Interactive Segmentation Paper Review

~~(not end game)~~

Fast Interactive Object Annotation with Curve-GCN

Taeu

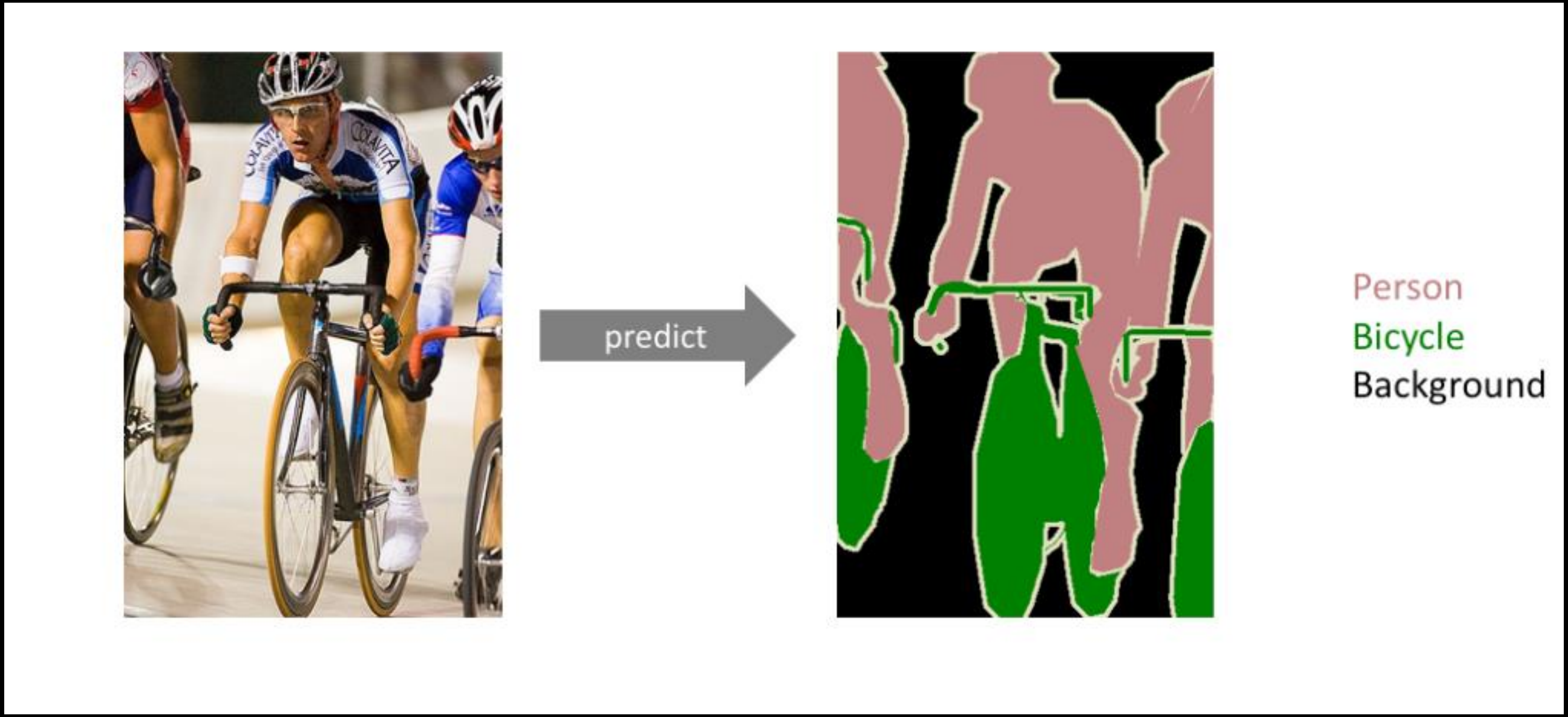
2020.04.20

What is an Interactive Segmentation?

