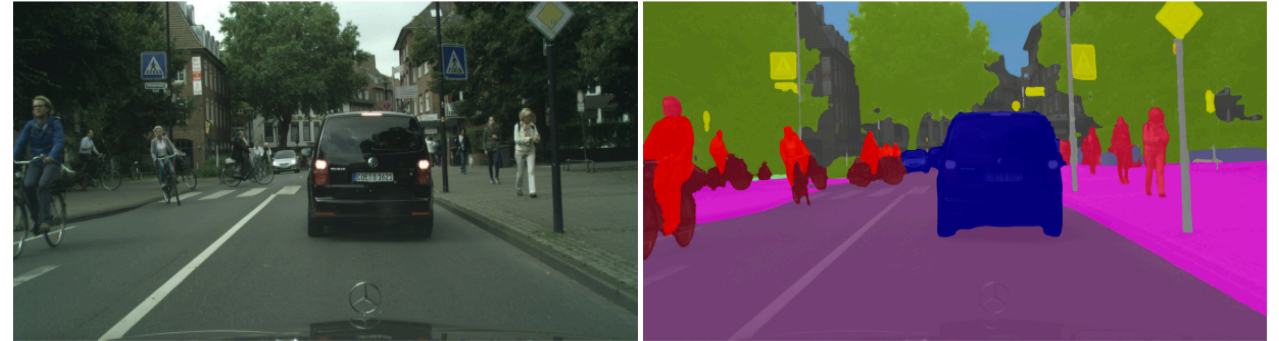


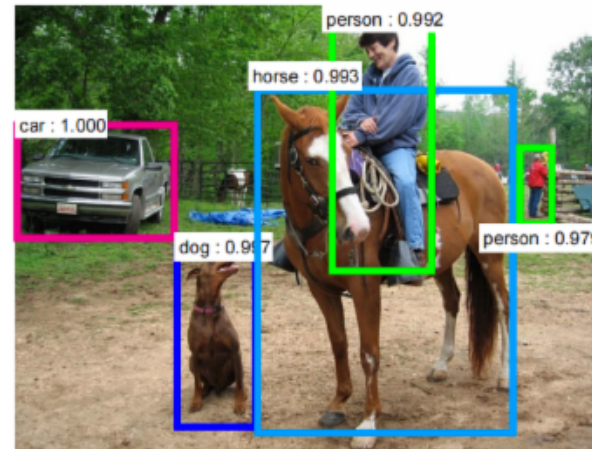
Image classification



Semantic segmentation



Pose recognition

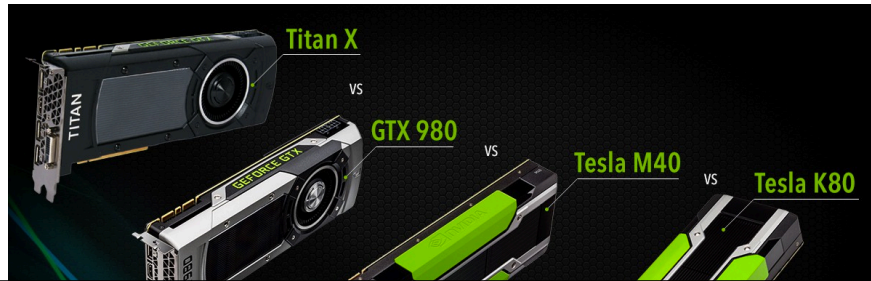


Object detection

... and many more

Ingredients for success today

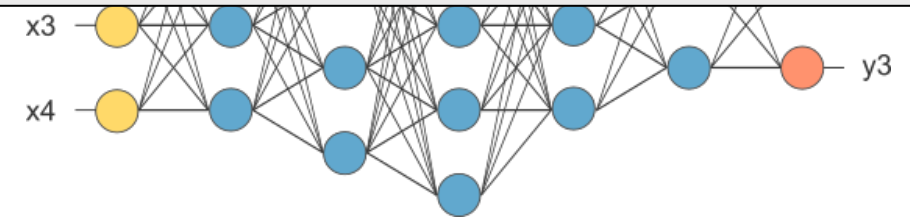
1. Big compute (GPUs)



3. Big models (deep neural nets)

Which ingredient will be the *bottleneck* for tomorrow's success?

2. Big labeled data



Ingredients for success today

1. Big compute (GPUs)



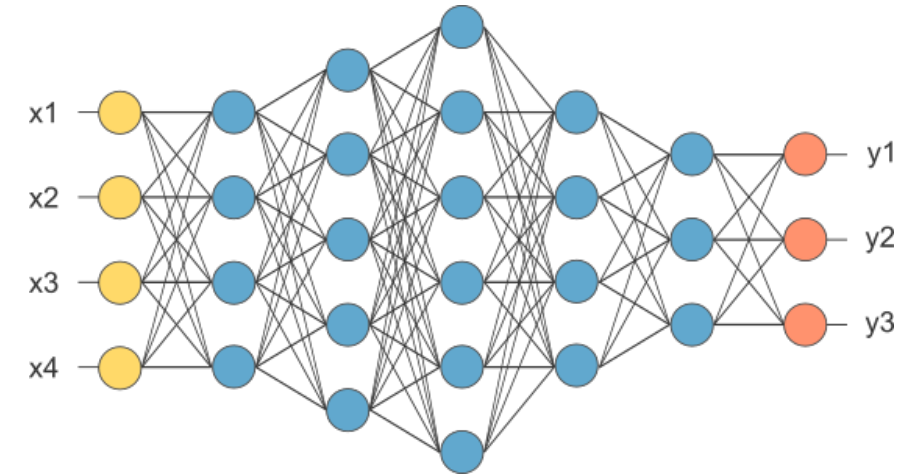
2. Big **labeled** data



Requires expensive, direct
human supervision



3. Big models (deep neural nets)



Direct supervision can be costly



70,000+ annotation hours for 328K images but *only* 80 object categories (MS COCO)



Requires pixel-level semantic labels

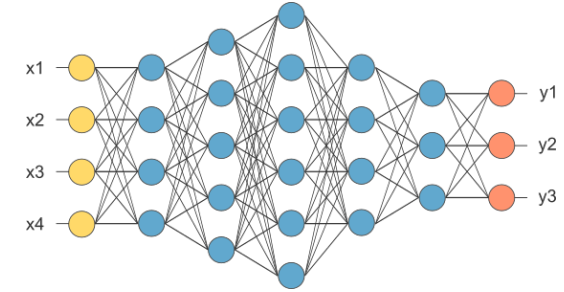
Direct supervision can be challenging



Ambiguity in what to label

right image more masculine?

Learning to understand visual data



Raw sensory input



Diverse data



Multi-modal sensory inputs



Dynamic environments



Minimal human supervision



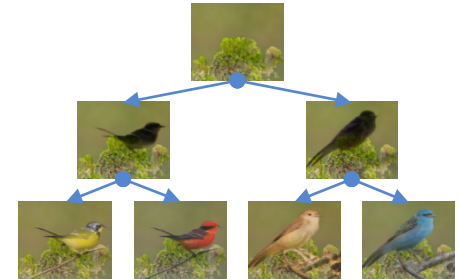
Outline

- Visual scene understanding with *minimal human supervision*

- *Localize* objects with only image-level tag annotations?



- *Generate* fine-grained object details without fine-grained annotations?

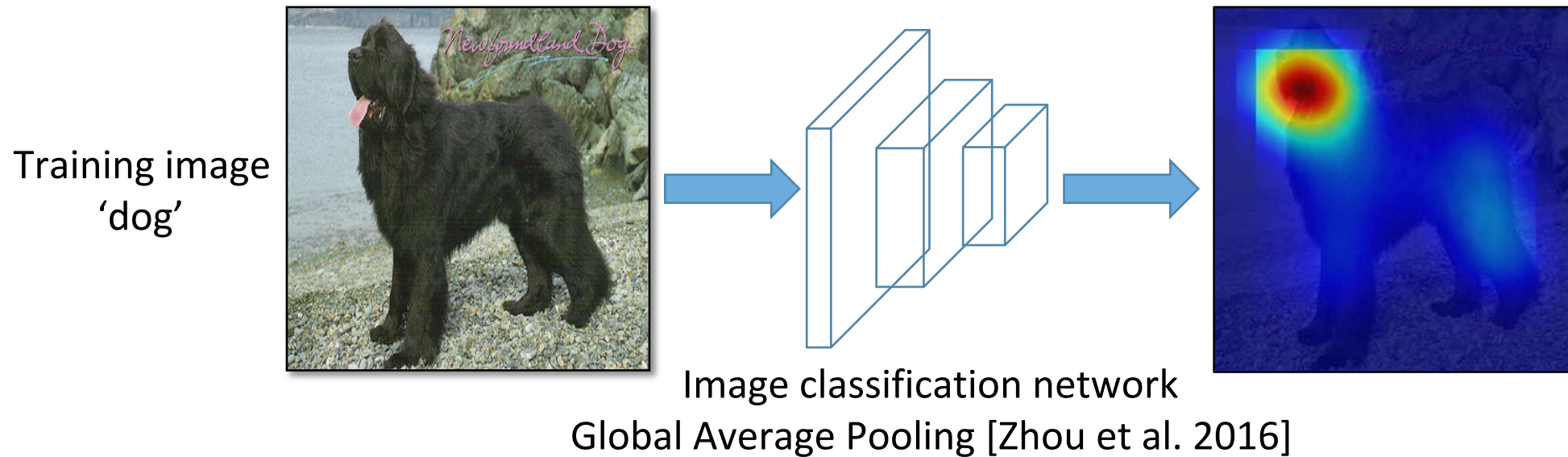


- Towards visual scene understanding in *dynamic environments*

- *Segment* object instances in real-time?



Learning to localize objects with image-label supervision



Model focuses only on the most discriminative part (i.e. dog's face) for image classification

[Weber et al. 2000, Pandey & Lazebnik 2011, Deselaers et al. 2012, Song et al. 2014, ...]

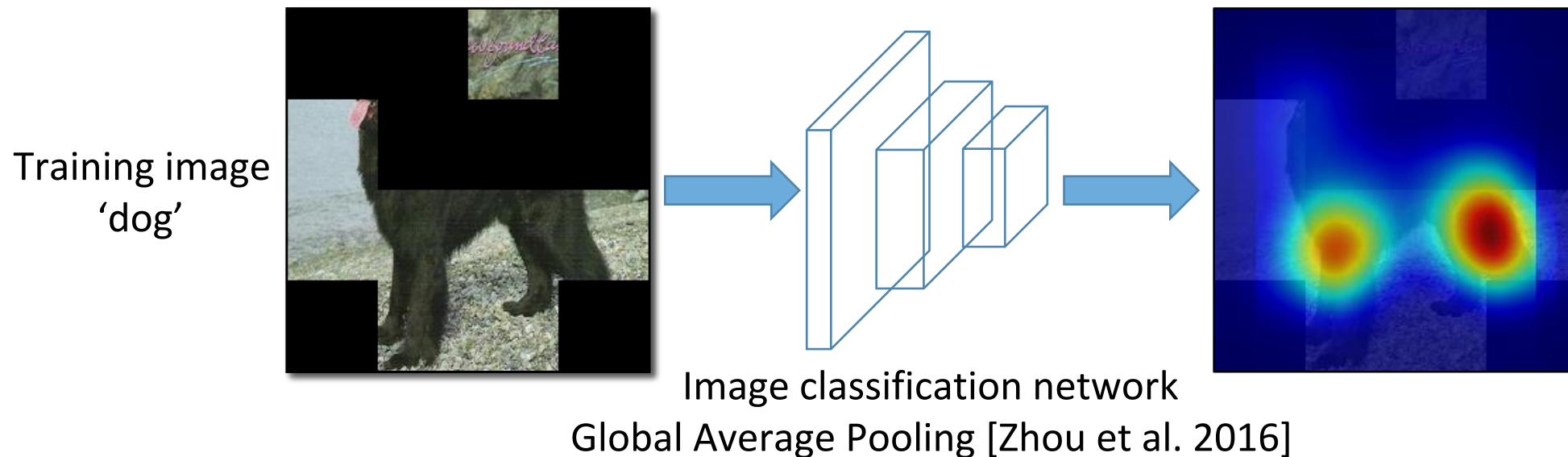
Our idea: Hide and Seek (HaS)

Training image
'dog'



[K. Singh and Y. J. Lee, "Hide-and-Seek", ICCV 2017]

Our idea: Hide and Seek (HaS)



Hide patches to force the network to *seek* other relevant parts

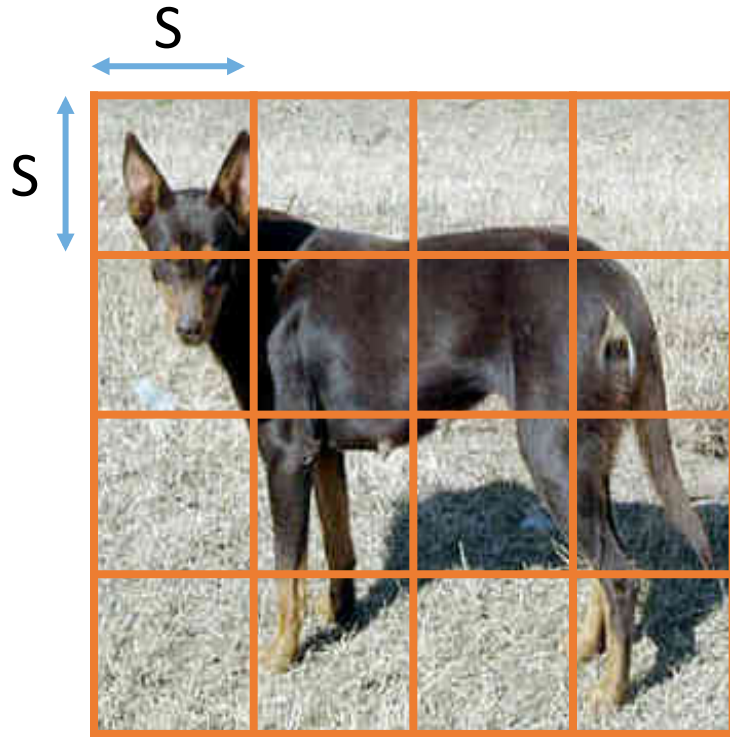
[K. Singh and Y. J. Lee, "Hide-and-Seek", ICCV 2017]

Divide the training image into a grid of patch size $S \times S$



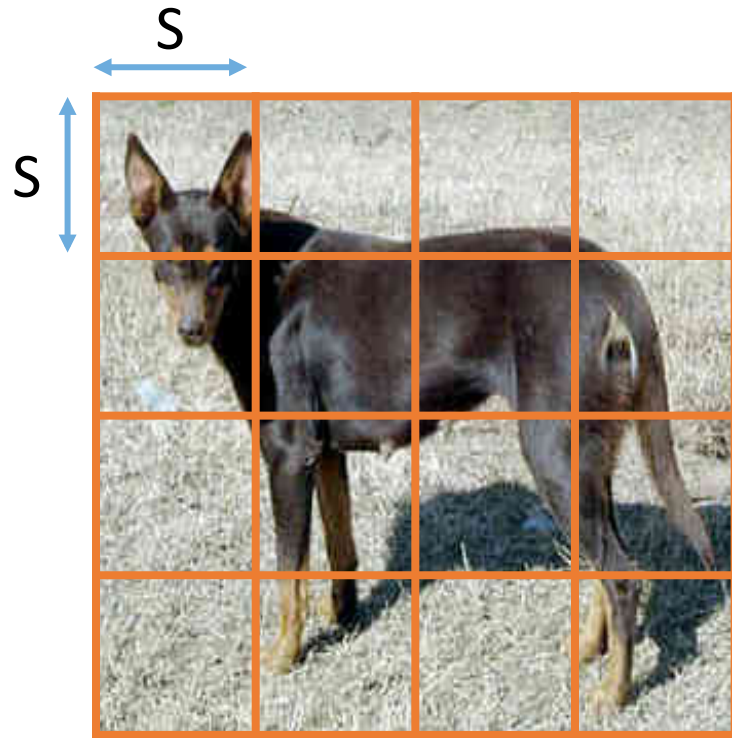
Training image
with label 'dog'

Divide the training image into a grid of patch size $S \times S$

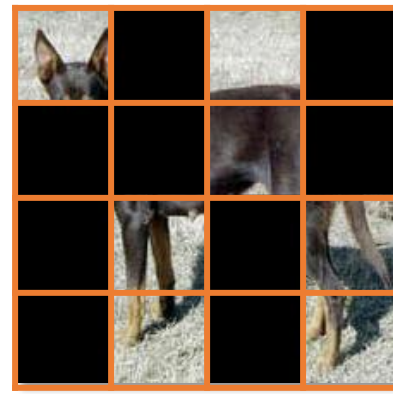


Training image
with label 'dog'

Randomly hide patches

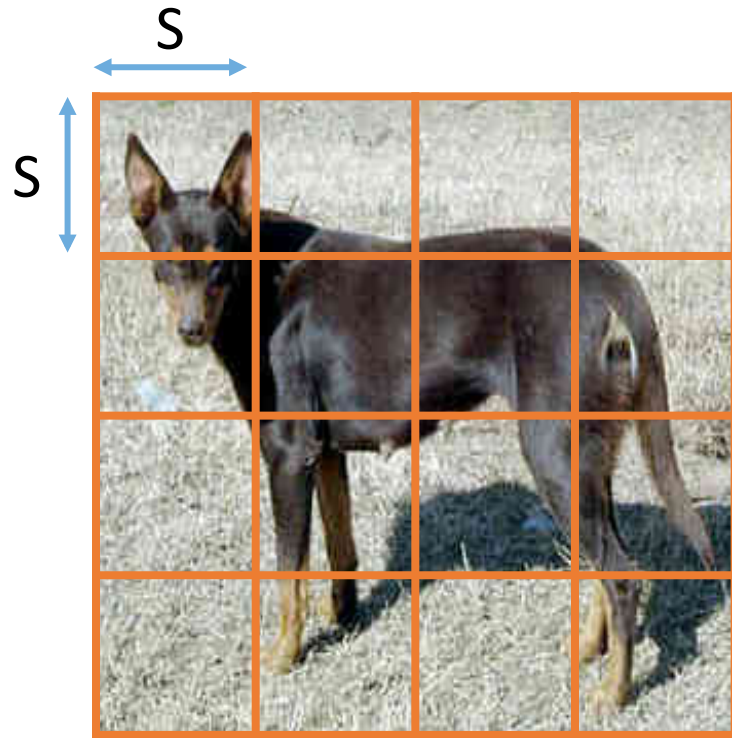


Training image
with label 'dog'

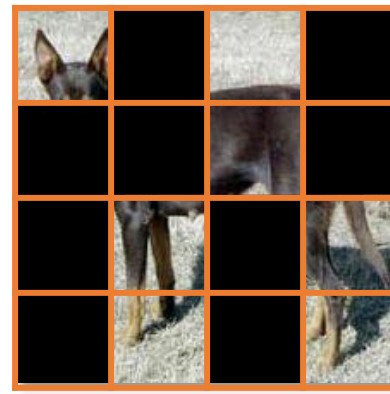
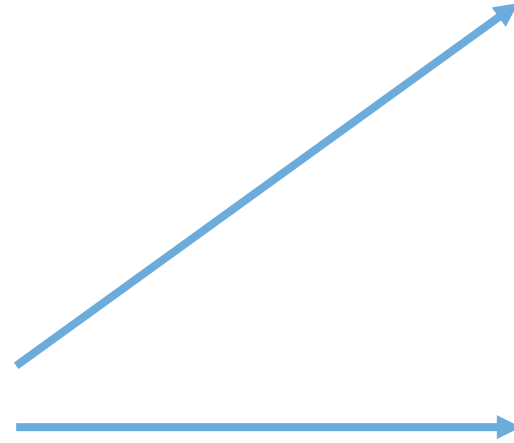


Epoch 1

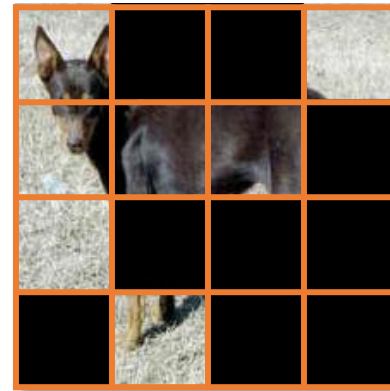
Randomly hide patches



Training image
with label 'dog'

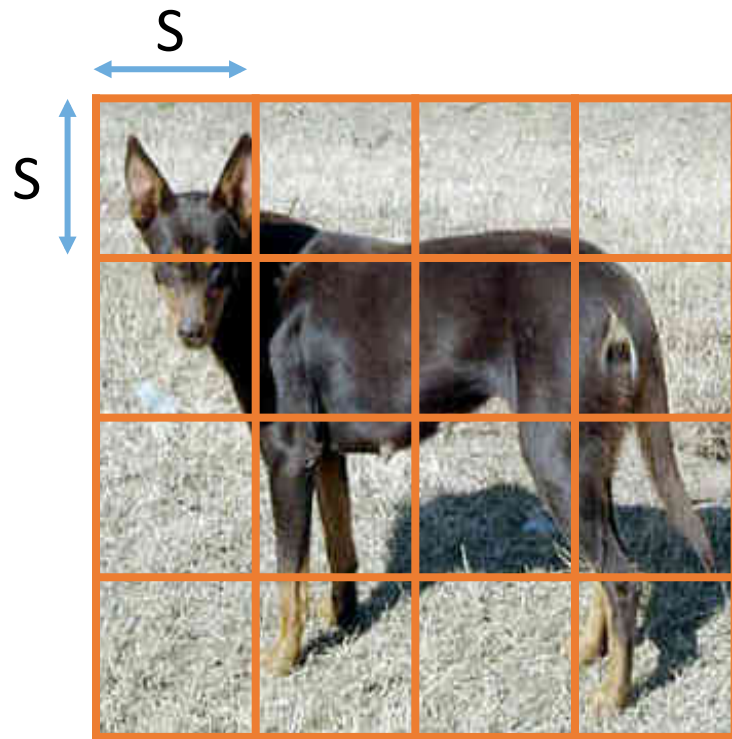


Epoch 1

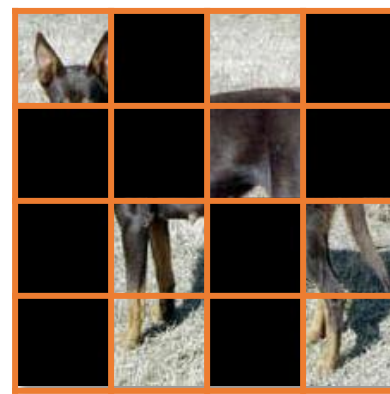


Epoch 2

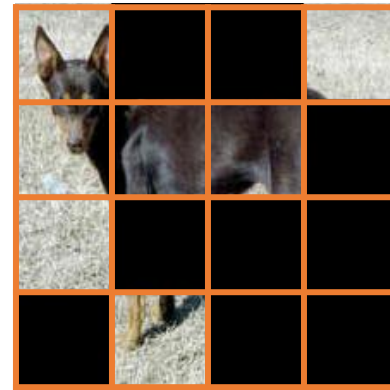
Randomly hide patches



Training image
with label 'dog'

