Interactive segmentation

- 1. Deep Extreme Cut: From Extreme Points to Object Segmentation
- 2. Interactive Image Segmentation with Latent Diversity
- 3. Fast Interactive Object Annotation with Curve-GCN
- 4. PhraseClick: Toward Achieving Flexible Interactive Segmentation by Phrase and Click
- 5. Deep Interactive Thin Object Selection
- 6. Efficient Full Image Interactive Segmentation by Leveraging Within-image Appearance Similarity

Deep Extreme Cut: From Extreme Points to Object Segmentation

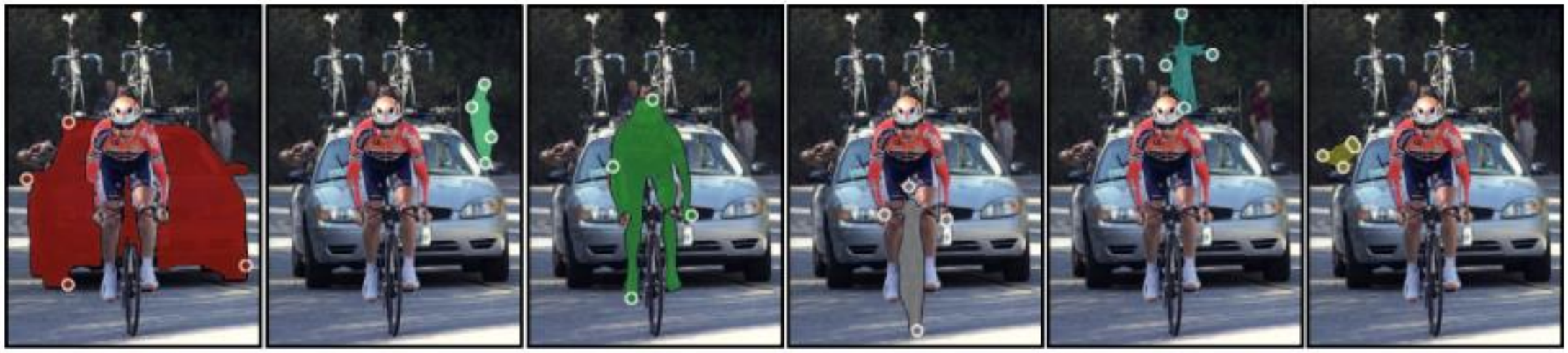


Figure 1. **Example results of DEXTR**: The user provides the extreme clicks for an object, and the CNN produces the segmented masks.

- *This paper use of extreme points in an object (left-most, right-most, top, bottom pixels) as input to obtain precise object segmentation for images and videos.*

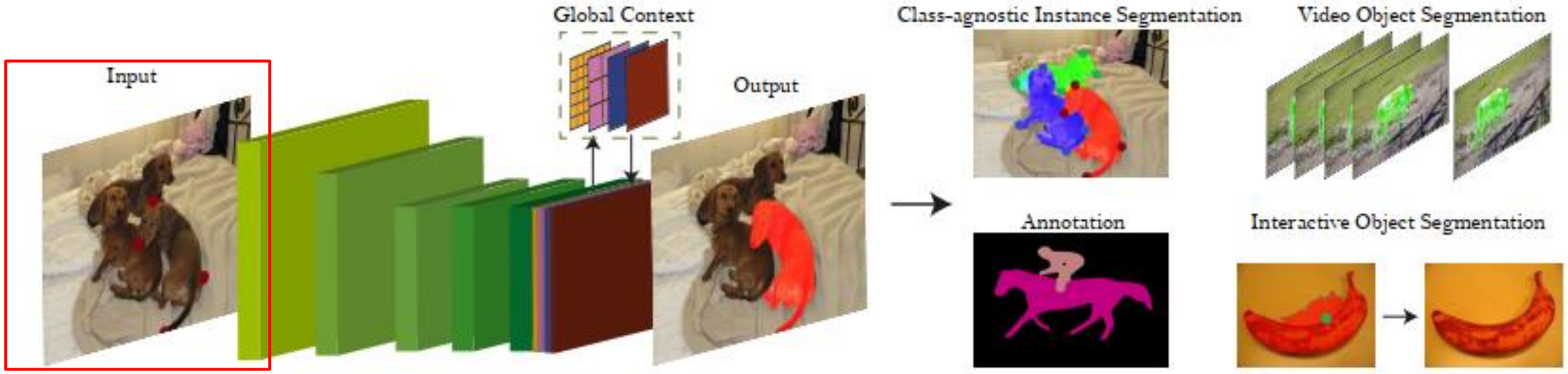
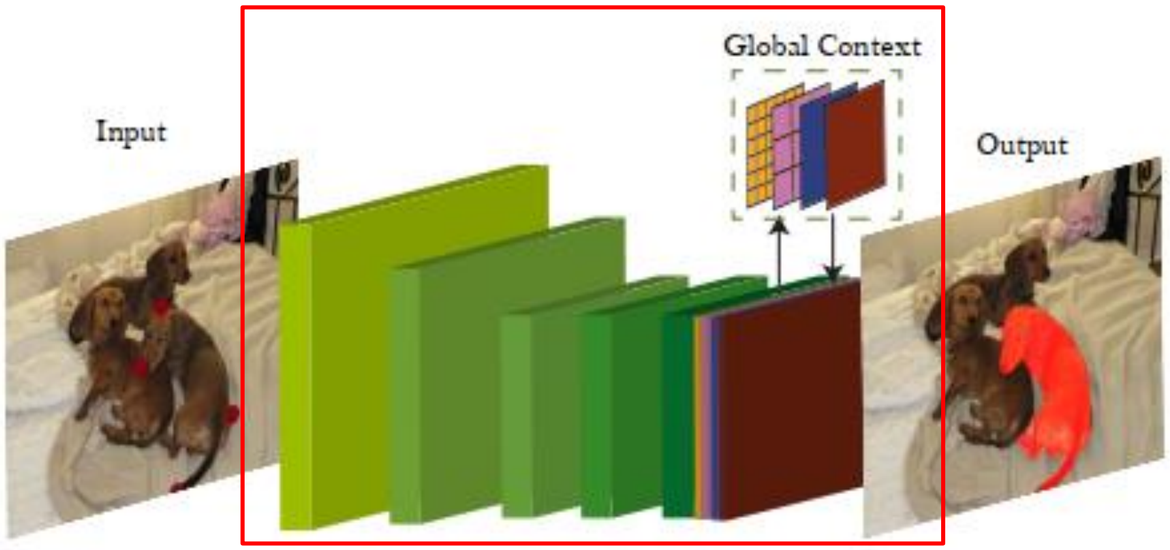


Figure 2. **Architecture of DEXTR**: Both the RGB image and the labeled extreme points are processed by the CNN to produce the segmented mask. The applicability of this method is illustrated for various tasks: Instance, Semantic, Video, and Interactive segmentation.

- 1. Input: a 4-channel input (RGB+ heatmap). We center a 2D Gaussian around each of the points, in order to create a single heatmap. The input is cropped by the bounding box, formed from the extreme point annotations.
-

pyramid scene parsing



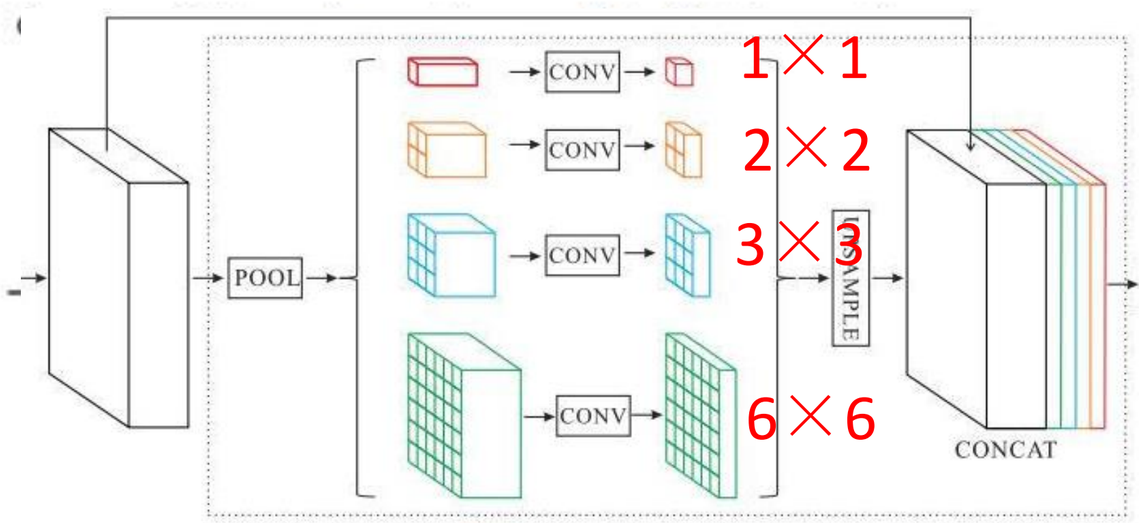


Figure 2. **Architecture of DEXTR**: Both the RGB image and the labeled extreme points are processed by the CNN to produce the segmented mask. The applicability of this method is illustrated for various tasks: Instance, Semantic, Video, and Interactive segmentation.