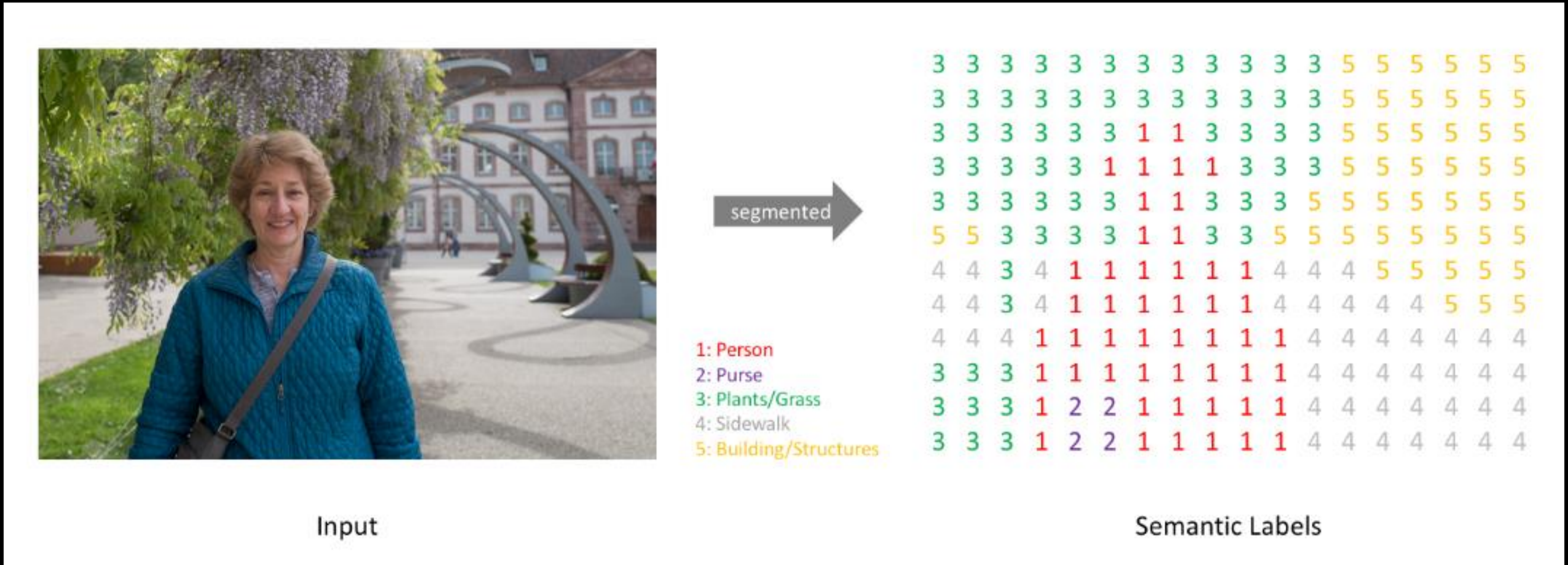
Interactive Segmentation

Interactive Segmentation

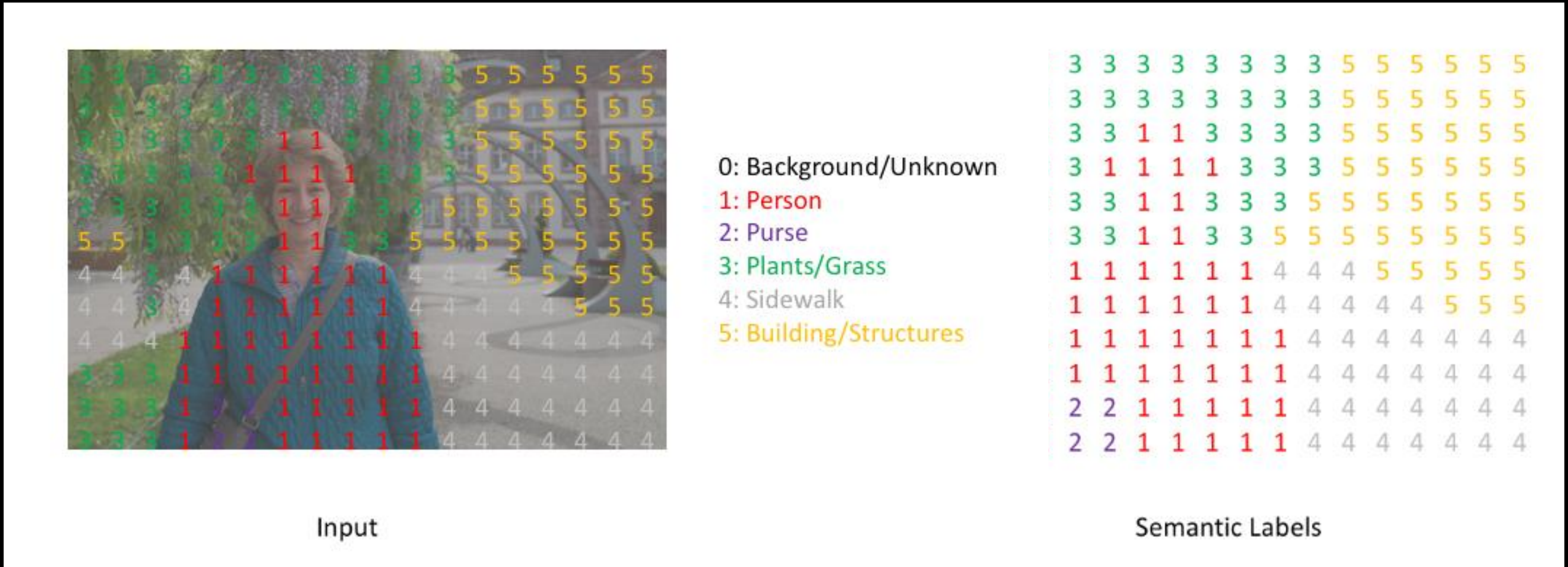
Interactive **Segmentation**



Interactive **Segmentation**

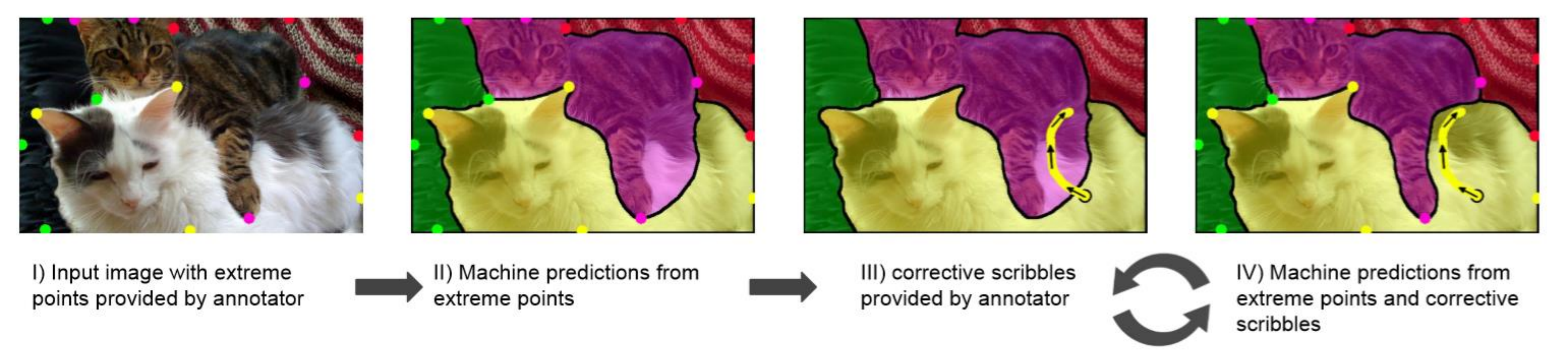


Interactive **Segmentation**

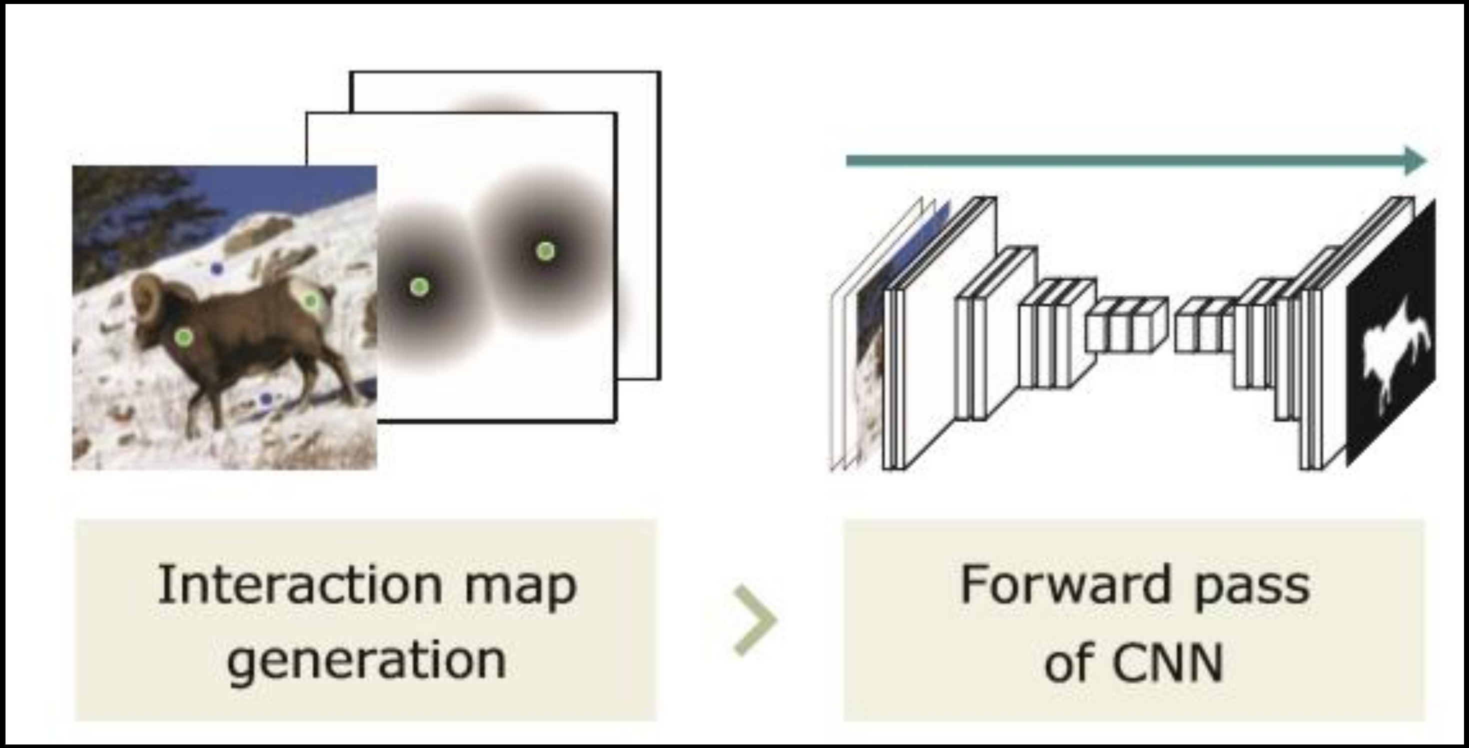


Interactive Segmentation

Interactive Segmentation



Interactive Segmentation



1. Paper search

CVPR 2019 ‘Interactive’ related work

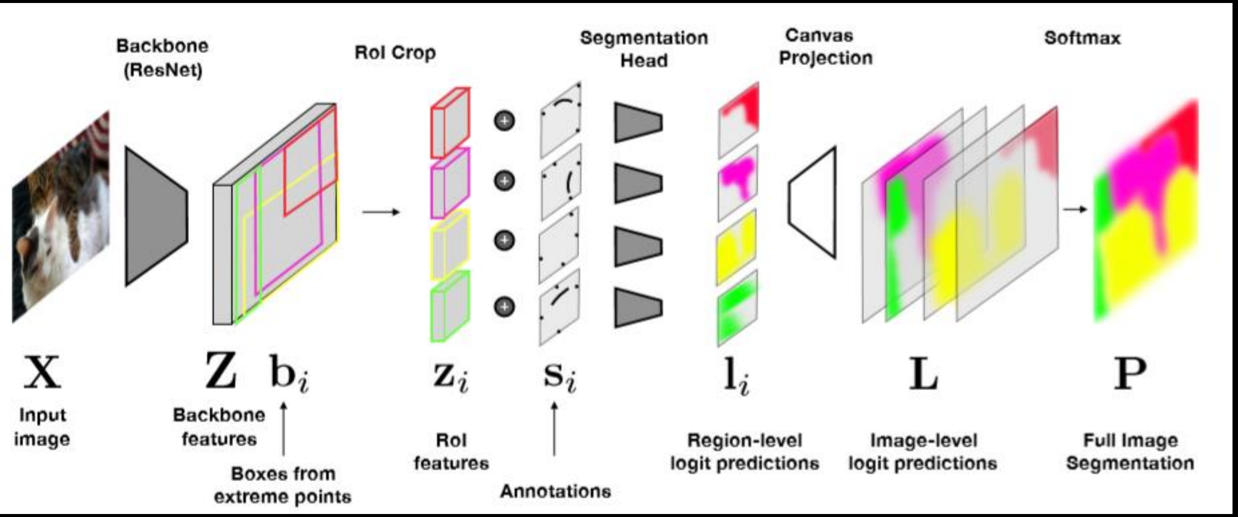
no	title	cited	-
1	**Interactive Full Image Segmentation by Considering All Regions Jointly**	7	google
2	Large-scale interactive object segmentation with human annotators	13	google
3	Constrained Generative Adversarial Networks for Interactive Image Generation	1	Air Force Research, USA
4	**Interactive Image Segmentation via Backpropagating Refinement Scheme**	7	havard, korea univ
5	**Fast Interactive Object Annotation with Curve-GCN**	21	toronto, NVIDIA
6	Content-Aware Multi-Level Guidance for Interactive Instance Segmentation	7	Bonn, Singapore univ

1. Paper search

CVPR 2019 'Interactive' related work

no	title	cited	-
1	**Interactive Full Image Segmentation by Considering All Regions Jointly**	7	google
2	Large-scale interactive object segmentation with human annotators	13	google
3	Constrained Generative Adversarial Networks for Interactive Image Generation	1	Air Force Research, USA
4	**Interactive Image Segmentation via Backpropagating Refinement Scheme**	7	havard, korea univ
5	**Fast Interactive Object Annotation with Curve-GCN**	21	toronto, NVIDIA
6	Content-Aware Multi-Level Guidance for Interactive Instance Segmentation	7	Bonn, Singapore univ

scribble-based, mask RCNN

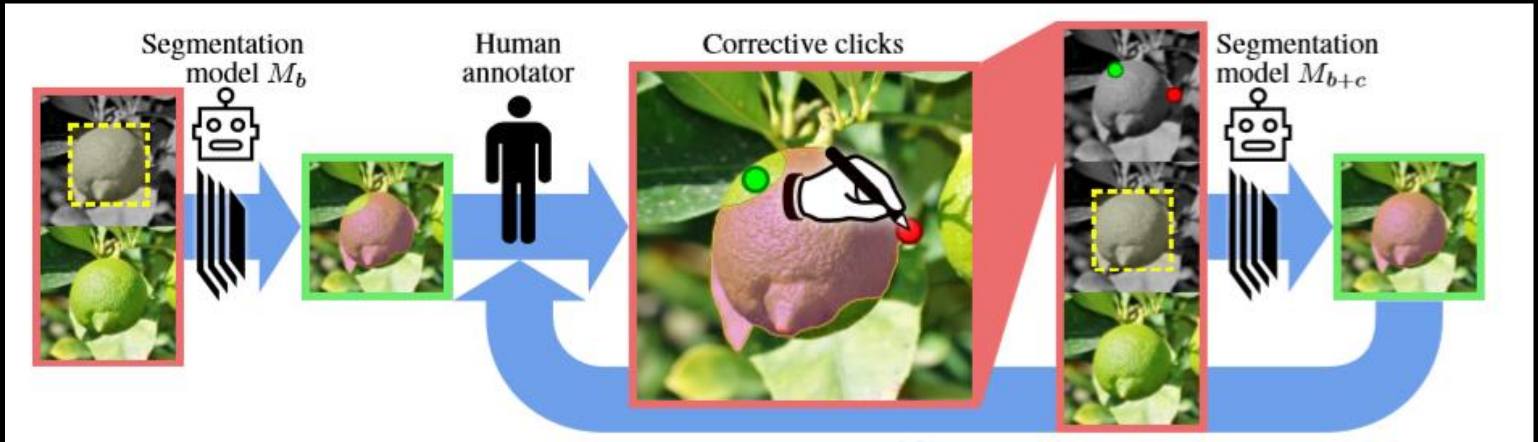


CVPR 2019 'Interactive' related work

no	title	cited	-
1	**Interactive Full Image Segmentation by Considering All Regions Jointly**	7	google
2	Large-scale interactive object segmentation with human annotators	13	google
3	Constrained Generative Adversarial Networks for Interactive Image Generation	1	Air Force Research, USA
4	**Interactive Image Segmentation via Backpropagating Refinement Scheme**	7	havard, korea univ
5	**Fast Interactive Object Annotation with Curve-GCN**	21	toronto, NVIDIA
6	Content-Aware Multi-Level Guidance for Interactive Instance Segmentation	7	Bonn, Singapore univ

scribble-based, mask RCNN

At scale, interaction behavior



we executed a large-scale annotation campaign, producing 2.5M instance masks on OpenImages.
we proposed a technique for automatically estimating the quality of individual masks.

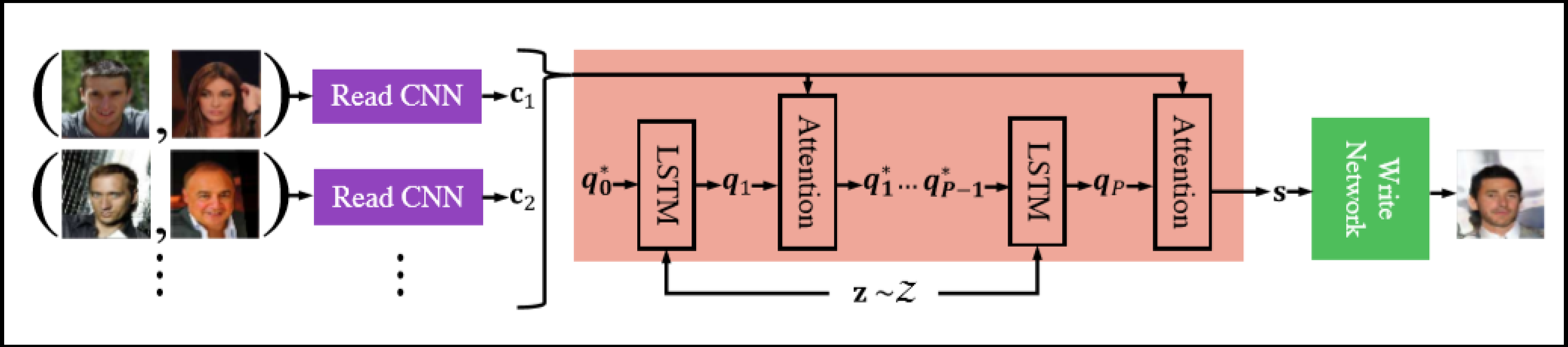
CVPR 2019 'Interactive' related work

no	title	cited	-
1	**Interactive Full Image Segmentation by Considering All Regions Jointly**	7	google
2	Large-scale interactive object segmentation with human annotators	13	google
3	Constrained Generative Adversarial Networks for Interactive Image Generation	1	Air Force Research, USA
4	**Interactive Image Segmentation via Backpropagating Refinement Scheme**	7	havard, korea univ
5	**Fast Interactive Object Annotation with Curve-GCN**	21	toronto, NVIDIA
6	Content-Aware Multi-Level Guidance for Interactive Instance Segmentation	7	Bonn, Singapore univ

scribble-based, mask RCNN

At scale, interaction behavior

Image generation, Adv.



1. Paper search

CVPR 2019 'Interactive' related work

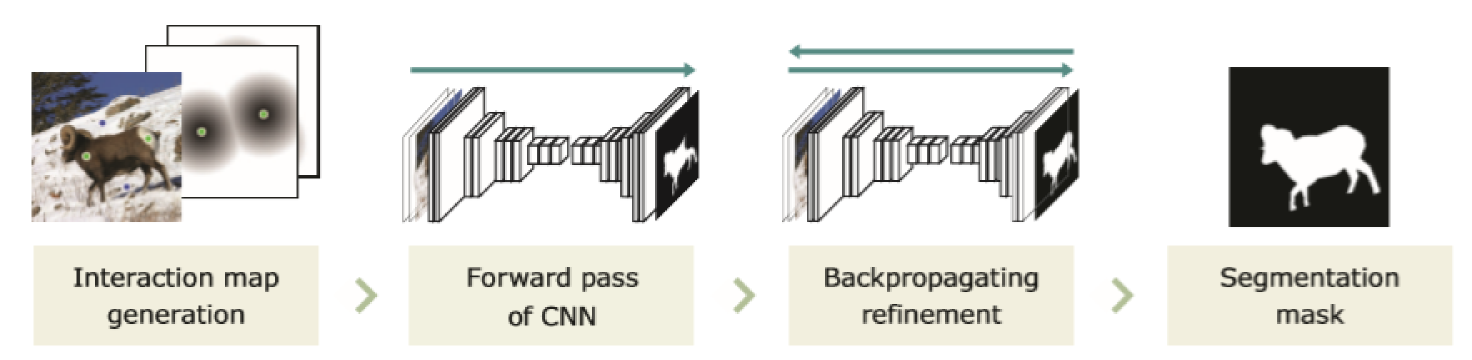
no	title	cited	-
1	**Interactive Full Image Segmentation by Considering All Regions Jointly**	7	google
2	Large-scale interactive object segmentation with human annotators	13	google
3	Constrained Generative Adversarial Networks for Interactive Image Generation	1	Air Force Research, USA
4	**Interactive Image Segmentation via Backpropagating Refinement Scheme**	7	havard, korea univ
5	**Fast Interactive Object Annotation with Curve-GCN**	21	toronto, NVIDIA
6	Content-Aware Multi-Level Guidance for Interactive Instance Segmentation	7	Bonn, Singapore univ

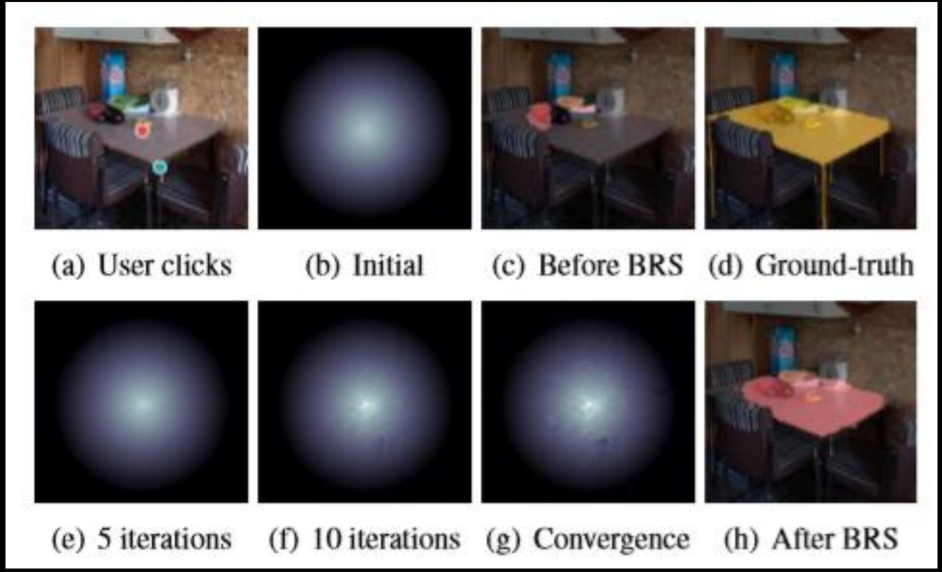
scribble-based, mask RCNN

At scale, interaction behavior

Image generation, Adv.

