Object Instance Annotation with Deep Extreme Level Set Evolution

Zian Wang[1] David Acuna[2,3,4*] Huan Ling[2,3*] Amlan Kar[2,3] Sanja Fidler[2,3,4]
[1]Tsinghua University [2]University of Toronto [3]Vector Institute [4]NVIDIA
wza15@mails.tsinghua.edu.cn {davidj, linghuan, amlan, fidler}@cs.toronto.edu

Abstract

In this paper, we tackle the task of interactive object segmentation. We revive the old ideas on level set segmentation which framed object annotation as curve evolution. Carefully designed energy functions ensured that the curve was well aligned with image boundaries, and generally "well behaved". The Level Set Method can handle objects with complex shapes and topological changes such as merging and splitting, thus able to deal with occluded objects and objects with holes. We propose Deep Extreme Level Set Evolution that combines powerful CNN models with level set optimization in an end-to-end fashion. Our method learns to predict evolution parameters conditioned on the image and evolves the predicted initial contour to produce the final result. We make our model interactive by incorporating user clicks on the extreme boundary points, following DEXTR [29]. We show that our approach significantly outperforms DEXTR on the static Cityscapes dataset [16] and the video segmentation benchmark DAVIS [34], and performs on par on PASCAL [19] and SBD [20].

1. Introduction

Interactive object segmentation is a well studied problem with the aim to reduce the time and cost of annotation. Having semi-automatic tools is particularly important when labeling large volumes of data such as video [34, 29], or 3D medical imagery [1]. While learning from weak labels such as image level tags or single point clicks is lending itself as a fruitful alternative direction [5], having access to more accurate annotations is still a necessity in order to achieve high-end performance on downstream tasks [41].

Early work on interactive annotation incorporated user strokes on both the foreground and background and used graph cut optimization to find objects [36, 6, 12]. Contour-based approaches such as Intelligent Scissors [31] traced object boundaries as optimal paths along edges, guided by the user's mouse cursor. In Polygon-RNN [9, 3], the authors predicted polygon annotations with a CNN-RNN architecture that outputs one vertex at a time. Human-level performance

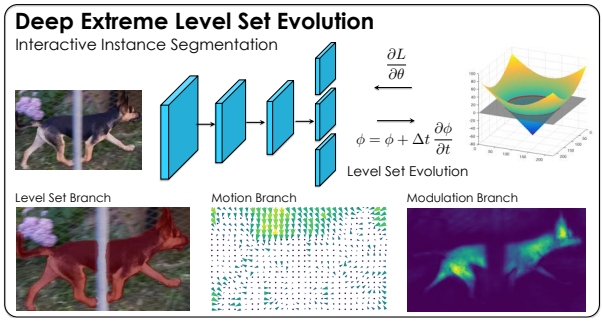

Figure 1: We introduce Deep Extreme Level Set Evolution (DELSE), which combines powerful CNN image feature extraction with Level Set Evolution. Our approach is end-to-end differentiable, and produces "well behaved" object contours. DELSE further exploits extreme points [29] for the purpose of interactive annotation.

was achieved with only a few user clicks per object. However, by assuming that an object is a closed cycle, the model cannot easily deal with objects cut in half due to occlusion, or wiry or donut like shapes that have holes.

An attractive alternative is to exploit the power of CNN-based architectures and treat object segmentation as a pixel-wise labeling task. In recent work [29], the authors proposed DEXTR, an approach that requires the user to only click on the four extreme boundary points. These are consumed as input to DeepLab, which is then shown to produce high quality object masks. However, pixel-wise approaches do not typically model pixel dependencies and may thus lead to spurious holes, small scattered islands, or overall irregular shapes. Fixing these is a time consuming task.

In our work, we revive the old idea of using level set methods for object segmentation [32, 8]. These find closed contours that align accurately with the image boundaries, and ensure that the resulting curve is regular and well behaved through a carefully designed energy function. Unlike parametric representations such as Active Contour Models [23, 8, 30], or approaches like Polygon-RNN++[3], level set methods can handle object boundaries with complex topologies such as corners and cusps, and are able to deal with topological changes like merging and splitting. While the literature on level set methods for segmentation is vast, most of the prior work either uses weak image gradients as observations [23, 7, 24], or exploit level sets as a postprocessing step to denoise CNN predictions [38].

*authors contributed equally

We propose an approach that combines a powerful CNN architecture with Level Set Evolution in an end-to-end fashion. Our model employs a multi-branch architecture that learns to predict level set evolution parameters conditioned on a given input image, and evolves a predicted initial contour to segment the object. We additionally make our methods interactive by incorporating both, extreme points following [29], and motion vectors by having annotators drag and drop erroneous points. We showcase our method on a variety of datasets and tasks, and show that it significantly outperforms state-of-the-art baselines on the Cityscapes dataset [16], and achieving state-of-the-art results for the task of video-based object annotation on the DAVIS dataset [34]. Code is available at https://github.com/fidler-lab/delse.

2. Related Work

Level Set Methods for Segmentation: Implicit representations of curves, rather than explicit (*i.e.* active contours models [23, 8]), naturally handle complex object topologies such as holes or splitting. In the Level Set formulation [32, 8], the curve iteratively evolves by moving along the descent of the level set energy, which includes the external energy coming from the data and internal energy coming from the curve. Edge based methods [23, 15, 7, 24, 8] mostly employ edge features in the external energy and evolve an initial curve to fit object boundaries. Instead of using edges, region based methods [35, 33, 10] utilize region homogeneity to segment objects. Exploiting texture, color and shape information has also been extensively studied [18]. Given the recent advances of deep learning in image segmentation [11, 13, 39, 4, 28], we proposed to combine a convolutional neural network with a carefully designed level set evolution scheme, thus exploiting the advantages of both.