- 1. Input: a 4-channel input (RGB+ heatmap).
- 2. Network: Deeplab-v2 model. choose *ResNet-101* as the backbone of our architecture. remove the fully connected layers as well as the max pooling layers in the last two stages. user pyramid scene parsing (PSP) module instead of Atrous spatial pyramid (ASPP).
- 3. Loss:

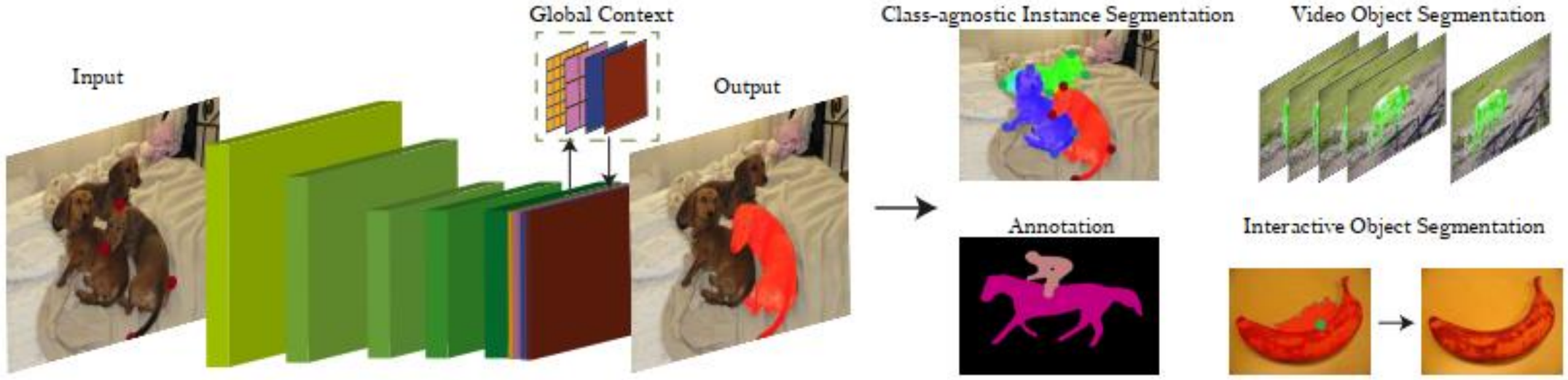


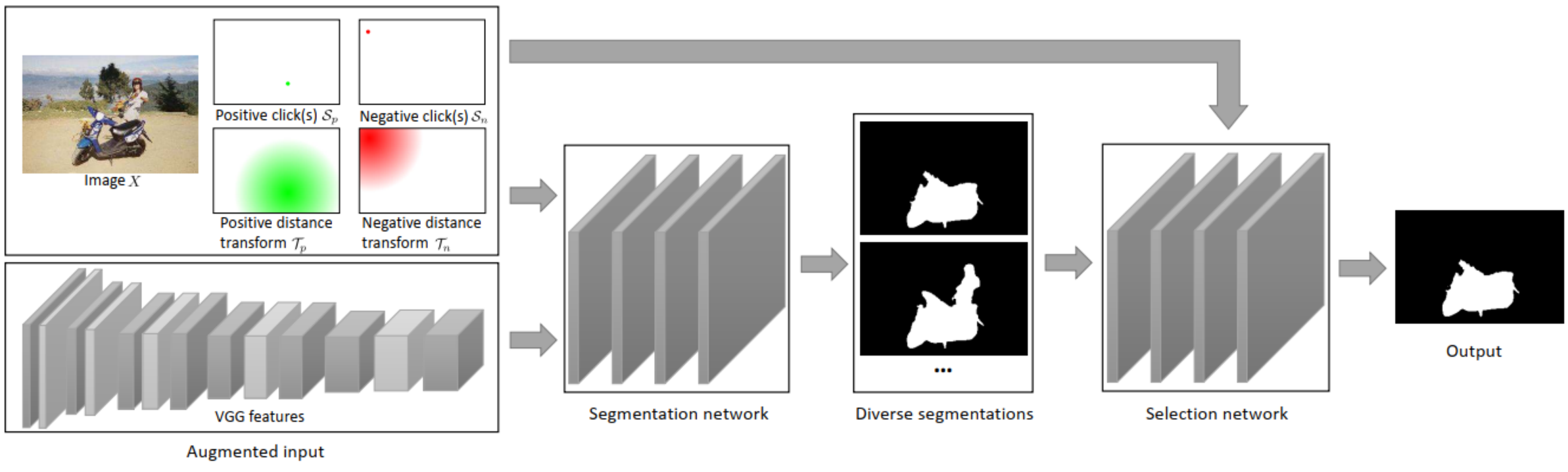
Figure 2. **Architecture of DEXTR**: Both the RGB image and the labeled extreme points are processed by the CNN to produce the segmented mask. The applicability of this method is illustrated for various tasks: Instance, Semantic, Video, and Interactive segmentation.

- 1. Input: a 4-channel input (RGB+ heatmap).
- 2. Network: Deeplab-v2 model.
- 3. Loss: class-balanced cross entropy loss , where the loss for each class in the batch is weighted by its inverse frequency.

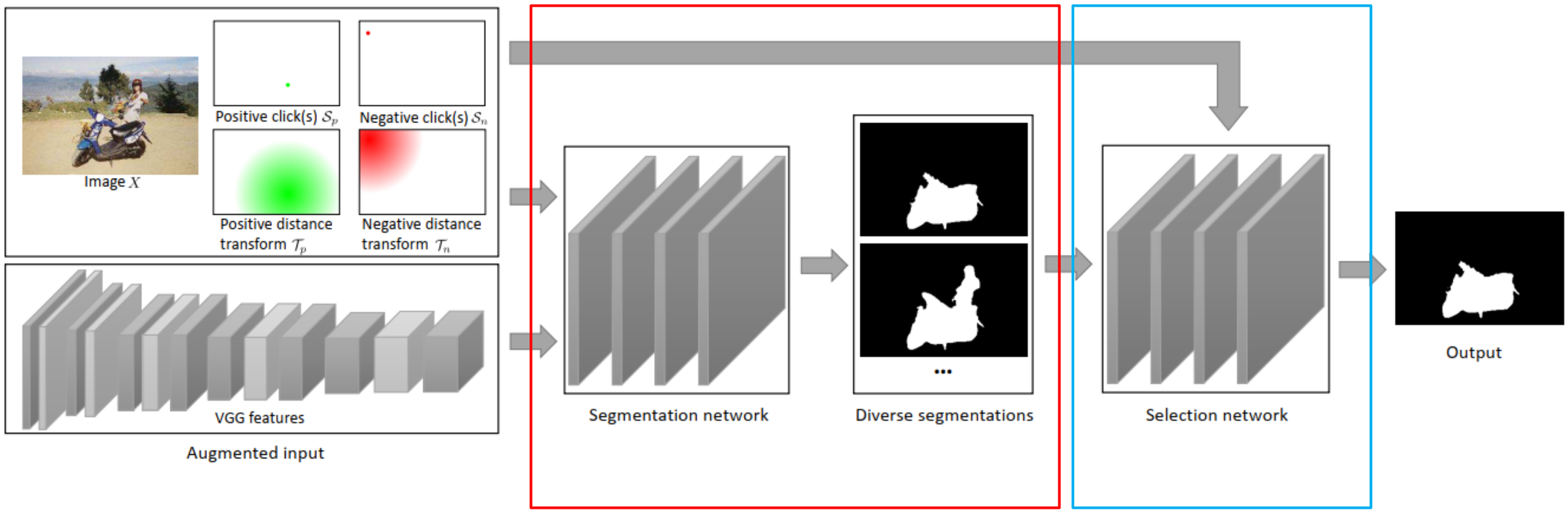
$$\mathcal{L} = \sum_{j \in Y} w_{y_j} C\left(y_j, \hat{y}_j\right), \qquad j \in 1, ..., |Y|$$
$$y_j \in \{0, 1\}$$

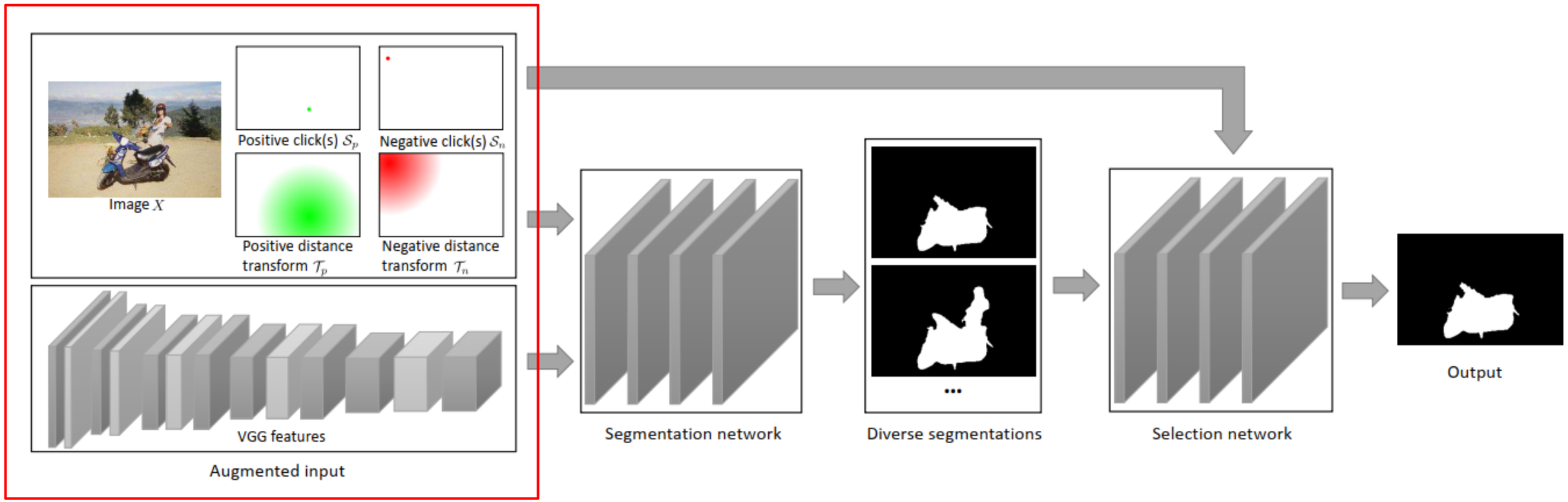
Interactive Image Segmentation with Latent Diversity

- When the user clicks on a door, do they intend to select the door or the whole house?



