A comparative study between Deep Active Contours and Gradient Vector Flow Snakes for interactive boundary extraction

Deepak Muralidharan
Electrical Engineering, UCLA
deepakmuralidharan2308@gmail.com

Abstract

In this project, we do a comparative study between the Deep Active Contour model and the classic Gradient vector flow (GVF) based Snake active contours for applications in interactive boundary extraction. As opposed to the GVF snakes contour model which is an energy minimization based approach, the Deep Active Contour model tries to learn the vector from the respective point on the evolving contour towards the closest point on the boundary of the object of interest. We train a convolutional neural network (CNN) model which learns to predict this direction vector using a deep patch based representation and evolve the contour in this direction. We provide a visual and quantitative evaluation of both these methods on a few sample images from the densely annotated DAVIS segmentation dataset.

1. Introduction

Snakes, or active Contours have become a popular tool for computer vision applications such as edge detection, segmentation [8], motion tracking [14]. Snakes are basically curves defined with an image which can confirm to object boundaries or other features in an image under the influence of an internal and external force. Ever since their introduction in 1998 [5], various efforts in deformable models literature have focused on improving the robustness of these contours and many approaches such as [7], [3], [4] have been proposed.

More recently, deep learning approaches to computer vision applications have become quite common. For example, in [16], the authors use a deep learning based approach for road scene segmentation. Many other papers have been published in the past few years [10], [1] which use convolutional neural networks for image segmentation. In this project, we combine the world of deep learning and active contours and implement a deep active contour model [12]. This model learns a deep patch based representation using a convolutional neural network and predicts a direction vector which is used to evolve the contour. We compare this deep learning based active contour approach with the gradient vector flow based Snakes method both visually and quantitatively.

This report is divided into three sections. In Section 2, we discuss about the theory and implementation of deep active contours. In Section 3, we give a brief overview of the gradient vector flow (GVF) based snakes algorithm. In Section 4, we briefly discuss the implementation details and finally in Section 5 we compare the performance of both methods.

2. Deep Active Contours

In this section, we explain the methodology of the deep active contour framework. As explained in the previous section, we train a convolutional neural network (CNN) which learns to predict a vector from the respective point on the evolving contour towards the closest point on the boundary of the object. We briefly give an overview of the convolutional neural network architecture that we used for this purpose, explain the deep active contour framework and finally describe our implementation methodology.

2.1. Convolutional Neural Networks

The (typical) CNN architecture that we used in this paper mainly consists of five layers **INPUT-CONV-RELU-POOL-FC** each of which is explained below:

- INPUT: dimension 64x64x3 holds the raw pixel value of the image. In our case, this is an image of width 64, height 64 and three color channels R,G,B.
- CONV: computes the output of neurons connected to local regions in the input, computing a dot product between their weights and a small region they are connected to in the input volume.
- RELU: applies an element-wise activation function such as max(0,x) thresholding at zero.

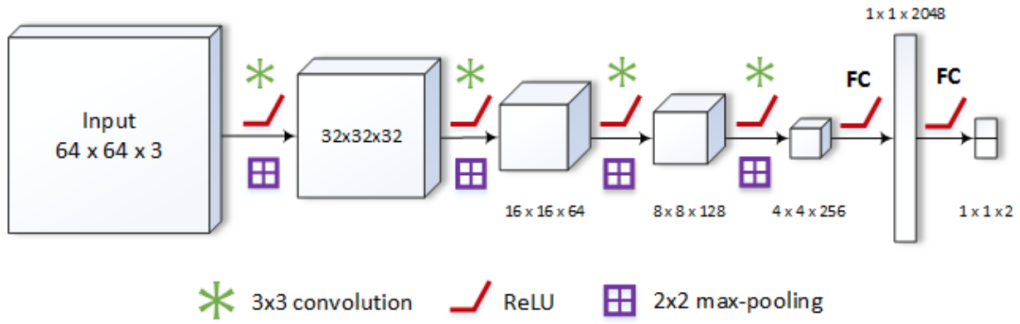


Figure 1: The convolutional neural network architecture that we used.