Iterative backpropagation





1. Paper search

CVPR 2019 'Interactive' related work

no	title	cited	-
1	**Interactive Full Image Segmentation by Considering All Regions Jointly**	7	google
2	Large-scale interactive object segmentation with human annotators	13	google
3	Constrained Generative Adversarial Networks for Interactive Image Generation	1	Air Force Research, USA
4	**Interactive Image Segmentation via Backpropagating Refinement Scheme**	7	havard, korea univ
5	**Fast Interactive Object Annotation with Curve-GCN**	21	toronto, NVIDIA
6	Content-Aware Multi-Level Guidance for Interactive Instance Segmentation	7	Bonn, Singapore univ

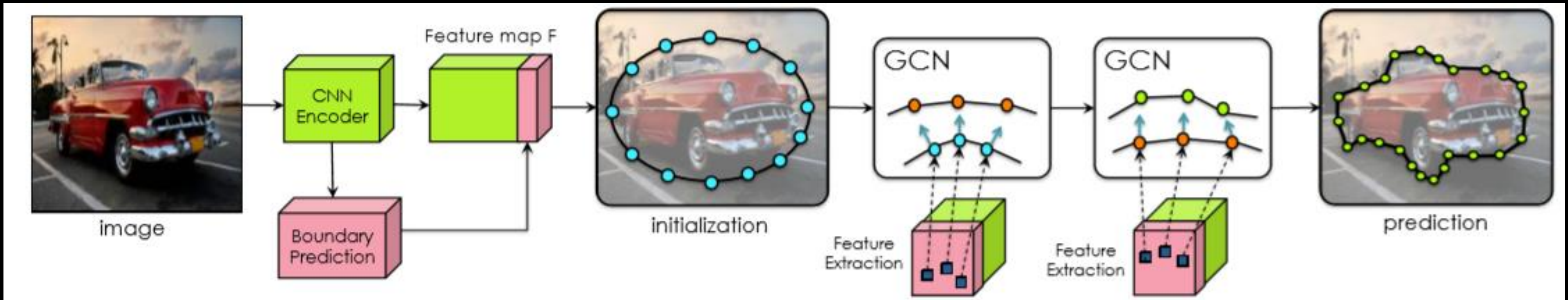
scribble-based, mask RCNN

At scale, interaction behavior

Image generation, Adv.

Iterative backpropagation

Graph Conv Network, +



1. Paper search

CVPR 2019 'Interactive' related work

no	title	cited	-
1	**Interactive Full Image Segmentation by Considering All Regions Jointly**	7	google
2	Large-scale interactive object segmentation with human annotators	13	google
3	Constrained Generative Adversarial Networks for Interactive Image Generation	1	Air Force Research, USA
4	**Interactive Image Segmentation via Backpropagating Refinement Scheme**	7	havard, korea univ
5	**Fast Interactive Object Annotation with Curve-GCN**	21	toronto, NVIDIA
6	Content-Aware Multi-Level Guidance for Interactive Instance Segmentation	7	Bonn, Singapore univ

scribble-based, mask RCNN

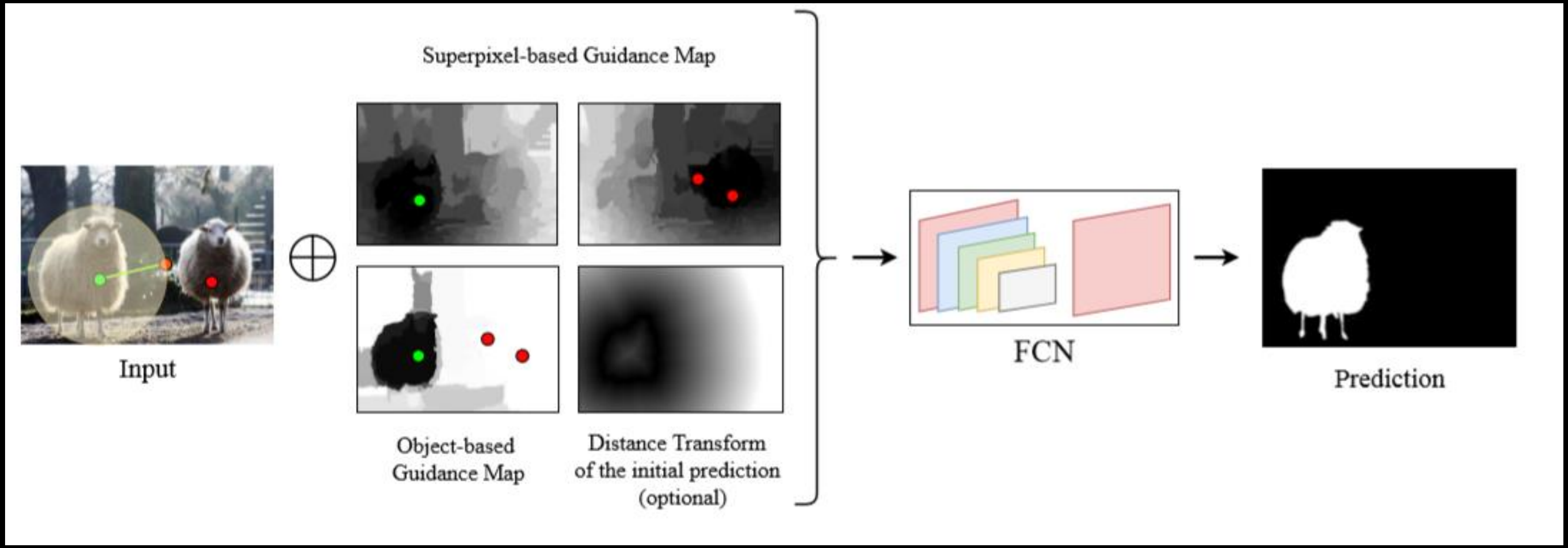
At scale, interaction behavior

Image generation, Adv.

Iterative backpropagation

Graph Conv Network, +

Guidance Map, FCN



1. Paper search

no	title	cited	-
1	**Interactive Full Image Segmentation by Considering All Regions Jointly**	7	google
2	Large-scale interactive object segmentation with human annotators	13	google
4	**Interactive Image Segmentation via Backpropagating Refinement Scheme**	7	havard, korea univ
5	**Fast Interactive Object Annotation with Curve-GCN**	21	toronto, NVIDIA
6	Content-Aware Multi-Level Guidance for Interactive Instance Segmentation	7	Bonn, Singapore univ

Segmentation

Interactive segmentation

1. Paper search

no	title	cited	-
1	Interactive Full Image Segmentation by Considering All Regions Jointly	7	google
2	Large-scale interactive object segmentation with human annotators	13	google
4	Interactive Image Segmentation via Backpropagating Refinement Scheme	7	havard, korea univ
5	Fast Interactive Object Annotation with Curve-GCN	21	toronto, NVIDIA
6	Content-Aware Multi-Level Guidance for Interactive Instance Segmentation	7	Bonn, Singapore univ

Segmentation

Interactive segmentation

Mask RCNN

Graph Convolutional Network

1. Paper search

no	title	cited	-
1	Interactive Full Image Segmentation by Considering All Regions Jointly	7	google
2	Large-scale interactive object segmentation with human annotators	13	google
4	Interactive Image Segmentation via Backpropagating Refinement Scheme	7	havard, korea univ
5	Fast Interactive Object Annotation with Curve-GCN	21	toronto, NVIDIA
6	Content-Aware Multi-Level Guidance for Interactive Instance Segmentation	7	Bonn, Singapore univ

Graph Convolutional Network

1. Paper search

no	title	cited	-
1	Interactive Full Image Segmentation by Considering All Regions Jointly	7	google
2	Large-scale interactive object segmentation with human annotators	13	google
4	Interactive Image Segmentation via Backpropagating Refinement Scheme	7	havard, korea univ
5	**Fast Interactive Object Annotation with Curve-GCN**	21	toronto, NVIDIA
6	Content-Aware Multi-Level Guidance for Interactive Instance Segmentation	7	Bonn, Singapore univ

Graph Convolutional Network

3D Computer graphics –
Spline parametrization
differentiable renderer

Graph Convolutional Network

Kipf & Welling (ICLR 2017),
Semi-Supervised Classification with Graph Convolutional Networks

Graph Convolutional Network

Graph Convolutional Network

Graph Convolutional Network

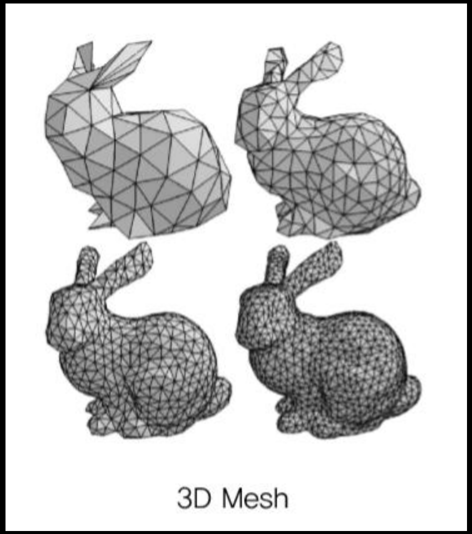
Graph **Convolutional** Network

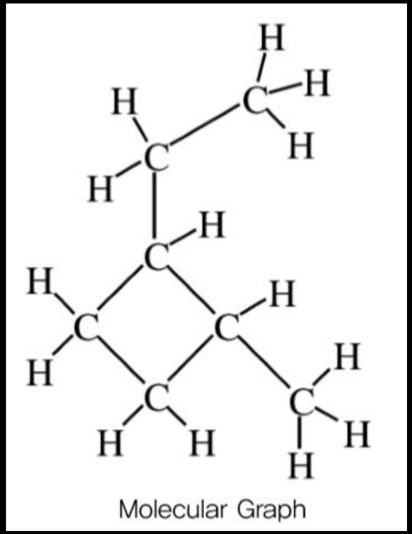
Graph Convolutional Network

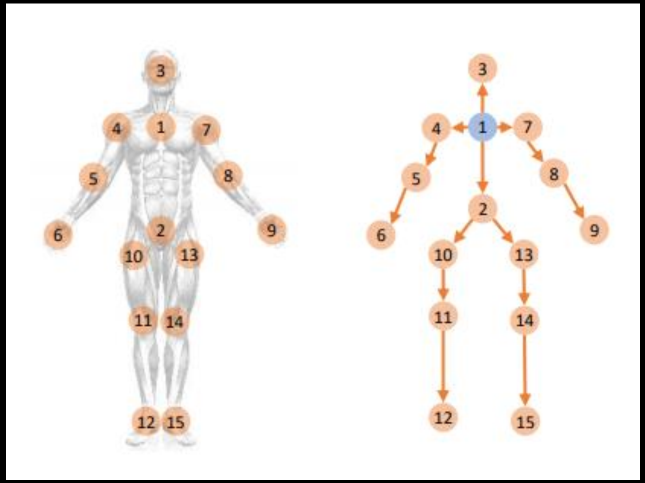
[Kaist 딥러닝 홀로서기 youtube 강의!](https://www.youtube.com/watch?v=YL1jGgcY78U&t=3223s)
https://www.youtube.com/watch?v=YL1jGgcY78U&t=3223s