Recent work on image segmentation has also combined the traditional active contour models with deep neural networks. [37] crops out patches around the initial curve and employ a CNN to predict the movement for curve evolution, patch by patch. For the task of building footprint extraction, [30] employs CNN features to predict the parameters of the active contour models. The authors propose a structured prediction formulation to train the model end-to-end by optimizing for an approximation to IoU. [14] extends this to encourage the active contour to match building boundaries. However, these methods need careful curve initialization and suffer from the typical drawbacks of parametric curves. [38] uses level set evolution as a postprocessing step to a CNN, and trains on unlabeled data processed in a semi-supervised fashion. [21] adds the level set energy in the loss function and uses a CNN to directly predict the level set function for salient object detection. Recent works also utilize motion of pixels for segmentation, which shares similarities with level set evolution. [27] employs deep CNNs to learn an affinity matrix and refines the segmentation output via spatial propagation. [22] adds a recurrent pathway onto deep CNNs and reconstructs neural cells by iterative extension. In [2], the authors use level set evolution during training to denoise object annotations. Our key contribution is a deep level set model for instance segmentation which can be trained end-to-end and naturally incorporates a human in the loop.

Interactive Annotation has been addressed with a variety of methods, ranging from graph-cut based approaches with simple image potentials such as [36, 6, 12], to recent work that employs powerful CNN architectures [9, 3, 29]. In [29], the authors incorporate user-provided extreme boundary points as input to DeepLab [11]. Contour-based interactive segmentation models include Intelligent Scissors [31], which find optimal paths along the boundary guided by the user's mouse cursor. Polygon-RNN [9, 3] predicts a polygon around the object with a CNN-RNN architecture. [26] predicts a spline outlining an object using Graph Convolutional Networks. In [17], the classical level set energy is augmented with user's clicks. We here build on top of the DeepLab-v2 architecture [11] and exploit user-clicked extreme points as in [29]. Our approach is an end-to-end trainable level set framework for interactive object segmentation.

3. Background on Level Sets

Let $\mathcal{C}(s) : \Omega_P \rightarrow \mathbb{R}^2$ denote a parametric curve, where $s \in \Omega_P = [0, 1]$ is the parameterization interval. The level set method implicitly represents a curve using the zero crossing of a level set function (LSF) $\phi(x, y)$:

$$\mathcal{C} = \{(x, y) | \, \phi(x, y) = 0\} \tag{1}$$

The minimization of the energy can be viewed as an evolution along the descent of the energy. Let $\mathcal{C}(s, t)$ denote a curve that depends on a time parameter $t \in \mathbb{R}$. The curve evolution can then be formally defined as:

$$\frac{\partial C(s,t)}{\partial t} = V\vec{N} \tag{2}$$

This can also be expressed by the evolution of the level set function $\phi(x, y, t)$ by

$$\frac{\partial \phi}{\partial t} = -V|\nabla\phi|, \tag{3}$$

where $\vec{N}$ is the unit vector in the inward normal direction of the curve and V indicates the velocity along the normal direction. Evolution of the LSF is performed iteratively. Let $\phi(x, y, t), t \in \mathbb{R}$ denote the evolution of LSF, where we use $\phi_i(x, y)$ to denote $\phi(x, y, i)$ for simplicity. For $i \in \{0, 1, \cdots, T-1\}$, the T-step iterative update is:

$$\phi_{i+1}(x, y) = \phi_i(x, y) + \Delta t \, \frac{\partial \phi_i}{\partial t}, \tag{4}$$

where Δt is the time step, $\frac{\partial \phi}{\partial t}$ is the update term derived from the level set energy, $\phi_0(x, y)$ is the initial LSF, and $\phi_T(x, y)$ is the corresponding output after T evolution steps.

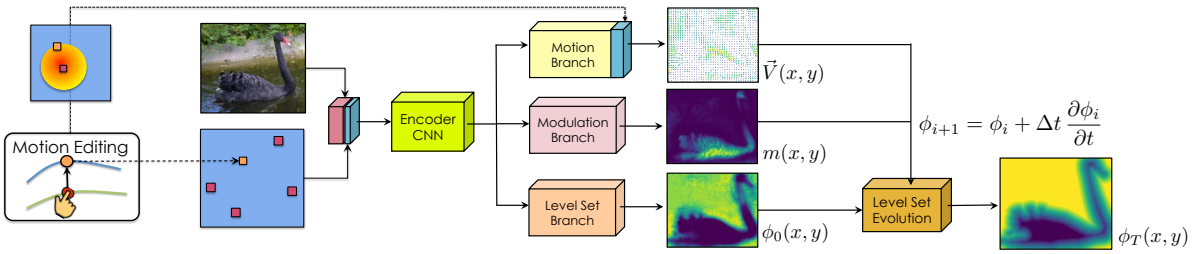

Figure 2: **Architecture of DELSE**: Extreme points are encoded as a heat map and concatenated with the image, and passed to the encoder CNN. A multi-branch architecture is used to predict the initial curve and parameters used in level set evolution. The Level Set Branch predicts initial level set function and evolve it using parameters predicted in the Motion and Modulation branches to get the final curve. The model is differentiable and trained end-to-end. For interactive annotation, we assume that the annotator drags and drops a wrong boundary point, producing a motion vector which is incorporated into the model.

For the task of object segmentation, one aims to find the boundary curve $\mathcal{C}$ that separates the foreground object from the background in an image. In the level set method, curve $\mathcal{C}$ is represented implicitly by LSF ϕ, and the foreground and background regions are denoted as $\{(x,y) \in \Omega_I | \, \phi(x,y) > 0\}$ and $\{(x,y) \in \Omega_I | \, \phi(x,y) < 0\}$, respectively.

4. Deep Extreme Level Set Evolution (DELSE)

Given an input image I, our goal is to employ a neural network to predict both the initial LSF ϕ_0 as well as the update terms used in level set evolution. We then evolve the initial LSF for T steps to generate the final LSF as our segmentation result. The whole evolution process is differentiable and thus can be trained end-to-end. To make our model interactive, we follow [29] and make use of extreme points P (*i.e.* left-most, right-most, top, and bottom pixels of an object) as an additional input to our model. Extreme points have been shown as minimal yet very effective input to guide the network. The proposed DELSE model for object instance segmentation is illustrated in Fig. 2.

In what follows, we describe the prediction of initial LSF in Section 4.1, and prediction of the level set terms in Section 4.2. Section 4.3 presents our training scheme.

4.1. Initial Level Set Function Prediction

Traditional level set methods require a human to label a rough boundary as an initialization, which is time-consuming. In our proposed model, we take the four extreme points as input and utilize the CNN model to automatically generate a rough estimate of the initial level set function ϕ_0, which is more efficient.