- POOL: performs a down-sampling operation along the spatial dimensions(width, height).
- FC: Fully connected layer gives the final output. In our case, the output is a gradient vector of size (2x1) pointing in the direction of the closest object.

The block diagram of our architecture is given in Fig 1.

2.2. Data Pre-processing

For training the CNN, we need to provide an input-output pair in the correct format. The dataset that we used for training the CNN is the DAVIS Segmentation dataset which is a densely annotated dataset. Each example in the dataset consists of a JPEG image along with its corresponding annotated version. An example is given in Fig 2. For our CNN input, we need to provide a patch of size 64x64x3 and for the corresponding output pair, we need to provide a gradient vector of size 2x1 pointing in the direction of the closest object.

For each example in the training dataset corresponding to a specific class (such as bear, blackswan etc.), we consider all level lines in the range of [-20,20] and at each point in a level line, we extract 64x64 patches surrounding that pixel. Using the ground truth segmentation of the image (Fig 2(b) we also generate a signed distance map (SDM) which defines the distance of each pixel to the closest boundary in the binary image. For example, a SDM value of 2 indicates that the pixel is 2 pixels away from the closest boundary. All pixels in the level line corresponding to 0 (ground truth boundary) will have a SDM value of 0, pixels outside the boundary will have positive SDM values and pixels within the boundary will have negative SDM values. In order to the find the direction of the pixel from the closest object boundary, we take the gradient of the signed distance map for each point on the level line (for which we already extracted 64x64 patches) which is a (2,1) vector $[G_x, G_y]$. This vector gives the x and y directions (in pixels) towards the closest object edge. An example of the input-output pair sent to the CNN is given in Fig 3.

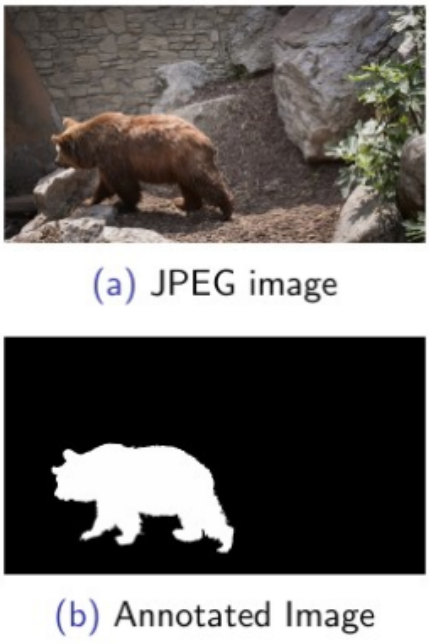


Figure 2: An example image pair from the DAVIS Segmentation Dataset. (a) denotes the JPEG image and (b) denotes the annotated version of the image.

2.3. Implementation methodology

As opposed to traditional active contour methods which are based on energy minimization, our deep active contour framework combines a deep patch-based representation with an active contour approach. We first initialize a contour around the image with K points. For each point

(a) Input: Patch of size 64x64x3

-12.0008
-12.9992

(b) Output: Direction to the closest object edge from center in pixels

Figure 3: An example of the input-output pair sent to the CNN. The input (a) is an image patch of 64x64x3 and the output (b) is a (2x1) direction vector pointing towards the closest object instance

$C_j \ \forall \ j = 1, \ldots, K$ on the contour, we extract a patch P_j around the point and send it through the trained CNN model which predicts a (2x1) direction vector v_j. We then evolve the contour using:

$$C_j^{i+1} = C_j^i + \tau v_j^i \eta_j^i \tag{1}$$

where τ is the step size and η_j is the normal vector at every point on the contour. We project the direction vector v_j onto the normal vector to ensure that the contour is not biased towards object edges but is propagated in the direction of the normal at each point on the contour. We perform this method for N iterations until the algorithm converges. Fig 4 gives an overview of the algorithm in a coherent way.