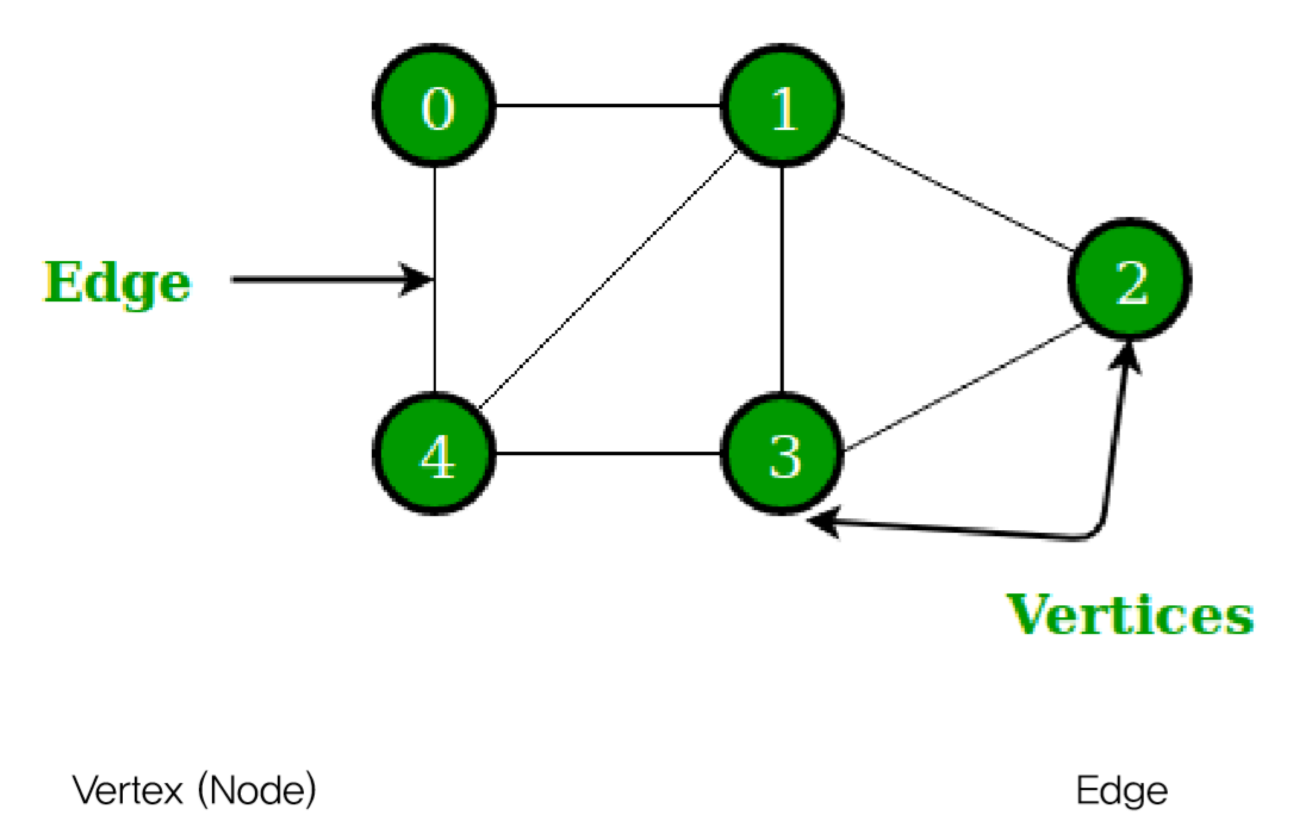
1. Paper search

Graph Convolutional Network



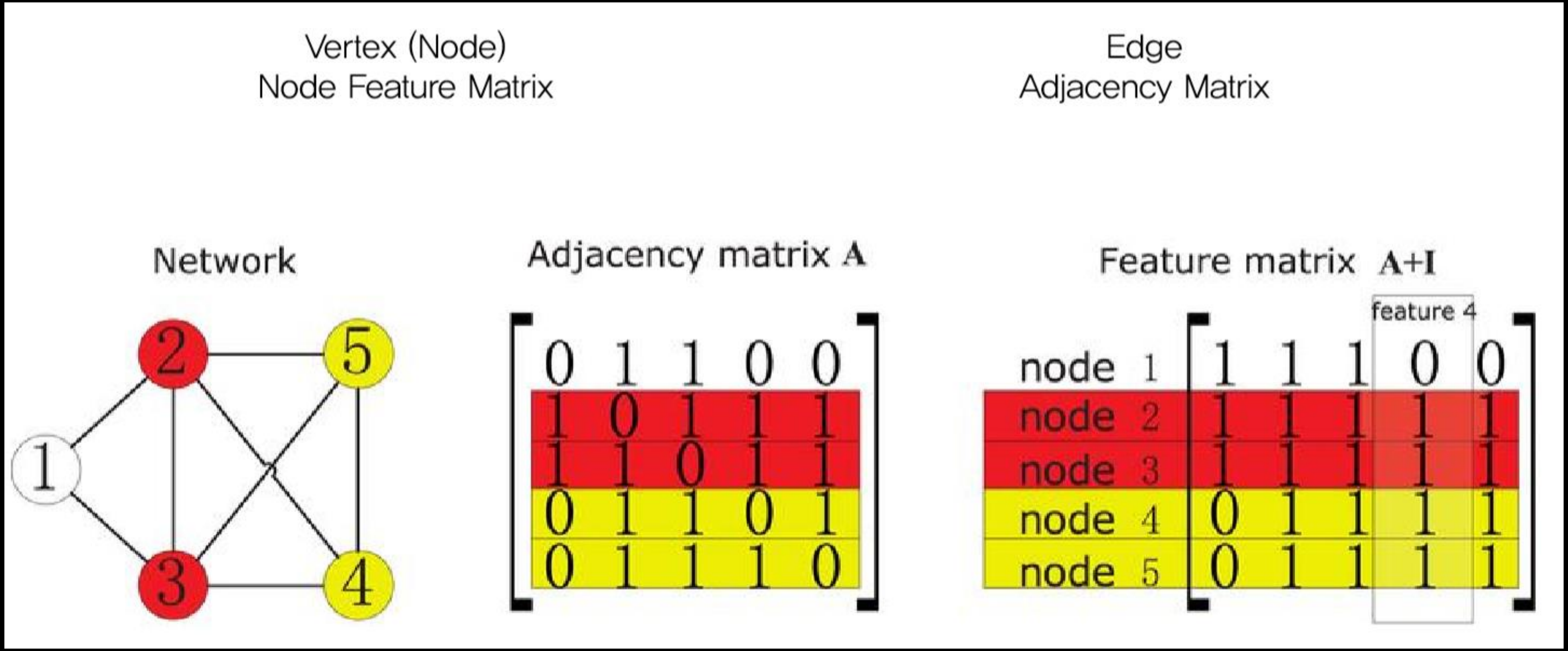




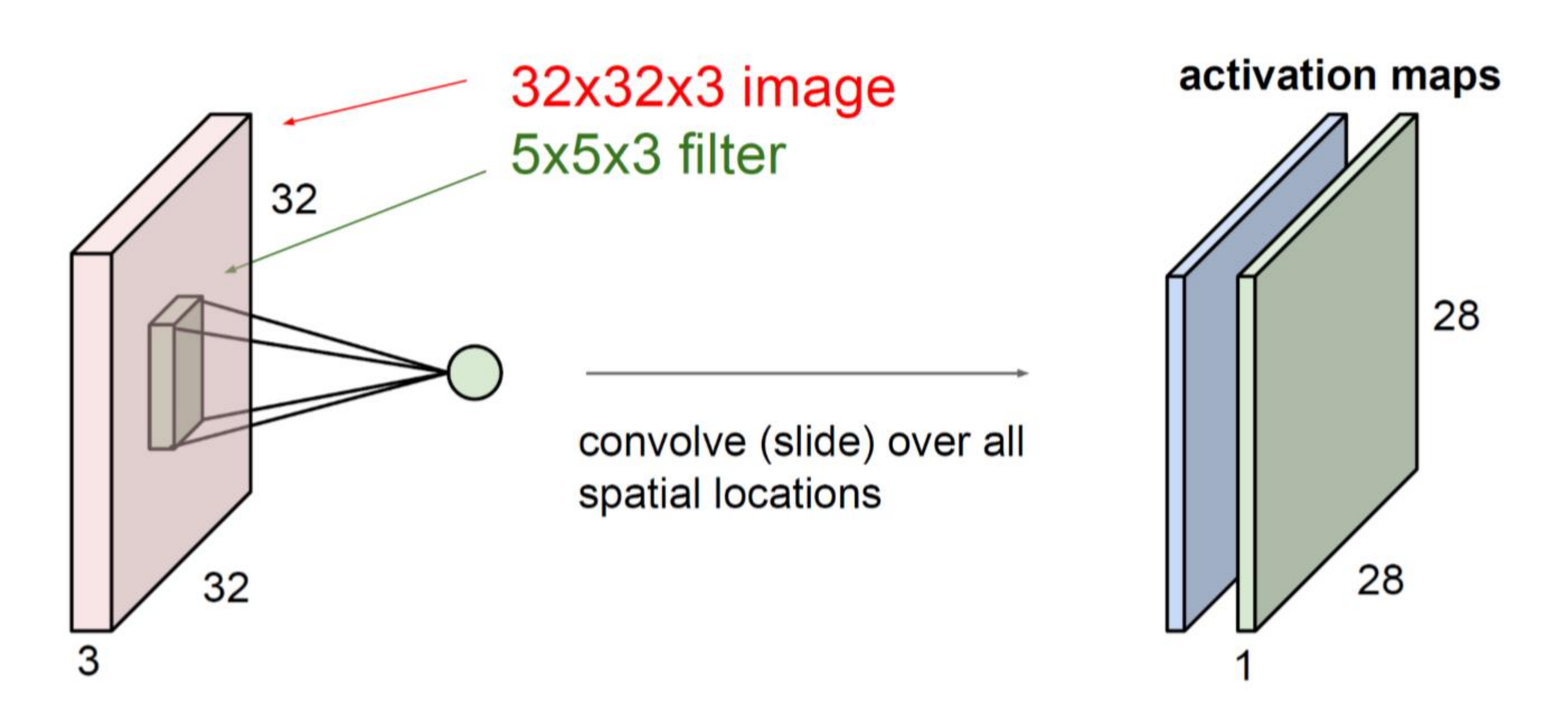


1. Paper search

Graph Convolutional Network



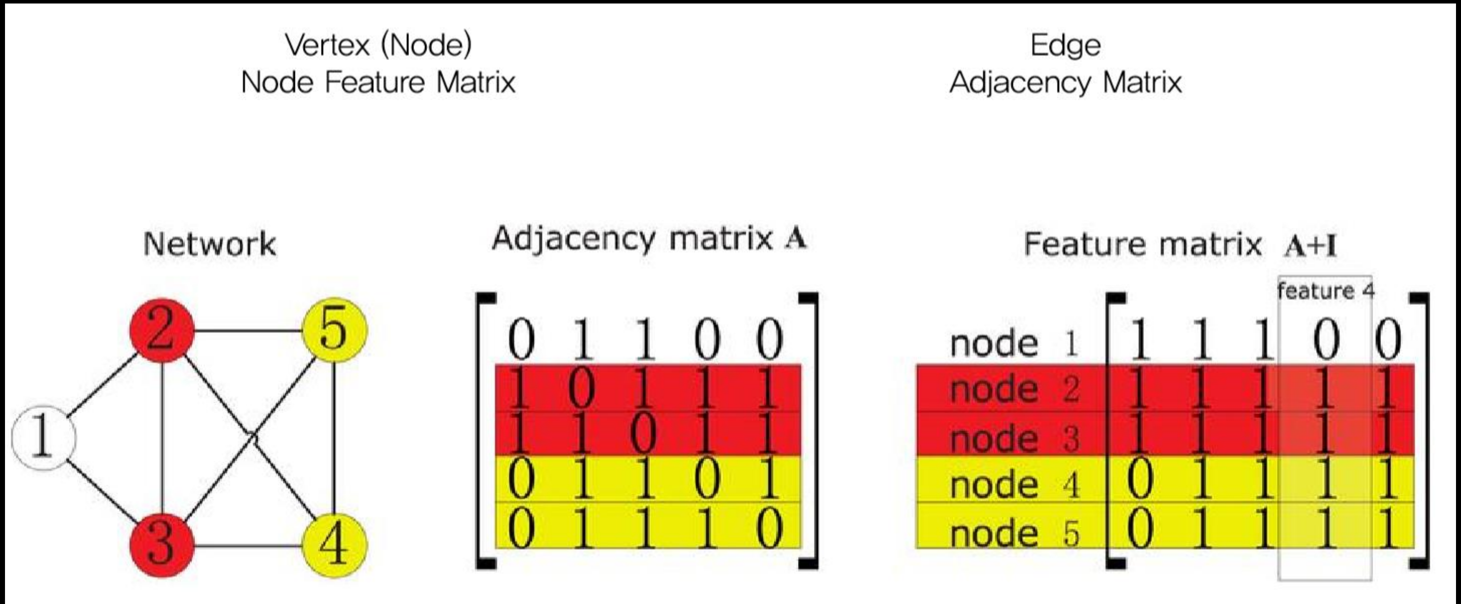
Graph **Convolutional** Network



Graph Convolutional Network

Propagation Rule

$$H^{(l+1)} = \sigma\left(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{(l)}W^{(l)}\right)$$

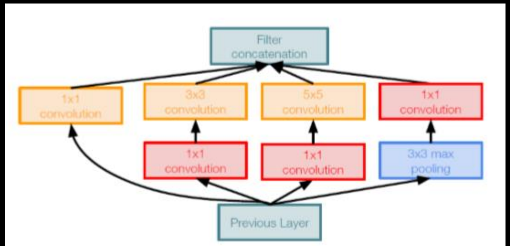


Graph Convolutional Network

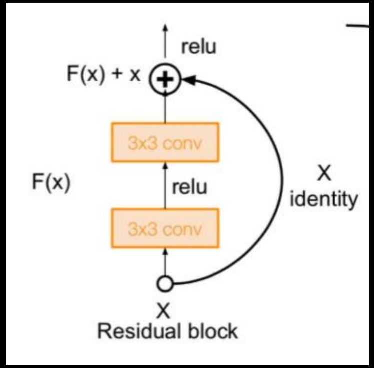
Propagation Rule

$$H^{(l+1)} = \sigma\left(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{(l)}W^{(l)}\right)$$

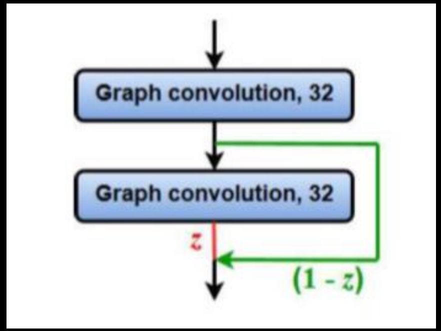
Inception



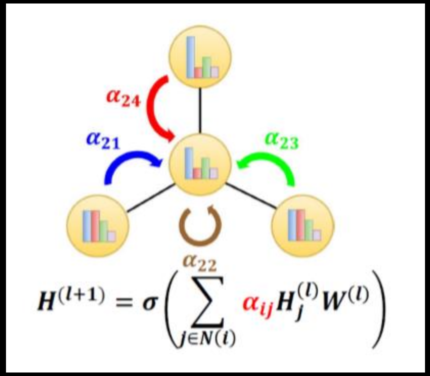
Skip connection



Gated skip connection



Attention



Fast Interactive Object Annotation with Curve-GCN

Huan Ling[1,2*] Jun Gao[1,2*] Amlan Kar[1,2] Wenzheng Chen [1,2] Sanja Fidler[1,2,3]

[1]University of Toronto, [2]Vector Institute, [3]NVIDIA

Fast Interactive Object Annotation with Curve-GCN
Model Architecture

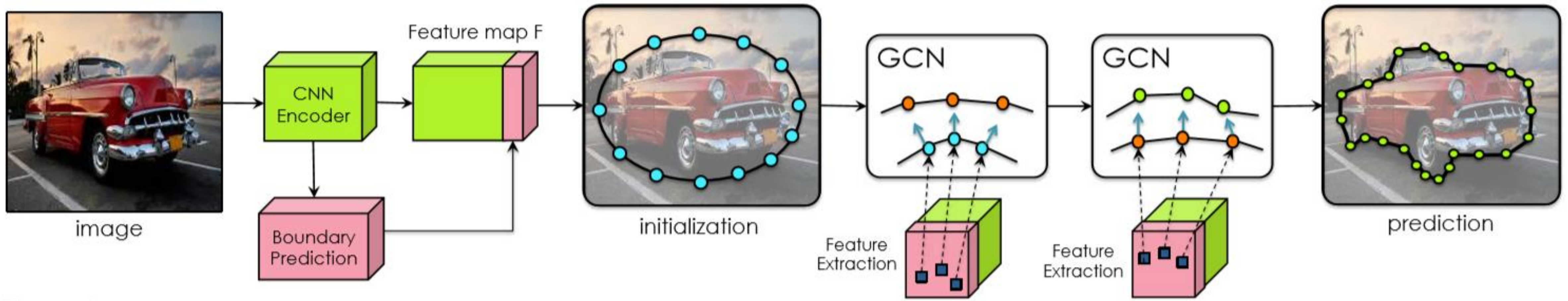


Figure 2: **Curve-GCN**: We initialize N control points (that form a closed curve) along a circle centered in the image crop with a diameter of 70% of image height. We form a graph and propagate messages via a Graph Convolutional Network (GCN) to predict a location shift for each node. This is done iteratively (3 times in our work). At each iteration we extract a feature vector for each node from the CNN's features F, using a bilinear interpolation kernel.

Inference → train → human-in-the-loop

[Previous Work]

- Manually tracing object boundaries is a laborious process → interactive segmentation
- DEXTR, pixel-wise approach, the worst case scenario still requires many clicks.
- Polygon-RNN, frames human-in-the-loop annotation as a recurrent process, which make model limit scalability to more complex shapes, longer time.