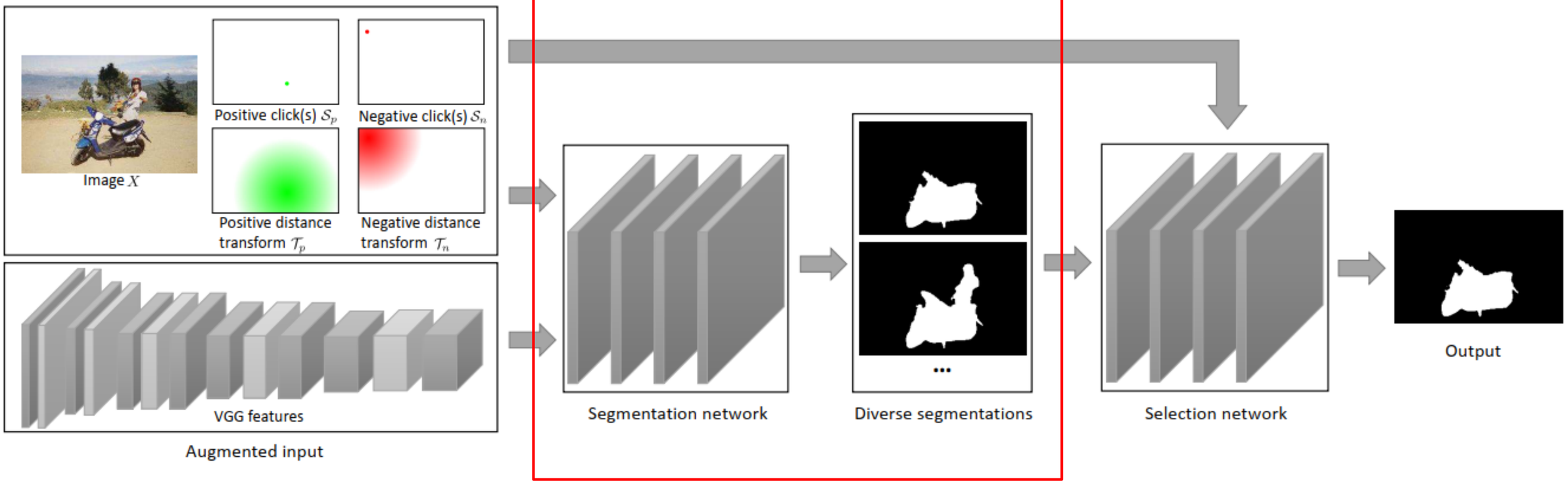
- *The first is trained to synthesize a diverse set of plausible segmentations that conform to the user's input.*
- *The second is trained to select among these.*





- *Input:* The total number of channels in the input is 1,477 (the image X, clicks Sp and Sn, distance maps, feature map)
- feature maps : Use VGG-19 network pretrained on the ImageNet to extract the feature maps from the following layers: ‘conv1 2’, ‘conv2 2’, ‘conv3 2’, ‘conv4 2’, and ‘conv5 2’.
- clicks & distance maps:

$$\mathcal{T}_p(\mathbf{p}) = \min_{\mathbf{q}\in\mathcal{S}_p} \|\mathbf{p}-\mathbf{q}\|_2$$
$$\mathcal{T}_n(\mathbf{p}) = \min_{\mathbf{q}\in\mathcal{S}_n} \|\mathbf{p}-\mathbf{q}\|_2.$$



- The output layer has M channels, one for each synthesized segmentation mask. The final nonlinearity is a sigmoid that maps each pixel to the range [0, 1].
- *Loss:*

$$\begin{aligned}\mathcal{L}_f(\boldsymbol{\theta}_f) &= \sum_i \min_m \Big\{\ell(Y_i, f_m(\mathbf{X}_i;\boldsymbol{\theta}_f)) \\ &+ \ell_c(\mathcal{S}_p^i, \mathcal{S}_n^i, f_m(\mathbf{X}_i;\boldsymbol{\theta}_f))\Big\}.\end{aligned}$$

- *Loss:*

$$\mathcal{L}_f(\boldsymbol{\theta}_f) = \sum_i \min_m \left\{ \ell(Y_i, f_m(\mathbf{X}_i; \boldsymbol{\theta}_f)) + \ell_c(\mathcal{S}_p^i, \mathcal{S}_n^i, f_m(\mathbf{X}_i; \boldsymbol{\theta}_f)) \right\}.$$

- Jaccard (IoU) distance：
- m: mask number (m=6)

$$\ell(A, B) = 1 - \frac{\sum_{\mathbf{p}} \min(A(\mathbf{p}), B(\mathbf{p}))}{\sum_{\mathbf{p}} \max(A(\mathbf{p}), B(\mathbf{p}))},$$

- *Loss:*

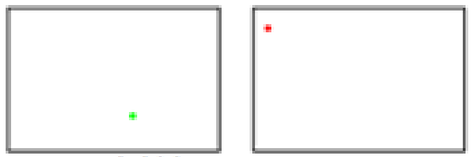
$$\begin{aligned}\mathcal{L}_f(\boldsymbol{\theta}_f) &= \sum_i \min_m \left\{ \ell(Y_i, f_m(\mathbf{X}_i; \boldsymbol{\theta}_f)) \right. \\ &+ \left. \ell_c(\mathcal{S}_p^i, \mathcal{S}_n^i, f_m(\mathbf{X}_i; \boldsymbol{\theta}_f)) \right\}.\end{aligned}$$

- soft constraints

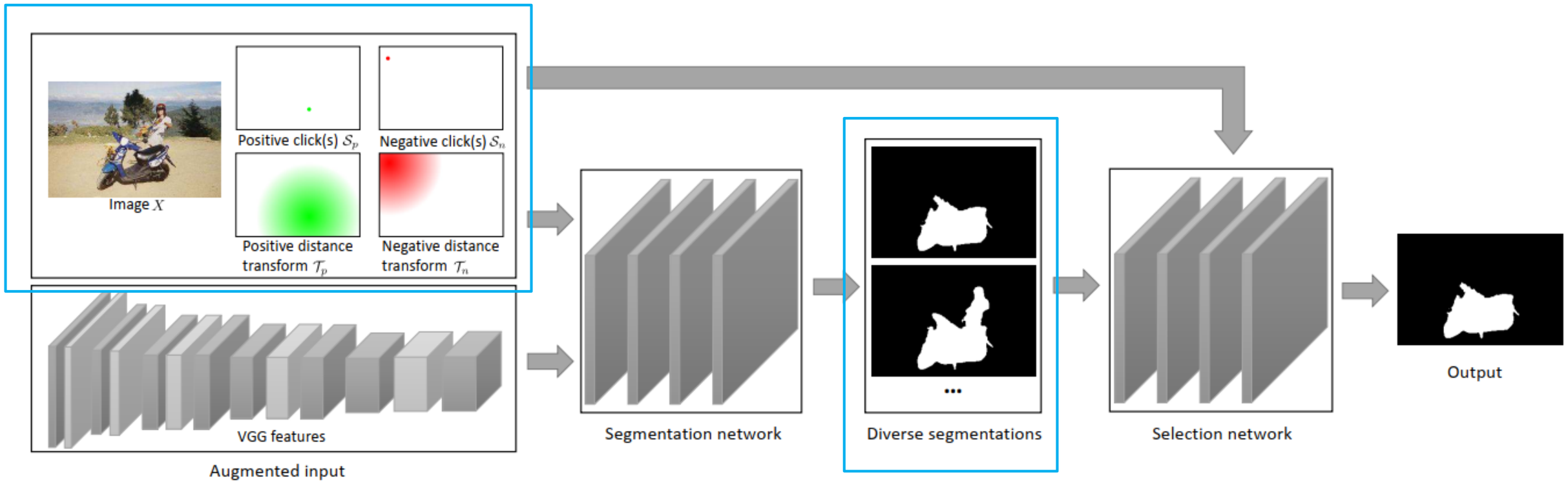
$$\begin{aligned}\ell_c(\mathcal{S}_p, \mathcal{S}_n, B) &= \|\mathcal{S}_p \odot (\mathcal{S}_p - B)\|_1 \\ &+ \|\mathcal{S}_n \odot (\mathcal{S}_n - (1 - B))\|_1,\end{aligned}$$

where $\odot$ denotes the Hadamard elementwise product.

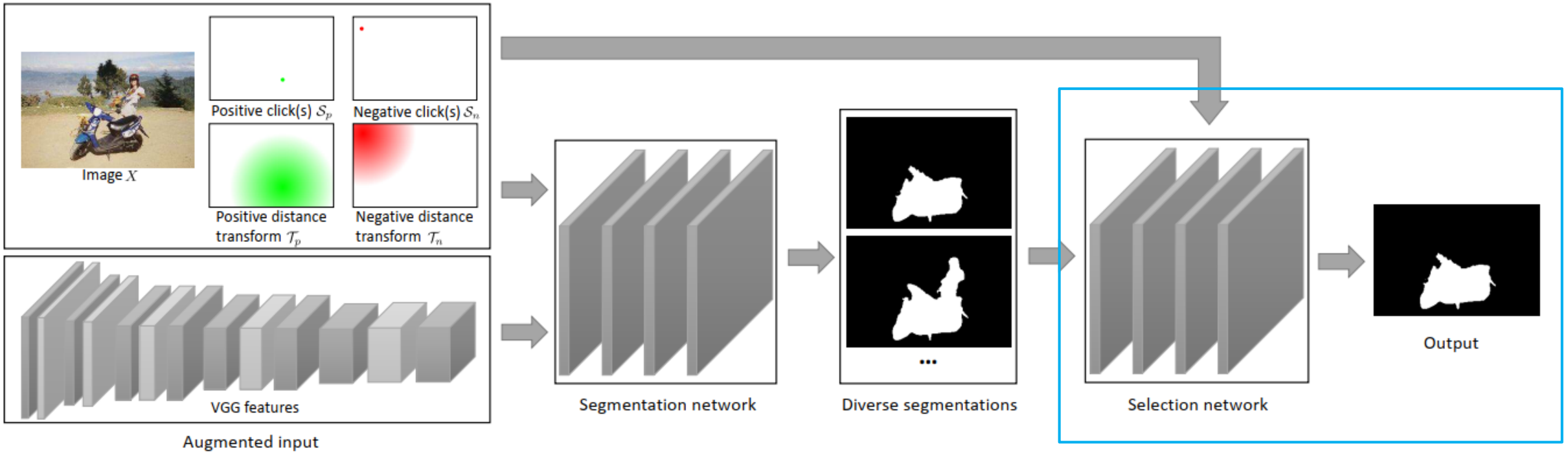
predicted mask $B \in [0, 1]^{w \times h}$.