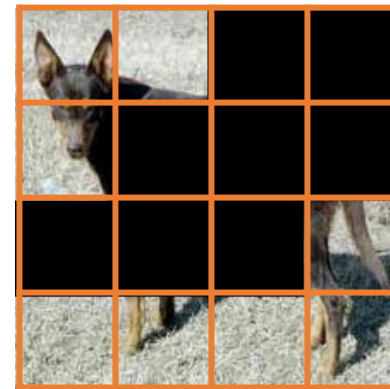


Epoch 1



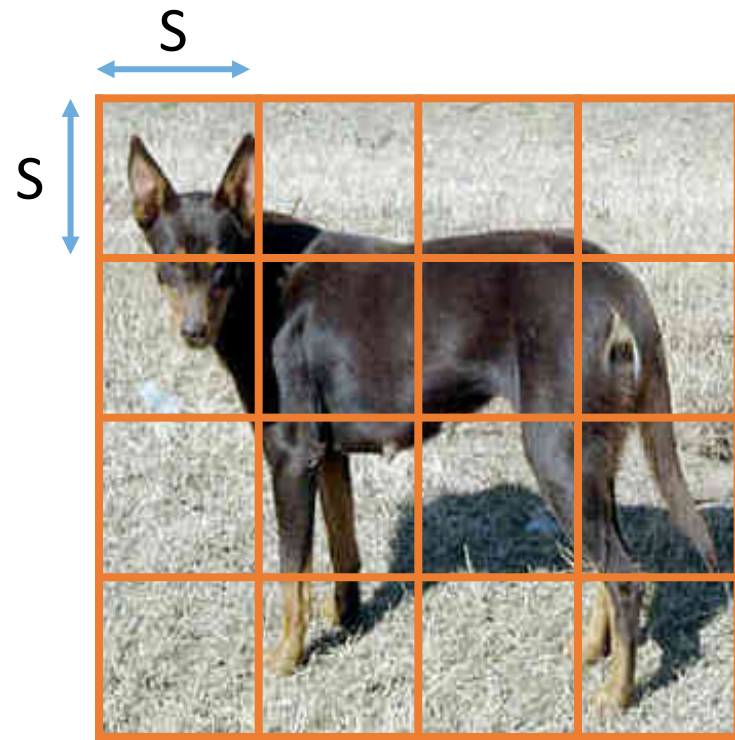
Epoch 2

⋮

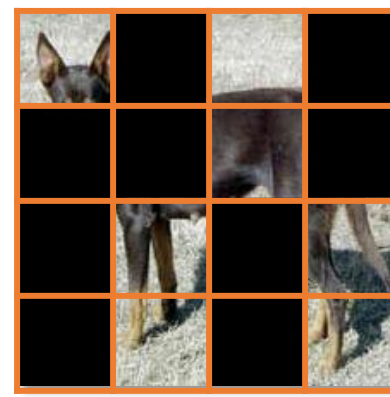


Epoch N

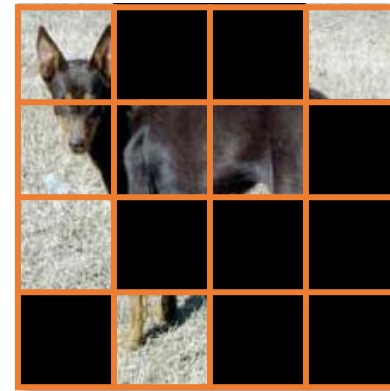
Feed each hidden image to image classification CNN



Training image
with label 'dog'

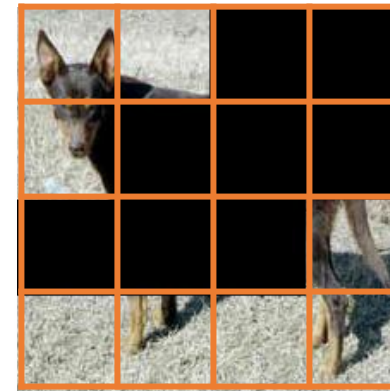


Epoch 1

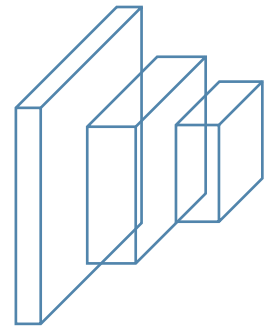


Epoch 2

⋮

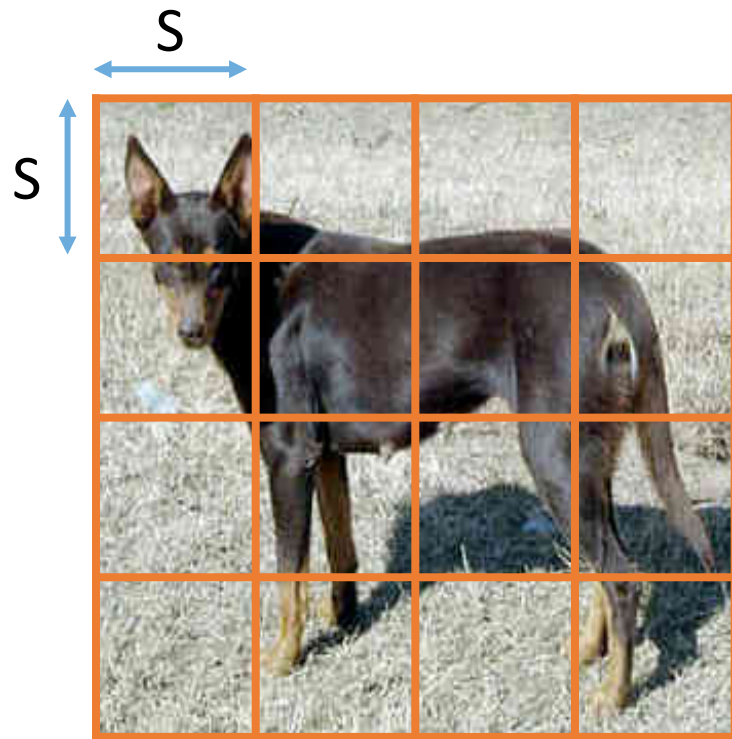


Epoch N

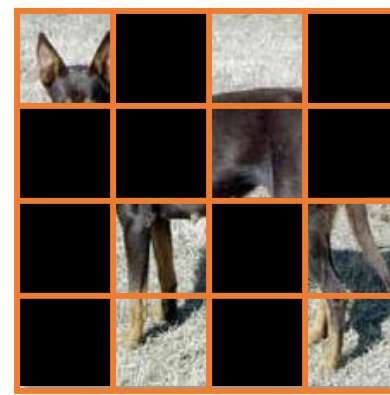


CNN

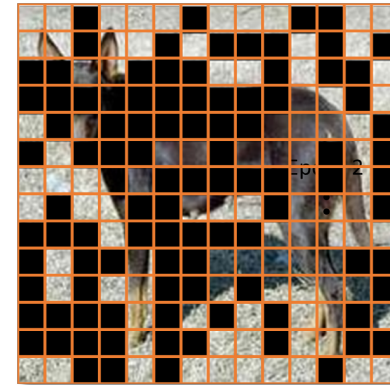
Feed each hidden image to
image classification CNN



Training image
with label 'dog'

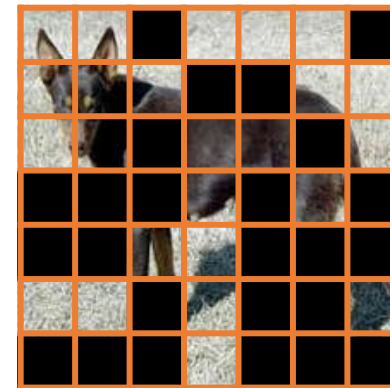


Epoch 1

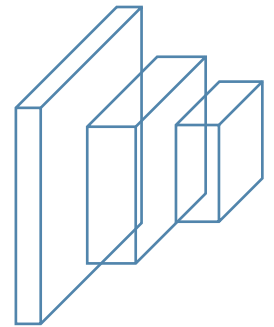


Epoch 2

⋮



Epoch N

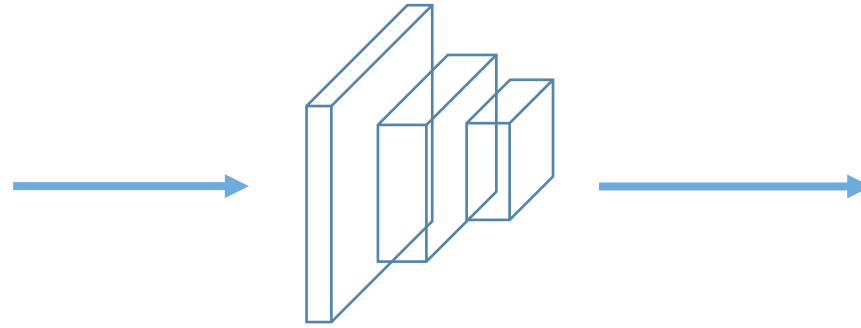


CNN

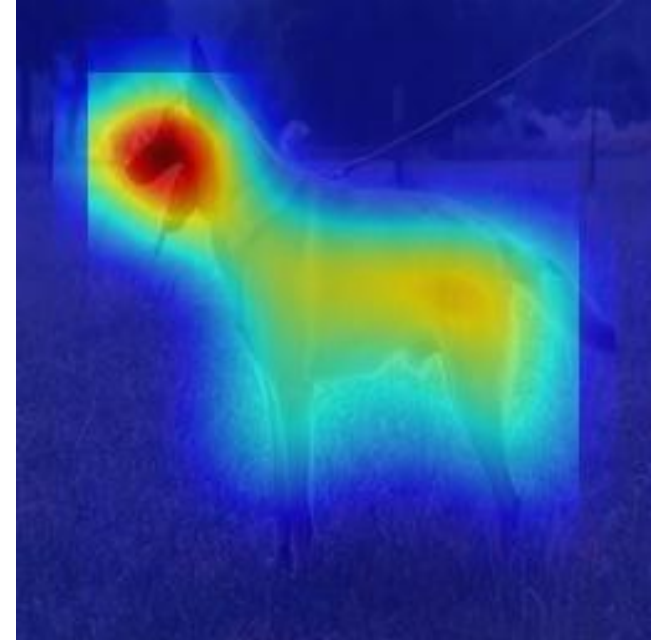
During testing feed full image into trained network



Test image



Trained CNN



Class Activation Map (CAM)

[Zhou et al. 2016]

Predicted label: 'dog'

Setting the hidden pixel values



Training



Testing

Need to assign mean RGB value (μ) to hidden pixels to ensure same filter activations in expectation during training and testing:

$$\mathbb{E}\left[\sum_{i=1}^{k \times k} \mathbf{w}_i^\top \mathbf{x}_i\right] = \sum_{i=1}^{k \times k} \mathbf{w}_i^\top \mu$$

Filter weights *RGB pixel value*

Experiments



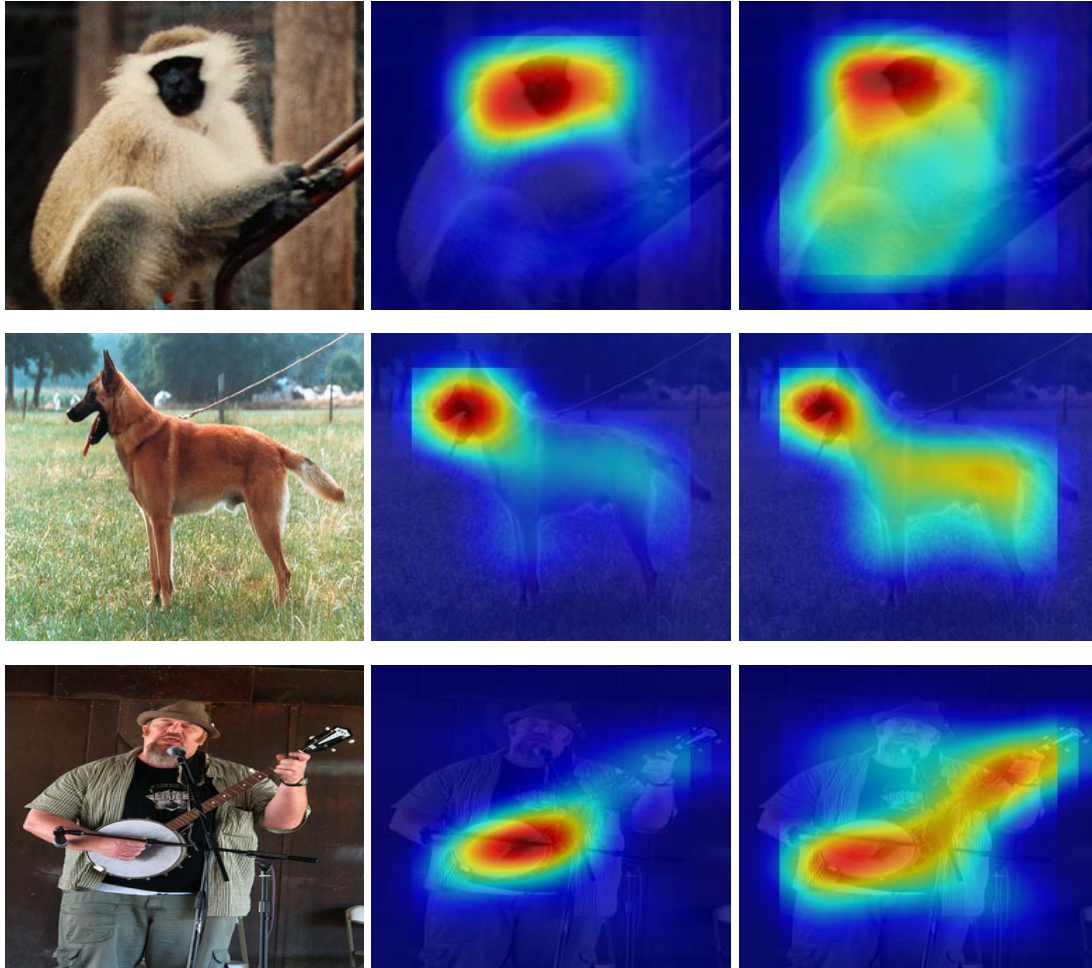
- ILSVRC 2016 for object localization
- 1000 categories
- 1.2 million training images, 50 thousand validation and test images

Hide-and-Seek localizes objects more fully

AlexNet-GAP

[Zhou et al.'16]

Ours



- Improvement of **27.66 to 30.04** pixel localization average precision (AP) for ResNet-50
- Generalizes across networks (AlexNet, GoogLeNet, ResNet)

Hide-and-Seek as *data augmentation*



Original Image



Horizontal Flip



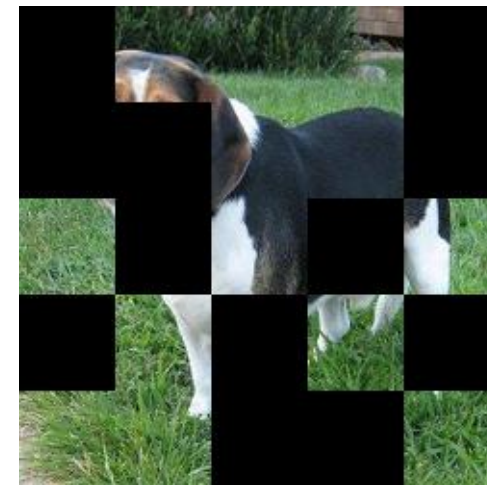
Color Jitter



Random Rotation



Random Crop



Hide-and-Seek

Hide-and-Seek as *data augmentation*

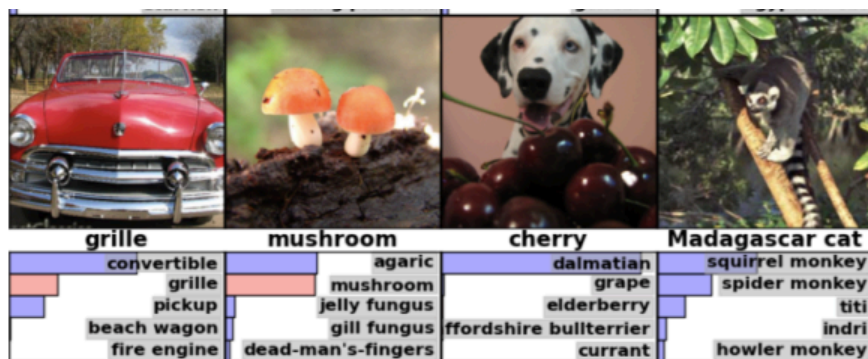


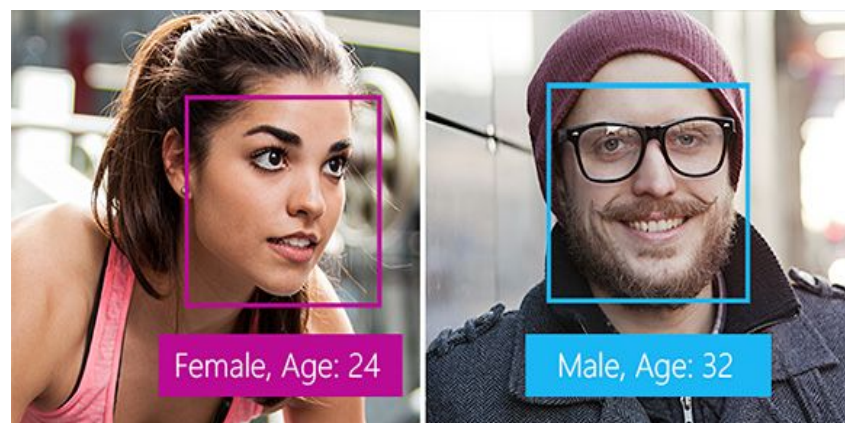
Image classification (76.1→77.2) +1.1%

ResNet-50 [He et al. 2015], ImageNet



Semantic segmentation (48.0→49.3) +1.3%

AlexNet FCN [Long et al. 2015], PASCAL 2011



Emotion/Age recognition (93.6→94.8) +1.2%

Custom network of [Khorrami et al. 2015], Cohn-Kanade+



Person reidentification (78.3→79.9) +1.6%

IDE+CamStyle [Zhong et al. 2018], DukeMTMC-reID

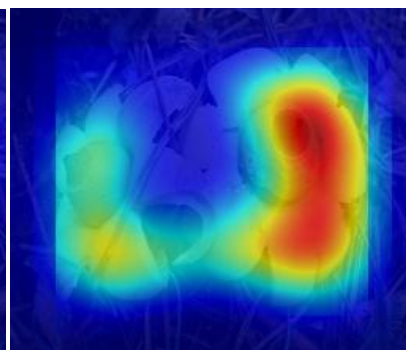
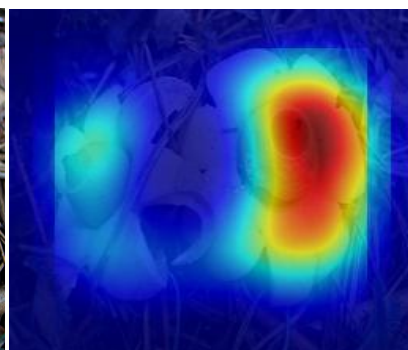
Independent, re-discoveries: Zhong et al. arXiv 2017, *Random erasing*; Ghiasi et al. NeurIPS 2018, *DropBlock*; ...

Limitations

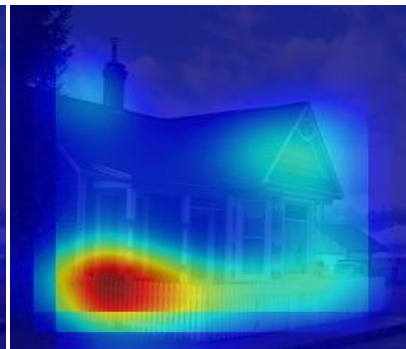
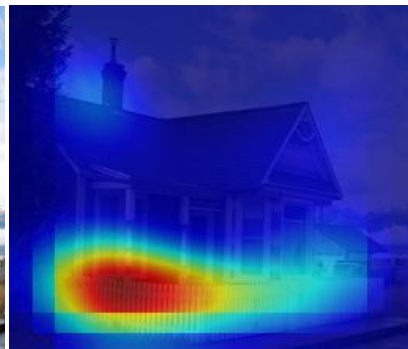
AlexNet-GAP

[Zhou et al.'16]

Ours

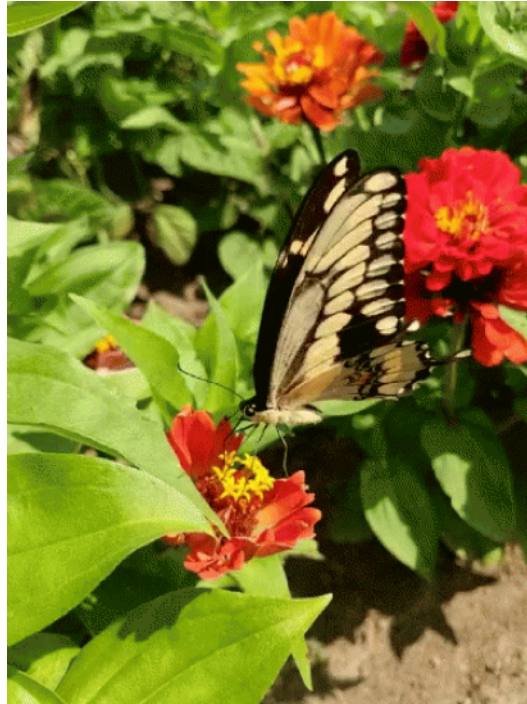


Merging spatially-close instances together



Localizing co-occurring context

Our visual world is dynamic

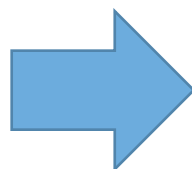


- Motion facilitates visual categorization and segmentation
- Video provides motion and temporal cues *for free!*

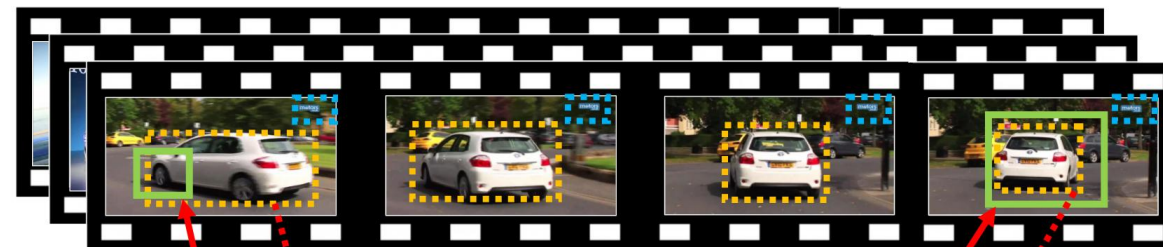
Our visual world is dynamic



Unsupervised Foreground Object Segmentation in Video



Videos tagged with "car"



Images tagged with "car"

match retrieve



...

match retrieve



Weakly-supervised object detection
[Singh, Xiao, Lee CVPR'17; Singh & Lee CVPR'19]

- Improvement of 5.0 AP on PASCAL '07 & '12 object detection for state-of-the-art weakly-supervised methods [Bilen '17, Tang '17]

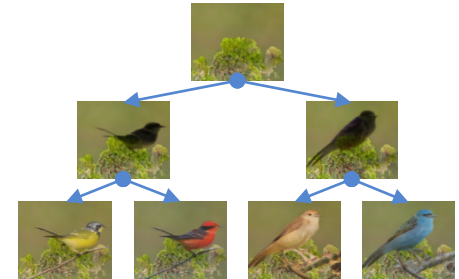
Outline

- Visual scene understanding with *minimal human supervision*

- *Localize* objects with only image-level tag annotations?



- *Generate* fine-grained object details without fine-grained annotations?

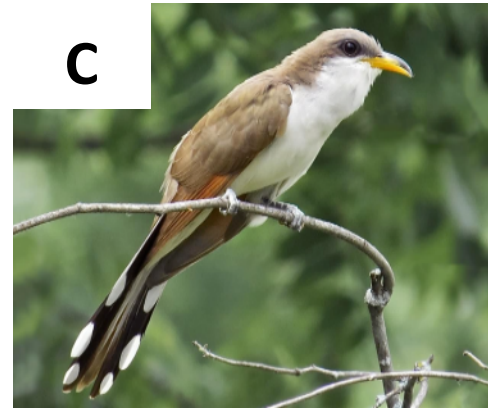
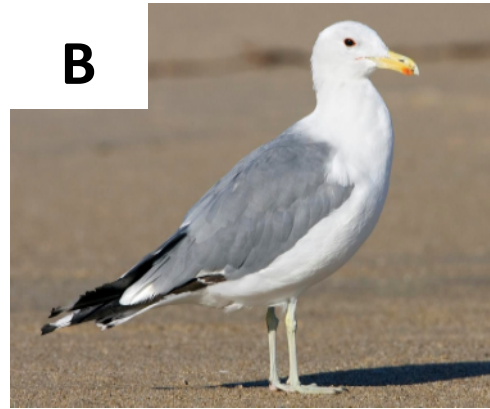


- Towards visual scene understanding in *dynamic environments*

- *Segment* object instances in real-time?



Task: Which birds belong to the same species?

