Organ Segmentation: A Journey from Level Sets to Shape Denoising

Adrian Barbu
Professor, Statistics Department
Florida State University

Acknowledgements: Orhan Akal, Cheng Long
Outline

Introduction

The Chan-Vese Neural Network
   CVNN Overview
   Backpropagation
   Pre-Processing

Experiments & Results

From CVNN to Shape Denoising

The Shape Denoising Problem
   Introducing Noise in Shapes

Experiments

Summary and Discussion
Introduction

Chan-Vese, CV:

- A level set method that minimizes a Mumford-Shah integral
- Simultaneously evolves a level set surface and fits locally constant intensity models for the interior and exterior regions
- Its length-based contour regularization is quite simple and too weak for many applications
- Performs poorly compared with the state of the art object segmentation methods

\[ C = \{(x, y) | \varphi(x, y) = 0\} \]
Chan-Vese Overview

The Chan-Vese Active contour [Chan and Vese, 2001] is aimed at minimizing the following Mumford-Shah energy

$$E(C) = \int_{C_i} (I(u) - \mu_i)^2 du + \int_{C_o} (I(u) - \mu_o)^2 du + \nu |C|$$

(1)

where

▶ $I$ denotes the image intensity,
▶ $C$ is the curve to be fitted,
▶ $C_i, C_o$ are the regions inside and respectively outside the curve $C$,
▶ $\mu_i$ and $\mu_o$ are the intensity averages of image $I$ inside and respectively outside the curve $C$. 


A direct approach to minimize the energy would be to derive the Euler-Lagrange equation and obtain a curve evolution equation:

$$\frac{\partial C}{\partial t} = f(\kappa)\vec{N}$$

(2)

where

- $\kappa$ is the curvature of $C$,
- $f$ is some function of the curvature,
- $\vec{N}$ is the normal vector to the curve.

Such a direct approach is difficult to implement because of topological changes:

- self intersection of the curve
- splitting into sub-objects, developing holes, or multiple objects merging into one

These make managing the curve representation quite challenging.
Chan-Vese

- The curve $C$, is represented as the 0 level set of a surface $\varphi$,
  \[ C = \{(x, y) | \varphi(x, y) = 0\} \].
- Usually $\varphi(x, y)$ is initialized as the signed Euclidean distance transform of $C$.
- The energy (1) is extended to an energy of the level set function $\varphi$:

  \[
  E(\varphi, I) = \int (I(u) - \mu_o(\varphi(u)))^2 (1 - H_\epsilon(\varphi(u))) \, du + \\
  \int (I(u) - \mu_i(\varphi(u)))^2 H_\epsilon(\varphi(u)) \, du + \nu \int \delta_\epsilon(\varphi(u)) |\nabla \varphi(u)| \, du \tag{3}
  \]

  where
  - $H_\epsilon$ is a smoothed Heaviside function and $\delta_\epsilon$ is its derivative.
  - The parameter $\nu$ controls the curve length regularization $\int |\nabla \varphi|$.
The following smoothed Heaviside function could be used

$$H_\epsilon(z) = \begin{cases} 
0 & \text{if } z < -\epsilon \\
1 & \text{if } z > \epsilon \\
\frac{1}{2} \left[ 1 + \frac{z}{\epsilon} + \frac{1}{\pi} \sin\left( \frac{\pi z}{\epsilon} \right) \right] & \text{if } |z| < \epsilon
\end{cases}$$  \hspace{1cm} (4)

Figure: Heaviside $H_\epsilon(z)$ and its derivative dirac delta function $\delta_\epsilon(z)$ for $\epsilon = 0.5$ and $\epsilon = 0.5$.5
The energy is minimized alternatively by updating $\mu_i, \mu_o$

$$
\mu_i^t = \frac{\int I(u) H_\epsilon(\varphi^t(u)) du}{\int H_\epsilon(\varphi^t(u)) du},
$$

$$
\mu_o^t = \frac{\int I(u)[1 - H_\epsilon(\varphi^t(u))] du}{\int [1 - H_\epsilon(\varphi^t(u))] du},
$$