Positive click(s) $\mathcal{S}_p$ Negative click(s) $\mathcal{S}_n$

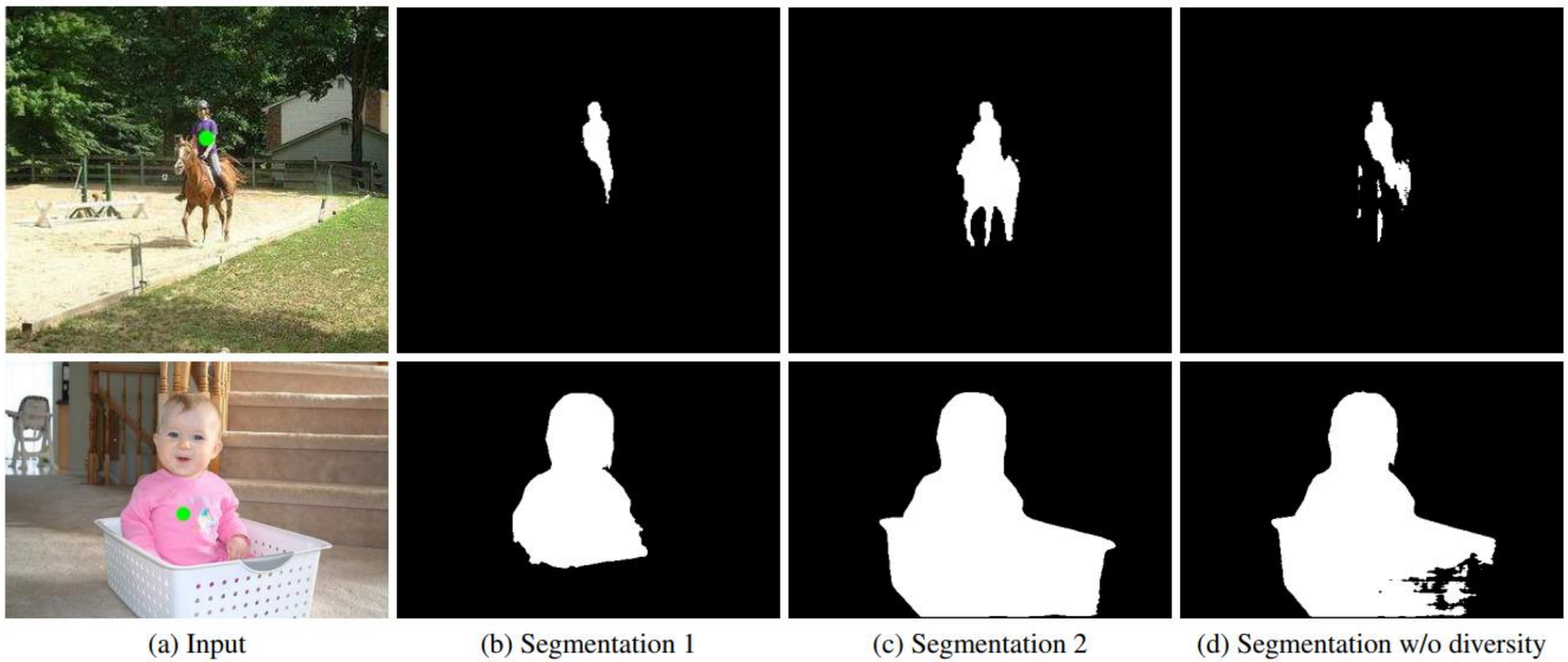


- *Input:* The the image X, clicks Sp and Sn, distance maps and M channels of Segmentation Masks



- The output : M-vector
- *Loss：* cross-entropy loss

$$\mathcal{L}_g(\boldsymbol{\theta}_g) = \sum_i \left(-g_{\phi_i}(\mathbf{Z}_i; \boldsymbol{\theta}_g) + \log \sum_{m=1}^{M} \exp\left(g_m(\mathbf{Z}_i; \boldsymbol{\theta}_g)\right) \right)$$



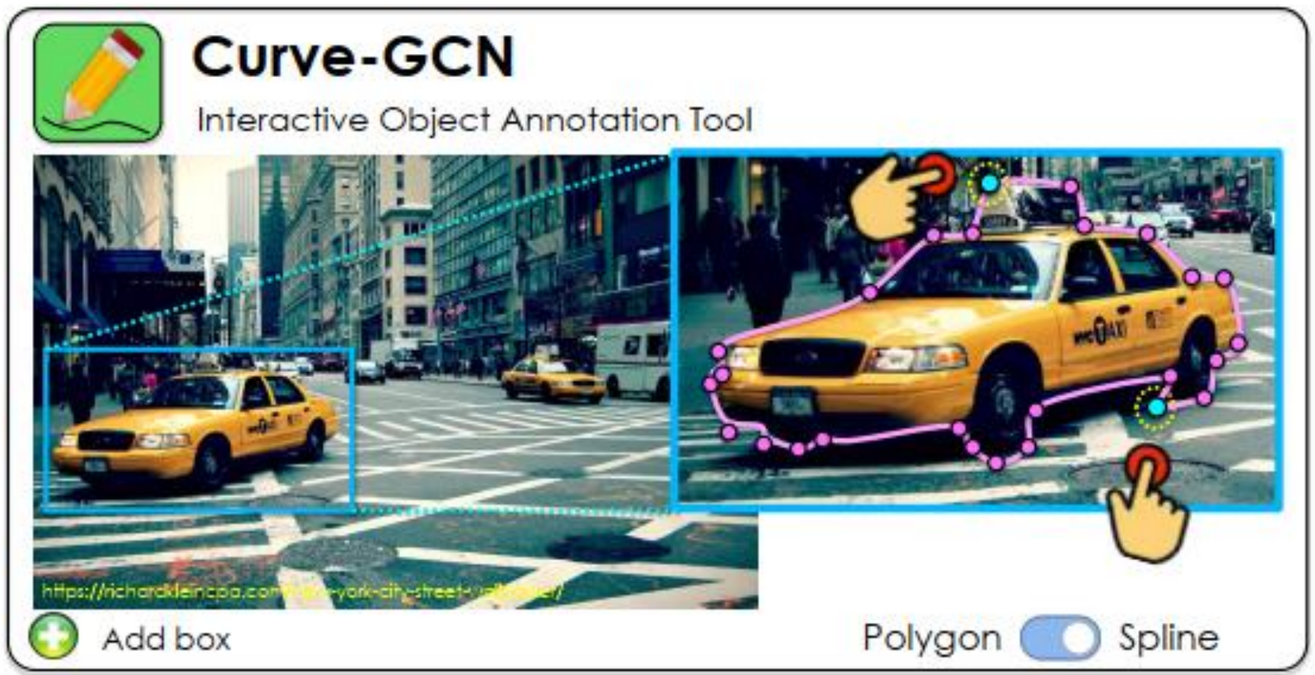
(a) Input (b) Segmentation 1 (c) Segmentation 2 (d) Segmentation w/o diversity

Figure 2. Illustration of diversity given one positive input click. (a) shows the input image with a positive click (green). (b) and (c) show two of the intermediate segmentations in $\mathcal{Y}$. (d) shows the segmentation that would have been produced by the same network f without diversity ($M = 1$).

- *testing:* The first click is positive and each subsequent click is placed on a pixel that is still misclassified .

Fast Interactive Object Annotation with Curve-GCN

- Predict all the vertices of a polygon using a Graph Convolutional Network simultaneously .
-



Extracting Features

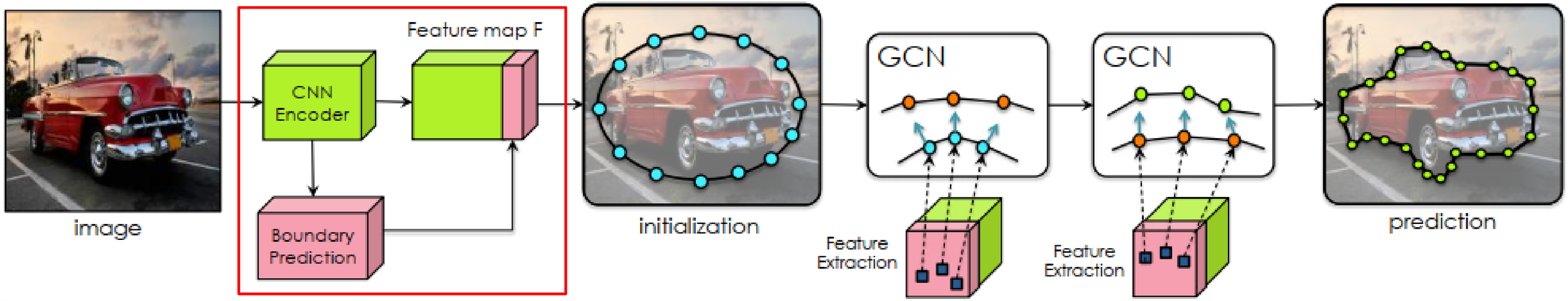


Figure 2: **Curve-GCN**: We initialize N control points (that form a closed curve) along a circle centered in the image crop with a diameter of 70% of image height. We form a graph and propagate messages via a Graph Convolutional Network (GCN) to predict a location shift for each node. This is done iteratively (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.

- 1. CNN serving as an image feature extractor.
  - additional branches are trained to predict the probability of existence of an object edge/vertex on a 28 $\times$ 28 grid. We train these two branches with the binary cross entropy loss.

initialize the GCN nodes

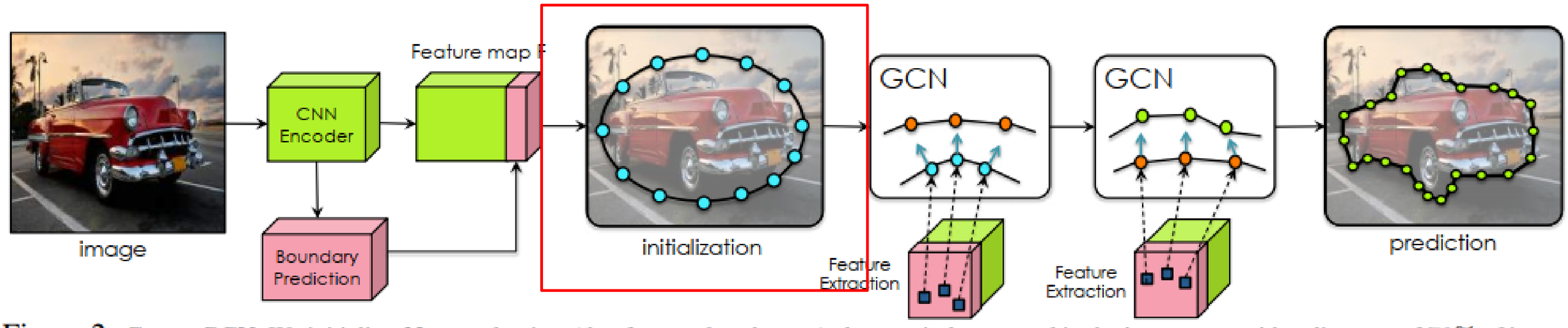


Figure 2: **Curve-GCN**: We initialize N control points (that form a closed curve) along a circle centered in the image crop with a diameter of 70% of image height. We form a graph and propagate messages via a Graph Convolutional Network (GCN) to predict a location shift for each node. This is done iteratively (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.

