SegDiff: Image Segmentation with Diffusion Probabilistic Models

Tomer Amit, Tal Shaharbany, Eliya Nachmani, and Lior Wolf
Tel-Aviv University
Facebook AI Research

Group 9
Chase Walker, Ilkin Sevqi Isler,
Dominic Simon, Shehreen Azad
Overview

- Related Work and Motivation
- Diffusion Model
- Architecture and Training
- Experiments
- Conclusion

SegDiff: Image Segmentation with Diffusion Probabilistic Models

Tomer Amit1, Tal Shaharbany1, Eliya Nachmanai1,2, and Lior Wolf4
1Tel-Aviv University
2Facebook AI Research
{tomeramit1,shaharbany,elihan,wolf}@mail.tau.ac.il

Abstract

Diffusion Probabilistic Methods are employed for state-of-the-art image generation. In this work, we present a method for extending such models for performing image segmentation. The method learns end-to-end, without relying on a pre-trained backbone. The information in the input image and in the current estimation of the segmentation map is merged by summing the output of two encoders. Additional encoding layers and a decoder are then used to iteratively refine the segmentation map, using a diffusion model. Since the diffusion model is probabilistic, it is applied multiple times, and the results are merged into a final segmentation map. The new method produces state-of-the-art results on the Cityscapes validation set, the Vaihingen building segmentation benchmark, and the MoNuSeg dataset.

I. Introduction

Diffusion methods, which iteratively improve a given image, obtain image quality that is on par with or better than other types of generative models, including other forms of log-likelihood models and adversarial models [16, 19]. Such methods have been shown to excel in many generation tasks, both conditional and unconditional.

The vast majority of diffusion models are applied in domains in which there is no absolute ground truth result and the output is evaluated either through a user study or using several quality and diversity scores. As far as we know, with the exception of super resolution [18, 27, 41], diffusion models have not been applied to problems in which the ground truth result is unique.

In this work, we tackle the problem of image segmentation. This problem is a cornerstone of both classical computer vision and the deep learning methods of the last decade. The leading methods in the field employ encoder-decoder networks of varied architectures [4, 31, 38, 50, 32, 33]. While adversarial methods have been attempted [12, 33, 49, 51], they do not constitute the current state of the art.

Therefore, it is uncertain whether diffusion models, which have been used primarily for GAN-like generation tasks, would be competitive in this domain. In this work, we propose applying a diffusion model to learn the image segmentation map. Unlike other recent improvements in the field of image segmentation [13, 32, 44], we train our method end-to-end, without relying on a pre-trained backbone network.

The diffusion model employs a denoising network conditioned on the input image only through a sum in which this information is aggregated with information arising from the current estimate 1. Specifically, the input image I and the current estimate 1 of the binary segmentation map are passed through two different encoders, and the sum of these multi-channel tensors is passed through a U-Net [38] to provide the next estimate 2.1.

Since the generation process is stochastic in its nature, one may obtain multiple solutions. As we show, merging these solutions, by simply averaging multiple runs, leads to an improvement in overall accuracy.

The novel method presented produces state-of-the-art results on multiple benchmarks: Cityscapes [1], building segmentation [39], and nuclei segmentation [25, 26].

Our main contributions are:

- We are the first to apply diffusion models to the image segmentation problem.
- We propose a new way to condition the model on the input image.
- We introduce the concept of multiple generations, in order to improve performance and calibration of the diffusion model.
**Image Segmentation**

- Image segmentation is a classical computer vision task
- Previous SOA models use encoder-decoder networks and a pre-trained backbone
  - FCNs, R-CNN, U-Net, etc.
- This is a **task with a unique ground truth** in its training data
- No prior attempt to use diffusion models
SegDiff: Segmentation with DPMs

- First application of DPMs to segmentation
- No pre-trained backbone required
  - End-to-end training
- A new conditioning method is proposed
- The concept of multiple generations creates sharper output
- New SOA performance is achieved
Diffusion Model Forward Process