3. Gradient Vector Flow based Snakes

3.1. Brief overview

Gradient Vector Flow based Snakes [15] have been a popular variant of Snakes (or active contours) and have been widely used in many computer vision and image processing applications, specifically in locating object boundaries. The difference between the traditional snakes and the GVF snakes is that the GVF snakes can converge to a concave boundary and need not be initialized close to the boundary. The original snake **v** is a dynamic two dimensional contour represented by **v**(s) = [x(s), y(s)] where s ∈ (0,1) minimizes the function:

$$E = \int_0^1 E_{int}(\mathbf{v}(s)) + E_{img}(\mathbf{v}(s)) + E_c(v(s))ds \tag{2}$$

DEEP SNAKE EVOLUTION

for $i = 1, \cdots,$ N do
  for $j = 1, \cdots,$ K do
    $P_j^i \leftarrow$ samplePatch(C_j^i,I)
    $v_j^i \leftarrow$ predictVector(P_j^i)
  end
  for $j = 1, \cdots,$ K do
    $C_j^{i+1} \leftarrow C_j^i + \tau v_j^i . \eta_j^i$
  end
  if converged(C_j^{i+1}, C_j^i) then
    break
  end
end

Figure 4: Algorithm for deep snake evolution

where E_{int} defines the energy of the contour due to bending, E_{img} represents the energy due to intensity of the image and E_c is the constraint energy established by the user. The GVF snake method uses the GVF field as the constraint. Many other constraint energies such as balloon snakes [3], distance potentials [4] have been been proposed in literature.

3.2. Implementation methodology

As the first step to obtaining the GVF field, we need to extract the edge map function *f(x,y)* from image $I(x, y)$. Although the DAVIS segmentation dataset that we are using for evaluation consists of RGB images, for the purpose of this experiment we convert it into grayscale. The edge map function for the grayscale image is given by the following equations:

$$f(x, y) = -|\Delta I(x, y)|^2 \tag{3}$$

$$f(x, y) = -|\Delta(G_\sigma(x, y) * I(x, y))|^2 \tag{4}$$

where Δ is the gradient operator and G_σ is a Gaussian kernel (blurring factor) used to increase the capture range of the GVF snake.

The gradient vector flow (GVF) field can be defined as the vector field **g**(x,y) = (u(x,y), v(x,y)) that minimizes the

energy functional:

$$\varepsilon = \int\int \mu(u_x^2+u_y^2+v_x^2+v_y^2)+|\Delta f|^2|\mathbf{g}-\Delta f|^2 dxdy \quad (5)$$

Here μ is the parameter that manages the trade-off between the smoothing term (the first term) and the data term (the second term) respectively.

The GVF can be found by solving the two Euler equations:

$$\mu\Delta^2 u - (u - f_x)(f_x^2 + f_y^2) = 0 \quad (6)$$

$$\mu\Delta^2 v - (v - f_x)(f_x^2 + f_y^2) = 0 \quad (7)$$

where Δ^2 is the Laplacian operator. Both the equations can then be solved by treating u and v as functions of time t and solving the following generalized diffusion equations for t $\rightarrow \infty$.

$$u_t(x,y,t) = \mu\Delta^2 u(x,y,t) - (u(x,y,t) - f_x(x,y)) (f_x(x,y)^2 + f_y(x,y)^2)$$

$$v_t(x,y,t) = \mu\Delta^2 v(x,y,t) - (v(x,y,t) - f_x(x,y)) (f_x(x,y)^2 + f_y(x,y)^2)$$

Once we obtain the gradient vector field **g**(x,y) by solving the above equations, we can substitute this as the energy constraint E_c and the snake can be computed in an iterative manner.

4. Training and Implementation Details

We evaluated both our deep active contour model and the gradient vector flow based snakes model on a few sample images i.e. bear, blackswan and sheep from the DAVIS[1] Segmentation dataset [11] . This dataset consists of images of around 60 classes with full-HD high quality pics and contains dense annotations. It is a very challenging dataset as it contains objects with occlusions, varied viewpoints etc.

We used Tensorflow [2] for training the CNN to predict the direction vector, and it took $\tilde{1}0$ hours to train the network on a 8GB RAM CPU with i5 processor. Fig 5 plots the CNN training and validation loss with each iteration and we can clearly observe that as the number of epochs increase, both the training and the validation loss decreases and we stop the training when the validation loss starts to increase (as any further training will result in overfitting). The deep active contour model was implemented from scratch in MATLAB and for the GVF snakes algorithm, the reference code provided in [6] was used.