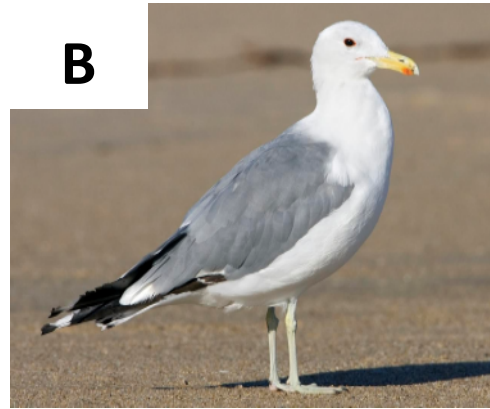


Easy to tell that **A** and **B** shouldn't be grouped with **C** and **D**
... *but how about C and D?*

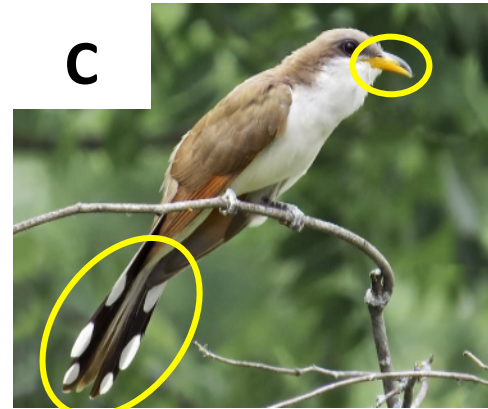
Task: Which birds belong to the same species?



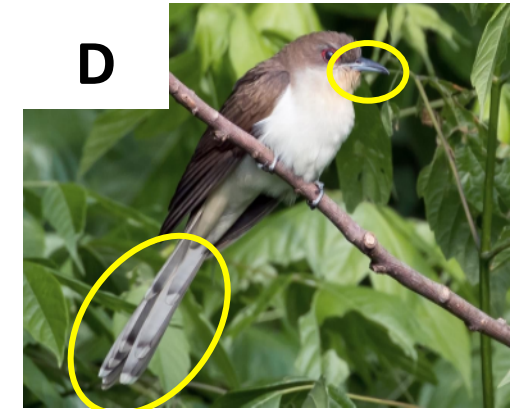
Barrow's Goldeneye



California Gull



Yellow-billed Cuckoo



Black-billed Cuckoo

Easy to tell that **A** and **B** shouldn't be grouped with **C** and **D**
... *but how about C and D?*

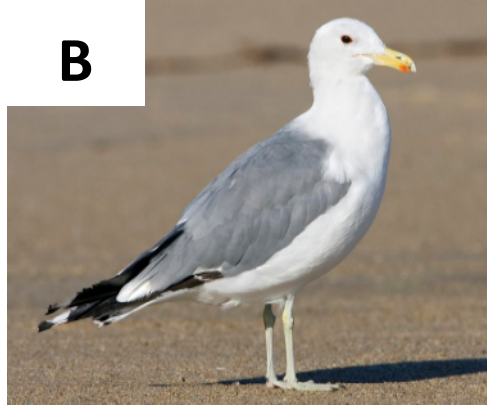
All birds belong to different fine-grained categories

What did we learn?

A



B



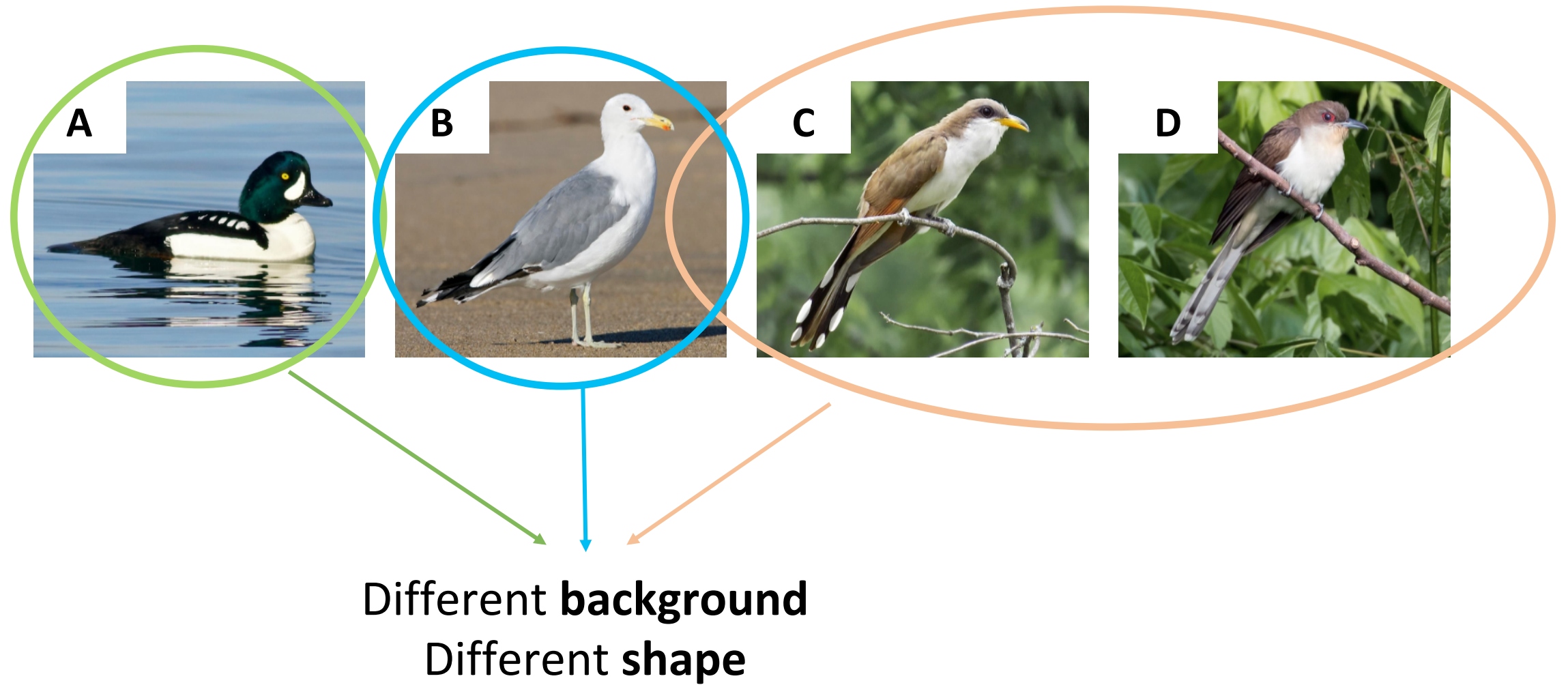
C



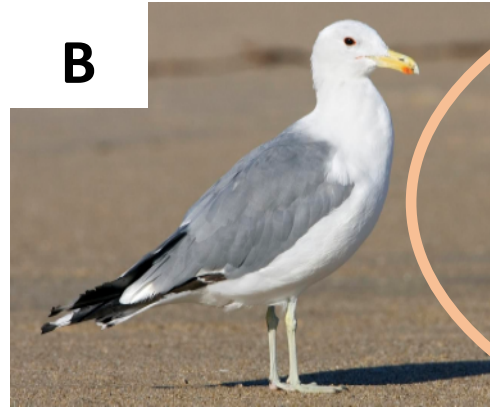
D



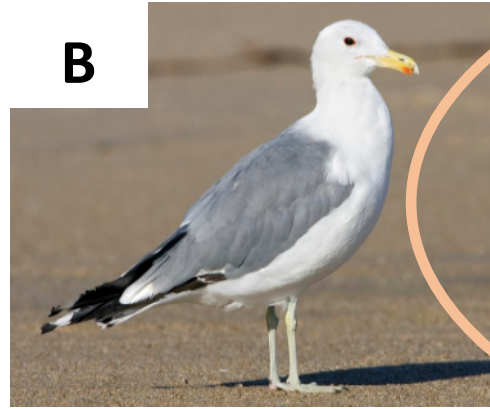
What did we learn? Multiple factors of variation



What did we learn? Existence of a natural hierarchy



What did we learn? Existence of a natural hierarchy



Same background
Same shape
Different color/texture

Goal: A generative model for fine-grained objects

- Generation requires a *deep understanding* of visual data
- Humans not only recognize patterns ...
- but can also *generate* new examples, *parse* an object into parts & relations, *combine* related concepts to generate new samples, etc.

[e.g., Lake et al. 2016, *Building Machines That Learn and Think Like People*]

Goal: A generative model for fine-grained objects

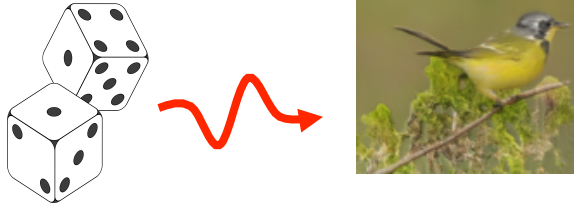
- **Disentangle** factors of variation (background, shape, appearance) *hierarchically* without:
 - (1) fine grained category labels
 - (2) part annotations or segmentation masks
 - (3) ground-truth hierarchy
- *Hypothesis*: Discovered representation will be useful for **unsupervised fine-grained clustering (“discovery”) of real images**

Disentangled representation learning: [Bengio et al. ‘14, Chen et al. ‘16, Yang et al. ‘17, Higgins et al. ‘17, Hu et al. ‘18 ...]
Unsupervised object category discovery: [Sivic et al. ‘05, Lee and Grauman ‘10, Xie et al. ‘16, Yang et al. ‘16, ...]

Unsupervised image generation

One-shot generation

Random noise

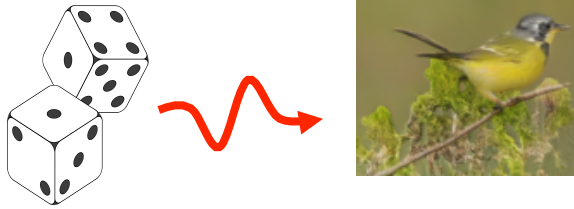


[Goodfellow et al. '14, Radford et al. '16,
Gulrajani et al. '17, ...]

Unsupervised image generation

One-shot generation

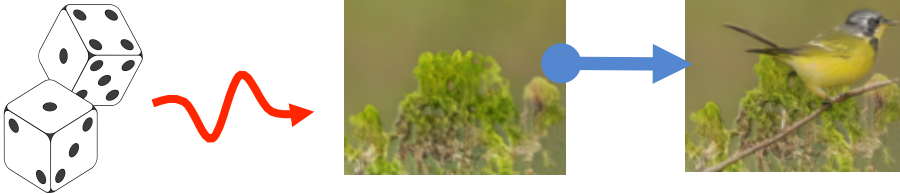
Random noise



[Goodfellow et al. '14, Radford et al. '16,
Gulrajani et al. '17, ...]

Stagewise generation

Random noise

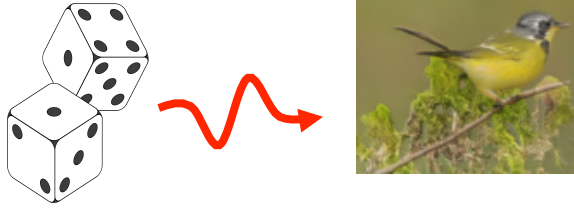


[Im et al. '16, Kwak and Zhang '16, Yang et al. '17, ...]

Unsupervised image generation

One-shot generation

Random noise



[Goodfellow et al. '14, Radford et al. '16, Gulrajani et al. '17, ...]

Stagewise generation

Random noise



[Im et al. '16, Kwak and Zhang '16, Yang et al. '17, ...]

Our idea: Hierarchical generation

Latent background code



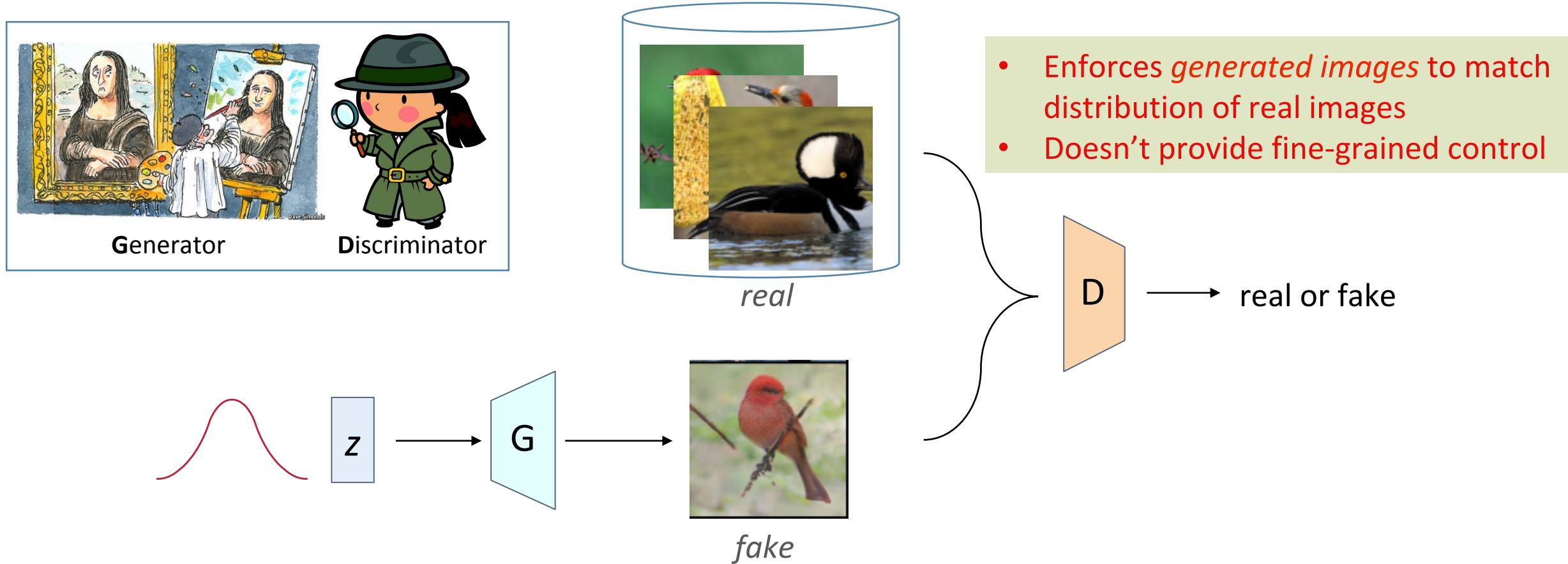
Latent parent code



Latent child code



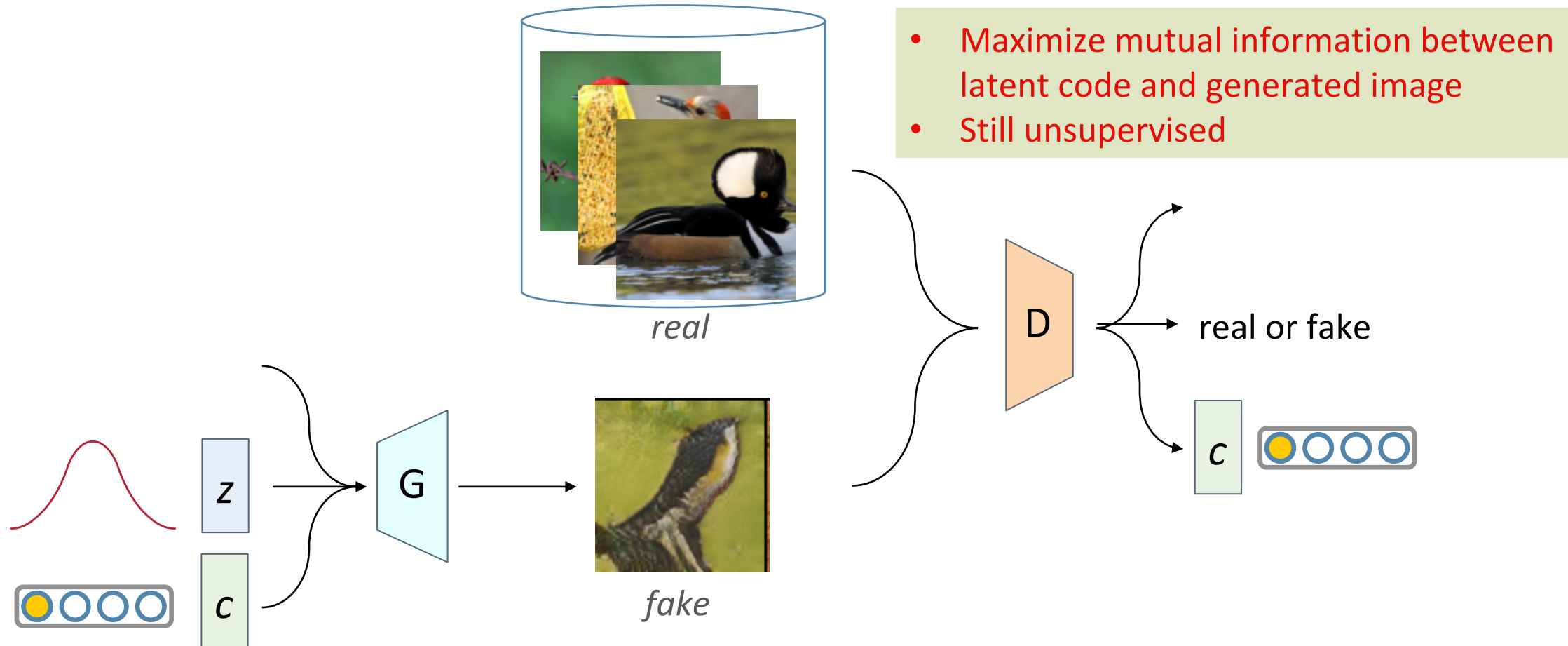
Generative Adversarial Networks (GANs)



$$\min_G \max_D V(D, G) = \mathbb{E}_{\mathbf{x} \sim p_{data}(\mathbf{x})} [\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}(\mathbf{z})} [\log(1 - D(G(\mathbf{z})))]$$

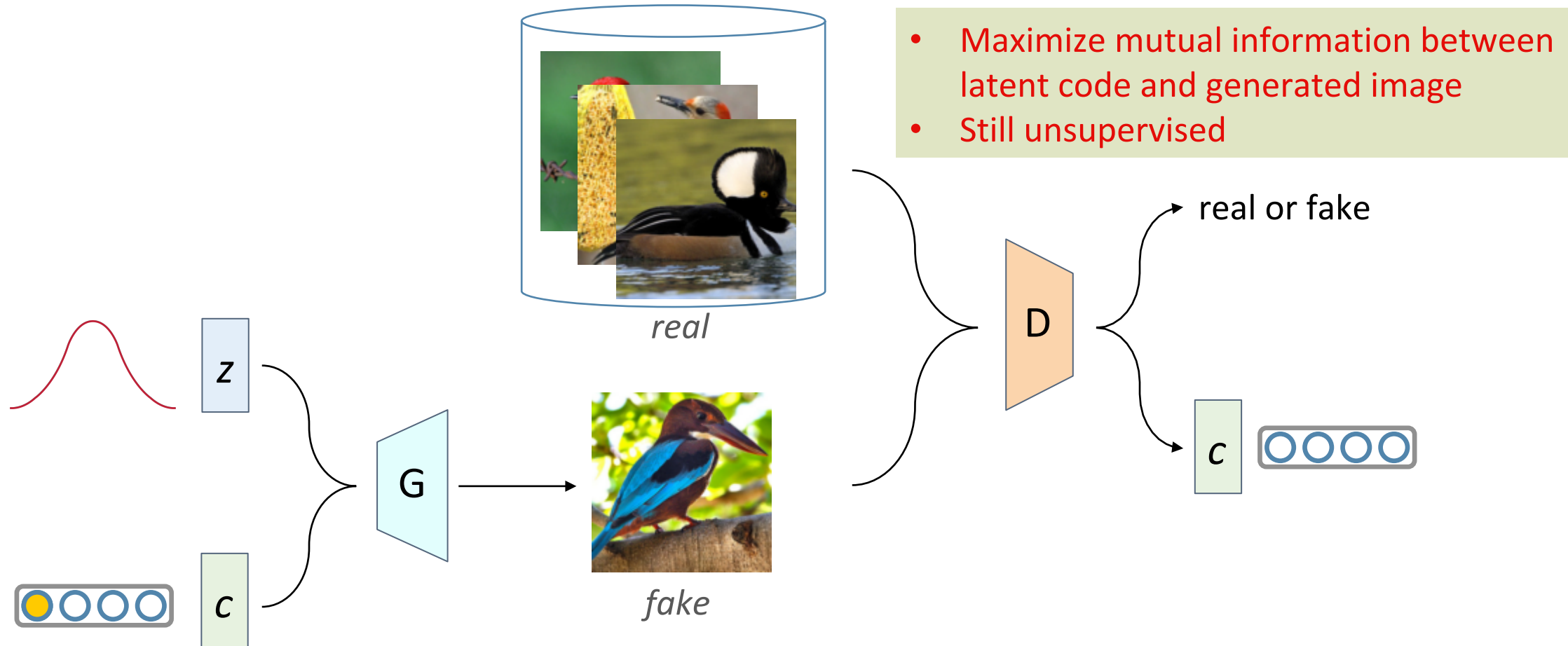
[I. Goodfellow et al., “Generative Adversarial Networks”, NIPS 2014]

Maximize Mutual Information (InfoGAN)



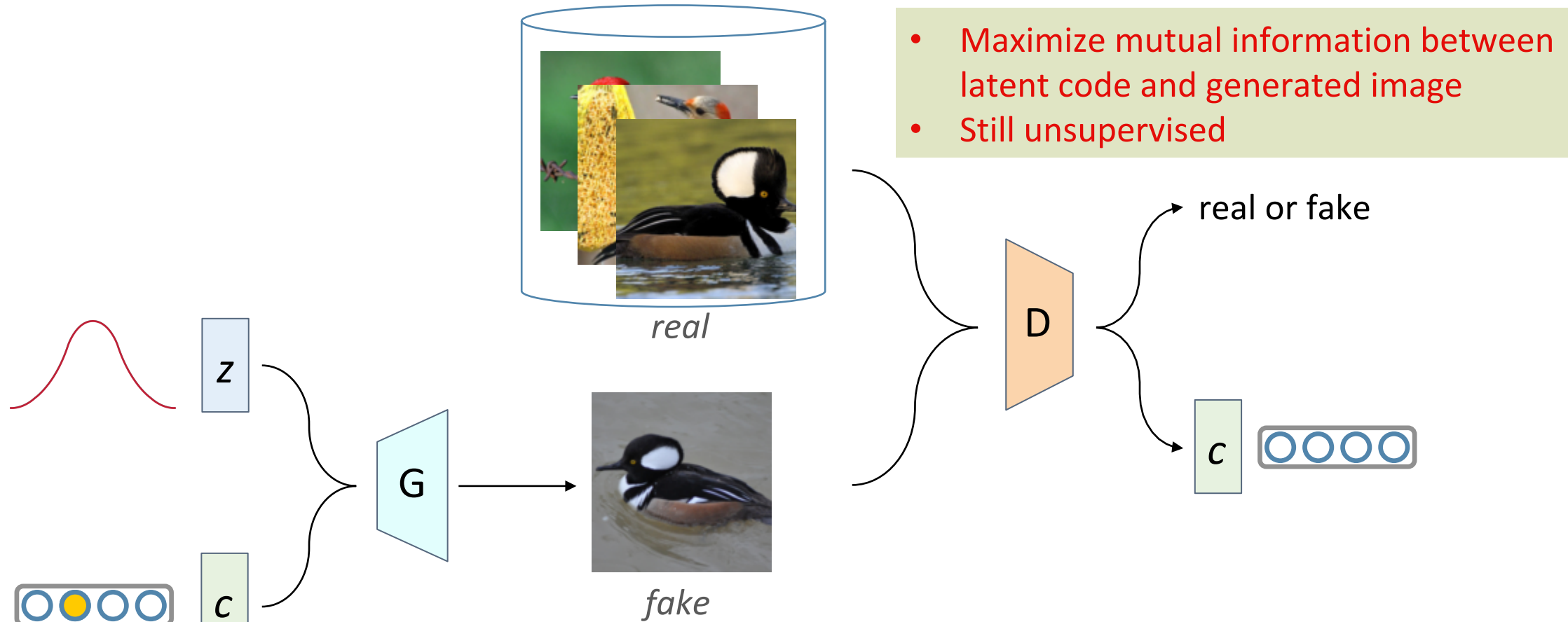
$$\min_G \max_D V(D, G) - \lambda \cdot I(\mathbf{c}, G(\mathbf{z}, \mathbf{c}))$$
$$I(\mathbf{c}, G(\mathbf{z}, \mathbf{c})) = \mathbb{E}_{\mathbf{x} \sim p_{\mathbf{c}}(\mathbf{c}), \mathbf{x} \sim G(\mathbf{z}, \mathbf{c})} [\log Q(\mathbf{c}|\mathbf{x})] + H(\mathbf{c})$$