(5)

then assuming $\mu_i^t, \mu_o^t$ fixed the solution $\varphi$ needs to satisfy the Euler-Lagrange equation:

$$
\delta_\epsilon(\varphi)[\nu(\frac{\nabla \varphi}{|\nabla \varphi|}) + (I - \mu_o^t)^2 - (I - \mu_i^t)^2] = 0
$$

(6)

which can be done iteratively:

$$
\varphi^{t+1} = \varphi^t + \eta[\kappa(\varphi^t) + (I - \mu_o^t)^2 - (I - \mu_i^t)^2]
$$

(7)
CVNN: Chan-Vese CNN

We generalize the Chan-Vese by replacing the curvature term $\kappa(\varphi) = \frac{\nabla \varphi}{|\nabla \varphi|}$ in the Euler-Lagrange equation (7) with a generic shape function $g(\varphi, \beta)$ with parameters $\beta$. Obtain the iterative algorithm

$$\varphi^{t+1} = \varphi^t + \eta \delta \epsilon(\varphi^t)(g(\varphi^t, \beta) + (I - \mu_0)^2 - (I - \mu_i)^2)$$  (8)

Figure: Our RNN model merging CNN with Chan-Vese
Proposed method

The problem that we are trying to address is
1. To impose better shape priors in the Chan-Vese formulation
2. To learn these shape priors using training examples instead of setting them by hand to a predefined form.

Our formulation uses
1. A Convolutional Neural Network (CNN) to encode the shape prior
2. The whole Chan-Vese evolution becomes a Recurrent Neural Network (RNN)
3. Training the shape model is done in an end-to-end fashion by backpropagation.
We are going to use the Combo loss [Taghanaki et al., 2019]

\[
L(\beta) = \alpha_1 \left( -\frac{1}{N} \sum_{i=1}^{N} \alpha_2 Y_i \ln \hat{R}_i - (1 - \alpha_2)(1 - Y_i) \ln(1 - \hat{R}_i) \right) \\
- (1 - \alpha_1) \frac{\sum_{i=1}^{N} Y_i \hat{R}_i + s}{\sum_{i=1}^{N} Y_i + \sum_{i=1}^{N} \hat{R}_i + s}
\]

where \( s \) is a small positive smoothing factor, \( \beta \) are the U-Net weights, \( \hat{R}_i \in [0, 1] \) is the prediction for voxel \( i \) after sigmoid normalization, and \( \alpha_1, \alpha_2 \in [0, 1] \) are tuning parameters, fixed as \( \alpha_1 = 0.7 \) and \( \alpha_2 = 0.5 \).
Backpropagation

Following in the footsteps of [Sun and Tappen, 2011] to obtain the gradient of the loss function $L$

$$\frac{\partial L}{\partial \beta} = \sum_{k=1}^{T} \frac{\partial L}{\partial \phi^k} \cdot \frac{\partial \phi^k}{\partial \beta} = \frac{\partial L}{\partial \phi^T} \cdot \sum_{k=1}^{T} \left\{ \frac{\partial \phi^k}{\partial \beta} \cdot \prod_{t=k}^{T-1} \frac{\partial \phi^{t+1}}{\partial \phi^t} \right\}. \quad (10)$$

**Figure:** For backpropagation, we unwind the model $T$ times and compute the gradient using Eq. (15).
Backpropagation

Using the the update (8), we get

$$\frac{\partial \varphi^{t+1}}{\partial \varphi^t} = 1 + \eta \left( \frac{\partial g(\varphi^t, \beta)}{\partial \varphi^t} - 2(1 - \mu_o) \cdot \frac{\partial \mu_o(\varphi^t)}{\partial \varphi^t} 
+ 2(1 - \mu_i) \cdot \frac{\partial \mu_i(\varphi^t)}{\partial \varphi^t} \right),$$

(11)

where

$$\frac{\partial \mu_i(\varphi^t)}{\partial \varphi^t} = \frac{\delta_\epsilon(\varphi^t) \cdot (1 - \mu_i)}{\int H_\epsilon(\varphi^t(x)) dx},$$

(12)

$$\frac{\partial \mu_o(\varphi^t)}{\partial \varphi^t} = \frac{\delta_\epsilon(\varphi^t) \cdot (\mu_o - 1)}{\int (1 - H_\epsilon(\varphi^t(x))) dx}$$