[1]This dataset can be downloaded from `https://graphics.ethz.ch/~perazzif/davis/code.html`

[2]`https://www.tensorflow.org/`

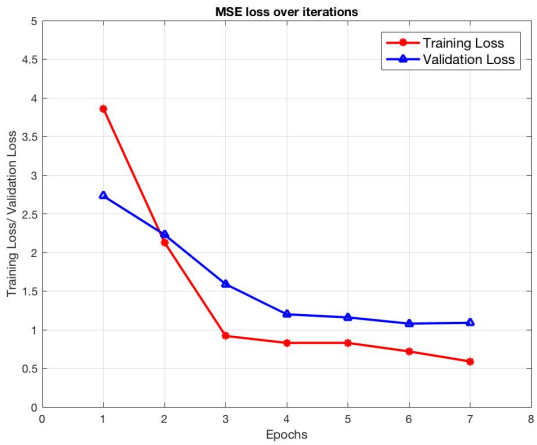


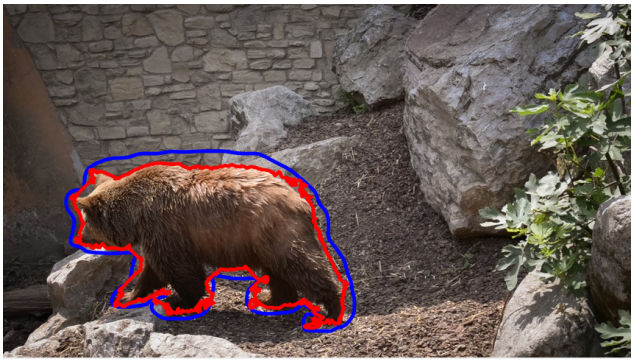
Figure 5: Training and validation loss for the CNN trained to predict the direction vector

5. Results and Comparison

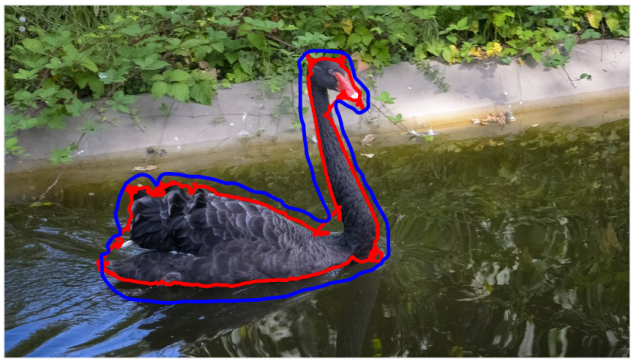
In this section, we discuss the results we obtained for the two methods and compare their performance. Fig 6 shows the performance of our deep active contour model on two images (bear and blackswan) from the DAVIS dataset. We manually initialize the contour and the deep active contour algorithm converges to extract the boundary of the desired object. It can be observed that the neural network is able to predict the direction vector very well even in difficult conditions such as varied lighting, shadow etc. However one disadvantage is that in regions where the edges are sharp, the contour starts twisting around itself leading to slightly incorrect segmentation in these contours. There are many regularization techniques which can be employed to smoothen this effect [2], [13], [9] and we propose this as future work.

Similarly, Fig 7 shows the performance of the Gradient vector flow based snakes method on a sample image from the same dataset. We can see that the snake evolves towards the object edge from a manual contour initialization. The algorithm performs very well for most part of the image, but in regions where the gradient map is not strong enough (such as the nose of the sheep), there are some mismatches in the segmentation. We also see that the regularization adds to the smoothening of the contour and the end segmentation is very good.