Maximize Mutual Information (InfoGAN)



$$\min_G \max_D V(D, G) - \lambda \cdot I(\mathbf{c}, G(\mathbf{z}, \mathbf{c}))$$
$$I(\mathbf{c}, G(\mathbf{z}, \mathbf{c})) = \mathbb{E}_{\mathbf{x} \sim p_{\mathbf{c}}(\mathbf{c}), \mathbf{x} \sim G(\mathbf{z}, \mathbf{c})} [\log Q(\mathbf{c}|\mathbf{x})] + H(\mathbf{c})$$

Maximize Mutual Information (InfoGAN)




$$\min_G \max_D V(D, G) - \lambda \cdot I(\mathbf{c}, G(\mathbf{z}, \mathbf{c}))$$
$$I(\mathbf{c}, G(\mathbf{z}, \mathbf{c})) = \mathbb{E}_{\mathbf{x} \sim p_{\mathbf{c}}(\mathbf{c}), \mathbf{x} \sim G(\mathbf{z}, \mathbf{c})} [\log Q(\mathbf{c}|\mathbf{x})] + H(\mathbf{c})$$

Maximize Mutual Information (InfoGAN)

Code 1

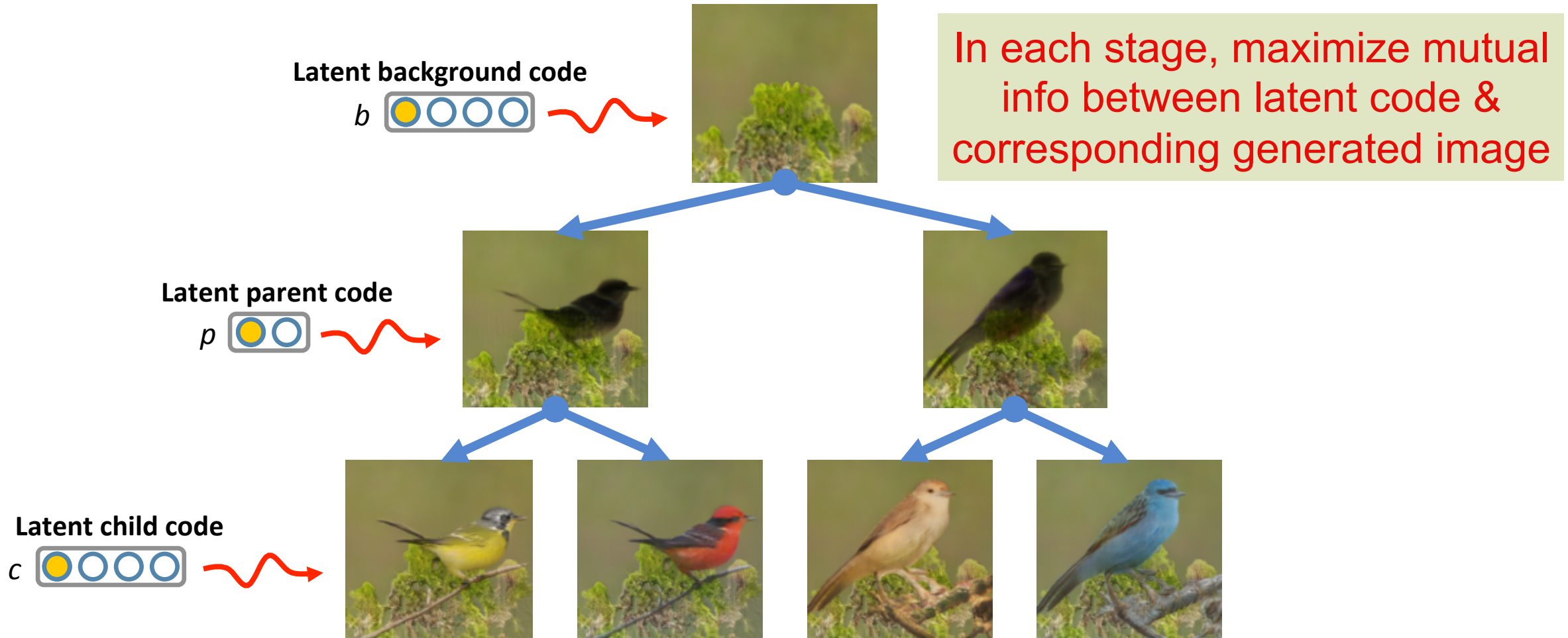



Code 2


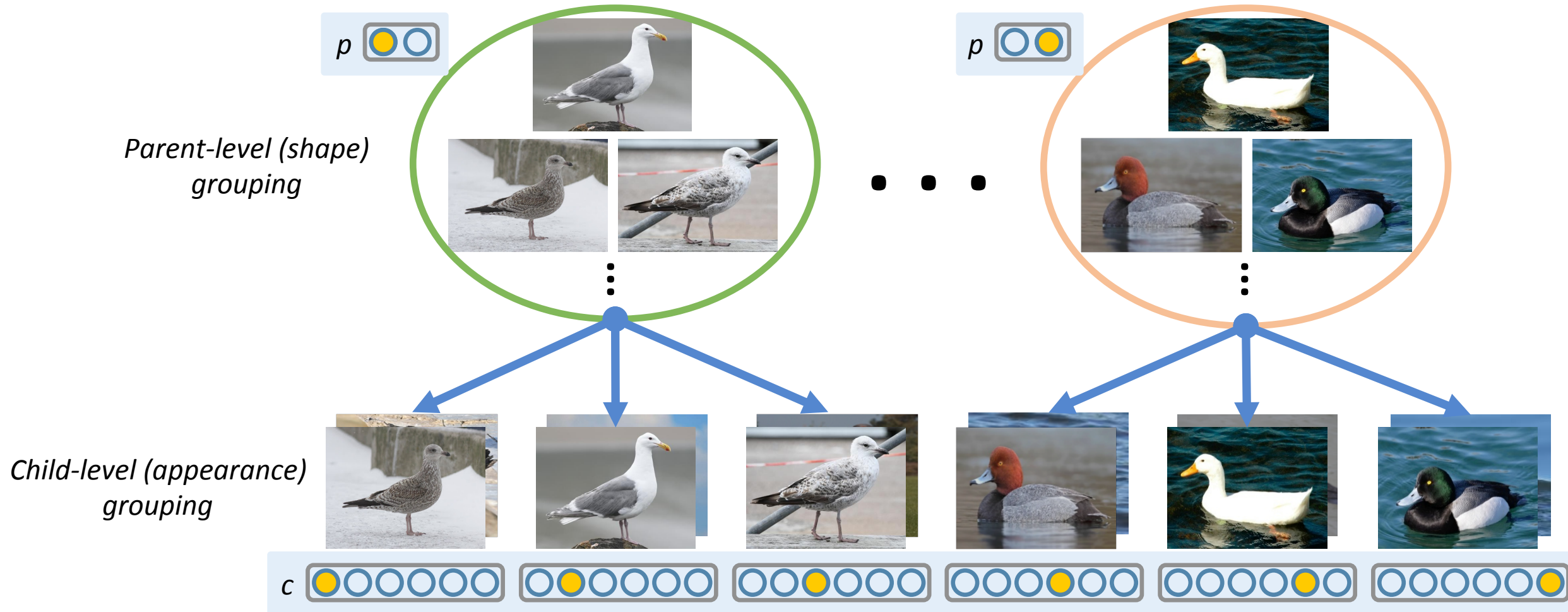


No fine-grained disentanglement of background, shape, appearance

Our Idea: Hierarchical, stagewise generation

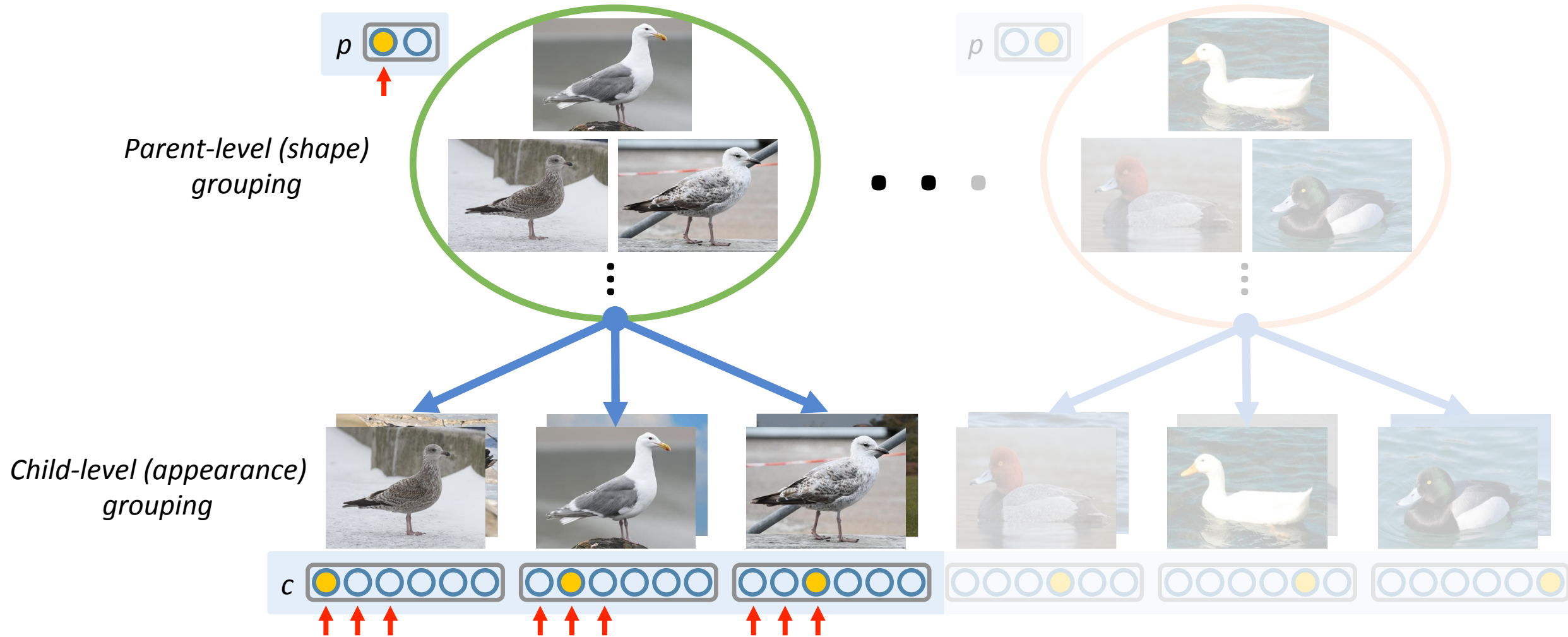


Fine-grained categories can be organized hierarchically



- # of parent (shape) codes \ll # of child (appearance) codes
- *a fixed group of children* share same parent code

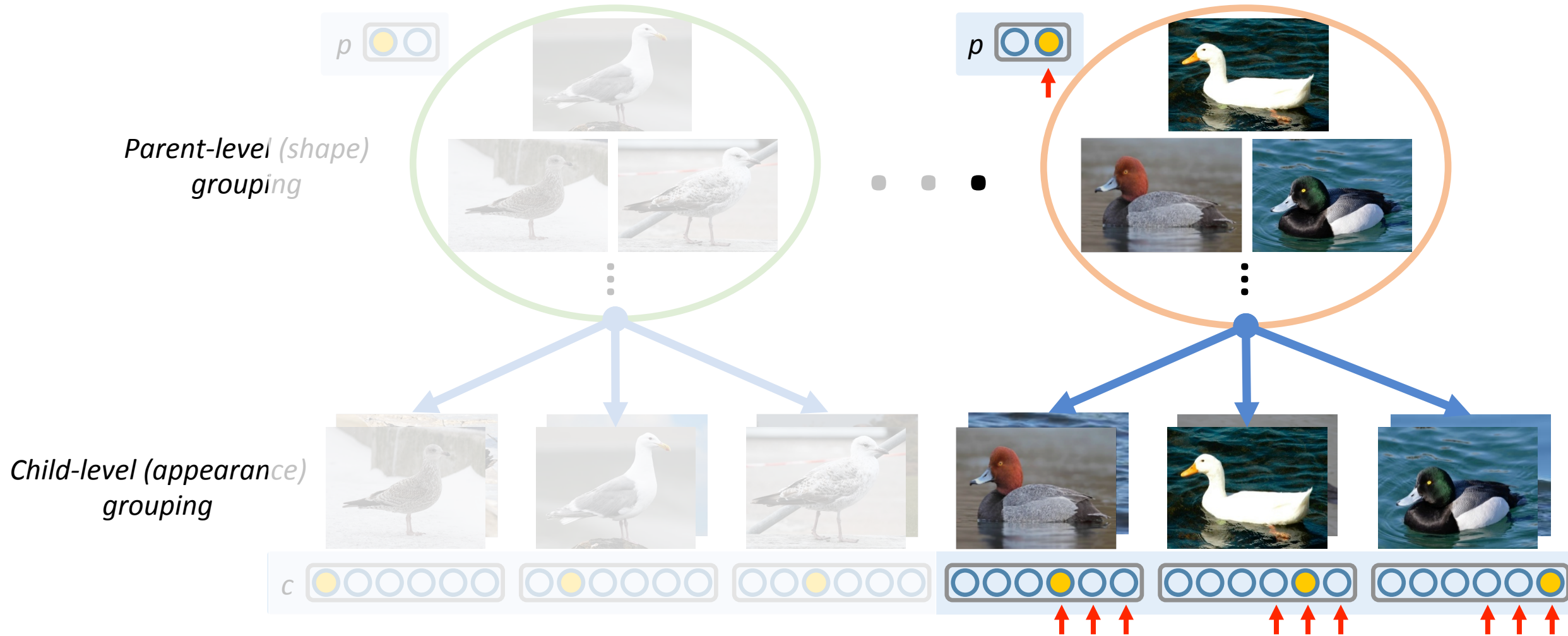
Fine-grained categories can be organized hierarchically



→ # of parent (shape) codes \ll # of child (appearance) codes

→ *a fixed group of children* share same parent code

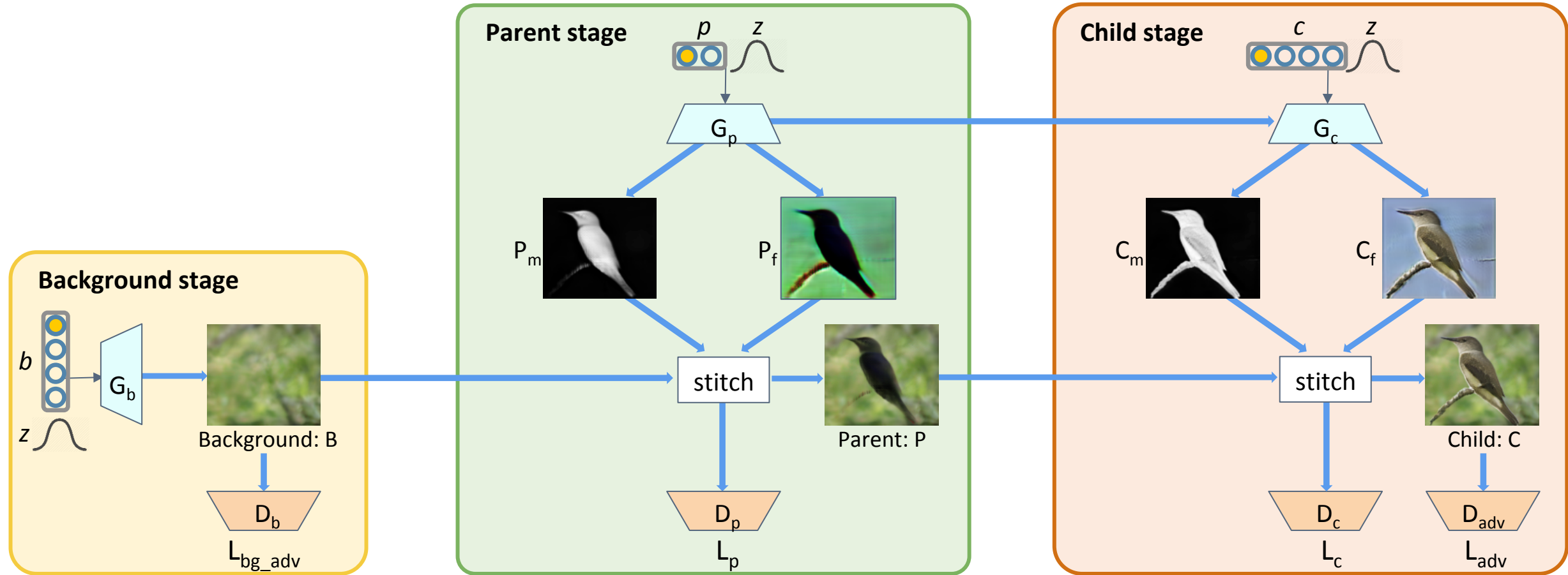
Fine-grained categories can be organized hierarchically



→ # of parent (shape) codes \ll # of child (appearance) codes

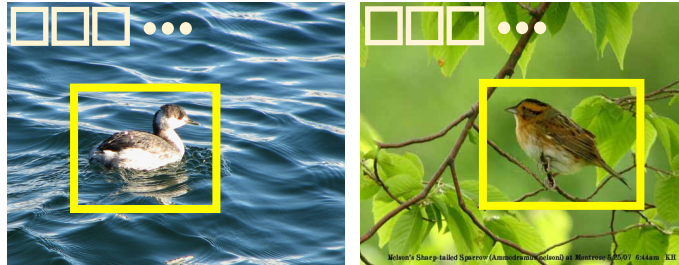
→ a fixed group of children share same parent code

FineGAN: Hierarchical, stagewise generation

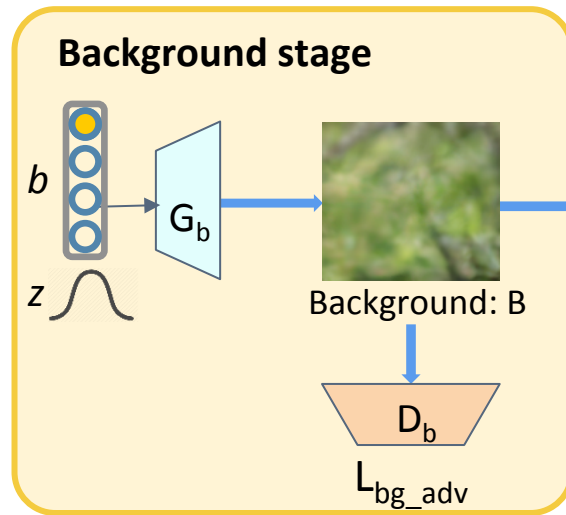


All stages trained end-to-end *without mask, fine-grained labels*

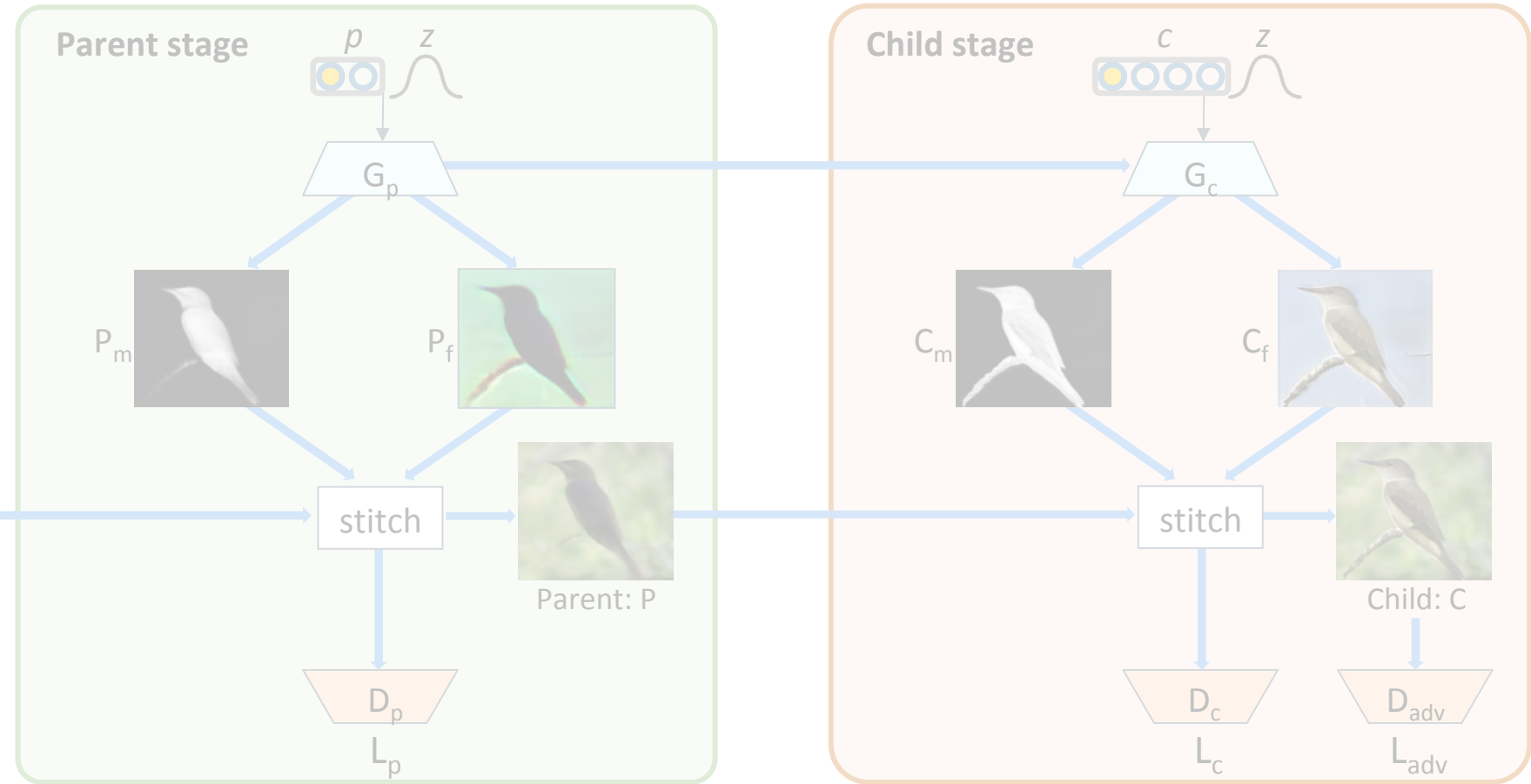
FineGAN: Hierarchical, stagewise generation