[Ours]

- In this paper, we frame interactive object annotation as a regression problem, using Graph Convolutional Network. → few clicks even in worst case, faster than Polygon-RNN
- We outperform Polygon-RNN++ and PSP-Deeplab/DEXTR in both automatic and interactive settings on the challenging Cityscapes dataset.
- Also, We outperform in cross-domain, general scene, aerial, and medical imagery.

2. Introduction

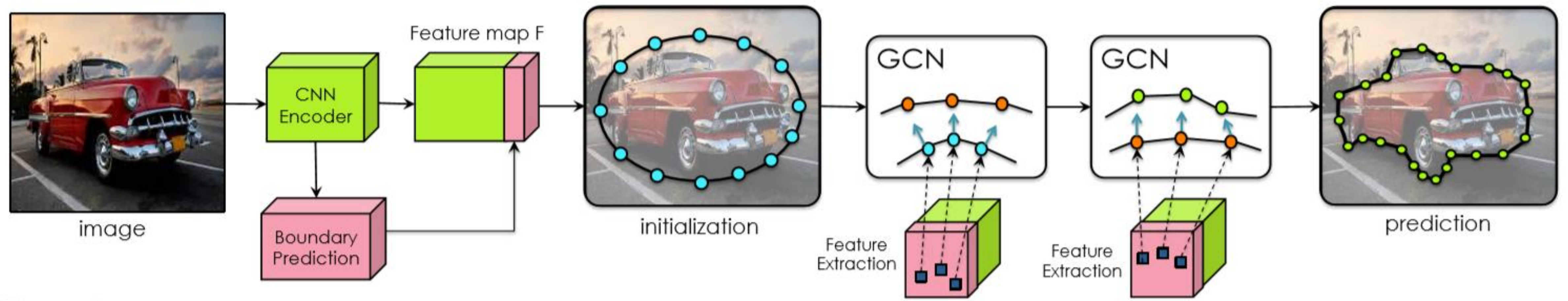


Figure 2: **Curve-GCN**: We initialize N control points (that form a closed curve) along a circle centered in the image crop with a diameter of 70% of image height. We form a graph and propagate messages via a Graph Convolutional Network (GCN) to predict a location shift for each node. This is done iteratively (3 times in our work). At each iteration we extract a feature vector for each node from the CNN's features F, using a bilinear interpolation kernel.

Notation

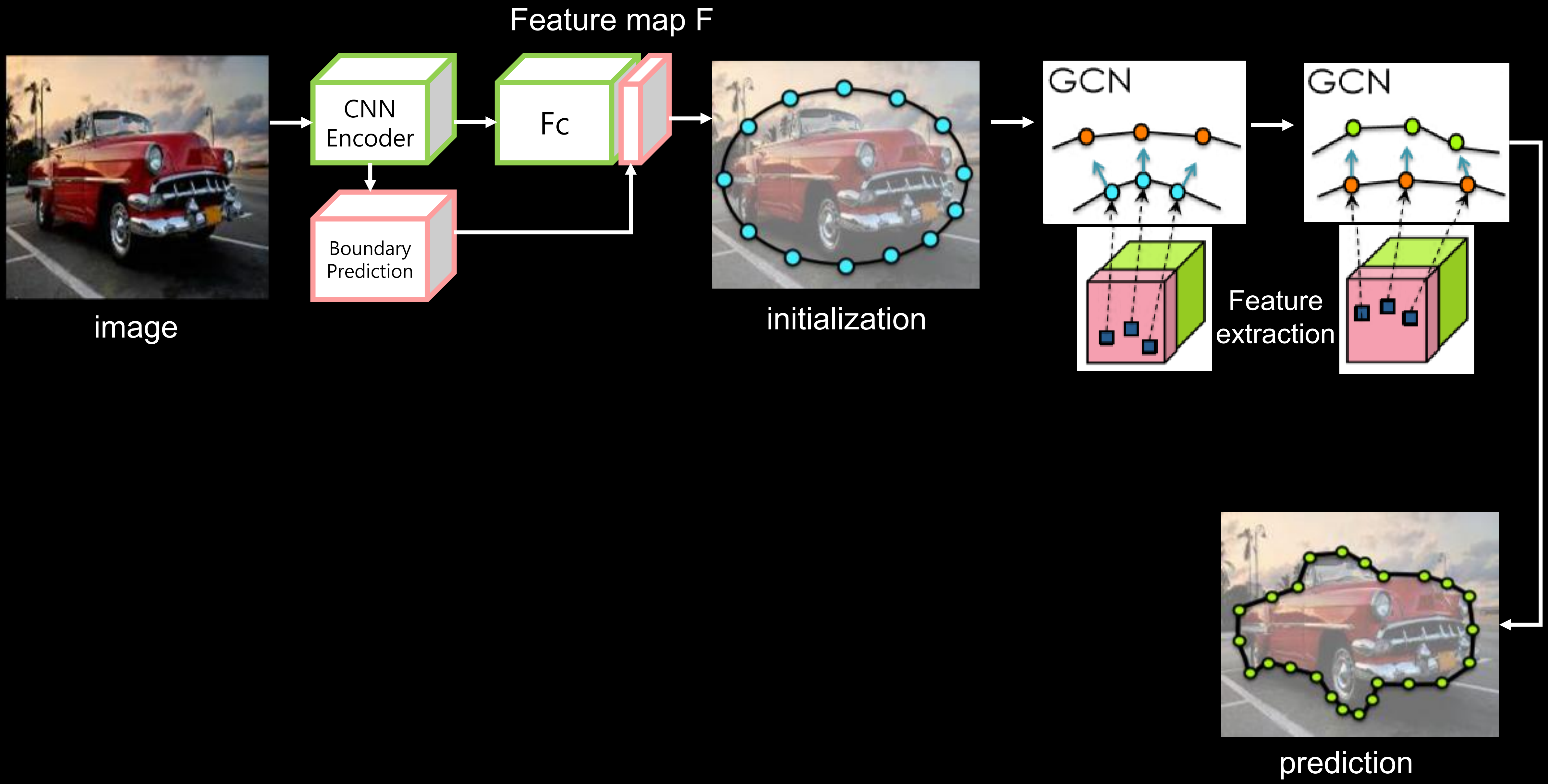
$cp_i = [x_i, y_i]^T$

$V = \{cp_0, \ldots, cp_{N-1}\}$ $, vertex, N\ control\ points, form\ a\ cricle.$

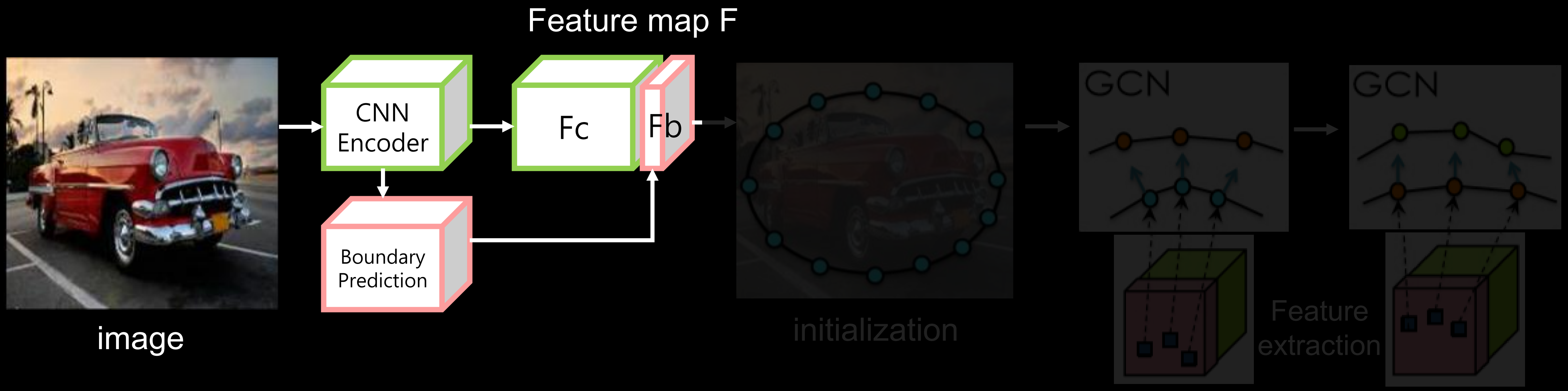
$G = (V, E)$

$E\ is\ edge\ by\ connecting\ each\ vertex,$ with its four neighboring vertices

▌3. Object Annotation via Curve-GCN

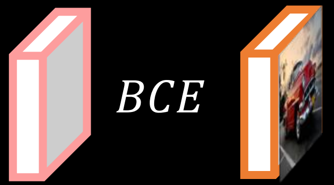


3. Object Annotation via Curve-GCN



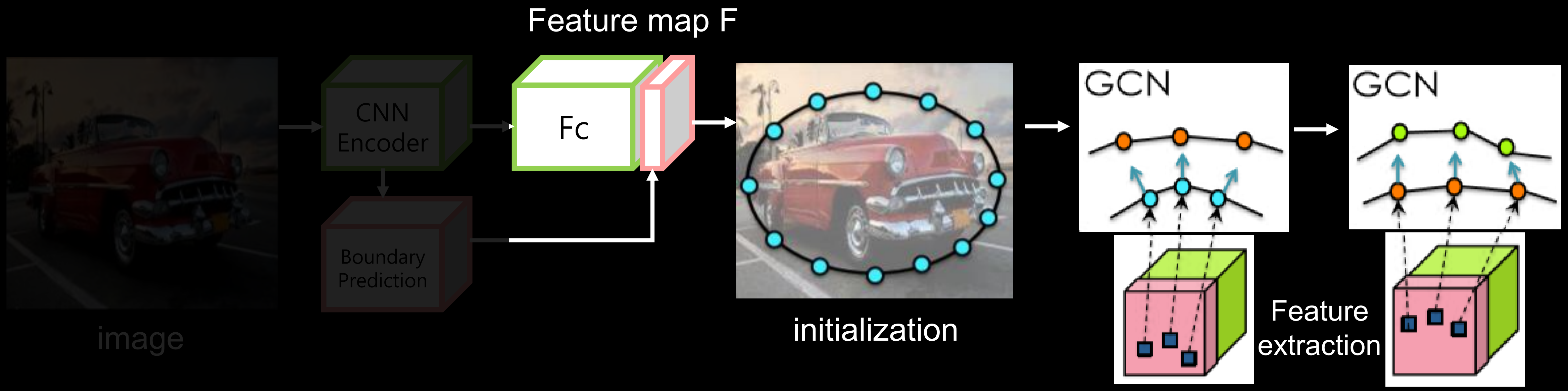
F_c : $image\ feature\ obtained\ from\ Image$, 28 x 28 grid

F_b : $edge, vertex\ branch\ obtained\ from\ Image$, 28 x 28 grid



$*\ BCE$: $binary\ cross\ entropy\ loss$

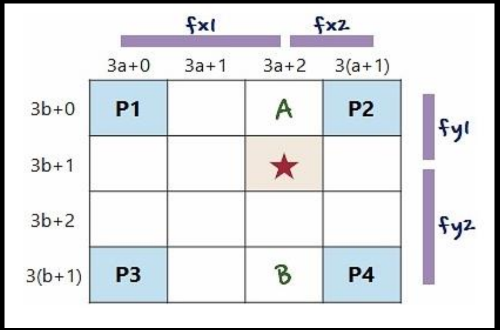
3. Object Annotation via Curve-GCN



GCN Input

*nodes : initialized by static central position

F : $f_i^0 = concat\{F(x_i, y_i), x_i, y_i\}$. $F(x_i, y_i)$ *is computed using* ***bilinear interpolation***



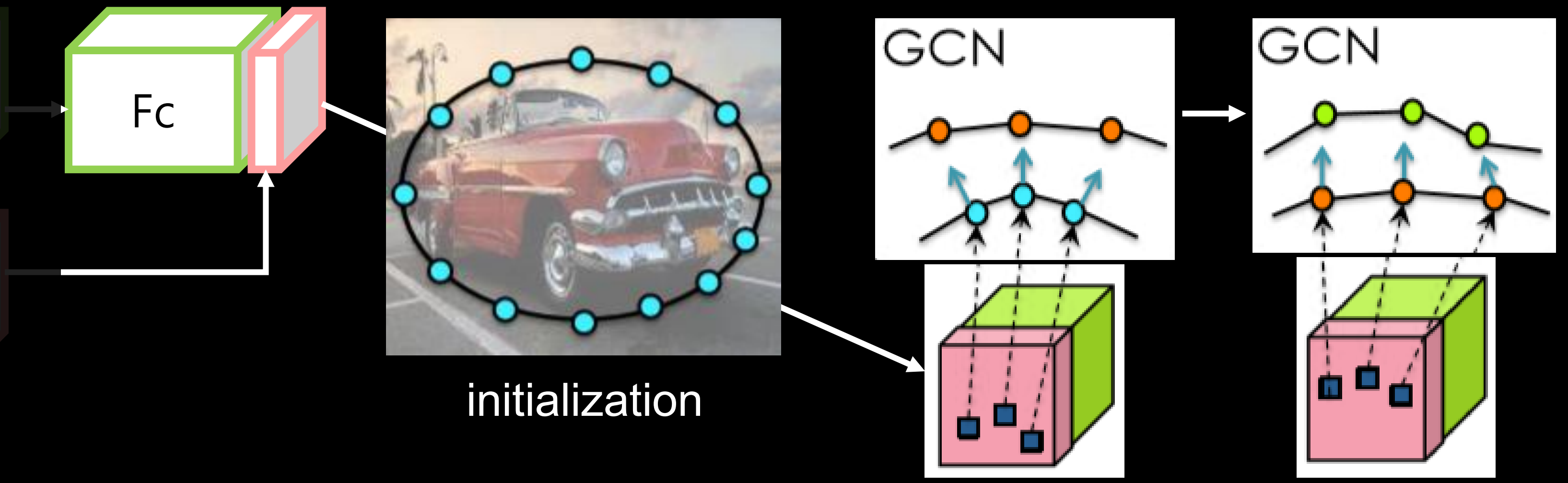
$$f_i^{l+1} = w_0^l f_i^l + \sum_{cp_j \in N(cp_i)} w_1^l f_j^l$$

$$r_i^l = ReLU\Big(w_0^l f_i^l + \sum_{\mathbf{cp}_j \in \mathcal{N}(\mathbf{cp}_i)} w_1^l f_j^l\Big) \quad (2)$$

$$r_i^{l+1} = \tilde{w}_0^l r_i^l + \sum_{\mathbf{cp}_j \in \mathcal{N}(\mathbf{cp}_i)} \tilde{w}_1^l r_j^l \quad (3)$$

$$f_i^{l+1} = ReLU(r_i^{l+1} + f_i^l), \quad (4)$$

3. Object Annotation via Curve-GCN



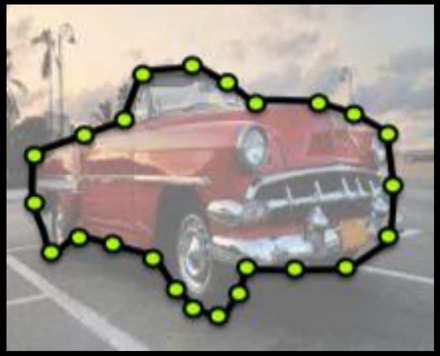
initialization

Feature extraction

single fully connected layer

predict a relative location shift, $(\Delta x_i, \Delta y_i)$

$(x_{i+1}, y_{i+1}) = (x_i + \Delta x_i,\ y_i + \Delta y_i)$



prediction

Another GCN
Coarse-to-fine

GCN Input

$F:\ f_i^0 = concat\{F(x_i, y_i), x_i, y_i\}$. $F(x_i, y_i)$ *is computed using* ***bilinear interpolation***