- The forward process \( q \) is described by the formulation:

\[
q(x_1:T|x_0) = \prod_{t=1}^{T} q(x_t|x_{t-1})
\]

- At each iteration Gaussian noise is added:

\[
q(x_t|x_{t-1}) = N(x_t; \sqrt{1 - \beta_t} x_{t-1}, \beta_t I_{n \times n})
\]

\[
\beta_t = \frac{10^{-4}(T - t) + 2 \times 10^{-2}(t - 1)}{T - 1}
\]
Diffusion Model Forward Process

- The forward process supports sampling at time $t$, with the formula:

$$q(x_t | x_0) = N(x_t; \sqrt{\alpha_t}x_0, (1 - \bar{\alpha}_t)I_{n \times n})$$

$$\alpha_t = 1 - \beta_t, \bar{\alpha}_t = \prod_{s=0}^{t} \alpha_s$$

- Which can be parametrized to:

$$x_t = \sqrt{\alpha_t}x_0 + \sqrt{(1 - \bar{\alpha}_t)}\epsilon, \epsilon \sim N(0, I_{n \times n})$$
Diffusion Model Reverse Process

- The reverse process is parametrized by $\theta$ and is defined by:

$$p_\theta(x_{0:T-1} | x_T) = \prod_{t=1}^{T} p_\theta(x_{t-1} | x_t)$$

- $p_\theta(x_0)$ is predicted from $p_\theta(x_T) = N(x_T; 0, I_{n \times n})$ using:

$$p_\theta(x_{t-1} | x_t) = N(x_{t-1}; \mu_\theta(x_t, t), \Sigma_t(x_t, t))$$

\[
\mu_\theta(x_t, t) = \frac{1}{\sqrt{\alpha_t}}(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}} \epsilon_\theta(x_t, t))
\]

\[
\sigma_t^2 I_{n \times n}
\]

\[
\sigma_t^2 = \hat{\beta}_t = \frac{1 - \bar{\alpha}_{t-1}}{1 - \bar{\alpha}_t} \beta_t
\]
The denoising NN is trained to predict the noise \( \epsilon_\theta(x_t, t) \) using KL divergence:

\[
E_{x_0, \epsilon, t} \left[ \left\| \epsilon - \epsilon_\theta(\sqrt{\alpha_t}x_0 + \sqrt{1 - \alpha_t}\epsilon, t) \right\|^2 \right]
\]

\( \epsilon \sim N(0, I_{n \times n}) \)
Diffusion Model Inference

For inference, we sample using:

\[ x_{t-1} = \frac{1}{\sqrt{\alpha_t}} (x_t - \frac{1 - \alpha_t}{\sqrt{1 - \bar{\alpha}_t}} \epsilon_{\theta}(x_t, t)) + \sigma_t z \]
The DPM Application

- Segmentation is viewed as a conditional generation
  - The input image is the conditioner
- This **conditioning is usually applied by concatenation**
  - Convolved input is concatenated with the estimation before U-Net
  - This is how super resolution is done
- In this work **summation is used instead of concatenation**
  - The task becomes learning the DPM of a residual model
SegDiff Architecture

- 2D-convolutional layer
- RRDB with a residual connection
- 2D-convolutional layer
- leaky RELU activation
- 2D-convolutional layer

![Diagram of SegDiff Architecture](https://www.researchgate.net/figure/The-residual-in-residual-dense-block-RRDB_fig5_337997289)
SegDiff Architecture
Method

● The diffusion model is modified by conditioning the step estimation function $\epsilon_\theta$ on an input tensor.