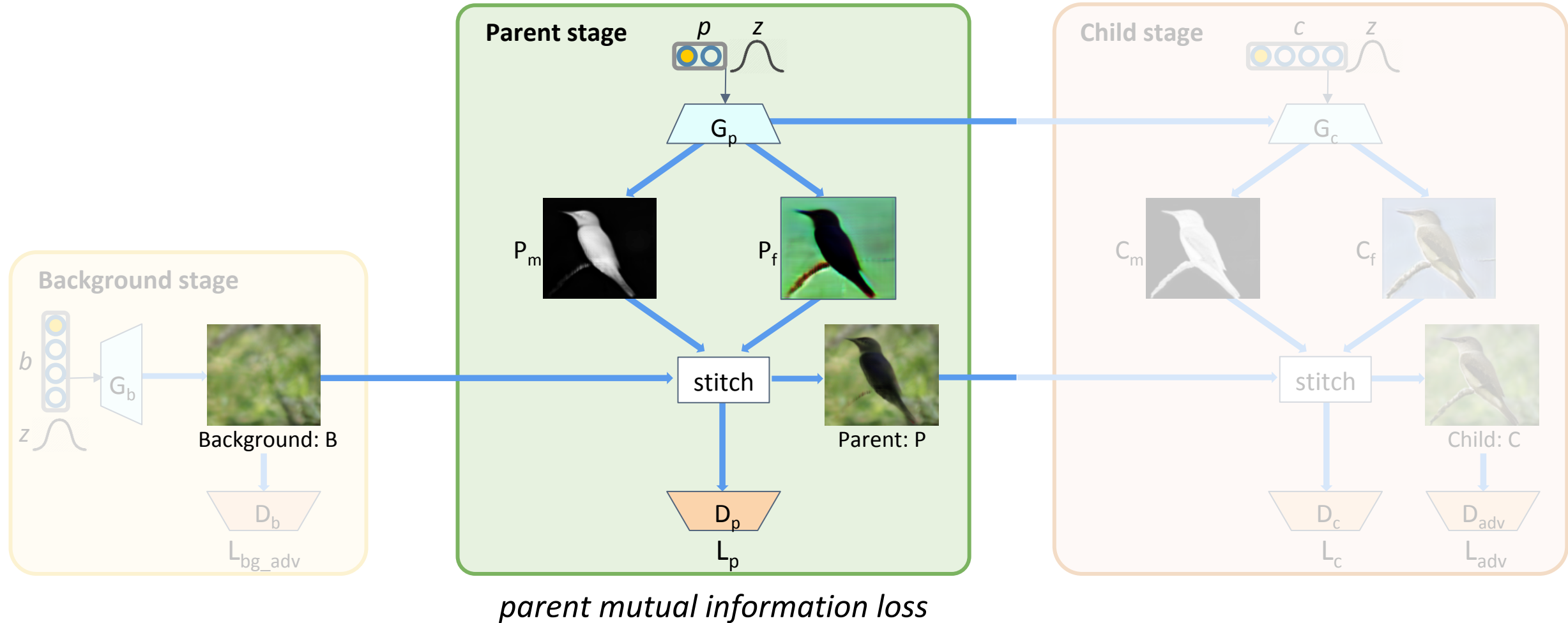
Real images with loose bounding box



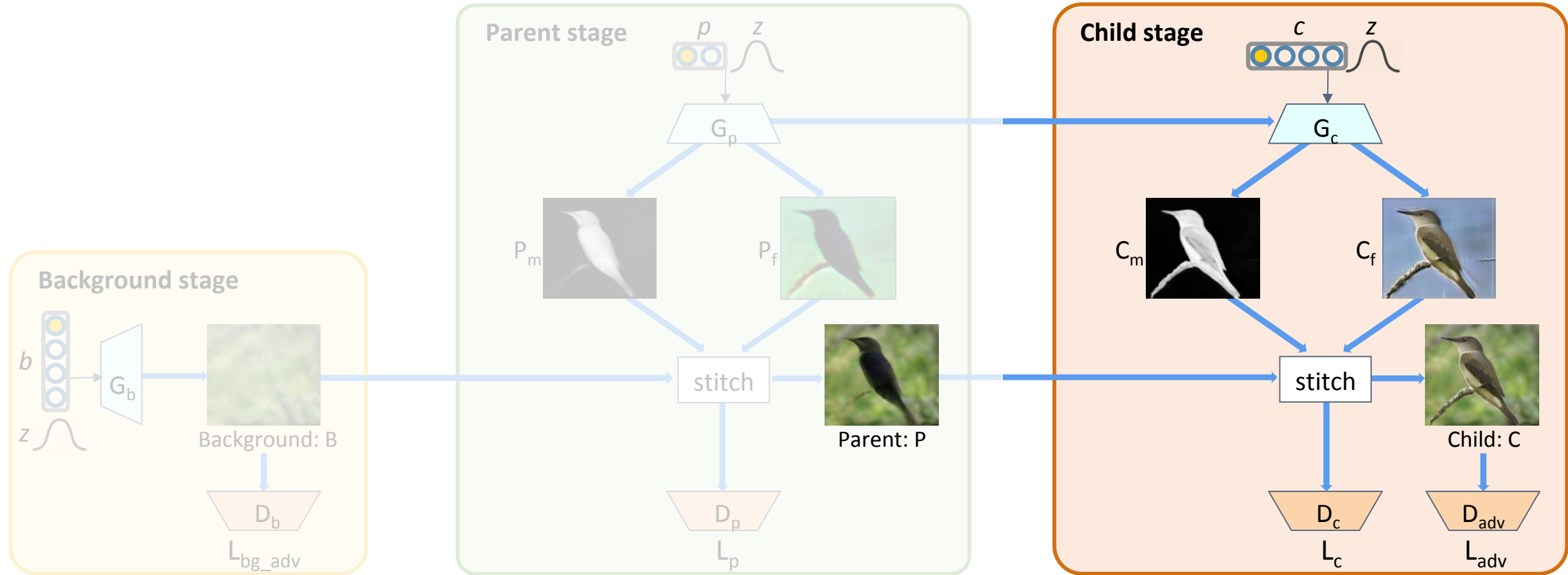
background adversarial loss



FineGAN: Hierarchical, stagewise generation



FineGAN: Hierarchical, stagewise generation

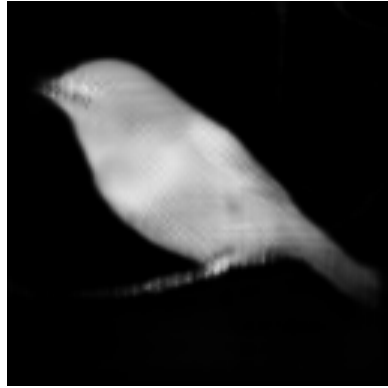


child mutual information loss
final image adversarial loss

FineGAN's stagewise image generation



Background



Parent Mask



Parent Image



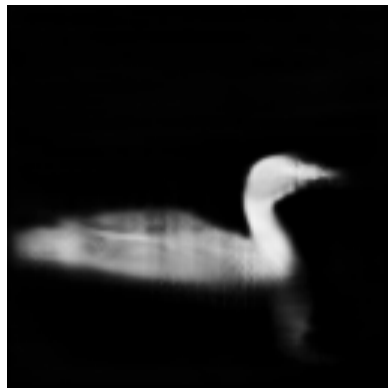
Child Mask



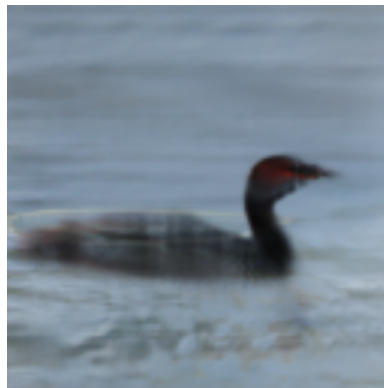
Child Image



Background



Parent Mask



Parent Image



Child Mask



Child Image

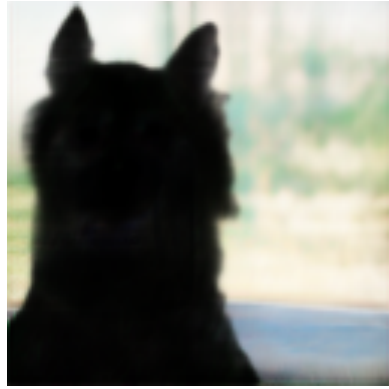
FineGAN's stagewise image generation



Background



Parent Mask



Parent Image



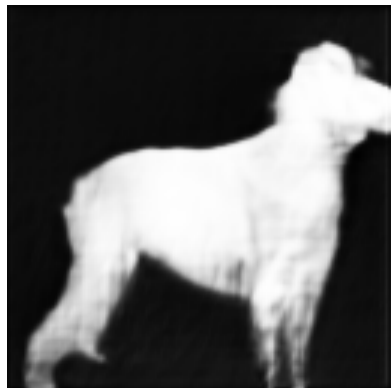
Child Mask



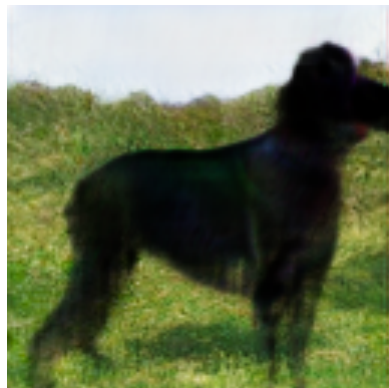
Child Image



Background



Parent Mask



Parent Image



Child Mask

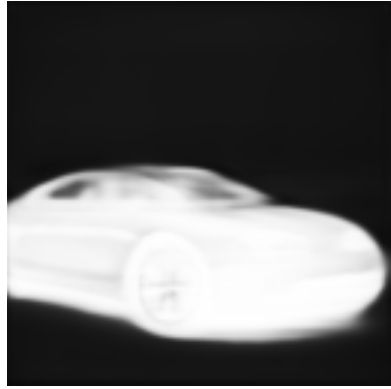


Child Image

FineGAN's stagewise image generation



Background



Parent Mask



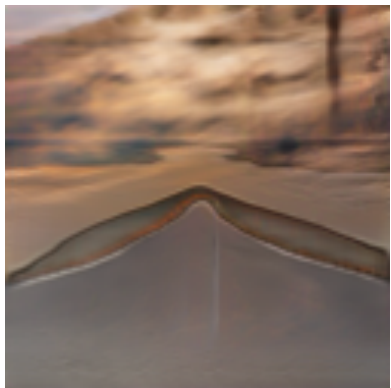
Parent Image



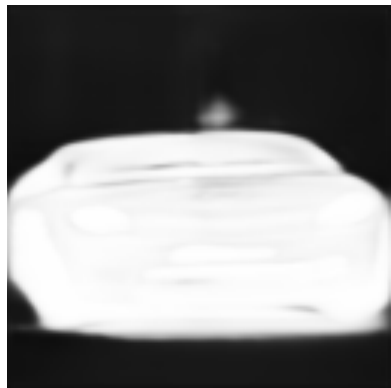
Child Mask



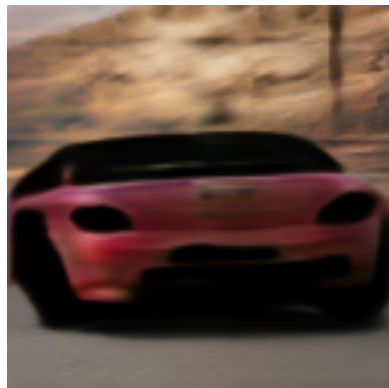
Child Image



Background



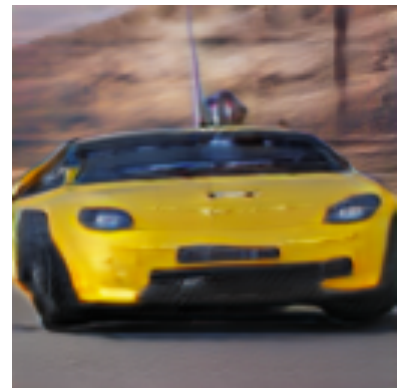
Parent Mask



Parent Image



Child Mask



Child Image

FineGAN's hierarchical disentanglement and grouping

Parent 1

c1



c2



c3



Parent 3

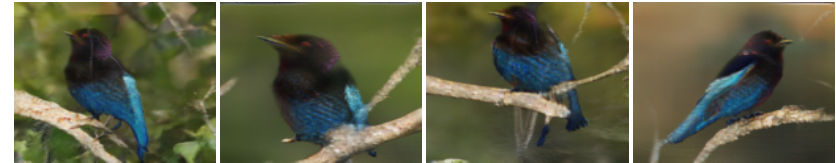
c7



c8



c9



Parent 2

c4



c5

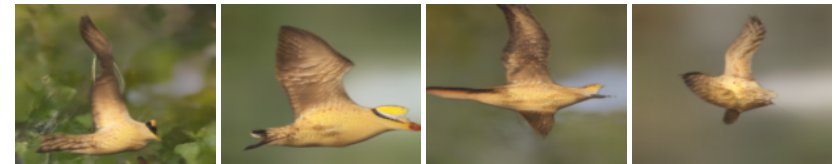


c6



Parent 4

c10



c11



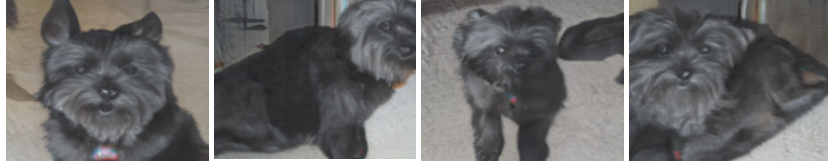
c12



FineGAN's hierarchical disentanglement and grouping

Parent 1

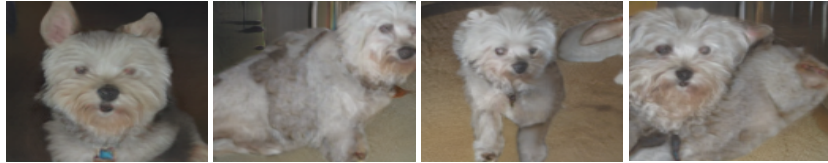
c1



c2



c3



Parent 3

c7



c8



c9



Parent 2

c4



c5



c6



Parent 4

c10



c11



c12



FineGAN's hierarchical disentanglement and grouping

Parent 1

c1



c2

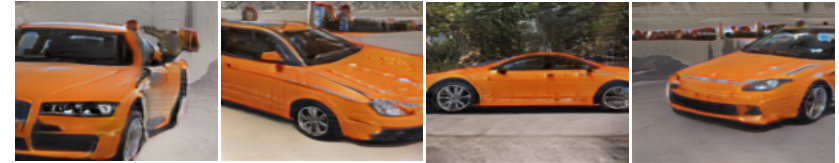


c3

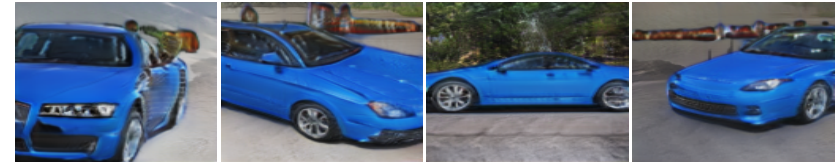


Parent 3

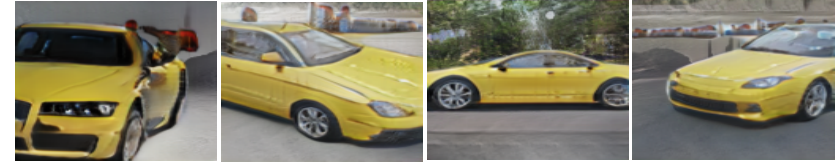
c7



c8



c9



Parent 2

c4



c5



c6



Parent 4

c10



c11



c12



Disentanglement of shape and appearance



FineGAN: Unsupervised Hierarchical Disentanglement for Fine-Grained Object Generation and Discovery

Krishna Kumar Singh*, Utkarsh Ojha*, and Yong Jae Lee

UC Davis

* equal contribution

How well does FineGAN model the distribution of fine-grained categories?

- Favorable Inception scores, Fréchet Inception Distance compared to state-of-the-art unconditional image generators

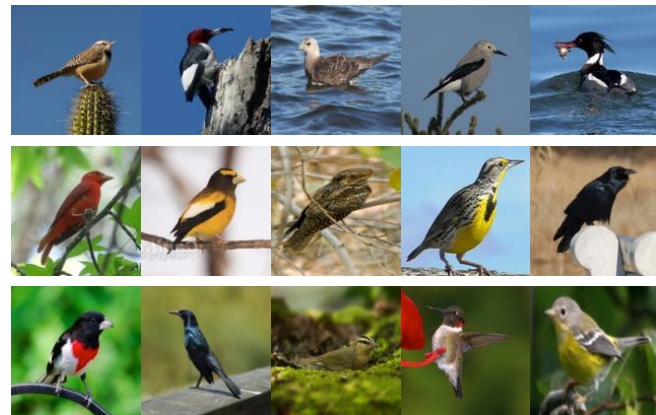
| | Fréchet Inception Distance | | |
|------------------------|----------------------------|--------------|--------------|
| | <i>Birds</i> | <i>Dogs</i> | <i>Cars</i> |
| InfoGAN [Chen '16] | 13.20 | 29.34 | 17.63 |
| LR-GAN [Yang '17] | 34.91 | 54.91 | 88.80 |
| StackGANv2 [Zhang '18] | 13.60 | 31.39 | 16.28 |
| Ours | 11.25 | 25.66 | 16.03 |

How useful is the learned representation?

- **Fine-grained real image clustering:**
Significant improvement over state-of-the-art deep clustering methods

| Clustering Accuracy (NMI) | | | |
|---------------------------|--------------|--------------|--------------|
| | <i>Birds</i> | <i>Dogs</i> | <i>Cars</i> |
| JULE [Yang '16] | 0.203 | 0.148 | 0.237 |
| DEPICT [Xie '16] | 0.297 | 0.183 | 0.329 |
| Ours | 0.403 | 0.233 | 0.354 |

JULE



DEPICT



Ours



Discussion

- **Limitations**

- # of parents, children are hyperparameters
 - Discovered latent modes of variation may not correspond to those annotated by a human
 - Still far behind fully-supervised fine-grained recognition accuracy
- Important initial step in tackling challenging problem of *unsupervised fine-grained object modeling*

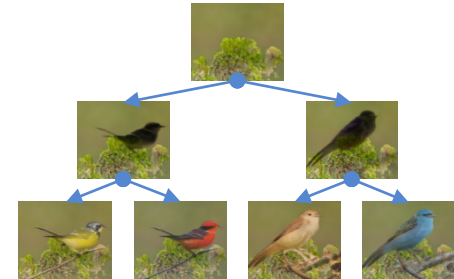
Outline

- Visual scene understanding with *minimal human supervision*

- *Localize* objects with only image-level tag annotations?



- *Generate* fine-grained object details without fine-grained annotations?



- Towards visual scene understanding in *dynamic environments*

- *Segment* object instances in real-time?



Real-time Instance Segmentation



Input Image



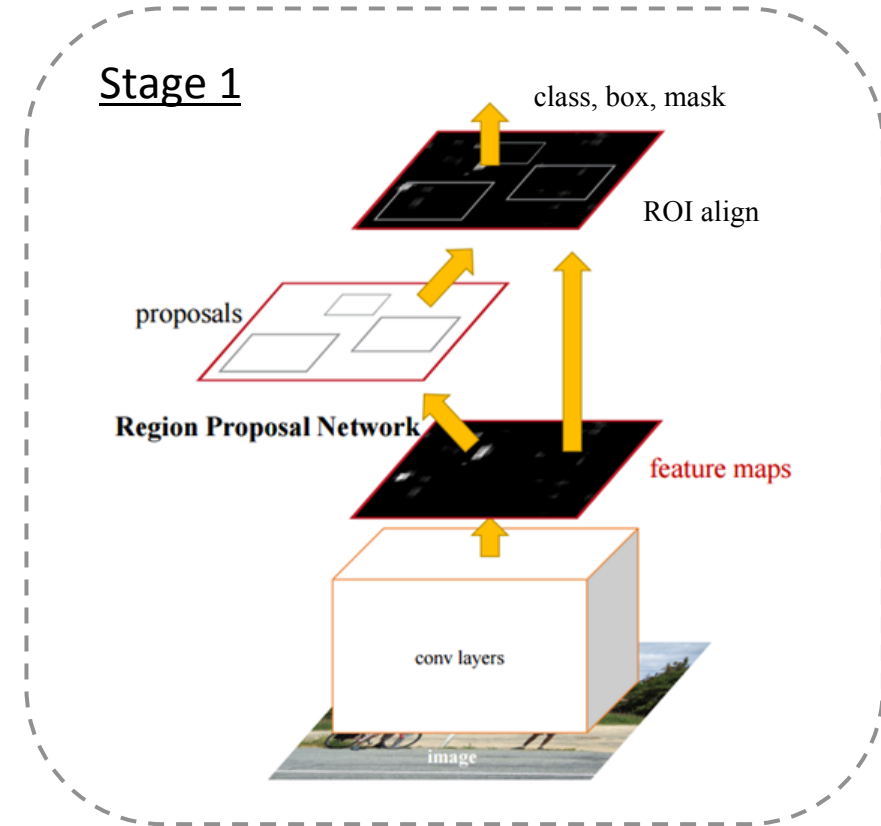
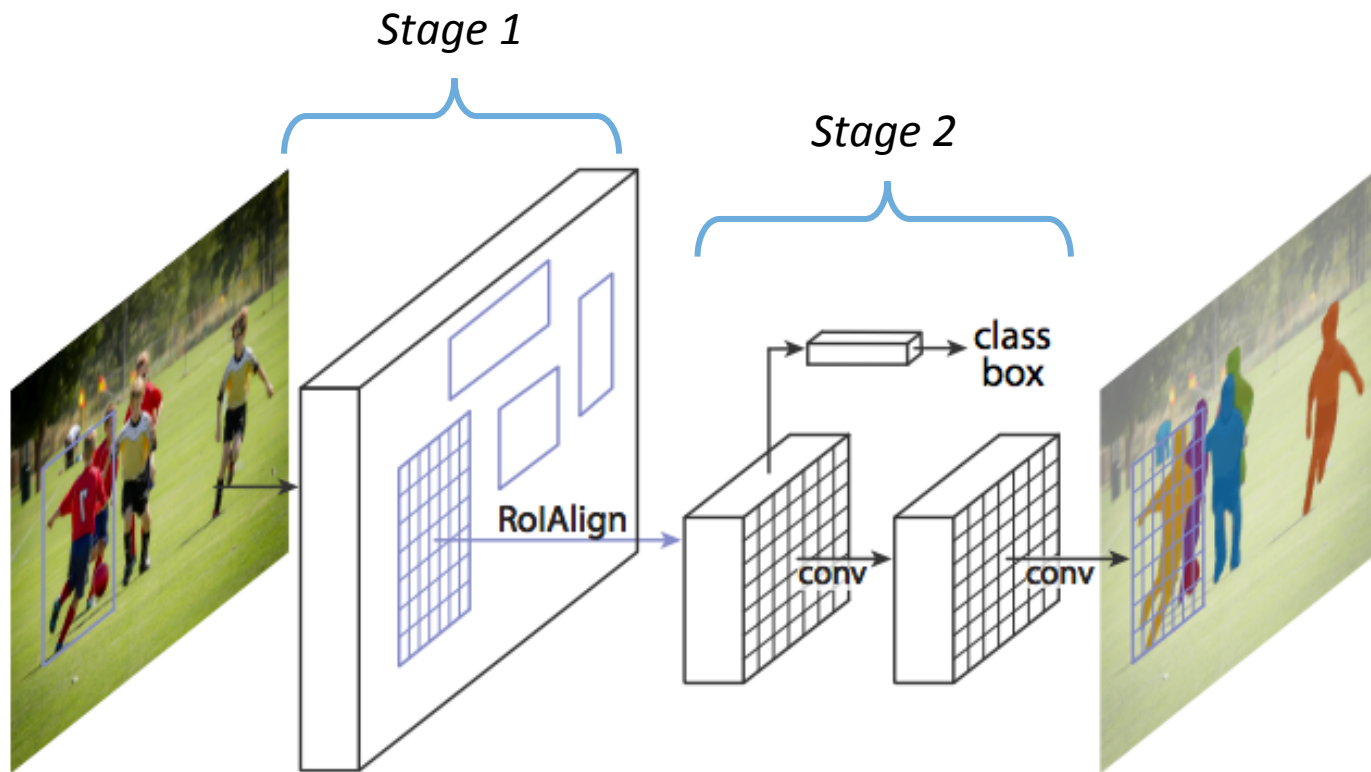
Semantic Segmentation