Following [29], we place a Gaussian around each extreme point to get a heatmap, and concatenate it with the RGB image as the fourth channel. The four-channel input is propagated through an encoder CNN and the extracted feature map is then fed into the *Level Set Branch* to regress to the initial LSF. A popular type of LSF is the signed distance function (SDF) of the curve. Instead of predicting the SDF, we choose the truncated signed distance function (TSDF) as our LSF with a threshold D ($D = 30$ in our work), where

$$\phi_{TSDF}(x,y) = \text{sgn}(\phi_{SDF}(x,y)) \, \min\{|\phi_{SDF}(x,y)|, D\}.$$

This reduces the variance of the output and makes the training process more stable. The Level Set Branch aims to predict the initial LSF $\phi_{0,\theta}(x,y)$ to be as close to $\phi_{TSDF}(x,y)$ as possible. We use the subscript θ to indicate that ϕ is predicted and thus a function of the network's parameters θ.

4.2. DELSE Components

The core of the level set methods is the definition of the level set terms, which define the rules for the level set evolution. The level set evolution typically consists of several different update terms which can be roughly divided into two categories: (1) External terms that attract the curve to a desired location based on the data evidence, such as edges with strong gradients; and (2) Internal regularization terms on the curve's shape, *e.g.* curvature and length of the curve.

In our work, we carefully design three different terms that best exploit deep neural networks to perform efficient level set evolution, which we describe next.

Motion Term: Since deep neural networks have the ability to extract both low-level details and high-level semantics, we employ a CNN to predict the external term used in level set evolution. Specifically, we feed the feature map into a branch which we refer to as the *Motion Branch*, to predict the *motion map* $\vec{V}_\theta(x,y)$. The motion map consists of a vector at each pixel, and forms a vector field indicating the direction and magnitude of the motion of the curve. Ideally, the direction of the curve's motion during evolution should efficiently minimize the level set energy. We use the negative gradient of the ground-truth distance function as the ground-truth direction $\vec{U}_{\text{gt}}$ of the motion map. We borrow this term from [4] who used it to help a CNN predict the energy of a watershed transform. We can compute it as:

$$\vec{U}_{\text{gt}}(x,y) = -\frac{\nabla \phi_{DT}(x,y)}{|\nabla \phi_{DT}(x,y)|} \tag{5}$$

where ϕ_{DT} denotes distance transform of GT boundary.

Consider a curve evolving in the vector field $\vec{V}_\theta$. According to the level set equation, the update term, *i.e.* motion term for LSF can be written as

$$\left[\frac{\partial \phi_i}{\partial t}\right]_{\text{motion}} = -\langle \vec{V}_\theta, \nabla \phi_i \rangle \tag{6}$$

We use a subscript to indicate that the gradient update will consist of several terms. Traditionally, edge based terms such as Laplacian of Gaussian features and expansion force such as the balloon term [15] have been used to attract curves to object boundaries. The motion term above has the functionality of both. It can learn to act as an edge detector to make the evolved boundary more precise, and can act as the expansion force that prevents the curve from collapsing. It also has the following advantages. Firstly, since the traditional active contours have the tendency to shrink, the initial LSF is usually initialized outside the object. The proposed motion term allows the initial LSF to be both inside and outside of the object. Secondly, people are usually interested in the geometry of the curve and the level set evolution only uses the projection of $\vec{V}_\theta$ onto the normal direction of the curve. Thus, small angular errors (smaller than $90°$) of $\vec{V}_\theta$ is tolerable and will still facilitate the evolution process.

Curvature Term: We further regularize the predicted curve by moving it in the direction of its curvature. In most cases this will help to smooth the curve and eliminate the noise on the boundary. However, in practice, some objects may have sharp corners, and thus directly applying this regularization may hurt the model's performance. To address this, we introduce the *Modulation Branch* to predict a modulation function $m_\theta(I, P) \in [0, 1]$ to selectively regularize the curve. Let κ denote the curvature. The curvature term for the level set can thus be written as

$$\begin{aligned}\Big[\frac{\partial \phi_i}{\partial t}\Big]_{\text{curvature}} &= m_\theta \, \kappa |\nabla \phi_i| \\ &= m_\theta \, |\nabla \phi_i| \, \text{div}\Big(\frac{\nabla \phi_i}{|\nabla \phi_i|}\Big).\end{aligned} \tag{7}$$

This gives the flexibility to the model to preserve the real sharp corners around the object and only remove the noise that damages the shape of the curve.

Regularization Term: A desirable shape of LSF $\phi(x, y)$ could be a signed distance function of the corresponding contour. During the evolution of LSF, however, irregularites may occur and will cause instability and numerical errors in the final result. In this paper, we follow the remedy proposed in [25], and introduce the distance regularization term to restrict the behavior of LSF. Mathematically, the additional regularization term is

$$\Big[\frac{\partial \phi_i}{\partial t}\Big]_{\text{reg}} = \text{div}\Big(p'(|\nabla \phi_i|)\frac{\nabla \phi_i}{|\nabla \phi_i|}\Big) \tag{8}$$

div is divergence and p a double-well potential function:

$$p(s) = \begin{cases} \frac{1}{(2\pi)^2}(1 - \cos(2\pi s)), & \text{if } s \le 1 \\ \frac{1}{2}(s-1)^2, & \text{if } s \ge 1 \end{cases}$$

The function $p(s)$ has two local minima at $s = 1$ and $s = 0$. With the regularization term, $|\nabla \phi|$ is regularized to be either close to 0 or close to 1, thus helping to maintain the signed distance property of the LSF. In our DELSE, the Level Set Branch aims to predict the truncated signed distance function as LSF, which exactly matches with the regularization term.