$$\epsilon_\theta(x_t, I, t) = D(E(F(x_t) + G(I), t), t)$$
Training Algorithm

**Algorithm 2 Training Algorithm**

**Input** total diffusion steps $T$, images and segmentation masks dataset $D = \{(I_k, M_k)\}_{k=1}^K$

**repeat**

Sample $(I_i, M_i) \sim D$, $\epsilon \sim N(0, I_{n \times n})$

Sample $t \sim \text{Uniform}\{1, \ldots, T\}$

$\beta_t = \frac{10^{-4}(T-t)+2\times10^{-2}(t-1)}{T-1}$

$\alpha_t = 1 - \beta_t$

$\bar{\alpha}_t = \prod_{s=0}^{t} \alpha_s$

Take gradient step on $\nabla_\theta \|\epsilon - \epsilon_\theta(x_t, I_i, t)\|$, $x_t = \sqrt{\bar{\alpha}_t} M_i + \sqrt{1-\bar{\alpha}_t} \epsilon$

**until** convergence

- Sample $\mathbf{X}_t$ according to:
  
  $x_t = \sqrt{\alpha_t} x_0 + \sqrt{1-\alpha_t}\epsilon, \epsilon \sim N(0, I_{n \times n})$

- Compute $F(x_t) + G(I_i)$

- Apply networks $E$ and $D$ to obtain: $\epsilon_\theta(x_t, I_i, t)$

**Loss function:**

$E_{x_0, \epsilon, x_t, t}[\|\epsilon - \epsilon_\theta(\sqrt{\alpha_t} x_0 + \sqrt{1-\alpha_t}\epsilon, I_i, t)\|^2]$ is a modified version of:

$E_{x_0, \epsilon, t}[\|\epsilon - \epsilon_\theta(\sqrt{\alpha_t} x_0 + \sqrt{1-\alpha_t}\epsilon, t)\|^2]$
Inference Algorithm

**Algorithm 1** Inference Algorithm

**Input** total diffusion steps $T$, image $I$

$x_T \sim N(0, I_{n \times n})$

for $t = T, T - 1, ..., 1$

- $z \sim N(0, I_{n \times n})$
- $\beta_t = \frac{10^{-4}(T-t)+2*10^{-2}(t-1)}{T-1}$
- $\alpha_t = 1 - \beta_t$
- $\tilde{\alpha}_t = \prod_{s=0}^{t} \alpha_s$
- $\tilde{\beta}_t = \frac{1 - \tilde{\alpha}_{t-1}}{1 - \tilde{\alpha}_t} \beta_t$

$x_{t-1} = \alpha_t^{-\frac{1}{2}} \left( x_t - \frac{1 - \alpha_t}{\sqrt{1 - \tilde{\alpha}_t}} \epsilon_\theta(x_t, I, t) \right) + \mathbb{1}_{[t>1]} \beta_t^\frac{1}{2} z$

return $x_0$
Employing multiple generations

- Run the algorithm multiple times, then average the results.
Experimental Setup

Three main datasets:

- Cityscapes
- Vaihingen
- MoNuSeg

Mean Intersection Over Union (mIoU)

- True Positive (TP)
- False Positive (FP)
- False Negative (FN)

\[
\text{IoU} = \frac{\text{Area of Intersection}}{\text{Area of Union}}
\]

\[
m\text{IoU}(y_i, \hat{y}_i) = \frac{\sum_{i=1}^{N} \frac{TP(y_i, \hat{y}_i)}{TP(y_i, \hat{y}_i) + FN(y_i, \hat{y}_i) + FP(y_i, \hat{y}_i)}}
\]

- TP = Predict an object where there is an object mask
- FP = Predict an object where there is no object mask
- FN = Don’t predict an object where there is an object mask
## Results on Cityscapes