Instance Segmentation

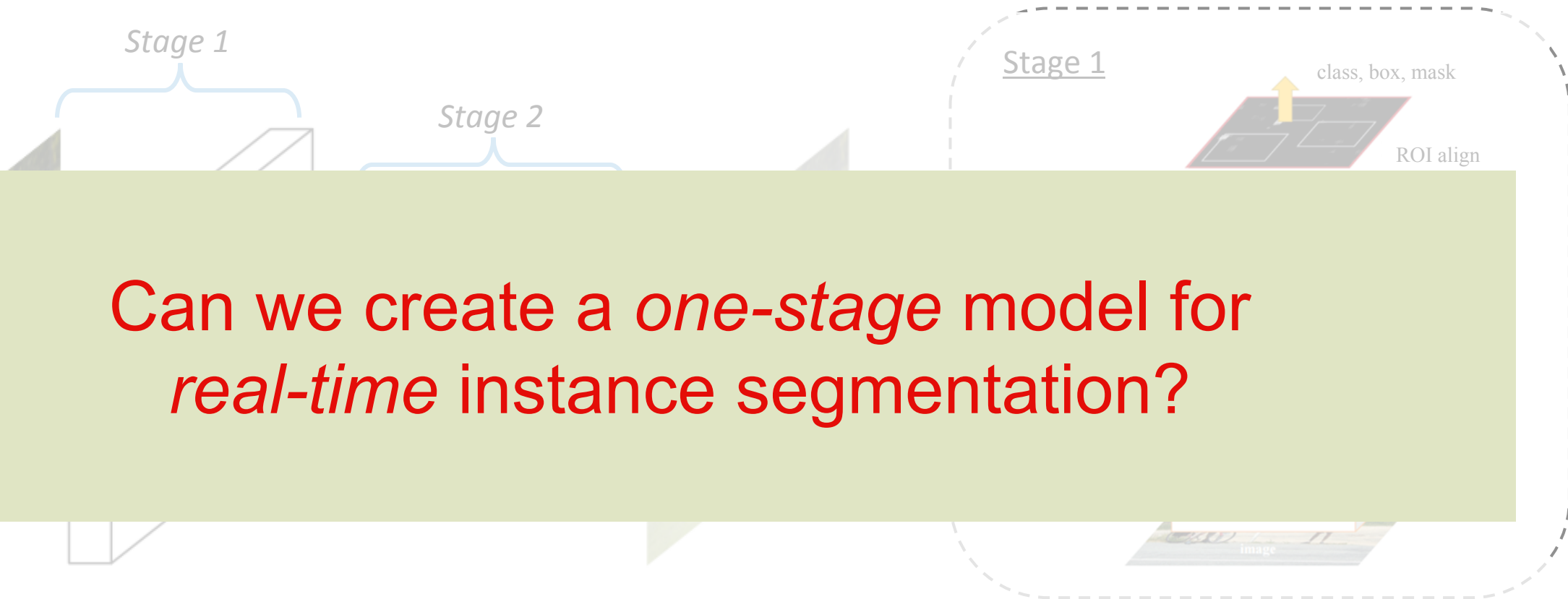
- So far, no robust *real-time* ($>30\text{ fps}$) algorithm exists
- **You Only Look At CoefficienTs** [Bolya, Zhou, Xiao, Lee, ICCV 2019]

Mask R-CNN: Accurate but not fast enough (<10 fps)



- *Stage 1*: use Region Proposal Network to generate region proposals
- *Stage 2*: pool features for each proposal (via ROI-align) and classify

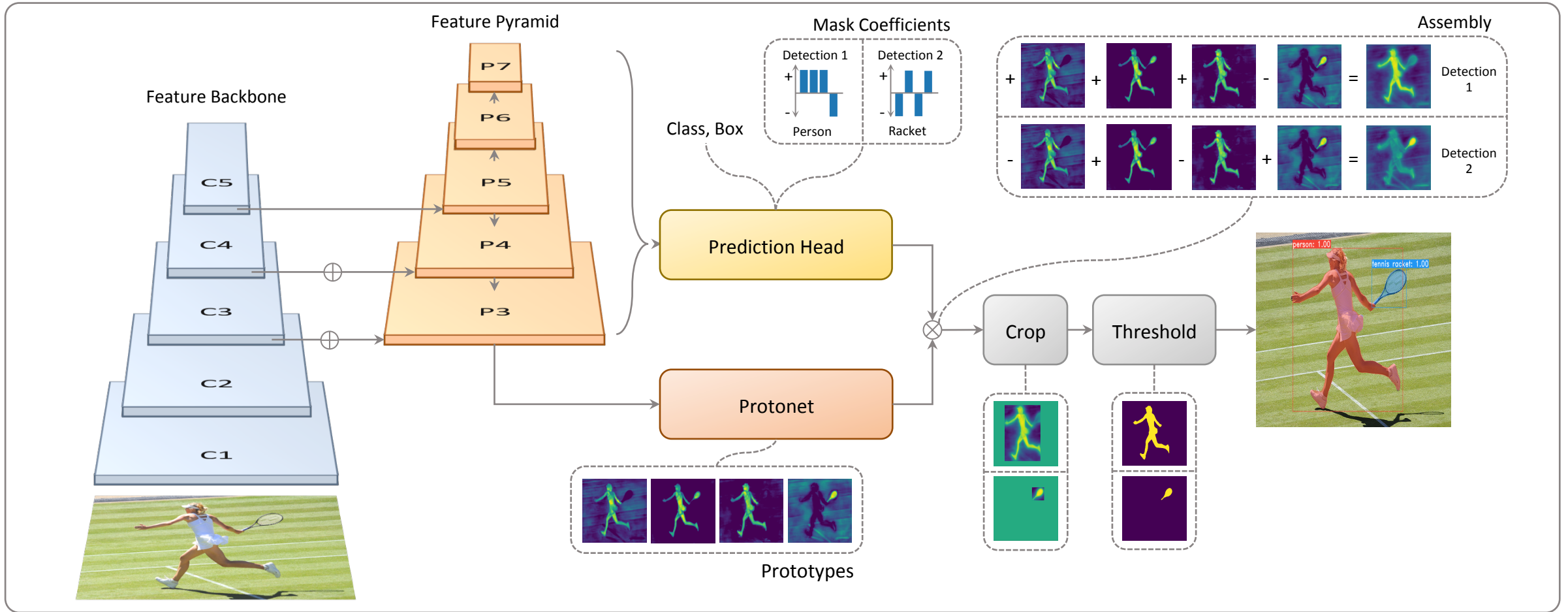
Mask R-CNN: Accurate but not fast enough (<10 fps)



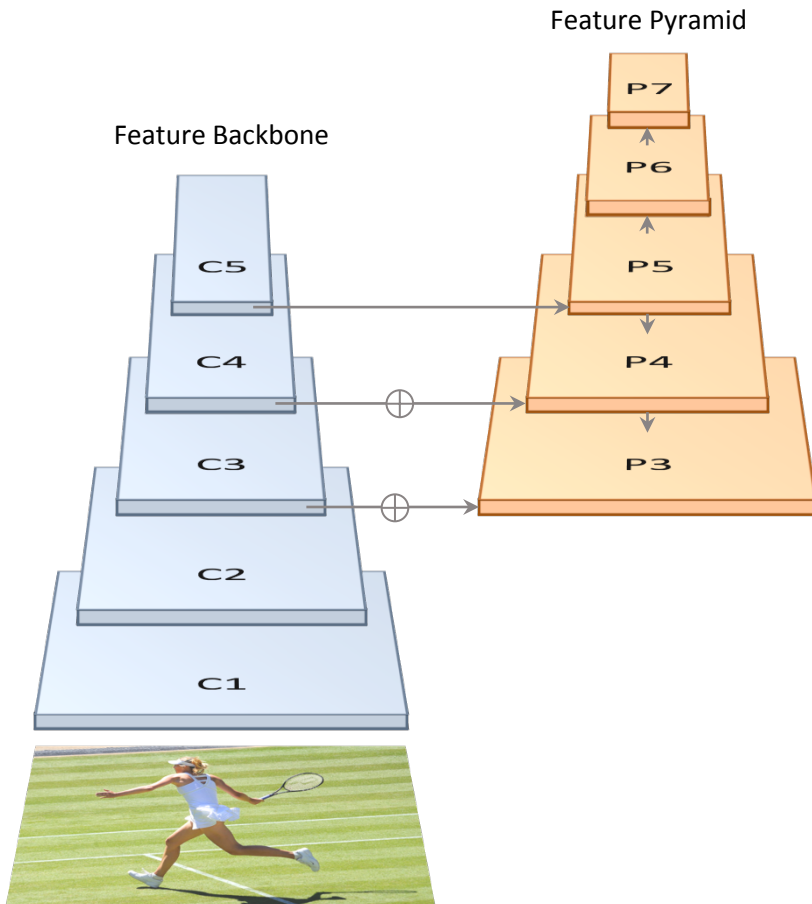
- *Stage 1*: use Region Proposal Network to generate region proposals
- *Stage 2*: pool features for each proposal (via ROI-align) and classify

[K. He et al., "Mask R-CNN", ICCV 2017]

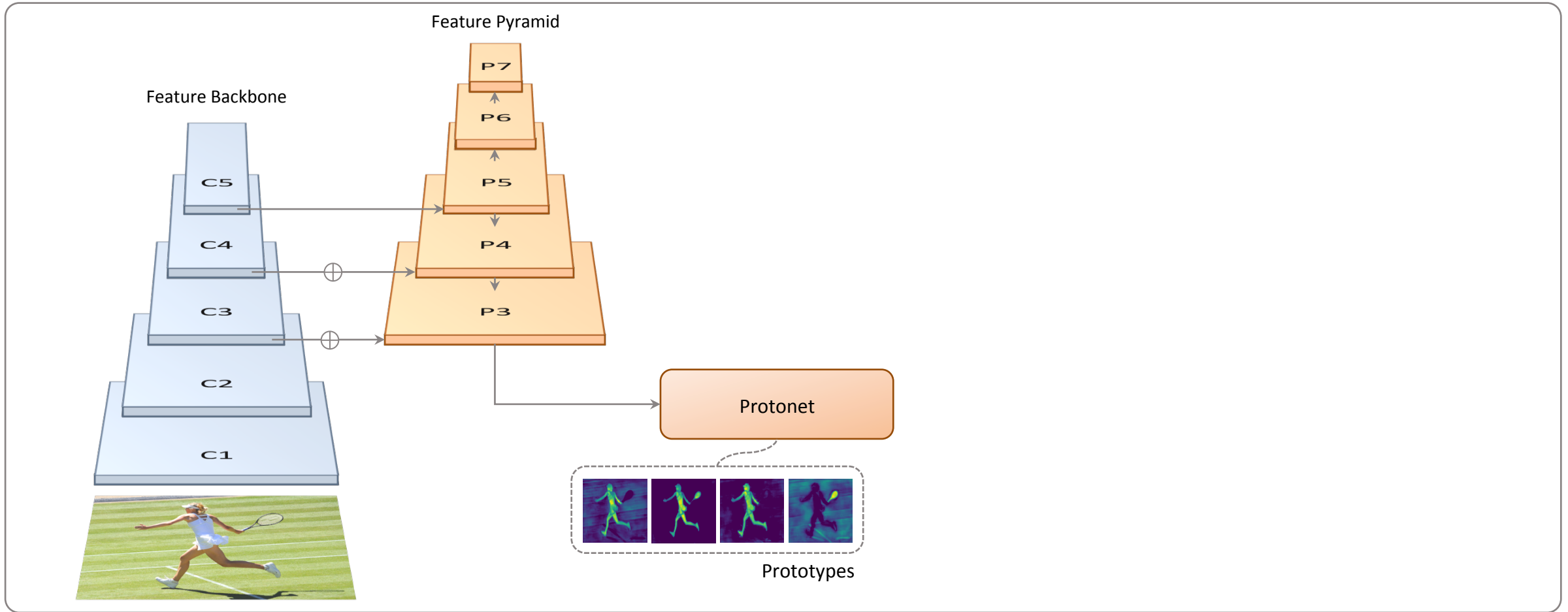
YOLACT architecture



YOLACT architecture

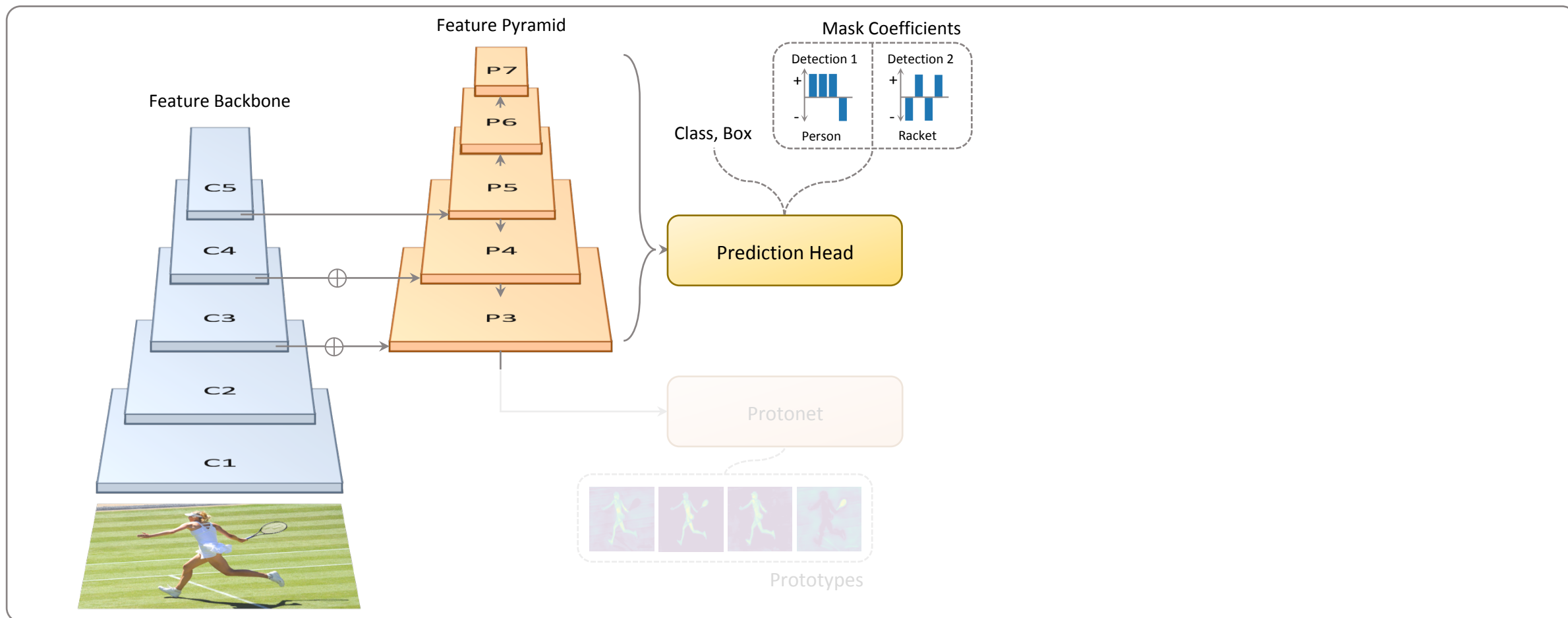


YOLACT architecture



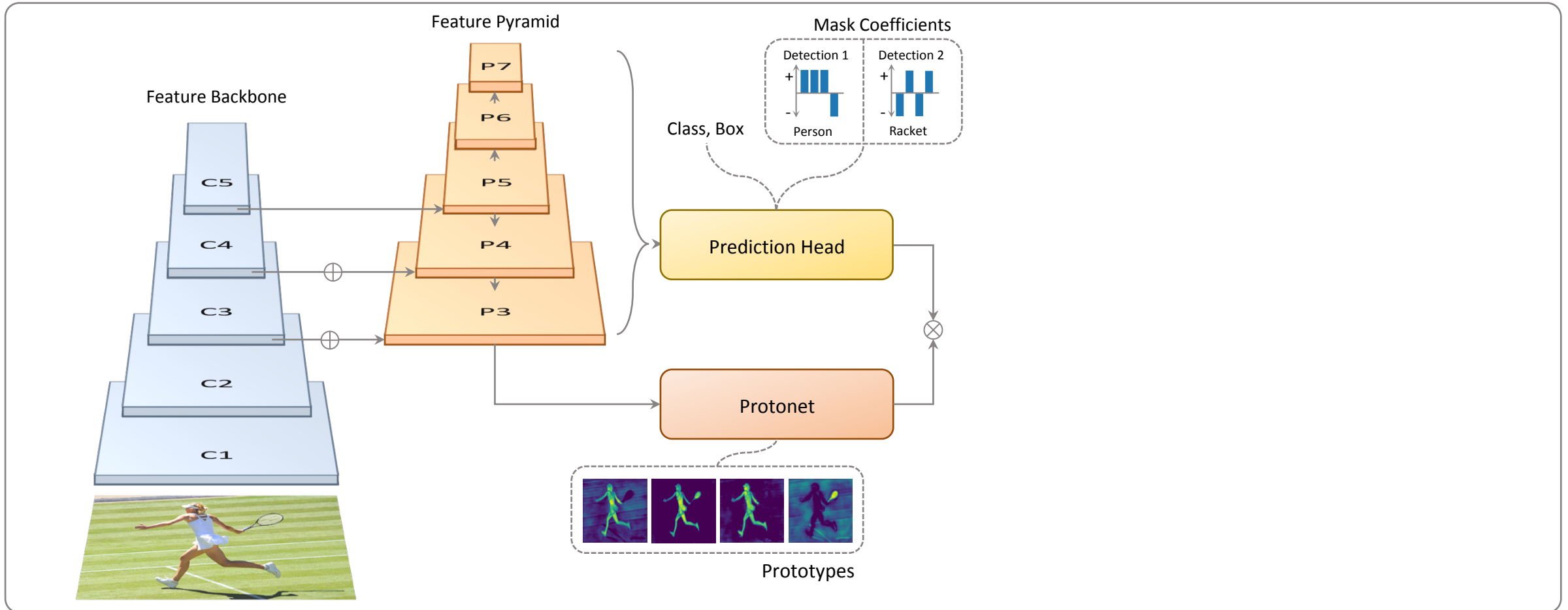
- Attach an FCN (“ProtoNet”) to the largest feature layer (P3) to produce k image-resolution prototype masks

YOLACT architecture



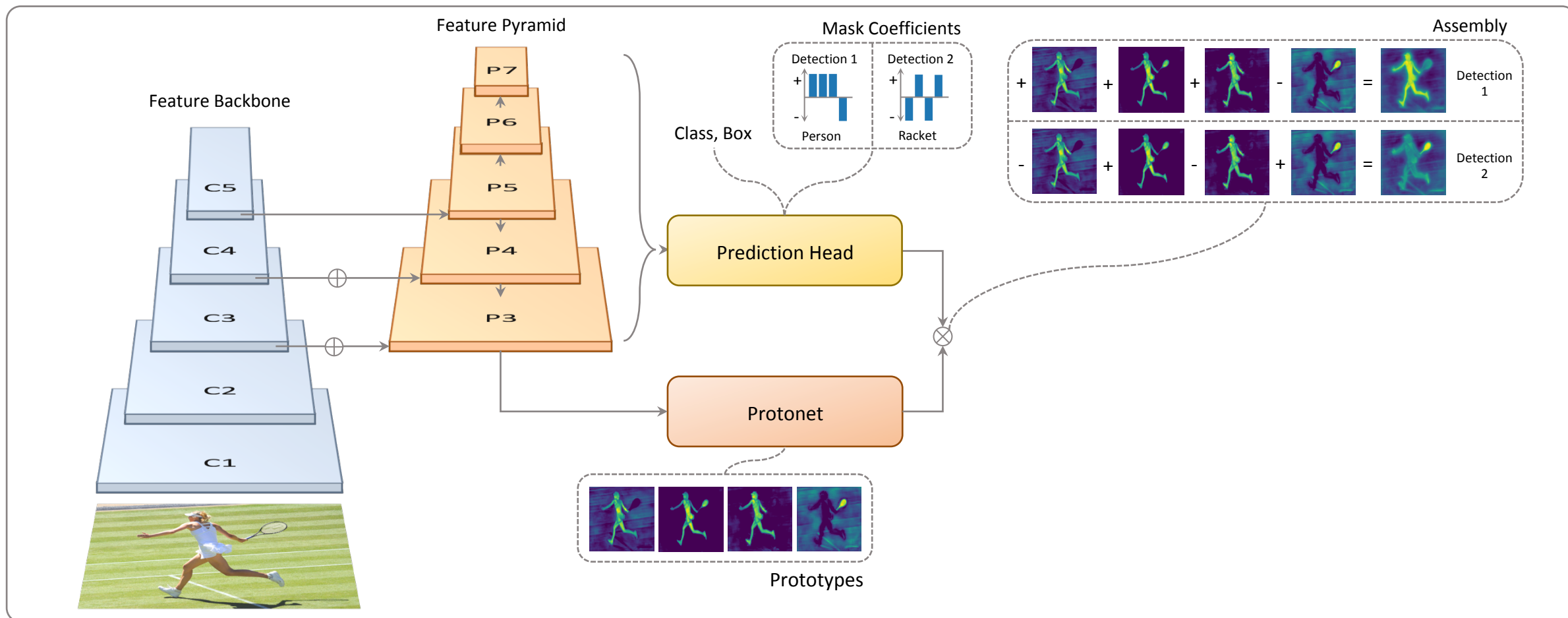
- In parallel, predict k mask coefficients for each anchor box (in addition to class confidences and box coefficients)

YOLACT architecture



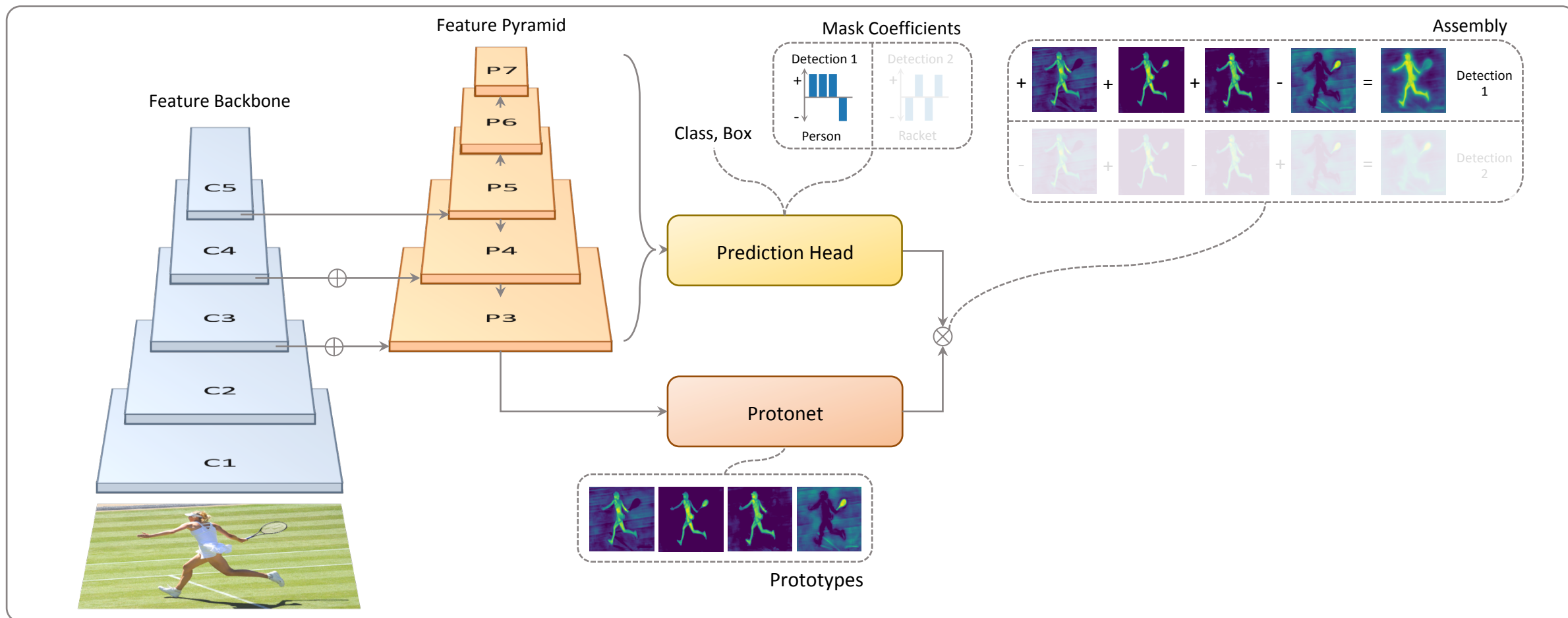
- For each instance, linearly combine prototypes using corresponding predicted coefficients