The level set evolution of our full DELSE can finally be described as the sum of all three terms:

$$\begin{aligned}\frac{\partial \phi_i}{\partial t} = &- \langle \vec{V}_\theta, \nabla \phi_i \rangle + \lambda \cdot m_\theta \, |\nabla \phi_i| \, \text{div}\Big(\frac{\nabla \phi_i}{|\nabla \phi_i|}\Big) \\ &+ \mu \cdot \text{div}\Big(p'(|\nabla \phi_i|)\frac{\nabla \phi_i}{|\nabla \phi_i|}\Big)\end{aligned} \tag{9}$$

where $\vec{V}_\theta$ is the direction map predicted by the network, m_θ is the predicted modulation function, and λ and μ weight different terms. With the initial LSF $\phi_{0,\theta}$ predicted by the network, the level set evolves T steps with a time step Δt as mentioned in Eq. 4. The evolution process is differentiable and can thus be trained end-to-end.

4.3. Network Training

To facilitate training, we first pre-train the three branches of our model, and then jointly train the model using our formulation. We provide details in this section. The training process is also summarized in Algorithm 1.

Multi-Task Pre-training: During pre-training, three types of losses are jointly optimized with a multi-task loss:

$$L_{\text{pre}}(\theta) = L_0(\theta) + \alpha \, L_T(\theta) + \beta \, L_{\text{direct}}(\theta) \tag{10}$$

where α, β are the weights of the different loss terms. We describe different loss terms next.

Level Set Branch Supervision: The level set branch predicts the initial LSF $\phi_{0,\theta}$. During the pre-training process, we employ the mean square error as our objective function

$$L_0(\theta) = \sum_{(i,j)} (\phi_{0,\theta}(i,j) - \phi_{\text{gt}}(i,j))^2 \tag{11}$$

where ϕ_{gt} is the truncated signed distance function of the ground-truth contour.

Modulation Branch Supervision: During pre-training, we simulate initial LSF $\tilde{\phi}_0$ to train this branch. We do this by shifting ground-truth LSF ϕ_{gt} with a distance Δh,

$$\tilde{\phi}_0(x, y) = \phi_{\text{gt}}(x, y) + \Delta h \tag{12}$$

where Δh is uniformly sampled from $[-5, 5]$. The randomly shifted LSF will zoom in or out of the ground-truth contour. We then evolve the $\tilde{\phi}_0$ for T steps and generate the output LSF $\tilde{\phi}_T$ after T steps based on the predicted term m_θ and $\vec{V}_\theta$. During pre-training, the model learns to fix the random shift and predicts the correct position of the object boundary.

We employ a weighted binary cross entropy loss to supervise the output $H(\tilde{\phi}_T)$, where

$$H(s) = \begin{cases} 1, & s \ge 0 \\ 0, & s < 0 \end{cases} \tag{13}$$

Algorithm 1 DELSE Training Algorithm

Input: $\mathcal{I}$ (images), $\mathcal{P}$ (extreme points), $\mathcal{M}$ (GT masks)

for $I_i, P_i, M_i \in \mathcal{I}, \mathcal{P}, \mathcal{M}$ **do** ▷ Pre-training Loop
  $\phi_0, \vec{V}, m \leftarrow CNN_\theta(I_i, P_i)$ ▷ Forward Pass
  $L_0 \leftarrow \text{LevelSetLoss}(\phi_0, M_i)$
  $L_{\text{direct}} \leftarrow \text{DirectionLoss}(\vec{V}, M_i)$

  $\tilde{\phi}_0 \leftarrow \text{LSFSimulation}(M_i)$
  for $j \leftarrow 0$ to $T-1$ **do** ▷ Level Set Evolution
    $\tilde{\phi}_{i+1} = \tilde{\phi}_i + \Delta t \frac{\partial \tilde{\phi}_i}{\partial t}$
  $L_T \leftarrow \text{EvolutionLoss}(\tilde{\phi}_T, M_i)$

  $L_{\text{pre}} \leftarrow L_0 + \alpha L_T + \beta L_{\text{direct}}$ ▷ Calculate Loss
  Update Network: $\theta \leftarrow \theta - \eta \frac{\partial L_{\text{pre}}}{\partial \theta}$

for $I_i, P_i, M_i \in \mathcal{I}, \mathcal{P}, \mathcal{M}$ **do** ▷ Joint Training Loop
  $\phi_0, \vec{V}, m \leftarrow CNN_\theta(I_i, P_i)$
  for $j \leftarrow 0$ to $T-1$ **do**
    $\phi_{i+1} = \phi_i + \Delta t \frac{\partial \phi_i}{\partial t}$
  $L_T \leftarrow \text{EvolutionLoss}(\phi_T, M_i)$
  Update Network: $\theta \leftarrow \theta - \eta \frac{\partial L_T}{\partial \theta}$

is the Heaviside function. Since it only has effect on the zero level set, we replace it with the approximated Heaviside function with a parameter ϵ

$$H_\epsilon(s) = \frac{1}{2}\Big(1 + \frac{2}{\pi}\arctan\big(\frac{s}{\epsilon}\big)\Big). \tag{14}$$

Thus the loss can be written as

$$\begin{aligned} L_T(\theta) = \sum_{(i,j)} & -w_p\, M_{\text{gt}}(i,j) \log\, H(\tilde{\phi}_{T,\theta}(i,j)) \\ & - w_n \big(1 - M_{\text{gt}}(i,j)\big) \log(1 - H(\tilde{\phi}_{T,\theta}(i,j))) \end{aligned} \tag{15}$$

where w_p and w_n are weights for the positive (foreground) and negative (background) classes, respectively. Here, M_{gt} denotes the ground-truth segmentation mask. Since the whole process is differentiable, the network learns to generate the modulation function m_θ for the curvature term.

Motion Branch Supervision: The gradient of the motion branch during pre-training comes from two parts. First, using the simulated initial LSF $\tilde{\phi}$ and the loss L_T mentioned above, the network can automatically learn the direction and magnitude of the vector field. Second, following [4], we utilize mean square error in the angular domain to enforce additional constraints on the direction of motion vectors,

$$L_{\text{direct}}(\theta) = \sum_{(i,j)} \Big(\cos^{-1} \Big\langle \frac{\vec{V}_\theta(i,j)}{|\vec{V}_\theta(i,j)|}, \vec{U}_{\text{gt}}(i,j) \Big\rangle \Big)^2. \tag{16}$$

This loss helps the network to learn the correct direction of the vector field, but leaves the magnitude unsupervised.