- 1.CNN serving as an image feature extractor.
- 2. GCN:
  - We initialize the nodes of the GCN to be at a static initial central position.
  - Represent object using *N* control points, which are connected to form a cycle, with straight lines (thus forming a polygon), or higher order curves (forming a spline).
  - GCN predicts a location offset for each node, aiming to move the node correctly onto the object's boundary .
  - We define the graph to be $G = (V; E)$

- **GCN Model** : multi-layer GCN

Input feature

$$F: f_i^0 = \text{concat}\{F(x_i, y_i), x_i, y_i\}$$

node $\mathbf{cp}_i$ at layer l is expressed as:

$$f_i^{l+1} = w_0^l f_i^l + \sum_{\mathbf{cp}_j \in \mathcal{N}(\mathbf{cp}_i)} w_1^l f_j^l$$

Graph-ResNet：

$$\begin{aligned} r_i^l &= ReLU(w_0^l f_i^l + \sum_{\mathbf{cp}_j \in \mathcal{N}(\mathbf{cp}_i)} w_1^l f_j^l) \\ r_i^{l+1} &= \tilde{w}_0^l r_i^l + \sum_{\mathbf{cp}_j \in \mathcal{N}(\mathbf{cp}_i)} \tilde{w}_1^l r_j^l \\ f_i^{l+1} &= ReLU(r_i^{l+1} + f_i^l), \end{aligned}$$

On top of the last GCN layer, apply a single FC layer to predict a relative location shift: $(\Delta x_i, \Delta y_i)$

$$[x_i^{'}, y_i^{'}] = [x_i + \Delta x_i, y_i + \Delta y_i]$$

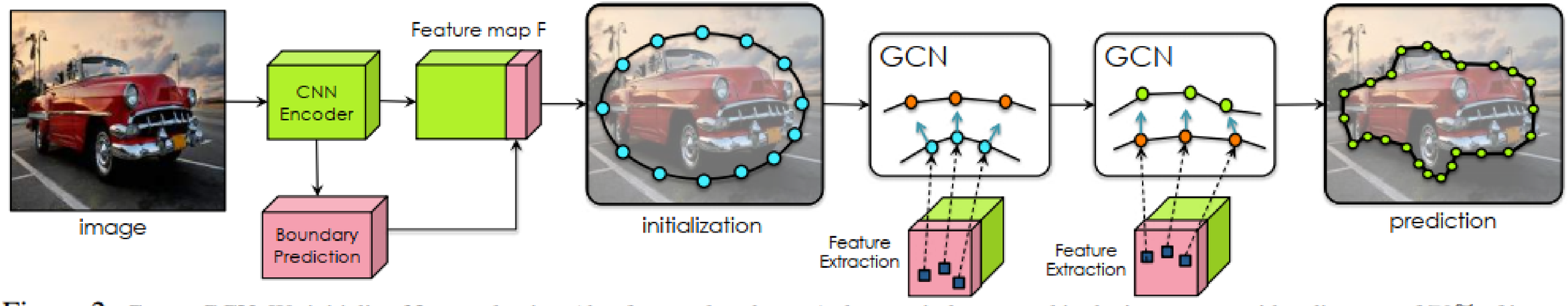


Figure 2: **Curve-GCN**: We initialize N control points (that form a closed curve) along a circle centered in the image crop with a diameter of 70% of image height. We form a graph and propagate messages via a Graph Convolutional Network (GCN) to predict a location shift for each node. This is done iteratively (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.

- 1.CNN serving as an image feature extractor.
- 2. GCN
- 3. Spline Parametrization

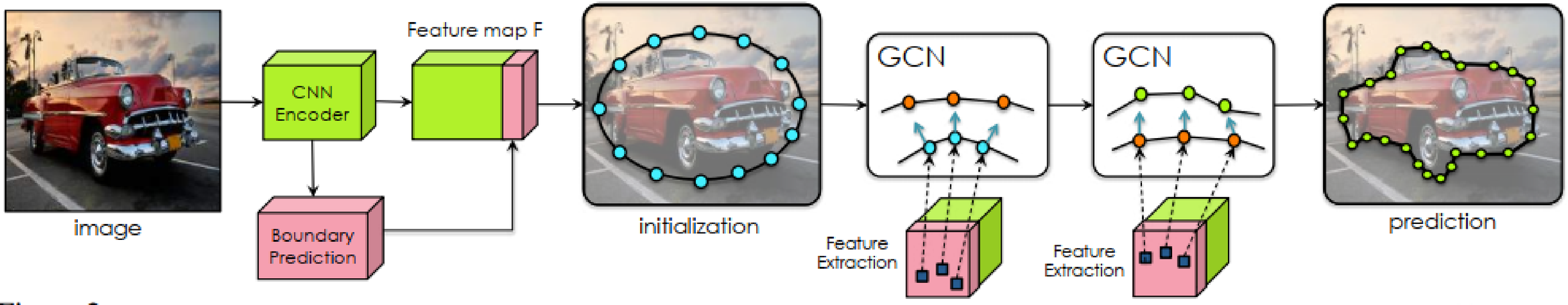


Figure 2: **Curve-GCN**: We initialize N control points (that form a closed curve) along a circle centered in the image crop with a diameter of 70% of image height. We form a graph and propagate messages via a Graph Convolutional Network (GCN) to predict a location shift for each node. This is done iteratively (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.

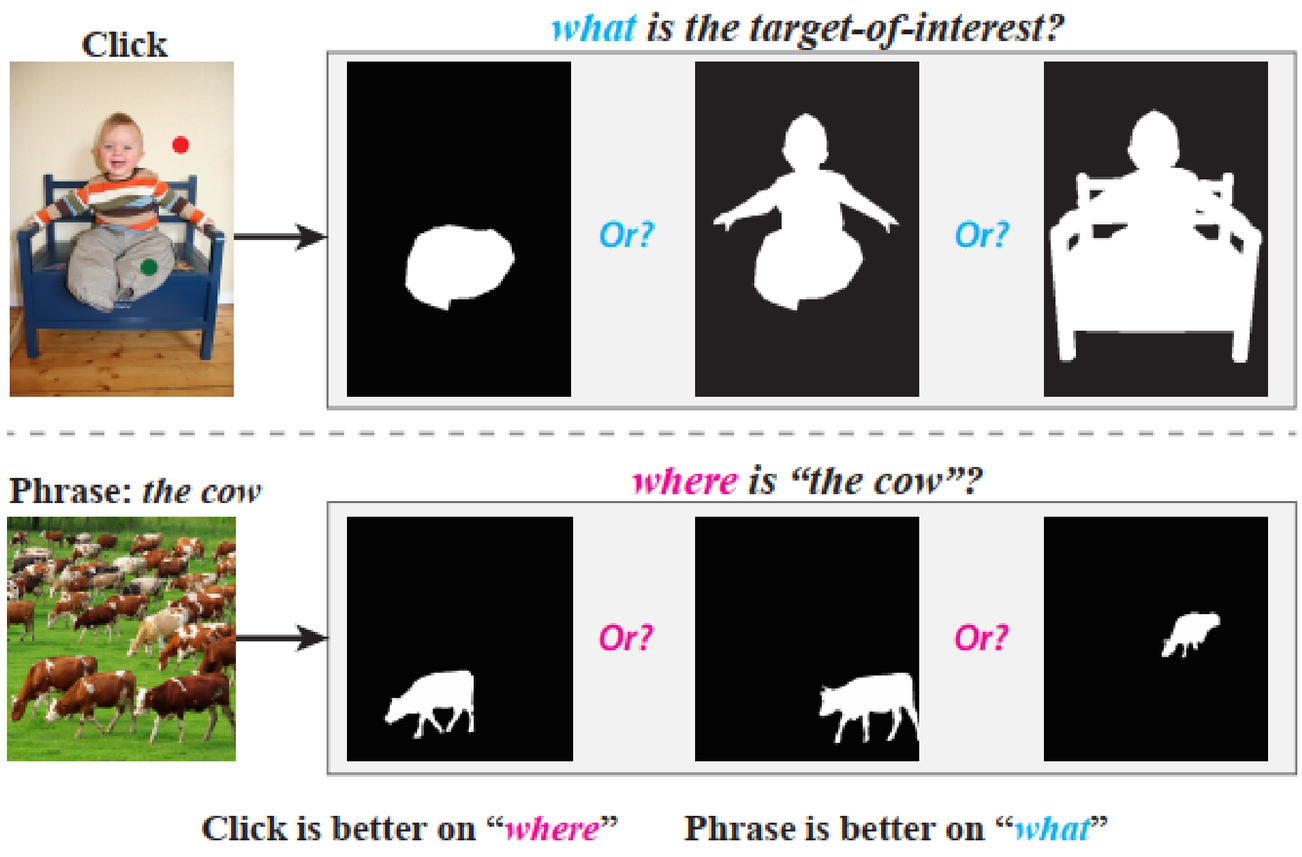
- 1.CNN serving as an image feature extractor.
- 2. GCN
- 3. Spline Parametrization
- 4. Loss

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

Differentiable Accuracy Loss: $$L_{\text{render}}(\theta) = \|M(\theta) - M_{\text{gt}}\|_1$$