<table>
<thead>
<tr>
<th>Method</th>
<th>Bicycle</th>
<th>Bus</th>
<th>Person</th>
<th>Train</th>
<th>Truck</th>
<th>M.cycle</th>
<th>Car</th>
<th>Rider</th>
<th>Mean</th>
</tr>
</thead>
<tbody>
<tr>
<td>Polygon-RNN++ [1]</td>
<td>63.06</td>
<td>81.38</td>
<td>72.41</td>
<td>64.28</td>
<td>78.90</td>
<td>62.01</td>
<td>79.08</td>
<td>69.95</td>
<td>71.38</td>
</tr>
<tr>
<td>PSP-DeepLab [5]</td>
<td>67.18</td>
<td>83.81</td>
<td>72.62</td>
<td>68.76</td>
<td>80.48</td>
<td>65.94</td>
<td>80.45</td>
<td>70.00</td>
<td>73.66</td>
</tr>
<tr>
<td>Polygon-GCN [29]</td>
<td>66.55</td>
<td>85.01</td>
<td>72.94</td>
<td>60.99</td>
<td>79.78</td>
<td>63.87</td>
<td>81.09</td>
<td>71.00</td>
<td>72.66</td>
</tr>
<tr>
<td>Spline-GCN [29]</td>
<td>67.36</td>
<td>85.43</td>
<td>73.72</td>
<td>64.40</td>
<td>80.22</td>
<td>64.86</td>
<td>81.88</td>
<td>71.73</td>
<td>73.70</td>
</tr>
<tr>
<td>SegDiff (ours)</td>
<td><strong>69.80</strong></td>
<td><strong>85.97</strong></td>
<td><strong>76.09</strong></td>
<td><strong>75.95</strong></td>
<td><strong>80.68</strong></td>
<td><strong>67.06</strong></td>
<td><strong>83.40</strong></td>
<td><strong>72.57</strong></td>
<td><strong>76.44</strong></td>
</tr>
<tr>
<td>Deep contour [14]</td>
<td>68.08</td>
<td>83.02</td>
<td>75.04</td>
<td>74.53</td>
<td>79.55</td>
<td>66.53</td>
<td>81.92</td>
<td>72.03</td>
<td>75.09</td>
</tr>
<tr>
<td>Segformer-B5 [50]</td>
<td>68.02</td>
<td>78.78</td>
<td>73.53</td>
<td>68.46</td>
<td>74.54</td>
<td>64.06</td>
<td>83.20</td>
<td>69.12</td>
<td>72.46</td>
</tr>
<tr>
<td>StdC1 [11]</td>
<td>67.86</td>
<td>80.67</td>
<td>74.20</td>
<td>69.73</td>
<td>77.02</td>
<td>64.52</td>
<td>83.53</td>
<td>69.58</td>
<td>73.39</td>
</tr>
<tr>
<td>StdC2 [11]</td>
<td>68.67</td>
<td>81.29</td>
<td>74.41</td>
<td>71.36</td>
<td>75.71</td>
<td>63.69</td>
<td>83.51</td>
<td>69.90</td>
<td>73.57</td>
</tr>
<tr>
<td>SegDiff (ours)</td>
<td><strong>69.62</strong></td>
<td><strong>84.64</strong></td>
<td><strong>75.18</strong></td>
<td><strong>74.89</strong></td>
<td><strong>80.34</strong></td>
<td><strong>67.75</strong></td>
<td><strong>83.63</strong></td>
<td><strong>73.49</strong></td>
<td><strong>76.19</strong></td>
</tr>
</tbody>
</table>


Results on Vaihingen

Additional Metrics:

- F1-Score
- Weighted Coverage
- Boundary F1-Score

<table>
<thead>
<tr>
<th>Method</th>
<th>F1-Score</th>
<th>mIoU</th>
<th>WCov</th>
<th>FBound</th>
</tr>
</thead>
<tbody>
<tr>
<td>FCN-UNet [38]</td>
<td>87.40</td>
<td>78.60</td>
<td>81.80</td>
<td>40.20</td>
</tr>
<tr>
<td>FCN-ResNet34</td>
<td>91.76</td>
<td>87.20</td>
<td>88.55</td>
<td>75.12</td>
</tr>
<tr>
<td>FCN-HarDNet [4]</td>
<td>93.97</td>
<td>88.95</td>
<td>93.60</td>
<td>80.20</td>
</tr>
<tr>
<td>DSAC [34]</td>
<td>-</td>
<td>71.10</td>
<td>70.70</td>
<td>36.40</td>
</tr>
<tr>
<td>DarNet [7]</td>
<td>93.66</td>
<td>88.20</td>
<td>88.10</td>
<td>75.90</td>
</tr>
<tr>
<td>Deep contour [14]</td>
<td>94.80</td>
<td>90.33</td>
<td>93.72</td>
<td>78.72</td>
</tr>
<tr>
<td>TDAC [15]</td>
<td>94.26</td>
<td>89.16</td>
<td>90.54</td>
<td>78.12</td>
</tr>
<tr>
<td>Segformer-B5 [50]</td>
<td>93.94</td>
<td>88.57</td>
<td>91.91</td>
<td>77.95</td>
</tr>
<tr>
<td>Stdc1 [11]</td>
<td>94.04</td>
<td>88.75</td>
<td>92.78</td>
<td>78.86</td>
</tr>
<tr>
<td>SegDiff (ours)</td>
<td><strong>95.14</strong></td>
<td><strong>91.12</strong></td>
<td><strong>93.83</strong></td>
<td><strong>85.09</strong></td>
</tr>
</tbody>
</table>
Results on MoNuSeg

<table>
<thead>
<tr>
<th>Method</th>
<th>Dice</th>
<th>mIoU</th>
</tr>
</thead>
<tbody>
<tr>
<td>FCN [3]</td>
<td>28.84</td>
<td>28.71</td>
</tr>
<tr>
<td>U-Net [38]</td>
<td>79.43</td>
<td>65.99</td>
</tr>
<tr>
<td>U-Net++ [53]</td>
<td>79.49</td>
<td>66.04</td>
</tr>
<tr>
<td>Res-UNet [48]</td>
<td>79.49</td>
<td>66.07</td>
</tr>
<tr>
<td>A.A U-Net [46]</td>
<td>76.83</td>
<td>62.49</td>
</tr>
<tr>
<td>MedT [45]</td>
<td>79.55</td>
<td>66.17</td>
</tr>
<tr>
<td><strong>Ours</strong></td>
<td><strong>81.59</strong></td>
<td><strong>69.00</strong></td>
</tr>
</tbody>
</table>
Effects of Number of Instances Generated on mIOU

(a) Train, Bus, Truck, M.cycle, Rider, Bicycle

(b) MoNuSeg, Valhingen
Effects of Number of Diffusion Steps on mIOU
Effect of Number of Diffusion Steps on Inference Time

![Graph showing the effect of diffusion steps on inference time for 'Bus' and 'Vaihingen' datasets.](image)
Effect of Number of RRDBs on mIoU
6 Different Architecture Variants

- \([F(x_t), G(I)]\) instead of \(F(x_t) + G(I)\)
- FC-HarDNet-70 V2 instead of RRDBs
- Concatenation of \(I\) to \(x_t\) without using an encoder
- Propagation of \(F(x_t)\) through U-Net and add to \(G(I)\) after 1st, 3rd and 5th downsample blocks instead of \(F(x_t) + G(I)\)
Results of Different Variants on Vaihingen Dataset
Limitations

- SegDiff requires multiple runs to get accurate segmentation mask
- Trained on a specific type of dataset and thus lacks generalizability
  - Limited evaluation on complex scenes
- Code is unavailable
Conclusion

- SegDiff generates accurate segmentations in much lesser diffusion steps
  - Exploration of model generalizability should be done
- Summation of the current segmentation estimate with encoded image generates better results
- Performing summation in the initial layers of the network is more impactful
- Using RRDBs in the input image encoder G provides a performance boost
References

- SegDiff: Image Segmentation with Diffusion Probabilistic Models, Amit et al., 2022
- Image Super-Resolution via Iterative Refinement, Saharia et al., 2021
- Image Segmentation Using Deep Learning: A Survey, Minaee et al., 2020