End-to-end Joint Training: After pre-training, we directly evolve the predicted initial LSF $\phi_{0,\theta}$ for T steps to get the final output $\phi_{T,\theta}$. Then we use the weighted binary cross entropy loss in Eq. 15 to supervise the output $\phi_{T,\theta}$.

4.4. Interactive Annotation by Motion Field Editing

We aim to give additional control to the annotator to interactively correct any mistakes produced by the model. In DEXTR [29], the annotator iteratively clicks on a correct boundary point in order to guide the model make a better prediction. The authors simply add corrected points to the channel containing extreme points. We here additionally enable the annotator to drag and drop an erroneous boundary point onto the correct one. In the language of the level set method, one can think of this as providing a corrected motion vector. This motion vector is then exploited in our model to re-predict the level set energy. We thus refer to this approach as *motion field editing*. We first describe how we simulate human interaction during training, and then provide details of how the model incorporates this information.

Human-in-Loop Simulation: To train our model to exploit human corrections we need to simulate these during training. Specifically, we find the most erroneous predicted boundary point $(x,y)^*_{\text{pred}}$ as $\arg\max_{\phi_T(x,y)=0} \phi_{DT}(x,y)$, with ϕ_{DT} a distance transform of the ground-truth contour. The corresponding GT contour point $(x,y)_{\text{gt}}$ is found as $\arg\min_{(x,y)_{\text{gt}}} ||(x,y)^*_{\text{pred}} - (x,y)_{\text{gt}}||^2$. Simulated correction is defined as dragging a point $(x,y)^*_{\text{pred}}$ to $(x,y)_{\text{gt}}$.

Motion Field Editing: We incorporate the resulting 2D vector into our model in two different places. First, we follow [29] and add the corrected point to the channel containing the extreme points. Secondly, we create two Gaussian heatmaps around the location of the erroneous point, encoded in two separate channels. The σ of the Gaussian is set to be $||(x,y)^*_{\text{pred}} - (x,y)_{\text{gt}}||$ in both channels, indicating the magnitude of the corrected vector. We then multiply the two channels with $|x^*_{\text{pred}} - x_{\text{gt}}|$ and $|y^*_{\text{pred}} - y_{\text{gt}}|$, respectively, where we normalize these values with the norm of the motion vector. This indicates the magnitude of change in each axis. We tried several different ways of encoding the motion vector, and this yielded the best results. The encoded vector is concatenated with the predicted motion field $\vec{V}_\theta(x,y)$, and a new motion field is re-predicted using a residual convolutional block. In particular, this block consists of 6 residual convolutional layers with 128 channels followed by a convolutional layer with 2 channels. With the newly predicted motion field, we simply re-run level-set evolution to get the repredicted segmentation mask. Fig. 2 visualizes the model.

5. Experimental Results

We evaluate our method on several datasets: PASCAL [19], SBD [20], Cityscapes [16], DAVIS 2016 [34].

Implementation Details: Our DELSE employs ResNet-101 as the encoder CNN and a PSP module [40] for each of the three prediction branches. Additionally, considering that curve evolution relies on both low-level information and high-level semantics, we follow [3] and further add skip-connections in the encoder CNN to aggregate both low-level

Model	Bicycle	Bus	Person	Train	Truck	Motorcycle	Car	Rider	**mIoU**
DEXTR*	71.92	87.42	78.36	78.11	84.88	72.41	84.62	75.18	79.11
DELSE*	**74.32**	**88.85**	**80.14**	**80.35**	**86.05**	**74.10**	**86.35**	**76.74**	**80.86**
DEXTR [29]	76.36	88.58	82.44	76.40	87.53	75.20	87.17	79.06	81.59
Level Set Regression	76.05	88.21	82.40	78.69	86.50	74.31	87.17	78.99	81.54
DELSE	**77.83**	**89.56**	**83.42**	**82.45**	**88.11**	**77.16**	**88.29**	**79.98**	**83.35**

Table 1: Performance (mIoU) on Cityscapes-Stretching. Method with * is without extreme points annoation.

Model	Bicycle	Bus	Person	Train	Truck	Motorcycle	Car	Rider	**mIoU**
Polygon-RNN++ [3]	63.06	81.38	72.41	64.28	78.90	62.01	79.08	69.95	71.38
DELSE*	**67.15**	**83.38**	**73.07**	**69.10**	**80.74**	**65.29**	**81.08**	**70.86**	**73.84**

Table 2: Performance (mIoU) on Cityscapes-Hard. DELSE* is without extreme points annotation.

Model	F mean(1 pix)	F mean(2 pix)
DEXTR*	54.00	68.60
DELSE*	**60.29**	**74.40**
DEXTR	60.65	73.85
Level Set Regression	58.87	72.08
DELSE	**64.35**	**77.62**

Table 3: Boundary evaluation on Cityscapes-Stretching. Method with * is without extreme points annotation.

Model	F mean(1 pix)	F mean(2 pix)
Polygon-RNN++ *	46.57	62.26
DELSE*	**48.59**	**64.45**

Table 4: Boundary evaluation on Cityscapes-Hard. DELSE* is without extreme points annotation.

Model	PASCAL	PASCAL + SBD
DEXTR	90.5	91.5
Level Set Regression	87.7	88.7
DELSE	90.5	91.3

Table 5: Performance (mean IoU) on PASCAL and SBD.

and high-level features. We use 3x3 conv filters with batch normalization and ReLU activations. We refer to the feature map with skip-connection architecture as *skip features*.

On DAVIS and PASCAL, we use image resolution of 512x512 following DEXTR [29]. We use the GT box to crop the image, where we expand it by 50 pixels in each direction as in [29]. On Cityscapes, we use input resolution of 224x224 following PolygonRNN++ [3] and we expand the box by 10 pixels due to the large number of tiny instances.

The derivatives in DELSE are computed by mimicking the central difference scheme. We use $T = 5$ evolution steps for both training and inference. The ratio of evolution terms is chosen to be $\lambda = 1, \mu = 0.04$, respectively, while the parameter ϵ for the approximated Heaviside function is set to 1. We weigh various losses during pre-training with $\alpha = 100, \beta = 1$, respectively. The model is pre-trained for 20 epochs and jointly trained for 60 epochs. The initial learning rate is set to 3e-4, and decayed by 0.3 every 20 epochs. We use SGD with momentum of 0.9 and use a batch size of 12. Training on PASCAL takes around 30 GPU hours, and inference time is 160 milliseconds per object instance.