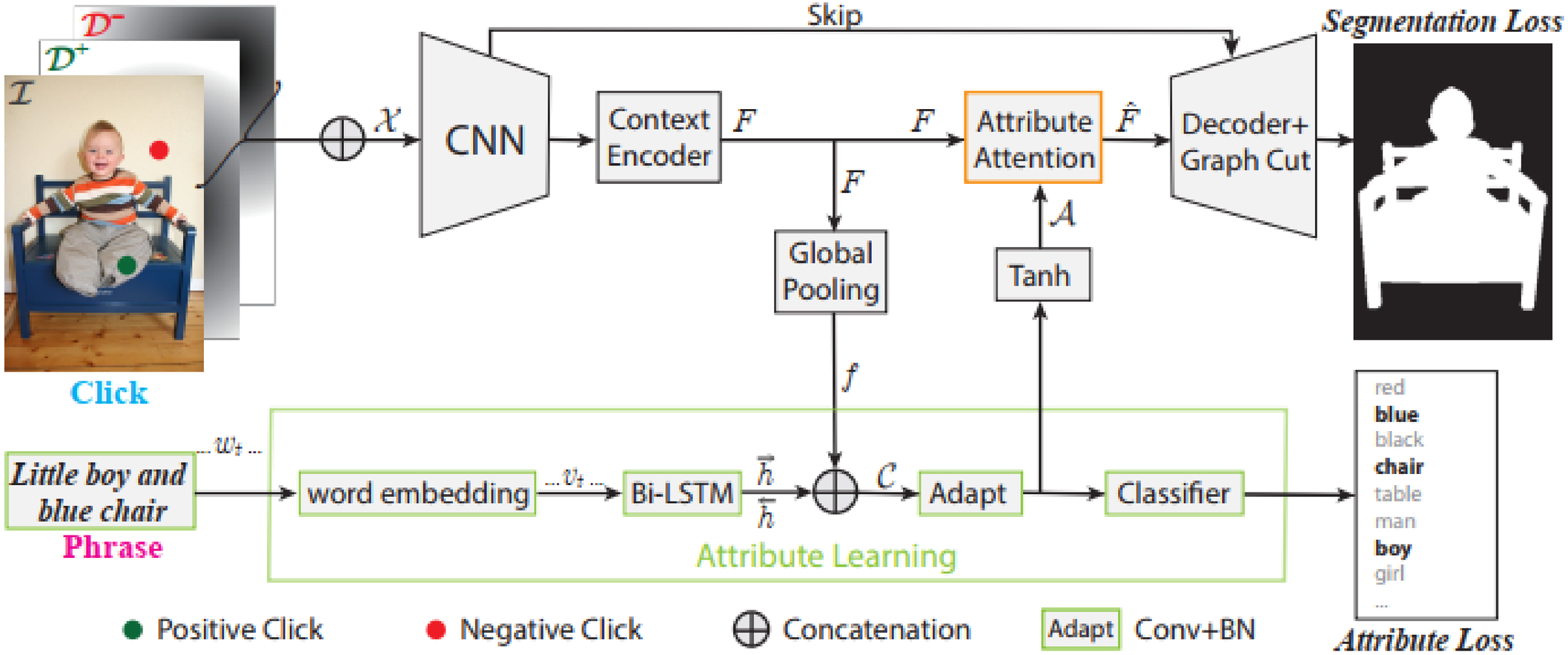
PhraseClick: Toward Achieving Flexible Interactive Segmentation by Phrase and Click

- propose to employ phrase expressions as another interaction input to infer the attributes of target object.



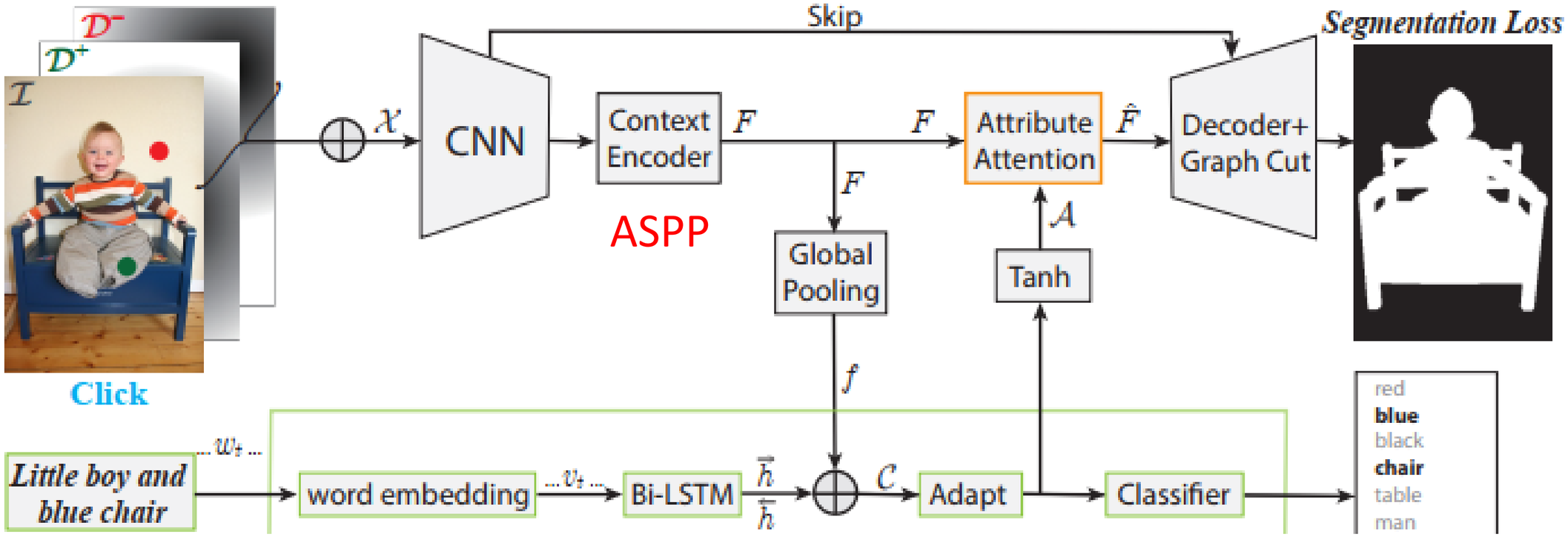
PhraseClick: Toward Achieving Flexible Interactive Segmentation by Phrase and Click

- propose to employ phrase expressions as another interaction input to infer the attributes of target object.



- Vision part Network: ResNet-101 based DeepLabv3+
- distance maps and concatenated with original image to form a 5-channel input .
- language part : word-to-vector model and bi-directional LSTM
- post-processing : graph cut

Vision part Network



$$\hat{F} = \beta \mathcal{A} * F + F$$

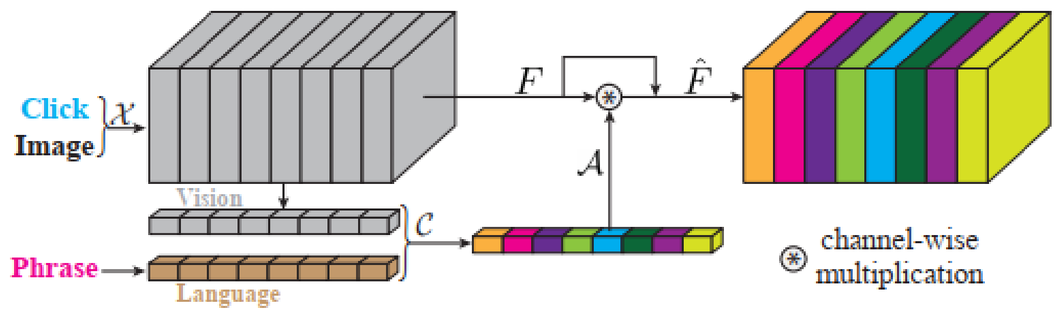
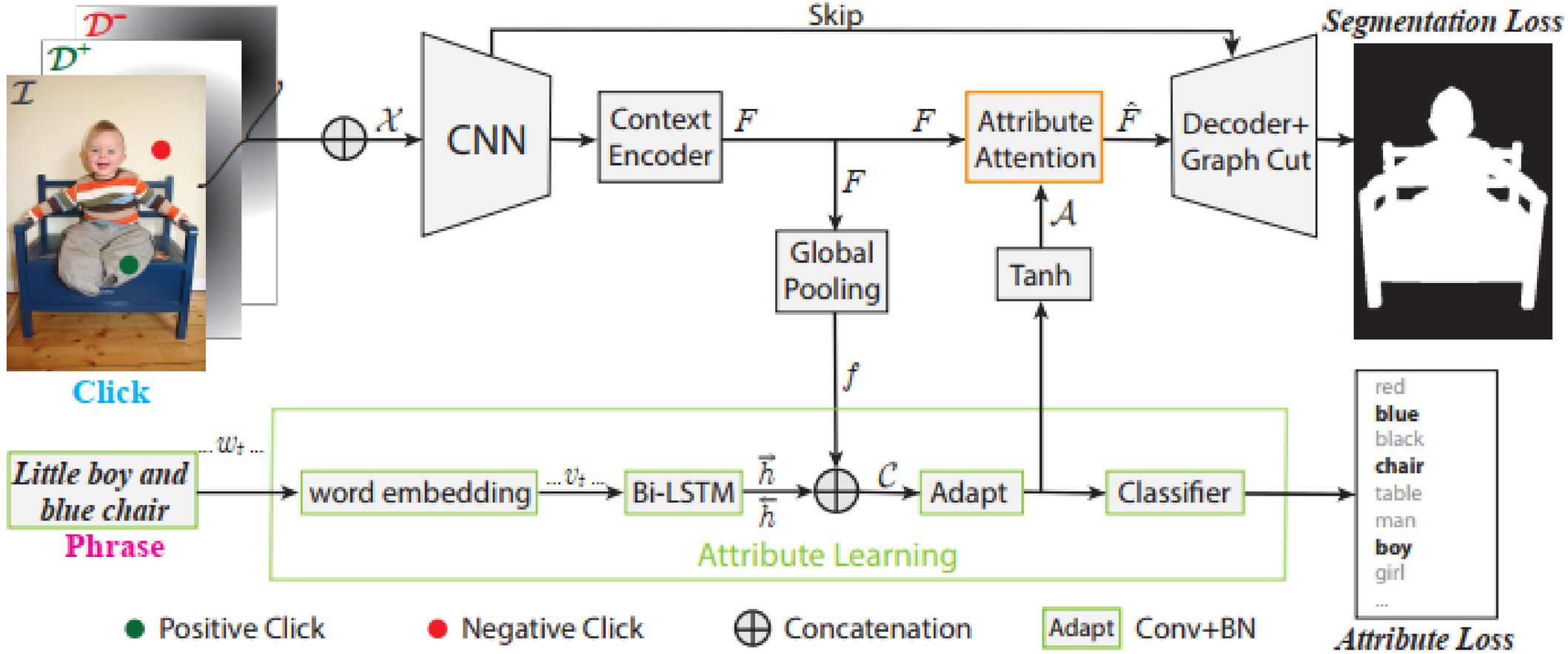


Fig. 4. Attribute Attention: It emphasizes feature channels that have larger response in semantic attribute learning, which is based on the phrase interaction, click interaction and visual patterns.