To compare both the methods, we evaluated both the methods on the same image shown in Fig 8. We can see that both the methods perform almost relatively well. The deep active contour method is able to fit the boundary more perfectly but does not perform well around sharp corners. On the contrary, the snakes method provides a nice smoothening effect around the corners but does not perform as well



(a)



(b)

Figure 6: Performance of the deep active contour model on two images i.e. (a) bear and (b) blackswan from the DAVIS dataset. In both the images, the initial contour is represented by blue color and the final contour is represented by red color.

when the gradient map is not strong enough in some regions of the image. In order to provide a quantitative evaluation of both the algorithms, we used the DICE index which compares the overlap of the final segmentation with the ground truth segmentation. The DICE index coefficient which varies between [0,1] is given by:

$$DICE = \frac{2|X \cap Y|}{|X| + |Y|} \tag{8}$$

where X is the binary ground truth segmentation, Y is the segmentation obtained from the contour model. A DICE index coefficient close to 1 indicates that the segmentation is good. The DICE index coefficients for the deep active contour method and the GVF snakes method is shown in Table 1. We see that the deep active contour method gives a better DICE coefficient compared to the GVF snakes method.

6. Conclusion

In this work, we implemented the deep active contour which combines a deep patch based representation with an active contour framework. We trained a class specific convolutional neural network (CNN) which learned to predict the direction of the closest object instance from each point on the contour and we used this direction vector to evolve the contour. For comparison, we also implemented the classic Gradient Vector Flow Snakes and compared both the methods. We evaluated the two methods on the newly released DAVIS segmentation dataset and observed that visually, both the methods perform equally well although quantitatively, the deep active contour method gives better DICE coefficients. From the results for the deep active contour model, we could observe that the deep patch based representation is able to different false edges such as shadows from the original object edge which is one of the advantages of this method. However, one disadvantage of the deep active contour method is that in a trade off for better segmentation accuracy, the neural network takes a long time (approximately 10 hours) to train. As future work, we can employ some regularization approaches to the deep active contour and possibly extend this concept to video segmentation with a better training time complexity.

Image	Deep Active Contours	GVF Snakes
Bear	0.9382	0.9387
Blackswan	0.9400	0.9000
Sheep	0.9678	0.9374

Table 1: A comparison of the DICE index values for Deep Active Contours and Gradient Vector Flow based snakes

7. Acknowledgment

I would like to thank Professor Demetri Terzopoulos for his excellent lectures from which I learnt a lot about deformable models and computer vision.

References

[1] V. Badrinarayanan, A. Handa, and R. Cipolla. Segnet: A deep convolutional encoderdecoder architecture for robust semantic pixel-wise labelling. arXiv preprint arXiv:1505.07293, 2015.

[2] G. Charpiat, P. Maurel, J. Pons, R. Keriven, and O. Faugeras. Generalized gradients: Priors on minimization flows. International Journal on Computer Vision (IJCV), 73(3):325-344, 2007.

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

[4] L. D. Cohen and I. Cohen. Finite-element methods for active contour models and balloons for 2-d and 3-d images. IEEE Trans. on Pattern Anal. Machine Intell., 15(11):1131-1147, 1993.

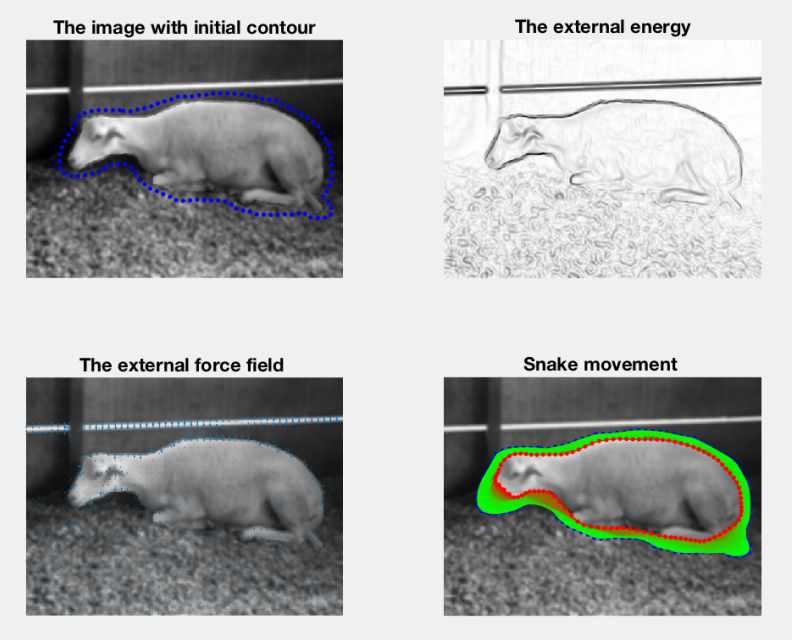


Figure 7: GVF algorithm on an example image. The image on the top left corner describes the grayscale image with the initial contour, the top right image is the edge map of the grayscale image, image on the bottom left shows the external force field, and the image on bottom right shows the snake movement.