Evaluation Metrics: We utilize two quantitative measures to evaluate our model: **1)** Intersection-over-union (IoU), and **2)** since getting accurate boundaries is crucial, we use the metric proposed in [34] to additionally evaluate accuracy around the boundaries. The boundary metric uses a given *threshold* to compute precision and recall along the contour of the predicted mask and then computes *F-measure* as a trade-off between both. The official boundary threshold in [34] is quite loose, thus not well reflecting the performance of different models. On DAVIS, in addition to the official boundary threshold, we perform a multi-scale evaluation by ranging the threshold from 1 to 10 pixels. Since the Cityscapes dataset is finely annotated and there is a considerable number of small instances, we set the boundary threshold to be 1 and 2 pixels.

Baselines: We compare DELSE with Polygon RNN++ [3] and DEXTR [29] as they represent the current state-of-the-art methods for object annotation. We also compare against our own baseline which we refer to as *Level Set Regression*, which regresses to LSF. Note that Polygon-RNN++ takes the bounding box as input (2 clicks), thus requiring less human guidance than our/DEXTR models which exploit extreme points (4 clicks). For a fair comparison, we run DEXTR and DELSE on Cityscapes with only the cropped image as input. We follow [29] to find the optimal segmentation threshold for both DEXTR and DELSE.

5.1. Datasets

Cityscapes: The Cityscapes dataset contains 5000 finely annotated images of driving scenes, including 2975 images for training, 500 for validation and 1525 for testing. Eight object classes are provided with per-instance annotation. To make a fair comparison, we follow the same split as in [9, 3]. We evaluate in two different settings. In one, we stretch the cropped bounding box image into a square, and denote this dataset as **Cityscapes-Stretching**. However, in Polygon-RNN++ [3], the authors enlarged the bounding box to be square and then cropped the image inside it. This makes for a considerably harder prediction task since the image crop may now contain multiple objects. In some cases this may lead to ambiguities about which object is required to be labeled, thus somehow artificially decreasing performance. We denote this setting as **Cityscapes-Hard**. We compare with PolygonRNN++ [9, 3] on **Cityscapes-Hard** to allow

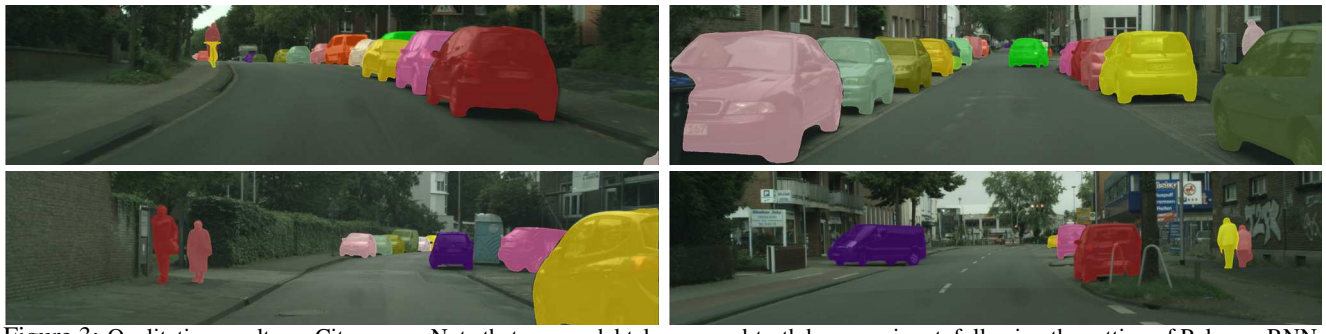

Figure 3: Qualitative results on Cityscapes. Note that our model takes ground-truth boxes as input, following the setting of Polygon-RNN.

Figure 4: Qualitative results for occluded objects on Cityscapes. **Top row**: ground-truth, **Bottom row**: Prediction from DELSE.

for a fair comparison, and compare with DEXTR [29] on the **Cityscapes-Stretching** dataset.

PASCAL and SBD: The PASCAL VOC dataset [19] contains 3507 instances in training and 3427 instances in the val set. Semantic Boundaries Dataset (SBD) [20] consists of instance-level annotation for 11355 images taken from the PASCAL VOC 2011 dataset. We follow [29] and used additional annotation from SBD to augment the PASCAL training set and evaluate on the PASCAL val set. The augmented training set contains 10582 images in total.

DAVIS 2016: The DAVIS dataset consists of 3455 finely annotated frames in 50 video sequences. We follow the official split and evaluation metrics and perform frame-by-frame segmentation to evaluate our method.

5.2. Evaluation in Automatic Mode

IoU metrics: The IoU performance of our model on Cityscapes is shown in Tables 1, 2. Results show that DELSE outperforms state-of-the-art methods in both the bounding box input and extreme points. On PASCAL, we follow [29] and conduct experiments on both official splits and splits augmented with additional annotation from SBD [20]. Performance is reported in Table 5. Our model significantly improves over our baseline, *i.e*. Level Set Regression method and performs on par with DEXTR. As shown in Fig. 6, labels in PASCAL are quite noisy which may harm the performance of our model for two main reasons. Firstly, as a curve-based method, DELSE is sensitive to the boundary quality of training samples. As such, the coarsely annotated boundaries on PASCAL makes learning a complex function even harder. Secondly, evaluation on PASCAL val dataset ignores pixels on the boundary which is indeed a crucial part to evaluate the performance of curve-based models.

Boundary metrics: The quality of the predicted boundaries is important for object annotation. Tables 3 and 4 compare our model vs state-of-the-art on the Cityscapes dataset. Results show that our model more accurately annotates object boundaries. However, we point out that we conduct a threshold search for both pixel-wise methods (DELSE and DEXTR), while this is not possible for Polygon-RNN++, leading to a disadvantage in this evaluation.

On DAVIS 2016, we perform multi-scale evaluation of different models, i.e. changing the threshold from 1 to 10 pixels. The results are shown in Fig. 5. Our method outperforms all baselines in the strictest regime. This shows that DELSE consistently produces better segmentation boundaries.