language part



$$\mathcal{C} = \overrightarrow{h} \oplus \overleftarrow{h} \oplus f$$

$tanh(x) = (e^x - e^{-x})/(e^x + e^{-x})$, $Adapt$ is `Conv+BN`

- Loss:
- the binary cross-entropy loss for the segmentation

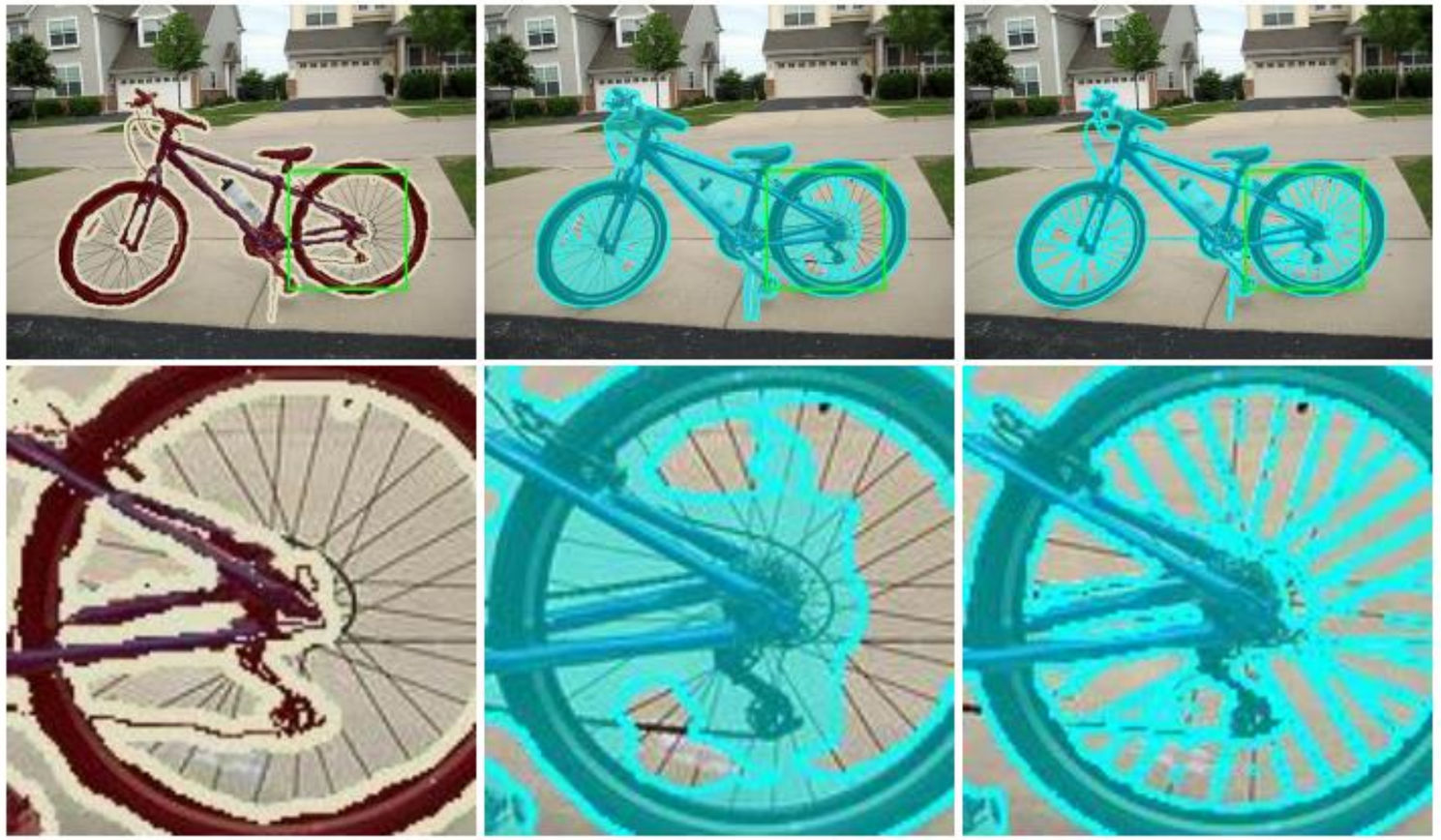
$$\mathcal{L}_m = -tlog(\sigma(s)) - (1-t)log(1-\sigma(s))$$

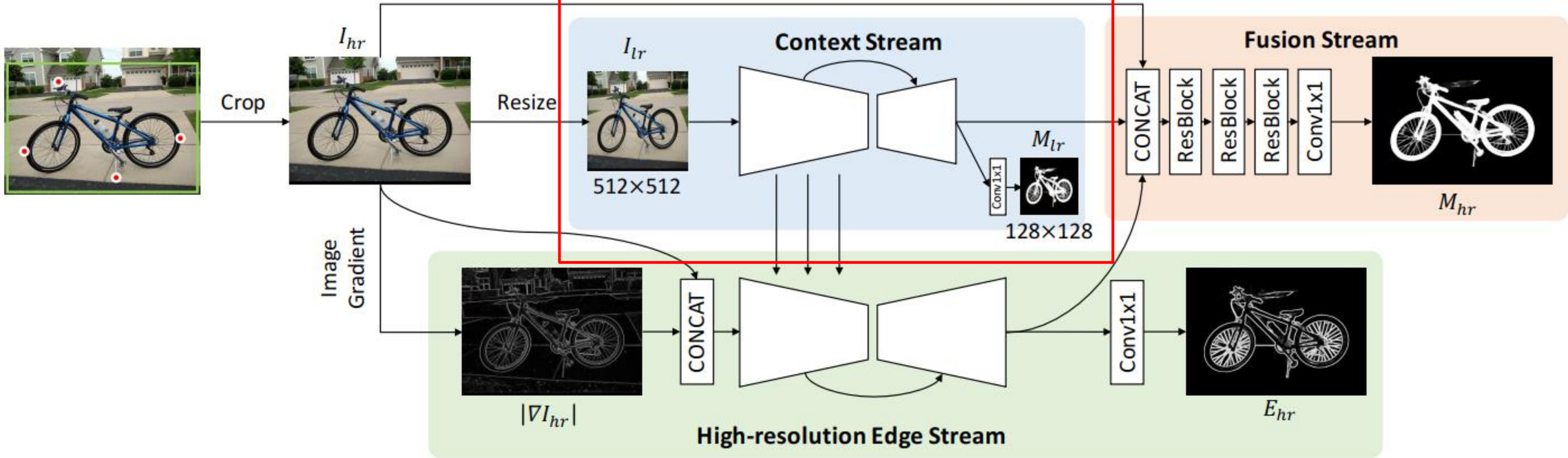
- the attribute learning :

$$\mathcal{L}_a = \sum_{i=1}^{N} w_i(a_i log(\sigma(p_i)) + (1-a_i)log(1-\sigma(p_i))$$

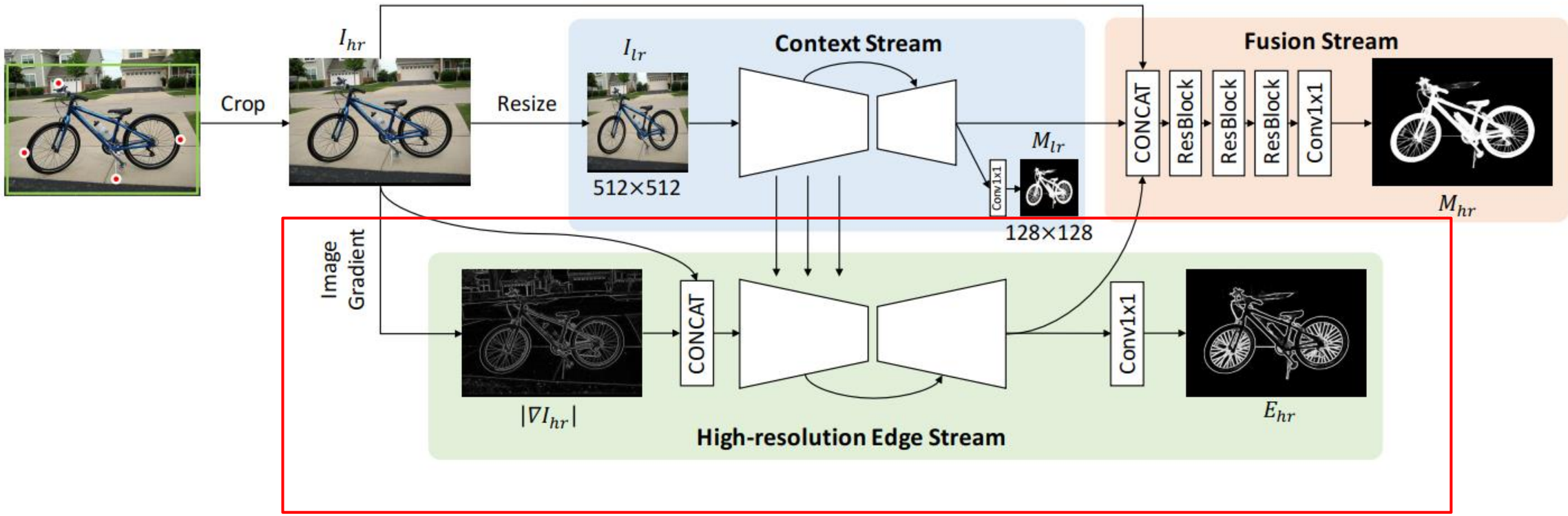
Fig. 8. Our interaction process is more flexible. The user can choose either