YOLACT architecture



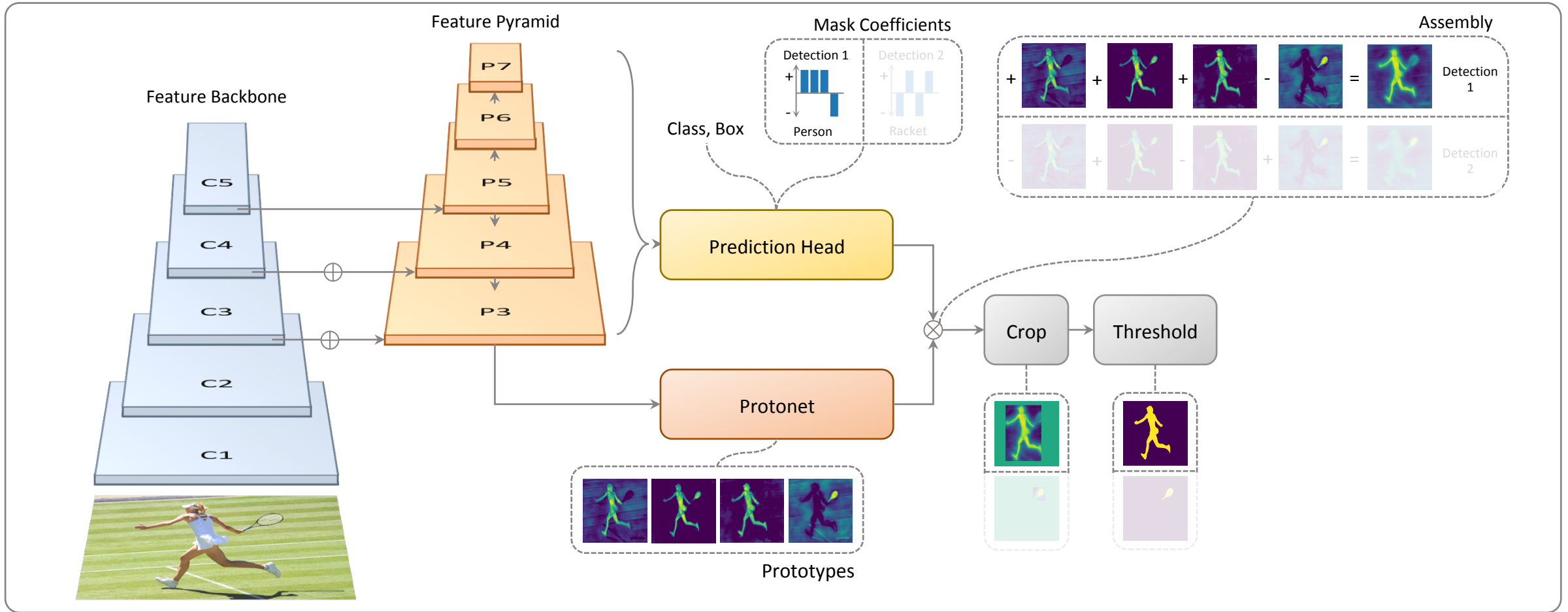
- For each instance, linearly combine prototypes using corresponding predicted coefficients

YOLACT architecture



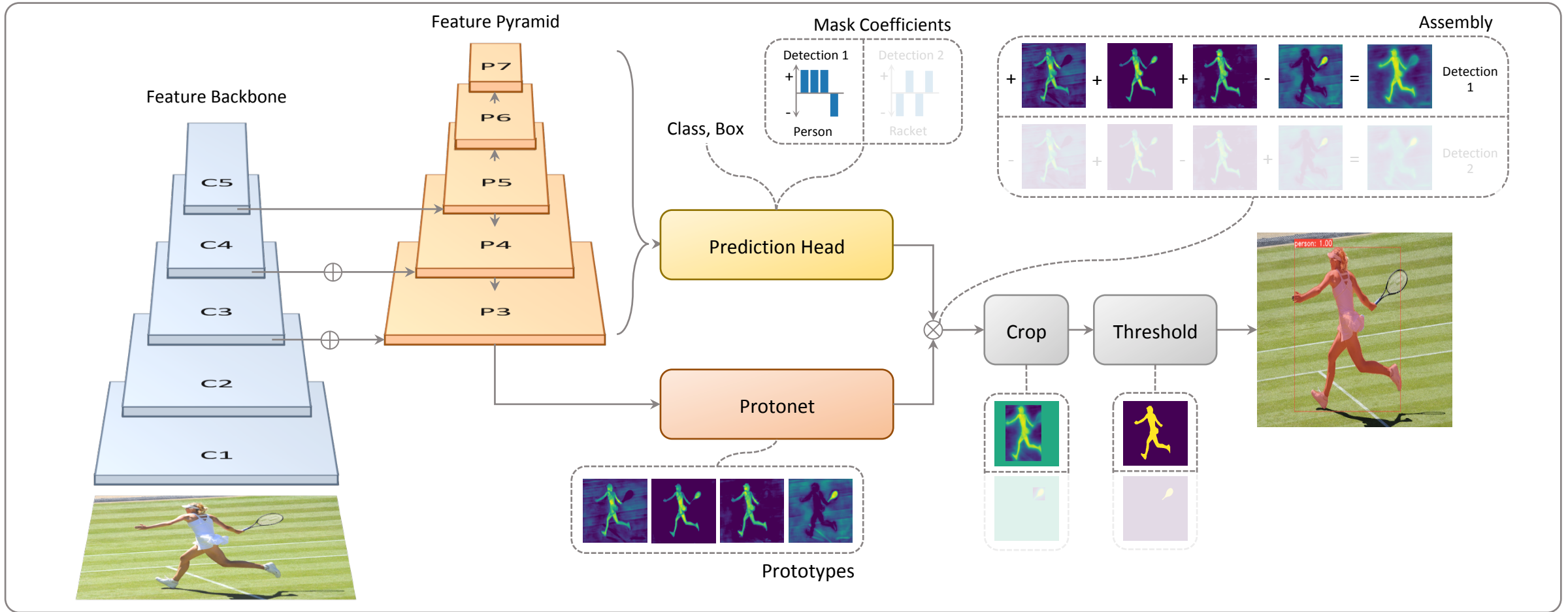
- For each instance, linearly combine prototypes using corresponding predicted coefficients

YOLACT architecture



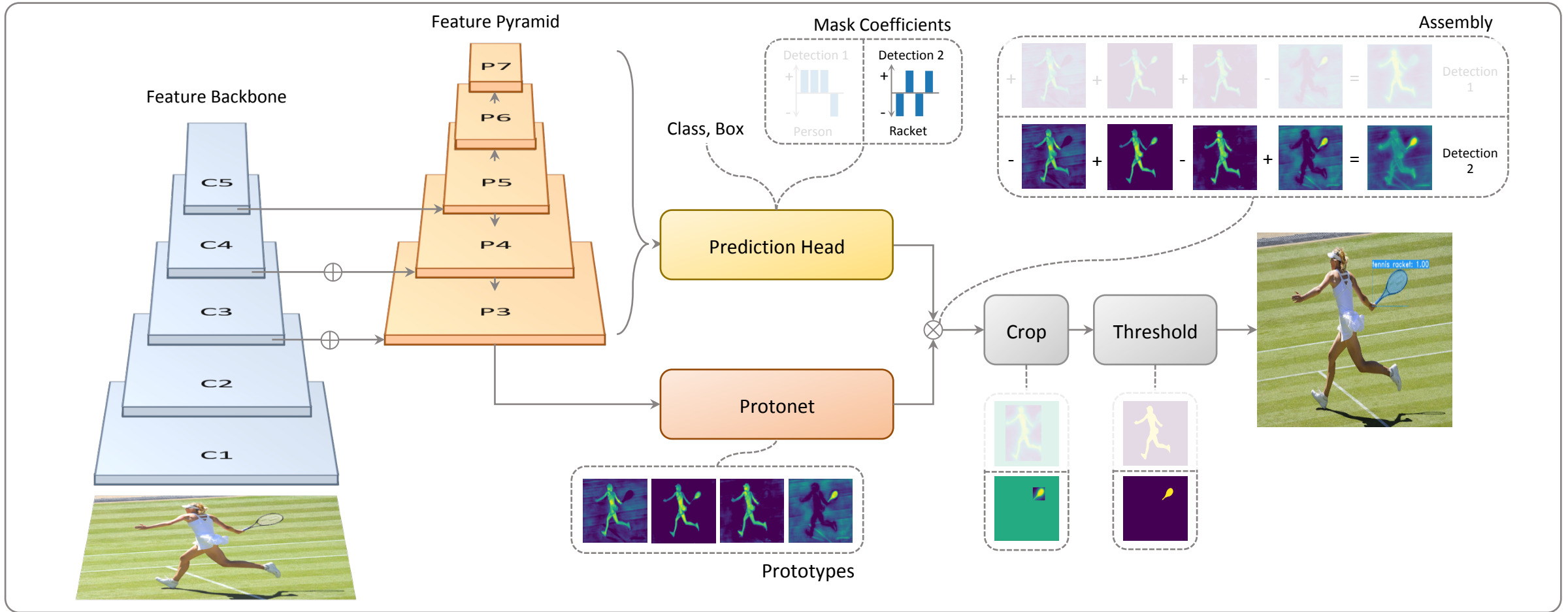
- Finally, crop with the predicted bounding box and threshold

YOLACT architecture



- Finally, crop with the predicted bounding box and threshold

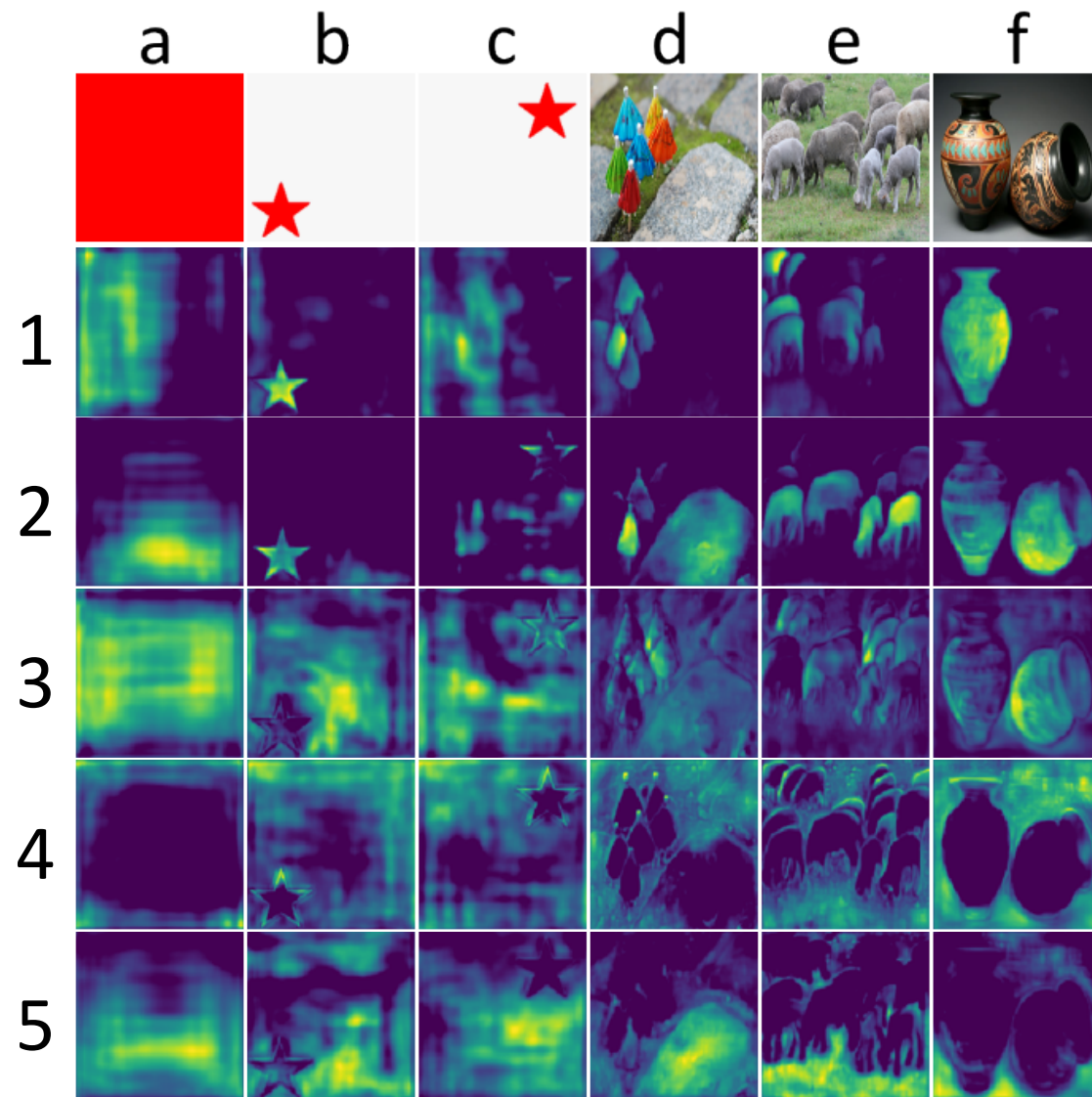
YOLACT architecture



- Finally, crop with the predicted bounding box and threshold

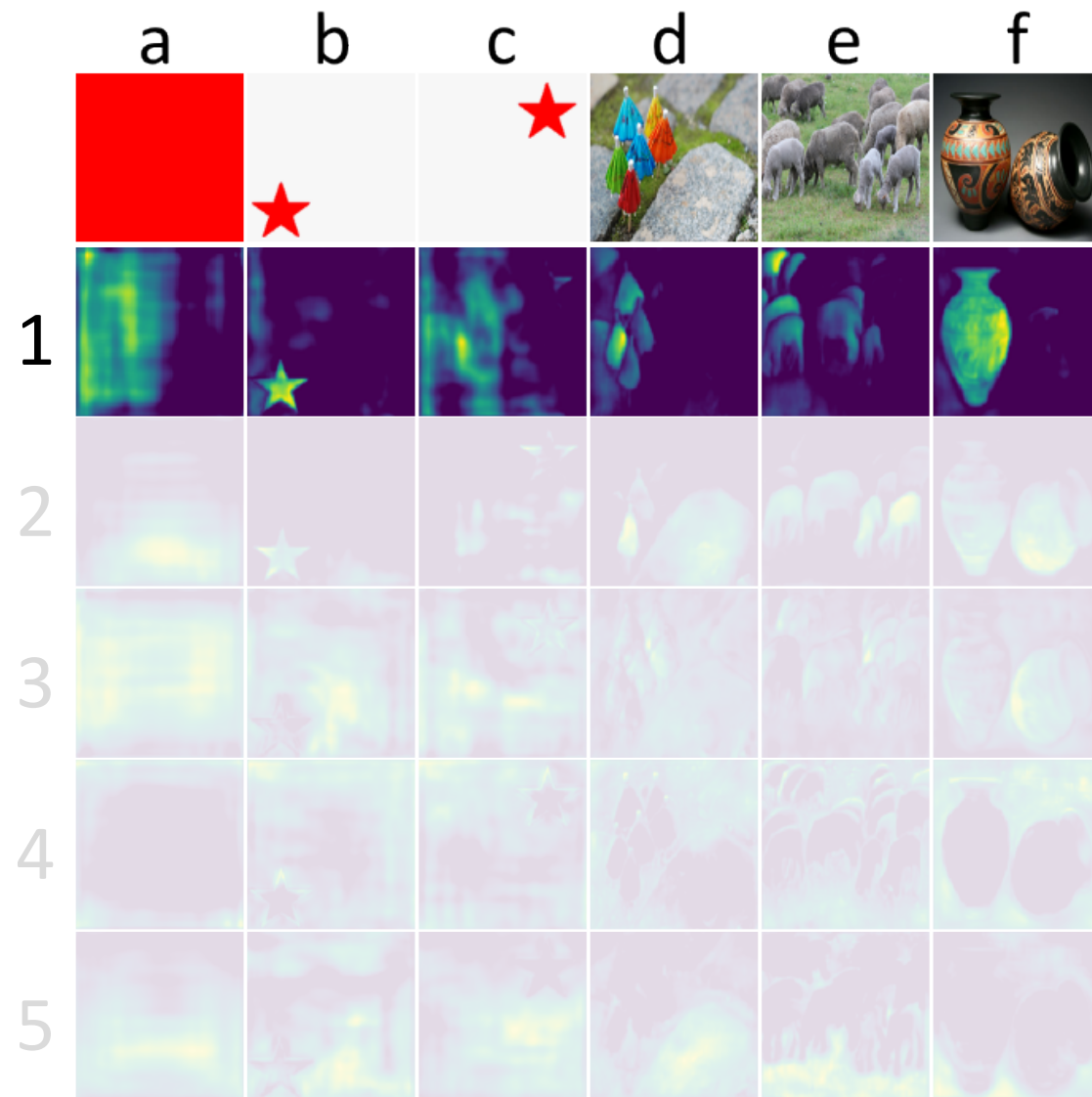
Prototype behavior

- Spatially partition the image
- Segment background
- Detect instance contours
- Encode position-sensitive directional maps
- Most do a combination



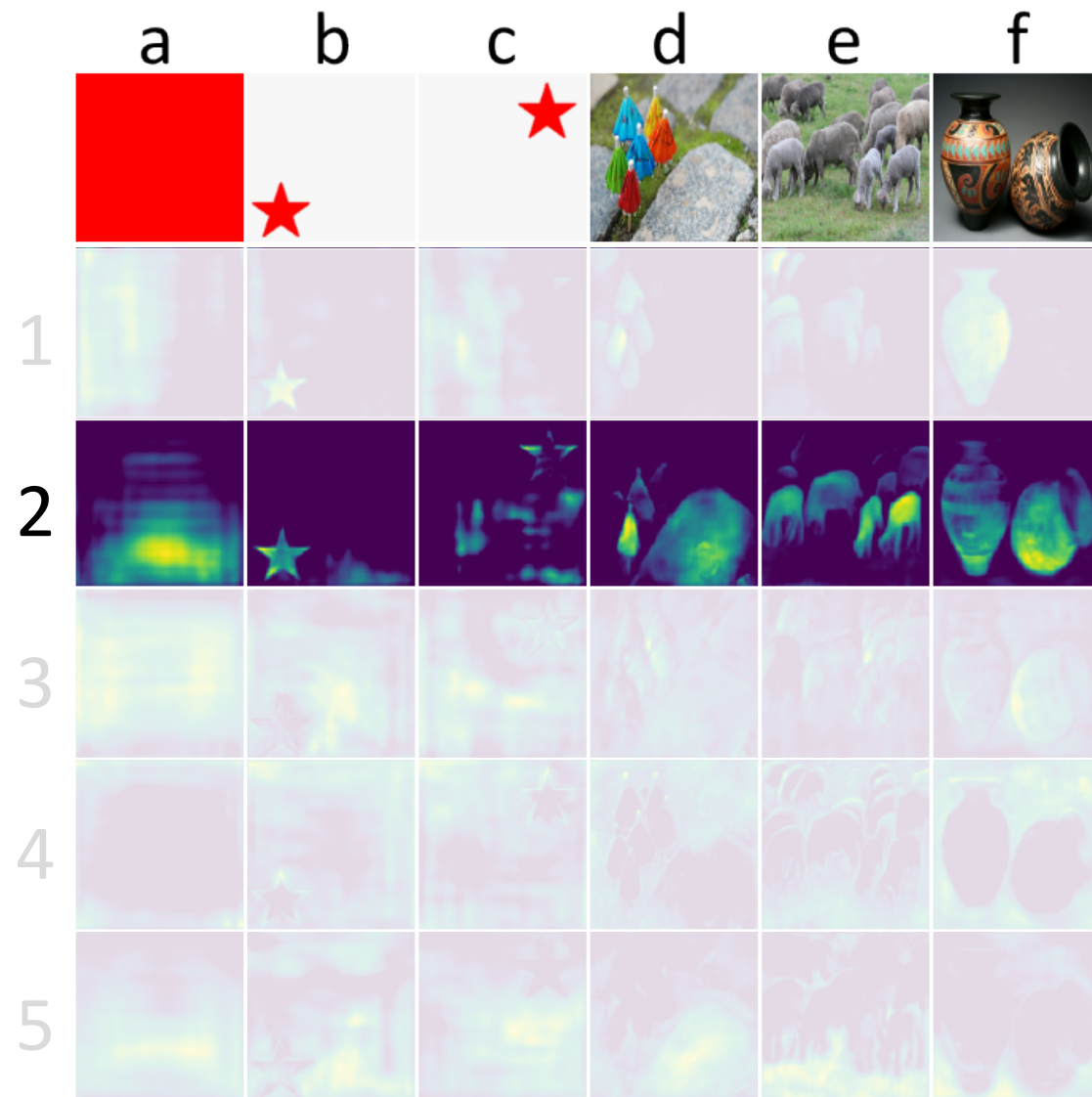
Prototype behavior

- Spatially partition the image
- Segment background
- Detect instance contours
- Encode position-sensitive directional maps
- Most do a combination



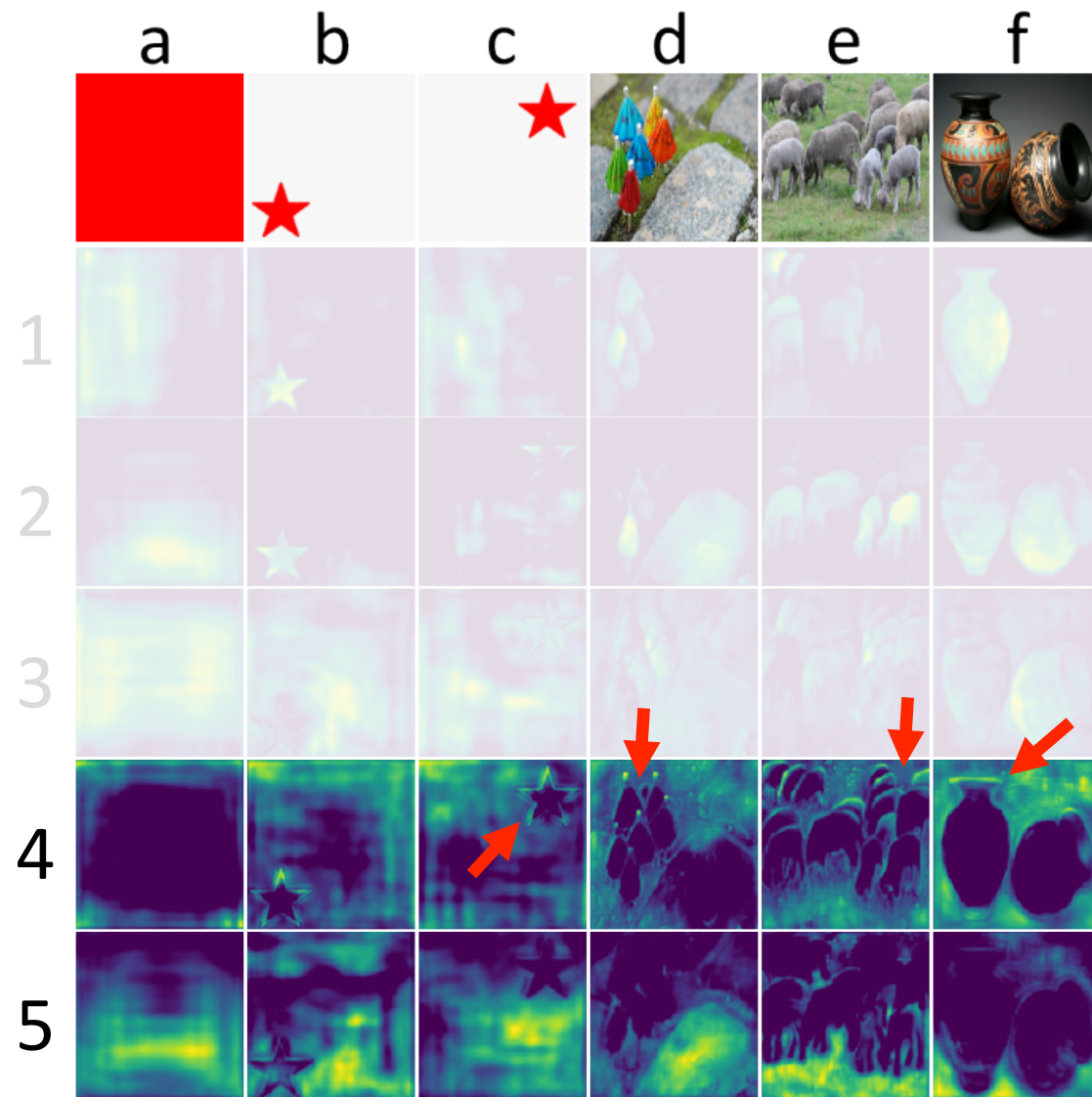
Prototype behavior

- Spatially partition the image
- Segment background
- Detect instance contours
- Encode position-sensitive directional maps
- Most do a combination