Qualitative Results: Examples of qualitative results on the Cityscapes dataset are shown in Fig. 3. We remind the reader that our model exploits ground-truth boxes for prediction. Compared to Polygon-RNN++ which predicts a single polygon around the object, our DELSE is able to successfully handle objects with multiple components. Fig. 4 visualizes a few examples. We show qualitative results of different evolution terms in Fig. 7. The motion map shows the ability to fit the curve to precise boundaries. It is interesting to note that the modulation function automatically learns to perform as an inverse edge detector on the salient object, which is similar to the terms used in traditional level set methods. Notice that the initial LSF is already quite close to the groundtruth, which largely reduces the steps required for evolution. With the curve evolution, some of the flaws are fixed and the boundaries are improved overall.

Ablation Study: We conduct an ablation study on individual components in DELSE including the motion and modulation terms, and skip features. The results are shown

Model	J mean	J recall	F mean	F recall
DEXTR	82.4	94.2	84.5	93.5
Level Set Regression	81.7	90.9	83.4	91.4
+ Motion Term	84.0	94.9	84.7	94.0
+ Modulation Term	84.8	95.0	87.5	**95.1**
+ Skip Features	**85.6**	**95.1**	**87.8**	94.8
– Extreme Points	83.9	93.6	85.0	92.7

Table 6: Performance and ablation study on DAVIS 2016. J is short for the Jaccard (IoU) metric.

Noise of Input	J mean	J recall	F mean	F recall
±0%	85.6	95.1	87.8	94.8
±5%	85.1	95.1	87.4	94.8
±10%	84.9	95.0	86.9	94.7

Table 7: Evaluation of noisy input on DAVIS 2016.

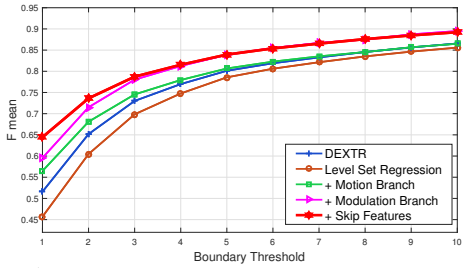

Figure 5: Multi-scale boundary evaluation on DAVIS 2016.

Model	mIOU	F mean
DELSE (Full data)	83.35	77.62
DELSE* (10 of 16 cities)	82.45	75.85
Motion Editing Clicks	**mIOU**	**F mean**
1	**84.73**	**79.64**
2	**85.97**	**81.34**
3	**86.83**	**82.52**
Extreme Points Clicks	**mIOU**	**F mean**
1	83.60	78.27
2	84.49	79.67
3	84.94	80.53

Table 8: Interactive correction on Cityscapes. Corrections are used with DELSE*, which is trained on 10 out of 16 cities.

in Table 6. We start with *Level Set Regression*, which contains only the Level Set Branch that directly regresses LSF. After adding the Motion Branch and using level set evolution to get the final result, the IoU performance increases by 2.3% and 1.3% w.r.t. mean boundary metric. Adding the Modulation Branch and applying selective regularization on the curvature term leads to a further boost of 0.8% in mIoU and 2.8% for the boundary metric. Results indicate that the motion term plays an important role in improving mIoU, and adding the modulation function increases the boundary quality. These results are consistent with the function of different terms, and prove the effectiveness of our proposed level set evolution scheme. Using skip features also increases the performance of the model. This is reasonable as the level set terms rely on both low-level details and high-level semantics to find the precise boundaries. Multi-scale evaluation results in Fig. 5 further show the impact of different components.

Figure 6: Qualitative results on PASCAL. (Left: GT, green indicates void pixels excluded in evaluation. Right: Prediction.)

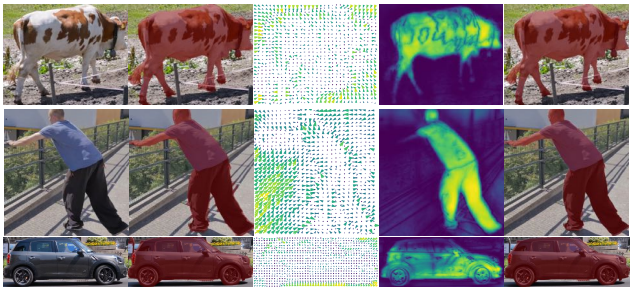

Figure 7: Qualitative results on DAVIS 2016. (From left to right: image, initial LSF, motion map, modulation function, final result.)

5.3. Interactive Image Annotation

We evaluate the interactive methods on the Cityscapes dataset. To train the human-in-the-loop model, we split the 16 cities from the training set into 10 used for training the original DELSE, and the remaining 6 to fine-tune the interactive model. We do this because DELSE trained on the full training set has too few errors on the same set.

We show results for up to 3 correction clicks, where we re-predict segmentation after every click. Results are reported in Table 8. Notice that both boundary clicks and motion editing increase performance, indicating ability of incorporating human guidance. In addition, motion editing outperforms the regime where only boundary clicks are provided, whereby adding only a small overhead on interaction.

Noisy Annotator: In this experiment we simulate a lazy annotator loosely clicking on extreme points. We do this by uniformly adding certain amount of noise to the ground-truth extreme points. Results shown in Table 7 indicate that our model is quite robust to noise.

6. Conclusion

We presented Deep Extreme Level Set Evolution for interactive object annotation. Our approach combines the power of convolutional networks with traditional curve evolution techniques in an end-to-end fashion. DELSE is shown to outperform existing state-of-the-art approaches on several benchmarks. It produces crisp object boundaries, highlighting its value as an interactive annotation tool.

Acknowledgements. We gratefully acknowledge support from Vector Institute, the Tsinghua University Initiative Scientific Research Program, and NVIDIA for donation of GPUs. S.F. also acknowledges the Canada CIFAR AI Chair award at Vector Institute. We thank Jun Gao for help and advice, and Relu Patrascu and Priyank Thatte for infrastructure support.

References

[1] http://medicaldecathlon.com/.

[2] D. Acuna, A. Kar, and S. Fidler. Devil is in the edges: Learning semantic boundaries from noisy annotations. In *CVPR*, 2019.