Deep Interactive Thin Object Selection

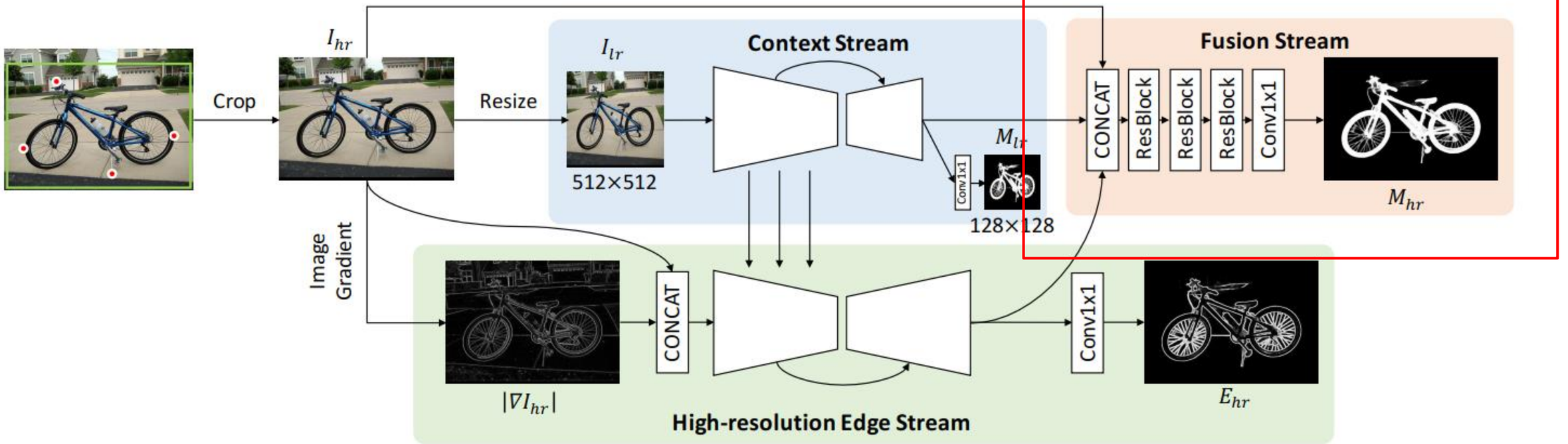




- 1. generate rough object segmentation



- 1. generate rough object segmentation
- 2. edge stream



- 1. generate rough object segmentation
- 2. edge stream
- 3. fusion

Efficient Full Image Interactive Segmentation by Leveraging Within-image Appearance Similarity

- interactive full-image semantic segmentation

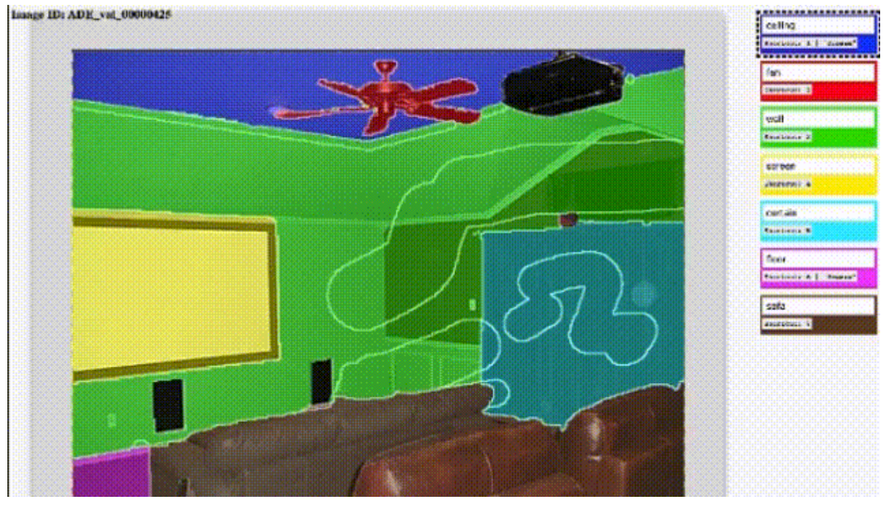
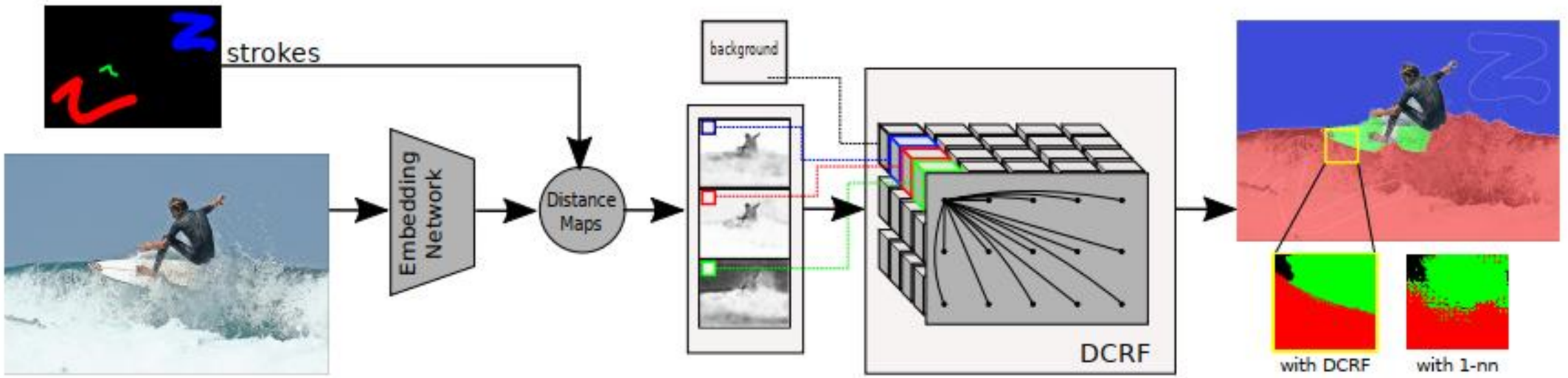
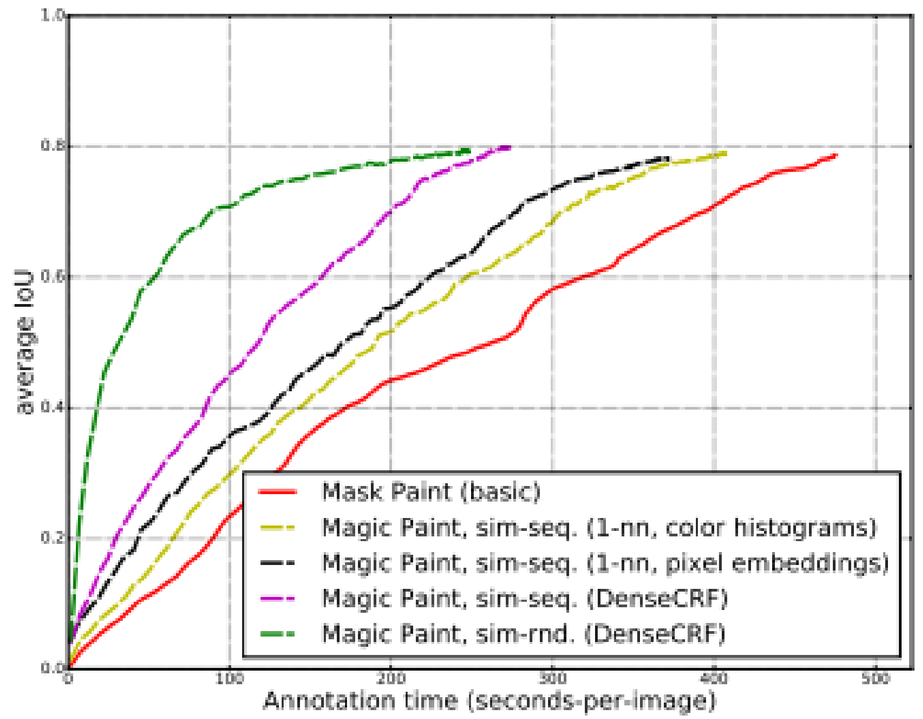


Fig. 1. An example of annotation using our Magic Paint approach. The user first draws a blue stroke on a person, which we automatically propagate to label the majority of that person and the other one (left). Next, the annotator draws a red stroke, which causes Magic Paint to both correct some minor leaks from the blue region and also propagates correctly to the wall region of the stadium (middle). Finally a single green stroke by the annotator gets correctly propagated to the entire ground (right). Note how just three simple strokes already lead to correctly annotating most of this image.



- we represent the output of the global pixel similarity computation between the labeled and unlabeled pixels in the form of distance maps.

Fig. 5. A few annotation examples of Magic Paint on ADE-20k (first column), Fashionista (fourth column) and COCO datasets (other columns). The top row shows the original image while the bottom row shows the annotation result, with the drawn strokes. The majority of the strokes are drawn in the center of the objects, while the border strokes are often in freeze foreground mode. Note how typically a large portion of the image is automatically labeled by Magic Paint (all pixels outside any stroke).



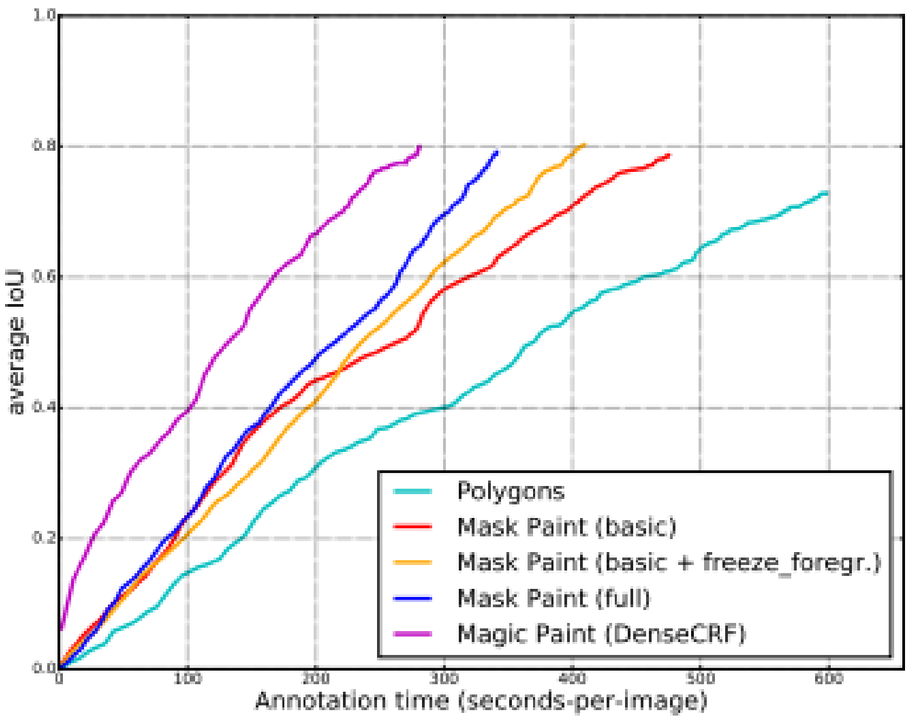


Fig. 6. Experimental results on COCO-val-100. Comparison of model variants in simulation (left) and evaluation with real human annotators (right).