$f_i^{l+1} = w_0^l f_i^l + \sum_{cp_j \in N(cp_i)} w_1^l f_j^l$

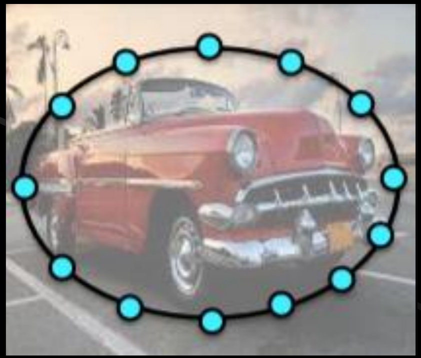
Utilize a Graph-ResNet

$$r_i^l = ReLU\Big(w_0^l f_i^l + \sum_{\mathbf{cp}_j \in \mathcal{N}(\mathbf{cp}_i)} w_1^l f_j^l\Big) \quad (2)$$

$$r_i^{l+1} = \tilde{w}_0^l r_i^l + \sum_{\mathbf{cp}_j \in \mathcal{N}(\mathbf{cp}_i)} \tilde{w}_1^l r_j^l \quad (3)$$

$$f_i^{l+1} = ReLU(r_i^{l+1} + f_i^l), \quad (4)$$

3. Object Annotation via Curve-GCN



initialization

Spline parametrization

- Choice of spline is important for the annotator's experience.
- Cubic Bezier, uniform B-Spline : don't' lie on the curve!
- we use the centripetal Catmull-Rom spline (CRS), which has control points along the curve.

For a curve segment $\mathbf{S}_i$ defined by control points $\mathbf{cp}_{i-1}$, $\mathbf{cp}_i$, $\mathbf{cp}_{i+1}$, $\mathbf{cp}_{i+2}$ and a knot sequence $t_{i-1}, t_i, t_{i+1}, t_{i+2}$, the CRS is interpolated by:

$$\mathbf{S}_i = \frac{t_{i+1}-t}{t_{i+1}-t_i} L_{012} + \frac{t-t_i}{t_{i+1}-t_i} L_{123} \tag{5}$$

where

$$L_{012} = \frac{t_{i+1}-t}{t_{i+1}-t_{i-1}} L_{01} + \frac{t-t_{i-1}}{t_{i+1}-t_{i-1}} L_{12} \tag{6}$$

$$L_{123} = \frac{t_{i+2}-t}{t_{i+2}-t_i} L_{12} + \frac{t-t_i}{t_{i+2}-t_i} L_{23} \tag{7}$$

$$L_{01} = \frac{t_i-t}{t_i-t_{i-1}} \mathbf{cp}_{i-1} + \frac{t-t_{i-1}}{t_i-t_{i-1}} \mathbf{cp}_i \tag{8}$$

$$L_{12} = \frac{t_{i+1}-t}{t_{i+1}-t_i} \mathbf{cp}_i + \frac{t-t_i}{t_{i+1}-t_i} \mathbf{cp}_{i+1} \tag{9}$$

$$L_{23} = \frac{t_{i+2}-t}{t_{i+2}-t_{i+1}} \mathbf{cp}_{i+1} + \frac{t-t_{i+1}}{t_{i+2}-t_{i+1}} \mathbf{cp}_{i+2}, \tag{10}$$

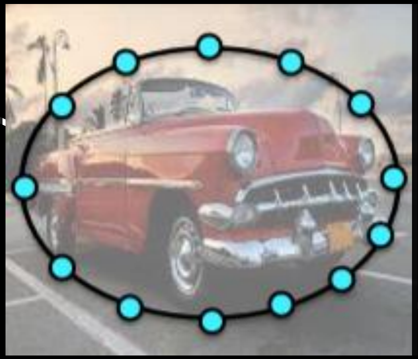
and $t_{i+1} = ||\mathbf{cp}_{i+1} - \mathbf{cp}_i||_2^\alpha + t_i$, $t_0 = 0$. Here, α ranges from 0 to 1. We choose $\alpha = 0.5$ following [32], which in theory produces splines without cusps or self-intersections [35]. To make the spline a closed and C^1-continuous curve, we add three additional control points:

$$\mathbf{cp}_N = \mathbf{cp}_0 \tag{11}$$

$$\mathbf{cp}_{N+1} = \mathbf{cp}_0 + \frac{||\mathbf{cp}_{N-1}-\mathbf{cp}_0||_2}{||\mathbf{cp}_1-\mathbf{cp}_0||_2}(\mathbf{cp}_1 - \mathbf{cp}_0) \tag{12}$$

$$\mathbf{cp}_{-1} = \mathbf{cp}_0 + \frac{||\mathbf{cp}_1-\mathbf{cp}_0||_2}{||\mathbf{cp}_{N-1}-\mathbf{cp}_0||_2}(\mathbf{cp}_{N-1} - \mathbf{cp}_0). \tag{13}$$

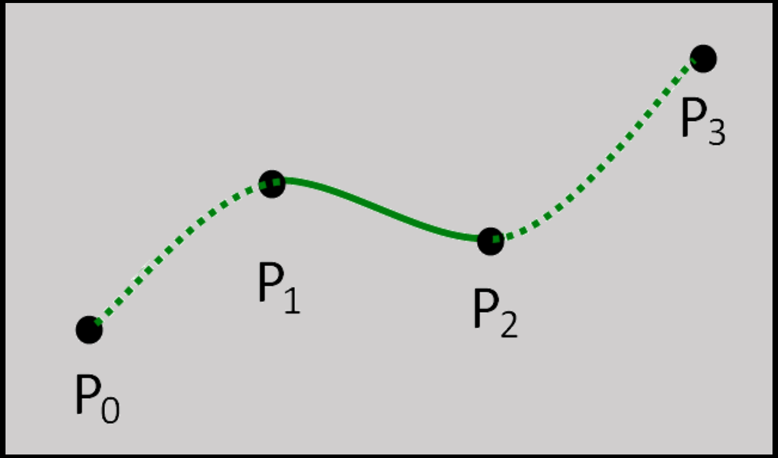
3. Object Annotation via Curve-GCN

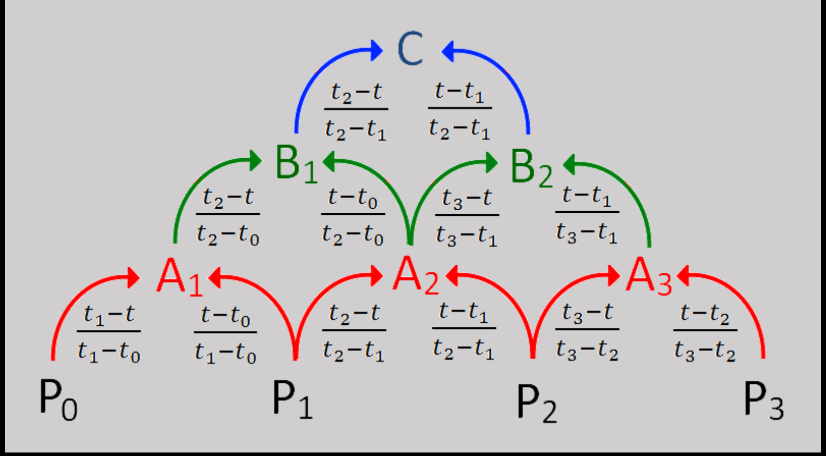


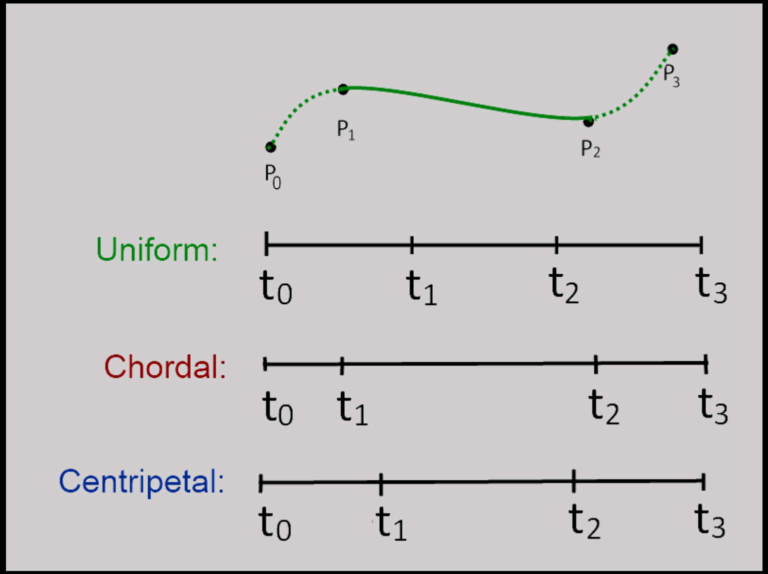
initialization

Spline parametrization

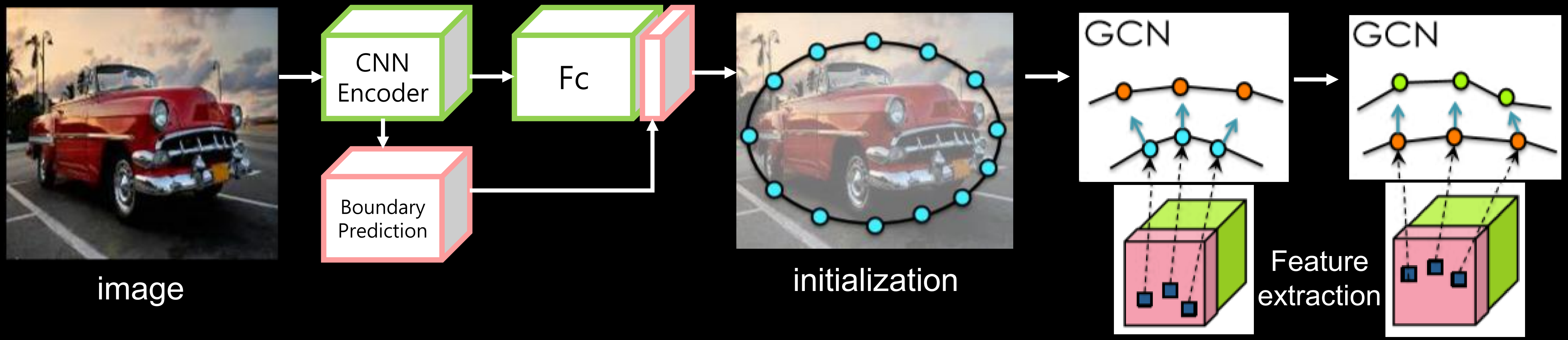
- Choice of spline is important for the annotator's experience.
- we use the centripetal Catmull-Rom spline (CRS), which has control points along the curve.







3. Object Annotation via Curve-GCN



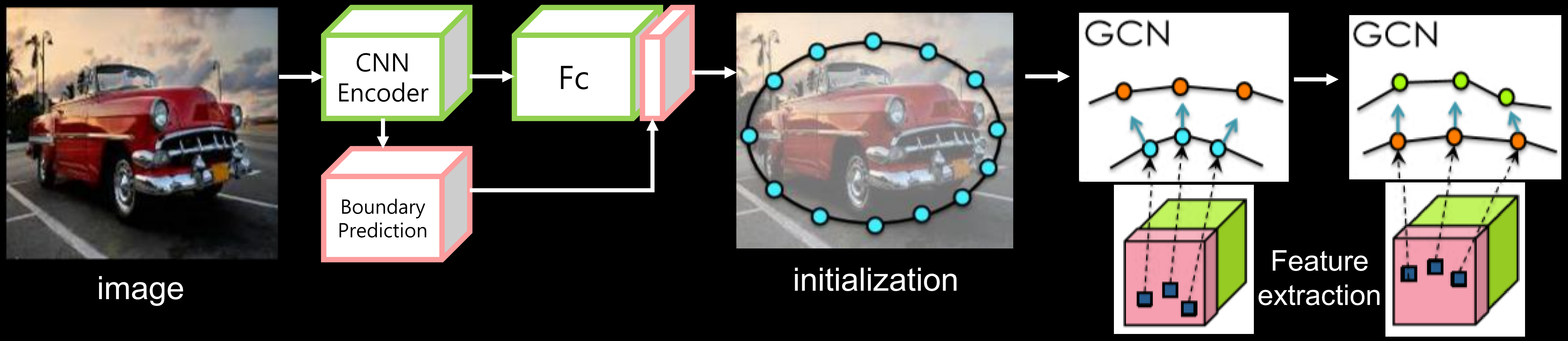
3.2.1. Point Matching Loss

$$p = \{p_0, p_1, \ldots, p_{K-1}\}, \textit{prediction point set}$$
$$p' = \{p'_0, p'_1, \ldots, p'_{K-1}\}, \textit{ground truth point set}$$

$$L_{\text{match}}(\mathbf{p}, \mathbf{p}') = \min_{j \in [0 \cdots, K-1]} \sum_{i=0}^{K-1} \left\| p_i - p'_{(j+i)\%K} \right\|_1 \quad (14)$$

- This loss explicitly ensures an order in the vertices in the loss computation.
- Sampling equal sized point sets. We sample K points along the spline by uniformly ranging t from t_i to t_{i+1}. Also uniformly sample the same number of points along the edges of the ground-truth polygon.
- We use K = 1280.
- More points, higher computational cost / fewer points, less accurate.