[3] D. Acuna, H. Ling, A. Kar, and S. Fidler. Efficient interactive annotation of segmentation datasets with polygon-rnn++. In *CVPR*, 2018.

[4] M. Bai and R. Urtasun. Deep watershed transform for instance segmentation. In *CVPR*, pages 2858–2866, 2017.

[5] A. Bearman, O. Russakovsky, V. Ferrari, and L. Fei-Fei. What's the point: Semantic segmentation with point supervision. *arXiv:1506.02106*, 2016.

[6] Y. Boykov and M.-P. Jolly. Interactive graph cuts for optimal boundary & region segmentation of objects in nd images. In *ICCV*, 2001.

[7] V. Caselles, F. Catté, T. Coll, and F. Dibos. A geometric model for active contours in image processing. *Numerische Mathematik*, 66(1):1–31, Dec 1993.

[8] V. Caselles, R. Kimmel, and G. Sapiro. Geodesic active contours. *IJCV*, 22(1):61–79, 1997.

[9] L. Castrejón, K. Kundu, R. Urtasun, and S. Fidler. Annotating object instances with a polygon-rnn. In *CVPR*, 2017.

[10] T. F. Chan and L. A. Vese. Active contours without edges. *IEEE Transactions on Image Processing*, 10(2):266–277, Feb 2001.

[11] L. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *CoRR*, abs/1606.00915, 2016.

[12] L.-C. Chen, S. Fidler, A. Yuille, and R. Urtasun. Beat the mturkers: Automatic image labeling from weak 3d supervision. In *CVPR*, 2014.

[13] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Semantic image segmentation with deep convolutional nets and fully connected crfs. In *ICLR*, 2015.

[14] D. Cheng, R. Liao, S. Fidler, and R. Urtasun. Darnet: Deep active ray network for building segmentation. In *CVPR*, 2019.

[15] L. D. Cohen. On active contour models and balloons. *CVGIP: Image Understanding*, 53(2):211 – 218, 1991.

[16] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. The cityscapes dataset for semantic urban scene understanding. In *CVPR*, 2016.

[17] D. Cremers, O. Fluck, M. Rousson, and S. Aharon. A probabilistic level set formulation for interactive organ segmentation. In *SPIE*, 2007.

[18] D. Cremers, M. Rousson, and R. Deriche. A review of statistical approaches to level set segmentation: Integrating color, texture, motion and shape. *IJCV*, 72(2):195–215, Apr 2007.

[19] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *IJCV*, 88(2):303–338, June 2010.

[20] B. Hariharan, P. Arbelaez, L. Bourdev, S. Maji, and J. Malik. Semantic contours from inverse detectors. In *ICCV*, 2011.

[21] P. Hu, B. Shuai, J. Liu, and G. Wang. Deep level sets for salient object detection. In *CVPR*, volume 1, page 2, 2017.

[22] M. Januszewski, J. Kornfeld, P. H. Li, A. Pope, T. Blakely, L. Lindsey, J. Maitinshepard, M. Tyka, W. Denk, and V. Jain. High-precision automated reconstruction of neurons with flood-filling networks. *Nature Methods*, 2018.

[23] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *IJCV*, 1(4):321–331, 1988.

[24] S. Kichenassamy, A. Kumar, P. Olver, A. Tannenbaum, and A. Yezzi. Conformal curvature flows: From phase transitions to active vision. *Archive for Rational Mechanics and Analysis*, 134(3):275–301, Sep 1996.

[25] C. Li, C. Xu, C. Gui, and M. D. Fox. Distance regularized level set evolution and its application to image segmentation. *IEEE Trans. Image Proc.*, 19(12):3243–3254, Dec 2010.

[26] H. Ling, J. Gao, A. Kar, W. Chen, and S. Fidler. Fast interactive object annotation with curve-gcn. In *CVPR*, 2019.

[27] S. Liu, S. De Mello, J. Gu, G. Zhong, M.-H. Yang, and J. Kautz. Learning affinity via spatial propagation networks. In *NIPS*, pages 1520–1530. 2017.

[28] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015.

[29] K.-K. Maninis, S. Caelles, J. Pont-Tuset, and L. Van Gool. Deep extreme cut: From extreme points to object segmentation. In *CVPR*, 2018.

[30] D. Marcos, D. Tuia, B. Kellenberger, L. Zhang, M. Bai, R. Liao, and R. Urtasun. Learning deep structured active contours end-to-end. In *CVPR*, 2018.

[31] E. N. Mortensen and W. A. Barrett. Intelligent scissors for image composition. In *SIGGRAPH*, pages 191–198, 1995.

[32] S. Osher and J. A. Sethian. Fronts propagating with curvature-dependent speed: Algorithms based on hamilton-jacobi formulations. *J. Comput. Phys.*, 79(1):12–49, Nov. 1988.

[33] N. Paragios and R. Deriche. Geodesic active regions for supervised texture segmentation. In *ICCV*, volume 2, pages 926–932 vol.2, Sept 1999.

[34] F. Perazzi, J. Pont-Tuset, B. McWilliams, L. Van Gool, M. Gross, and A. Sorkine-Hornung. A benchmark dataset and evaluation methodology for video object segmentation. In *CVPR*, 2016.

[35] R. Ronfard. Region-based strategies for active contour models. *IJCV*, 13(2):229–251, Oct 1994.

[36] C. Rother, V. Kolmogorov, and A. Blake. Grabcut: Interactive foreground extraction using iterated graph cuts. In *SIGGRAPH*, 2004.

[37] C. Rupprecht, E. Huaroc, M. Baust, and N. Navab. Deep active contours. *arXiv preprint arXiv:1607.05074*, 2016.

[38] M. Tang, S. Valipour, Z. V. Zhang, D. Cobzas, and M. Jägersand. A deep level set method for image segmentation. *CoRR*, abs/1705.06260, 2017.

[39] Z. Zhang, S. Fidler, and R. Urtasun. Instance-level segmentation for autonomous driving with deep densely connected mrfs. In *CVPR*, 2016.

[40] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia. Pyramid scene parsing network. *CoRR*, abs/1612.01105, 2016.

[41] A. Zlateski, R. Jaroensri, P. Sharma, and F. Durand. On the importance of label quality for semantic segmentation. In *CVPR*, June 2018.