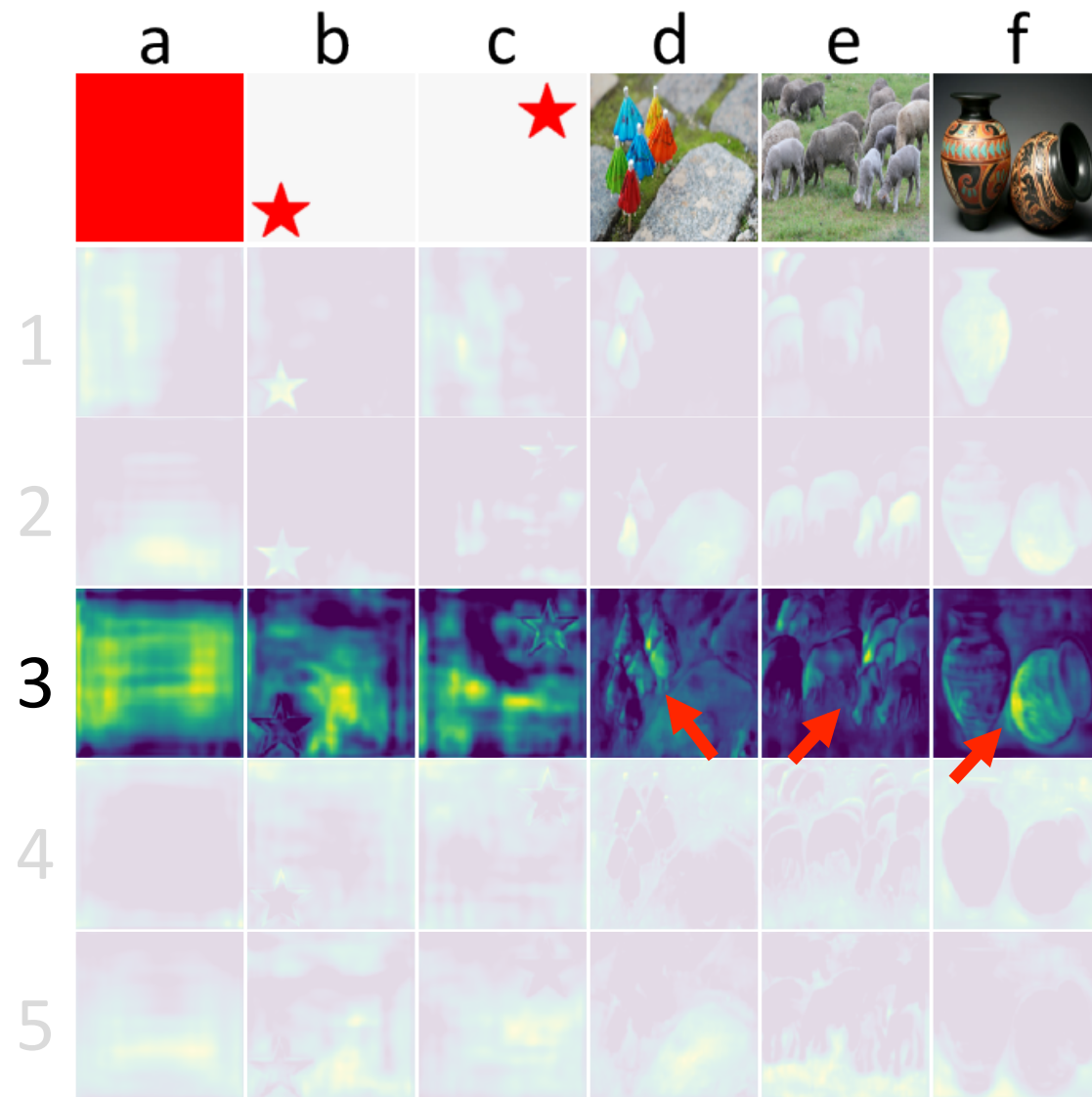
Prototype behavior

- Spatially partition the image
- Segment background
- Detect instance contours
- Encode position-sensitive directional maps
- Most do a combination



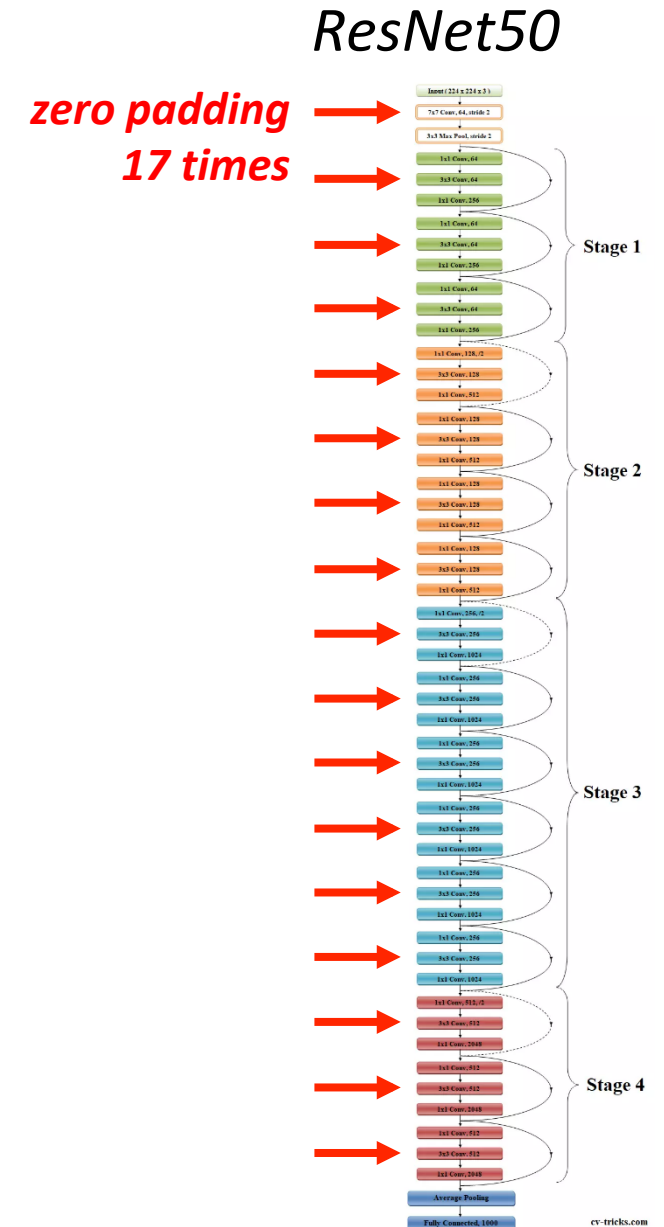
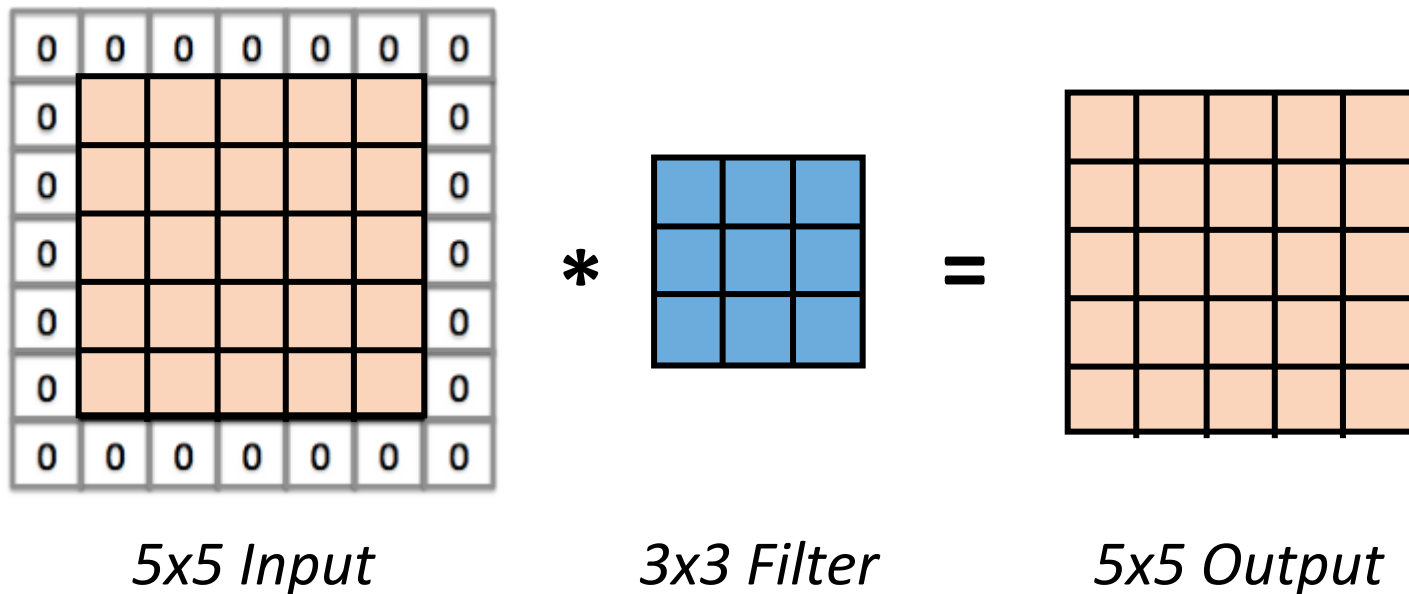
Prototype behavior

- Spatially partition the image
- Segment background
- Detect instance contours
- Encode position-sensitive directional maps
- Most do a combination



Zero-padding in ResNets

- Needed to keep input and output spatial resolution same



Results

- First *real-time* (> 30 fps) instance segmentation algorithm with competitive results on the challenging MS COCO dataset

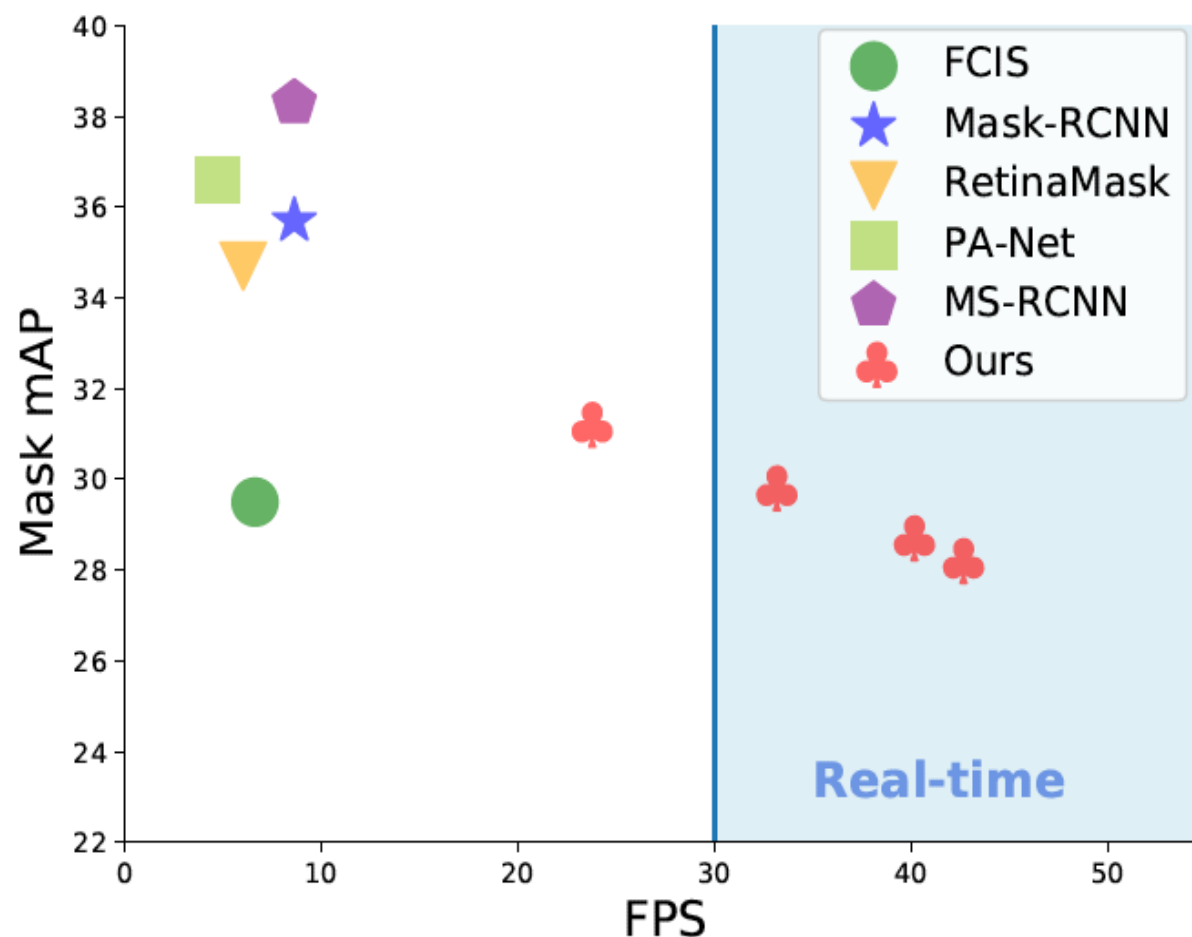
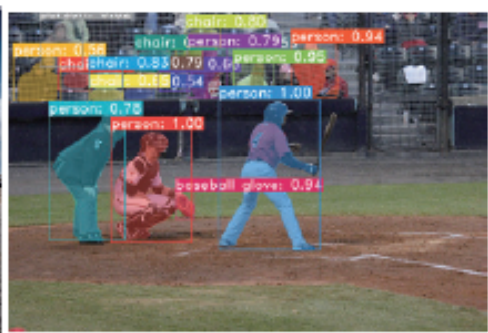
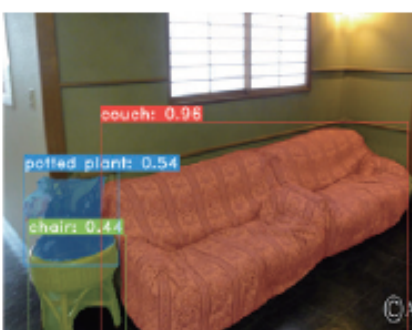
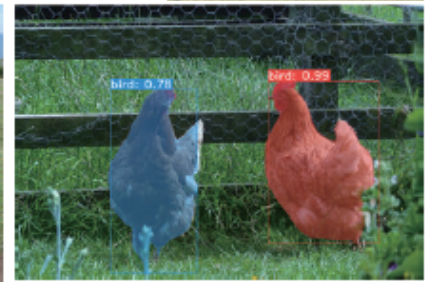
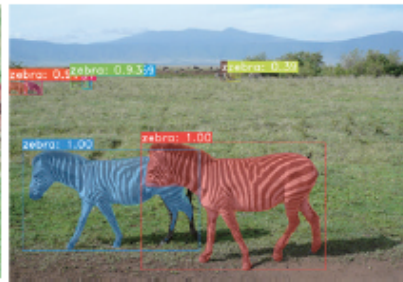
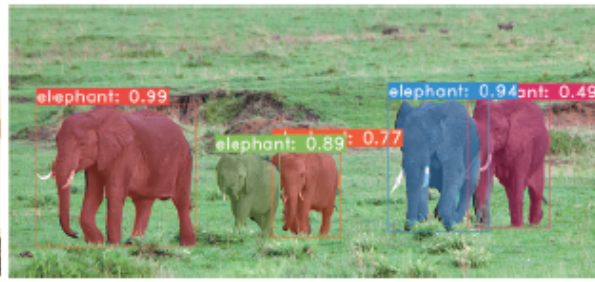
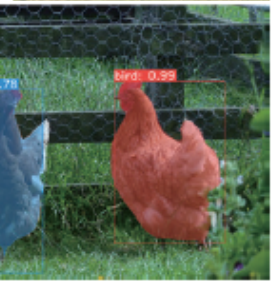
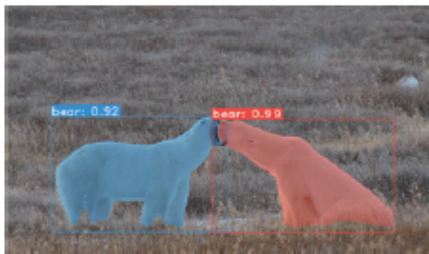
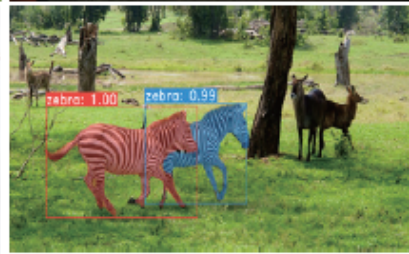
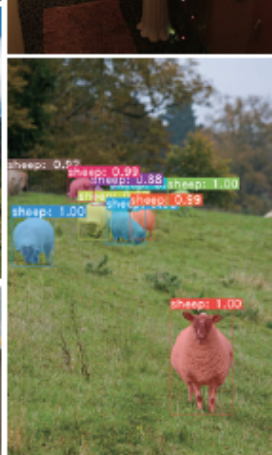
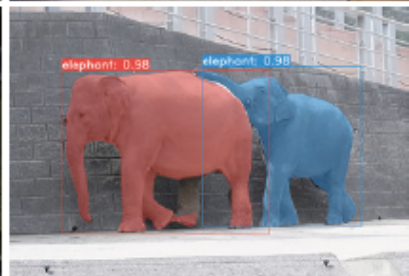
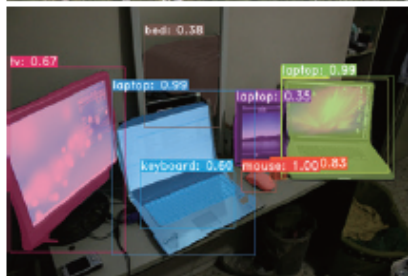
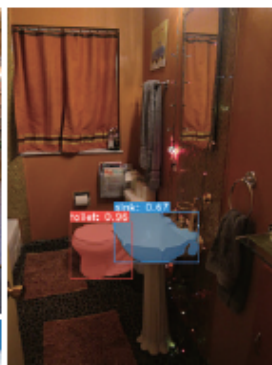
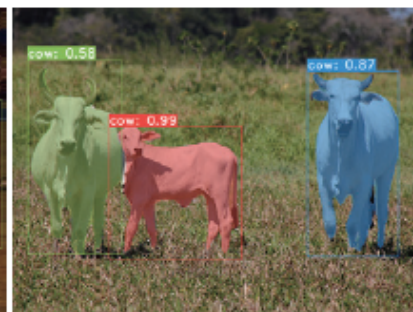


Figure 1: Speed-performance trade-off for various instance segmentation methods on COCO.





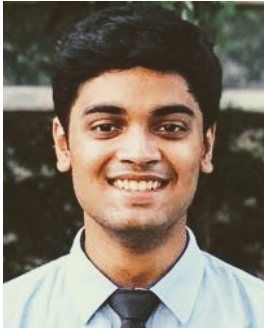
Conclusions

- Tremendous success stories in computer vision, but mostly limited to **specific domains with lots of labeled data** 😞
- Learn to understand visual data with **minimal human supervision**
 - *Challenging* since there's no direct supervision
 - *But with the right constraints*, can push the algorithm to behave in desirable ways with little to no supervision
 - Handling *dynamic environments* requires fast learning and inference
- Code, additional results available: <http://web.cs.ucdavis.edu/~yjlee/>

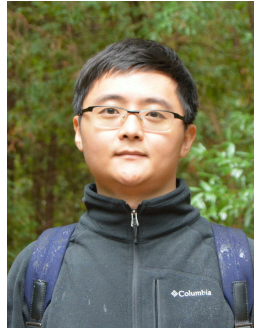
Acknowledgements



Krishna K. Singh



Utkarsh Ojha



Fanyi Xiao



Maheen Rashid



Xueyan Zou



Daniel Bolya



Chong Zhou

Funding agencies: National Science Foundation, Army Research Office, Hellman Foundation, UC Davis, Intel, Adobe, Nvidia, Google, Amazon

| Method | Backbone | FPS | Time | AP | AP ₅₀ | AP ₇₅ | AP _S | AP _M | AP _L |
|-----------------|-----------|-------------|-------------|-------------|------------------|------------------|-----------------|-----------------|-----------------|
| PA-Net [29] | R-50-FPN | 4.7 | 212.8 | 36.6 | 58.0 | 39.3 | 16.3 | 38.1 | 53.1 |
| RetinaMask [14] | R-101-FPN | 6.0 | 166.7 | 34.7 | 55.4 | 36.9 | 14.3 | 36.7 | 50.5 |
| FCIS [24] | R-101-C5 | 6.6 | 151.5 | 29.5 | 51.5 | 30.2 | 8.0 | 31.0 | 49.7 |
| Mask R-CNN [18] | R-101-FPN | 8.6 | 116.3 | 35.7 | 58.0 | 37.8 | 15.5 | 38.1 | 52.4 |
| MS R-CNN [20] | R-101-FPN | 8.6 | 116.3 | 38.3 | 58.8 | 41.5 | 17.8 | 40.4 | 54.4 |
| YOLACT-550 | R-101-FPN | 33.5 | 29.8 | 29.8 | 48.5 | 31.2 | 9.9 | 31.3 | 47.7 |
| YOLACT-400 | R-101-FPN | 45.3 | 22.1 | 24.9 | 42.0 | 25.4 | 5.0 | 25.3 | 45.0 |
| YOLACT-550 | R-50-FPN | 45.0 | 22.2 | 28.2 | 46.6 | 29.2 | 9.2 | 29.3 | 44.8 |
| YOLACT-550 | D-53-FPN | 40.7 | 24.6 | 28.7 | 46.8 | 30.0 | 9.5 | 29.6 | 45.5 |
| YOLACT-700 | R-101-FPN | 23.4 | 42.7 | 31.2 | 50.6 | 32.8 | 12.1 | 33.3 | 47.1 |

Table 1: **MS COCO [28] Results** We compare to state-of-the-art methods for mask mAP and speed on COCO `test-dev` and include several ablations of our base model, varying backbone network and image size. We denote the backbone architecture with `network-depth-features`, where R and D refer to ResNet [19] and DarkNet [36], respectively. Our base model, YOLACT-550 with ResNet-101, is 3.9x faster than the previous fastest approach with competitive mask mAP.

| Method | NMS | AP | FPS | Time |
|------------|----------|-------------|-------------|--------------|
| YOLACT | Standard | 30.0 | 24.0 | 41.6 |
| | Fast | 29.9 | 33.5 | 29.8 |
| Mask R-CNN | Standard | 36.1 | 8.6 | 116.0 |
| | Fast | 35.8 | 9.9 | 101.0 |

(a) **Fast NMS** Fast NMS performs only slightly worse than standard NMS, while being around 12 ms faster. We also observe a similar trade-off implementing Fast NMS in Mask R-CNN.

| k | AP | FPS | Time |
|-----|-------------|-------------|-------------|
| 8 | 26.8 | 33.0 | 30.4 |
| 16 | 27.1 | 32.8 | 30.5 |
| *32 | 27.7 | 32.4 | 30.9 |
| 64 | 27.8 | 31.7 | 31.5 |
| 128 | 27.6 | 31.5 | 31.8 |
| 256 | 27.7 | 29.8 | 33.6 |

(b) **Prototypes** Choices for k . We choose 32 for its mix of performance and speed.

| Method | AP | FPS | Time |
|---------------------------------|-------------|-------------|-------------|
| FCIS w/o Mask Voting | 27.8 | 9.5 | 105.3 |
| Mask R-CNN (550×550) | 32.2 | 13.5 | 73.9 |
| <i>fc</i> -mask | 20.7 | 25.7 | 38.9 |
| YOLACT-550 (Ours) | 29.9 | 33.0 | 30.3 |

(c) **Accelerated Baselines** We compare to other baseline methods by tuning their speed-accuracy trade-offs. *fc*-mask is our model but with 16×16 masks produced from an *fc* layer.

Table 2: **Ablations** All models evaluated on COCO `val2017` using our servers. Models in Table 2b were trained for 400k iterations instead of 800k. Time in milliseconds reported for convenience.