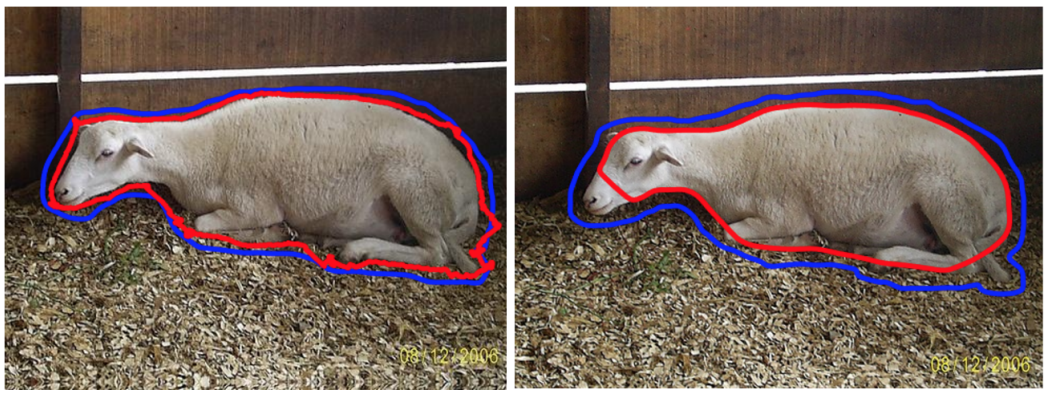


Figure 8: Comparison of the boundary extraction performance for deep active contours (left) and Gradient Vector Flow based snakes (right). In both the images, the initial contour is represented by blue color and the final contour is represented by red color.

[5] M. Kass, A. Witkin, and D.Terzopoulos. Snakes: Active contour models. International Journal of Computer Vision, 321–331, 1998.

[6] D. Kroon. https://www.mathworks.com/matlabcentral/ fileexchange/ 28149-snake-active-contour.

[7] B. Leroy, I. Herlin, and L. D. Cohen. Multi-resolution algorithms for active contour models. 12th International Conference on Analysis and Optimization of Systems, 58-65, 1996.

[8] F. Leymarie and M. D. Levine. Tracking deformable objects in the plane using an active contour model. IEEE Trans. on Pattern Anal. Machine Intell., 15(6):617-634, 1993.

[9] J. Melonakos, E. Pichon, S. Angenent, and A. Tannenbaum. Finsler active contours. IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI), 30(3):412-423, 2008.

[10] O.Ronneberger, P.Fischer, and T.Brox. U-net: Convolutional networks for biomedical image segmentation. Medical Image Computing and Computer-Assisted Intervention (MICCAI), 234-241, 2015.

[11] F. Perazzi, J. Pont-Tuset, B. McWilliams, L. V. Gool, M. Gross, and A. Sorkine-Hornung. A benchmark dataset and evaluation methodology for video object segmentation. Computer Vision and Pattern Recognition (CVPR), 2016.

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

[13] G. Sundaramoorthi, A. Yezzi, and A. Mennucci. Sobolev active contours. International Journal of Computer Vision (IJCV), 73(3):345-366, 2007.

[14] D. Terzopoulos and R. Szeliski. Tracking with kalman snakes. Active Vision, Artificial Intelligence. The MIT Press, Cambridge, Massachusetts,1992.

[15] C. Xu and J. L. Prince. Snakes, shapes, and gradient vector flow. IEEE Transactions on Image Processing, 7(3):359-369, 1998.

[16] Z.Zhang, A.G.Schwing, S.Fidler, and R.Urtasun. Monocular object instance segmentation and depth ordering with cnns. IEEE International Conference on Computer Vision (ICCV), 2015.