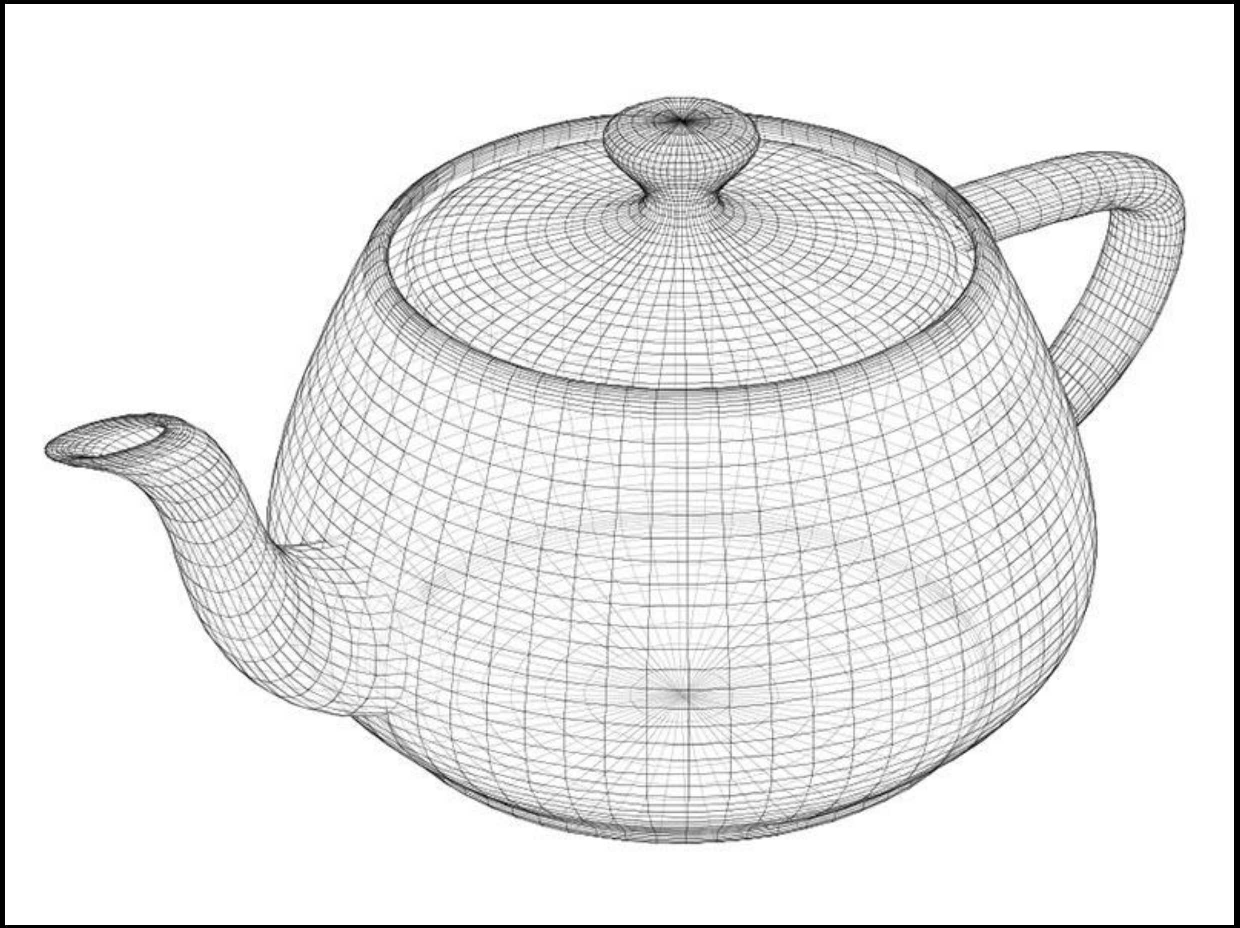
3. Object Annotation via Curve-GCN



3.2.2. Differentiable Accuracy Loss

- A differentiable rendering loss
- This has been used previously to optimize 3D mesh vertices to render correctly onto a 2D image
- What is the rendering process??
- How can it be differentiable??

3.2.2. Differentiable Accuracy Loss – Rendering process



3.2.2. Differentiable Accuracy Loss – Rendering process

$M(\theta) = R(p(\theta))$, where p is the sampled point sequence, R is rendering function and M is the corresponding mask rendered from p.

$$L_{render}(\theta) = \left\| M(\theta) - M_{gt} \right\|_1$$

L_{render} is exactly the pixel-wise accuracy of the predicted mask $M(\theta)$ with respect to the ground truth M_{gt}.

Forward Pass

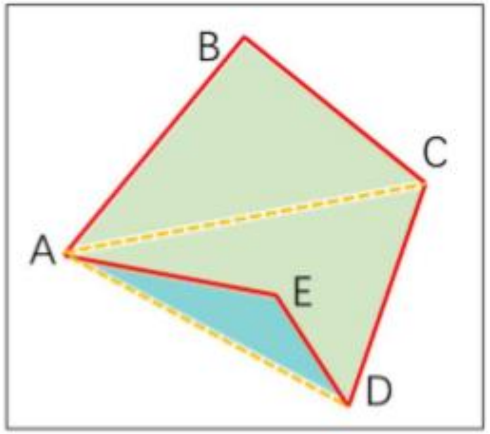


Figure 3: We decompose the polygon ABCDE into 3 triangle fans ABC, ACD and ADE, and render them separately. We assign positive value for clock wise triangles (ABC, ACD) and negative value for the others (ADE). Finally we sum over all the renderings. The sum retains only the interior of the polygon.

Backward Pass

$$\frac{\partial M_j}{\partial \mathbf{f}_j} = \frac{R(\mathbf{f}_j + \Delta t) - R(\mathbf{f}_j)}{\Delta t},$$

$$\frac{\partial M_j}{\partial \mathbf{f}_{j,k}} = \sum_i w_k^i \frac{\partial M_j^i}{\partial \mathbf{f}_j} \quad k \in [0, 1, 2]$$

3. Object Annotation via Curve-GCN

3.2.2. Differentiable Accuracy Loss – Rendering process

Forward Pass

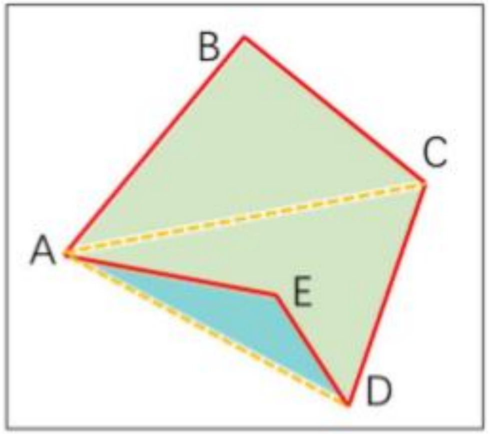


Figure 3: We decompose the polygon ABCDE into 3 triangle fans ABC, ACD and ADE, and render them separately. We assign positive value for clock wise triangles (ABC, ACD) and negative value for the others (ADE). Finally we sum over all the renderings. The sum retains only the interior of the polygon.

- Using OpenGL
- Decompose the shape into triangle fans f_j
- Sum over all the triangles to get the final mask

3.2.2. Differentiable Accuracy Loss – Rendering process

Backward Pass

- **Rendering process** is non-diff. due to rasterization, which truncates all float values to integers.
- Compute its gradient with first order Taylor expansion
- Analyze each triangle fan separately.
- A small shift of the fan f_j , calculate →
- $\Delta \boldsymbol{t}$ to be a 1 pixel shift, which alleviates the need to render twice
- M_j : mask corresponding to the fan f_j
- Calculate gradients by subtracting neighboring pixels.

$$\frac{\partial M_j}{\partial \mathbf{f}_j} = \frac{R(\mathbf{f}_j + \Delta t) - R(\mathbf{f}_j)}{\Delta t},$$

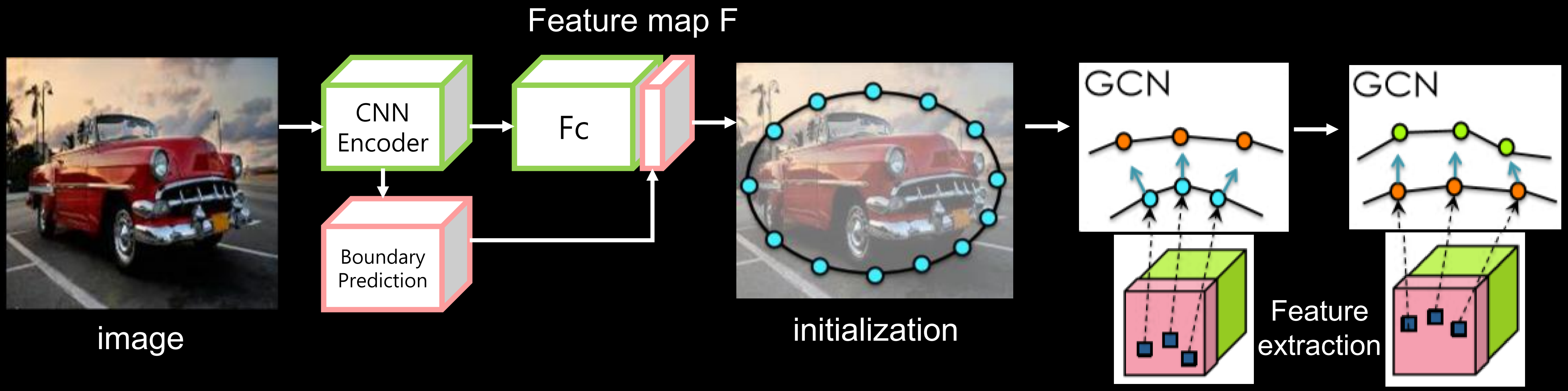
- Next, pass the above gradient to its three vertices $f_{j,0}$, $f_{j,1}$, $f_{j,2}$ →

$$\frac{\partial M_j}{\partial \mathbf{f}_{j,k}} = \sum_i w_k^i \frac{\partial M_j^i}{\partial \mathbf{f}_j} \qquad k \in [0, 1, 2]$$

where we sum over all pixels i. For the i-th pixel M_j^i in the rendered image M_j, we compute its weight w_0^i, w_1^i and w_2^i with respect to the vertices of the face $\mathbf{f}_j$ as its barycentric coordinates. For more details, please refer to [22].

[22] Opendr: An approximate differentiable renderer. ECCV 2014. / Neural 3D Mesh Renderer CVPR 2019

3. Object Annotation via Curve-GCN



3.3. Annotator in the Loop

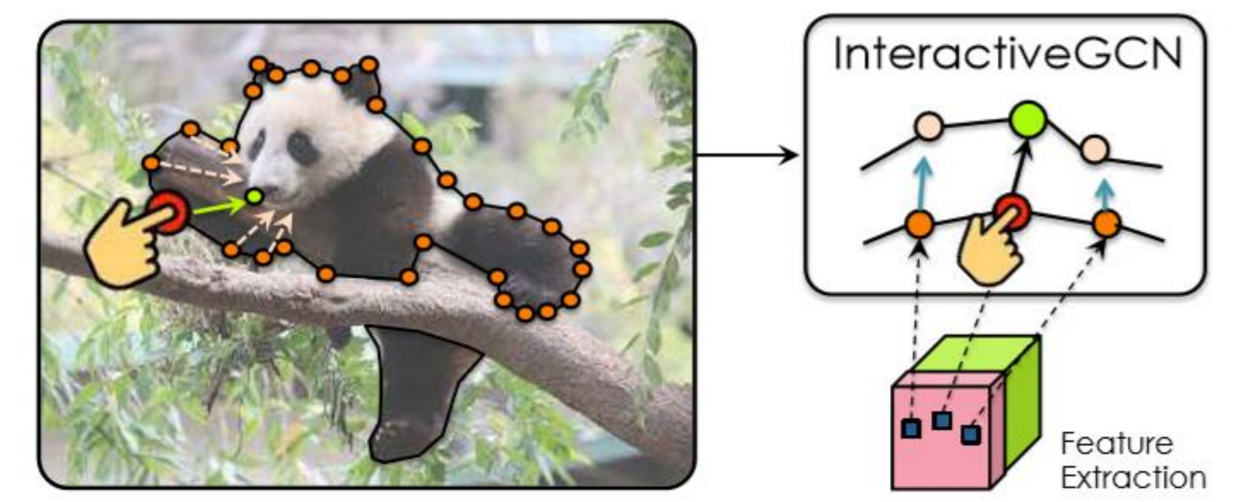


Figure 4: **Human-in-the-Loop**: An annotator can choose any wrong control point and move it onto the boundary. Only its immediate neighbors ($k = 2$ in our experiments) will be re-predicted based on this interaction.

3. Object Annotation via Curve-GCN

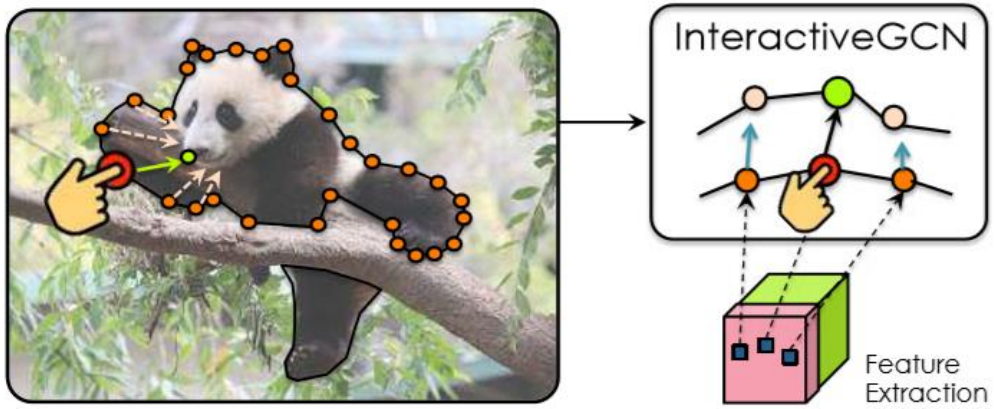
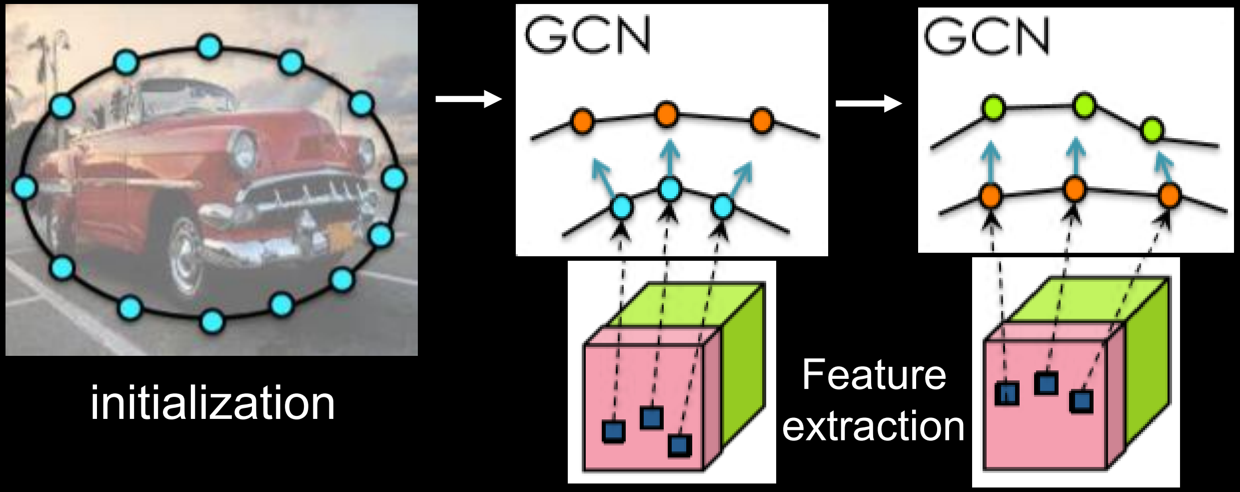


Figure 4: **Human-in-the-Loop**: An annotator can choose any wrong control point and move it onto the boundary. Only its immediate neighbors ($k = 2$ in our experiments) will be re-predicted based on this interaction.



3.3. Annotator in the Loop

Interactive GCN Input

$F: f_i^0 = concat\{F(x_i, y_i), x_i, y_i, \Delta x_i, \Delta y_i\}$. $where\ (\Delta x_i, \Delta y_i)\ is\ the\ shift\ given\ by\ the\ annotator.$

$cp_{(i-k)\%N}, \ldots, cp_{(i-1)\%N}, cp_{(i+1)\%N}, \ldots, cp_{(i+k)\%N}$ to be predicted, k=2, the annotator could vary k at test time.

assume that the annotator always chooses to correct the worst predicted point.

then find the point with the **largest manhattan distance** to the corresponding GT point.

then iterate between the annotator choosing the worst prediction, and training to correct its neighbors

3.3. Annotator in the Loop

Algorithm 1 Learning to Incorporate Human-in-the-Loop

```
while not converged do
    (rawImage, gtCurve) = Sample(Dataset)
    (predCurve, F) = Predict(rawImage)
    data = []
    for i in range(c) do
        corrPoint = Annotator(predictedCurve)
        data += (predCurve, corrPoint, gtCurve, F)
        predCurve = InteractiveGCN(predCurve, corrPoint)
                                        ▷ Do not stop gradients
    TrainInteractiveGCN(data)
```

In every iteration, the GCN first predicts the correction for the neighbors based on the last annotator's correction, and then the annotator corrects the next worst point.

c denotes the number of iterations

-. Training Detail

1. first train our model via the matching loss
2. followed by fine-tuning with the differentiable accuracy loss
3. For speed issues we use the matching loss to train the InteractiveGCN

On Cityscapes dataset

1. the matching loss operates on single polygons,

we train our model on single component instances first.

2. fine-tune with the differentiable accuracy loss on all instances.

Automatic Mode

Model	Bicycle	Bus	Person	Train	Truck	Motorcycle	Car	Rider	Mean
Polygon-RNN++	57.38	75.99	68.45	59.65	76.31	58.26	75.68	65.65	67.17
Polygon-RNN++ (with BS)	63.06	81.38	72.41	64.28	78.90	62.01	79.08	69.95	71.38
PSP-DeepLab	67.18	83.81	72.62	**68.76**	**80.48**	**65.94**	80.45	70.00	**73.66**
Polygon-GCN (MLoss)	63.68	81.42	72.25	61.45	79.88	60.86	79.84	70.17	71.19
+ DiffAcc	66.55	85.01	72.94	60.99	79.78	63.87	81.09	71.00	72.66
Spline-GCN (MLoss)	64.75	81.71	72.53	65.87	79.14	62.00	80.16	70.57	72.09
+ DiffAcc	**67.36**	**85.43**	**73.72**	64.40	80.22	64.86	**81.88**	**71.73**	**73.70**

Table 1: **Automatic Mode on Cityscapes.** We compare our Polygon and Spline-GCN to Polygon-RNN++ and PSP-DeepLab. Here, *BS* indicates that the model uses beam search, which we do not employ.

Model	mIOU	F at 1px	F at 2px
Polyrnn++ (BS)	71.38	46.57	62.26
PSP-DeepLab	73.66	47.10	62.82
Spline-GCN	**73.70**	**47.72**	**63.64**
DEXTR	79.40	55.38	69.84
Spline-GCN-EXTR	**79.88**	**57.56**	**71.89**

Table 2: **Different Metrics**. We report IoU & F boundary score. We favorably cross-validate PSP-DeepLab and DEXTR *for each metric* on val. *Spline-GCN-EXTR* uses extreme points as additional input as in DEXTR.

Model	Spline	Polygon
GCN	68.55	67.79
+ Iterative Inference	70.00	70.78
+ Boundary Pred.	72.09	71.19
+ DiffAcc	**73.70**	72.66

Table 3: **Ablation study** on Cityscapes. We use 3 steps when performing iterative inference. *Boundary Pred* adds the boundary prediction branch to our CNN.

Model	Time(ms)
Polygon-RNN++	298.0
Polygon-RNN++ (Corr.)	270.0
PSP-Deeplab	71.3
Polygon-GCN	28.7
Spline-GCN	29.3
Polygon-GCN (Corr.)	2.0
Spline-GCN (Corr.)	2.6

Table 4: **Avg. Inference Time** per object. We are $10\times$ faster than Polygon-RNN++ in forward pass, and $100\times$ for every human correction.

4. Experimental Results

Interactive Mode

Model	Bicycle	Bus	Person	Train	Truck	Mcycle	Car	Rider	Mean	# clicks
Spline-GCN-BOX	69.53	84.40	76.33	69.05	85.08	68.75	83.80	73.38	76.29	2
PSP-DEXTR	74.42	87.30	79.30	73.51	85.42	73.69	85.57	76.24	79.40	4
Spline-GCN-EXTR	**75.09**	87.40	**79.88**	72.78	**86.76**	73.93	**86.13**	**77.12**	79.88	4
Spline-GCN-MBOX	70.45	88.02	75.87	76.35	82.73	70.76	83.32	73.49	77.62	2.4
+ One click	73.28	**89.18**	78.45	**79.89**	85.02	**74.33**	85.15	76.22	**80.19**	3.6

Table 5: **Additional Human Input.** We follow DEXTR [23] and provide a budget of 4 clicks to the models. Please see text for details.

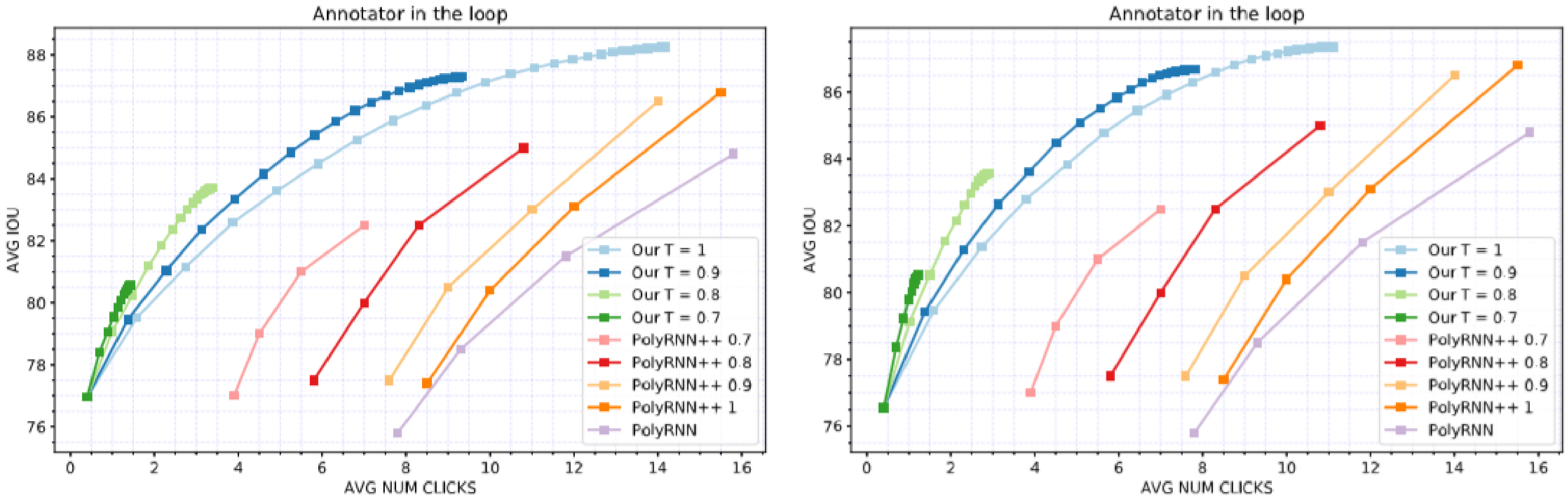


Figure 8: **Interactive Mode on Cityscapes**: (**left**) 40 control points, (**right**) 20 control points.

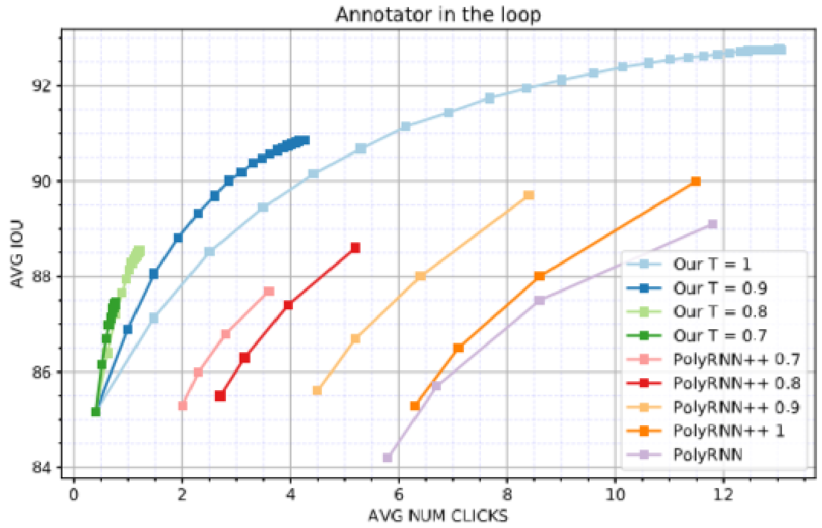


Figure 9: **Inter. Mode on KITTI**: 40 cps

4. Experimental Results

Cross Domain

Model	KITTI	ADE	Rooftop	Card.MR	ssTEM
Square Box (Perfect)	-	69.35	62.11	79.11	66.53
Ellipse (Perfect)	-	69.53	66.82	92.44	71.32
Polygon-RNN++ (BS)	83.14	71.82	65.67	80.63	53.12
PSP-DeepLab	83.35	72.70	57.91	74.11	47.65
Spline-GCN	**84.09**	**72.94**	**68.33**	78.54	58.46
+ finetune	**84.81**	77.35	**78.21**	**91.33**	-
Polygon-GCN	83.66	72.31	66.78	**81.55**	**60.91**
+ finetune	84.71	**77.41**	75.56	90.91	-

Table 6: **Automatic Mode on Cross-Domain**. We outperform PSP-DeepLab out-of-the-box. Fine-tuning on 10% is effective.

Table 7: **Automatic Mode for Cross-Domain**. (**top**) Out-of-the-box output of Cityscapes-trained models, (**bottom**) fine-tuned with 10% of data from new domain.

https://www.youtube.com/watch?v=ycD2BtO-QzU

Fast Interactive Object Annotation with Curve-GCN
Model Architecture

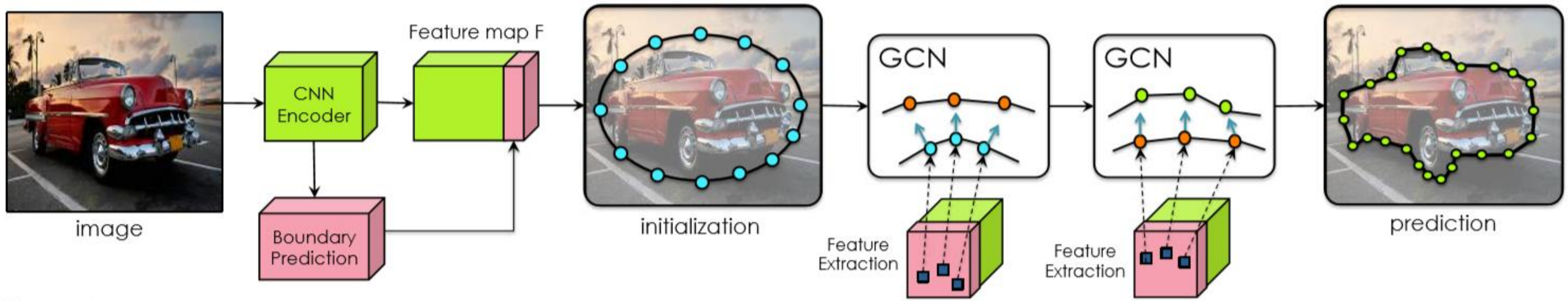


Figure 2: **Curve-GCN**: We initialize N control points (that form a closed curve) along a circle centered in the image crop with a diameter of 70% of image height. We form a graph and propagate messages via a Graph Convolutional Network (GCN) to predict a location shift for each node. This is done iteratively (3 times in our work). At each iteration we extract a feature vector for each node from the CNN's features F, using a bilinear interpolation kernel.

Thank you

~~(not end game)~~