(13)

and $H_\epsilon$ has been defined in Eq. (4) and $\delta_\epsilon$ is its derivative.
Backpropagation

We also get

\[
\frac{\partial \varphi^{t+1}}{\partial \beta} = \frac{\partial \varphi^{t+1}}{\partial g} \cdot \frac{\partial g}{\partial \beta} = \eta \cdot \frac{\partial g}{\partial \beta}
\] (14)

Consequently

\[
\frac{\partial L}{\partial \beta} = \frac{\partial L}{\partial \varphi^T} \cdot \eta \cdot \sum_{k=1}^{T} \left\{ \frac{\partial g(\varphi^{k-1}, \beta)}{\partial \beta} \cdot \prod_{t=k}^{T-1} \frac{\partial \varphi^{t+1}}{\partial \varphi^t} \right\}
\] (15)
Use a trained classifier to find organ pixels.

Yields a very coarse initial segmentation i.e. detection map.

Total number of parameters is 204033.
Pre-Processing: Probability map

Algorithm 2 Probability Map Computation

Input: CT scan $C$, initial binary mask $D$, number of bins $N^{bins}$.
Output: Probability map $I$.

1: Extract pixels inside the mask $u = C(D > 0)$
2: Construct $N^{bins}$ equally spaced bins in the range $[\min(u), \max(u)]$
3: Compute counts $n = \text{histogram}(u, \text{bins})$
4: Obtain $I(x) = n(C(x)) / \max(n)$

detection map  probability map  segmentation and ground truth
CNN architecture

For CNN we used a network with

- 3 convolutional layers with 3 filters of size $3 \times 3$ with padding,
- A convolutional layer with 3 filters of size $1 \times 1$,
- Exponential linear unit (ELU), instead of ReLU activation,
- A convolutional layer with 1 filter of size $1 \times 1$.

The filters of size $3 \times 3$ used padding such that the size of the output was kept the same as the size of the input.
Multiple Initializations $\varphi^0$ for Training

To avoid overfitting, we need to train the CNN to recover from different initializations. We initialize with:

- Detection map
- Thresholded probability map with different thresholds and some connected components removed
- Distorted ground truth $Y$ as shown below

Figure: Left: ground truth. Middle: distorted by added or punched semicircles with random radius at random border locations. Right: The middle image is corrupted by adding Gaussian noise and used as initialization for training.
φ’s after 3 iterations

Figure: From left to right: CT Image (I), probability map, φ₀(initialization), φ³ (segmentation), ground truth mask.

Figure: Level surface evolution through 3 iterations of CVNN. Left to right: φ⁰, φ¹, φ², φ³
Figure: 2D segmentation results. Top: Chan-Vese with 100 iteration. Bottom: CVNN with 3 iterations.
Dice Coefficient

We will use the Dice coefficient in our experiments

\[
Dice(A, B) = \frac{2|A \cap B|}{|A| + |B|}.
\]

**Figure:** Computation of $Dice(A, B) = \frac{2|A \cap B|}{|A| + |B|}$. 
## Experiments & Results

<table>
<thead>
<tr>
<th></th>
<th>CV-1\textsuperscript{it}</th>
<th>CV-10\textsuperscript{it}</th>
<th>CV-100\textsuperscript{it}</th>
<th>CVNN-1\textsuperscript{it}</th>
<th>CVNN-2\textsuperscript{it}</th>
<th>CVNN-3\textsuperscript{it}</th>
</tr>
</thead>
<tbody>
<tr>
<td>Liver</td>
<td>90.85</td>
<td>91.72</td>
<td>92.69</td>
<td>92.48</td>
<td>93.56</td>
<td>94.04</td>
</tr>
<tr>
<td>Horse</td>
<td>50.61</td>
<td>53.83</td>
<td>67.66</td>
<td>68.72</td>
<td>68.36</td>
<td>68.10</td>
</tr>
</tbody>
</table>

**Table:** Dice coefficients on the test set obtained with 4-fold cross validation. 'CV - \textsuperscript{n}it' stands for standard Chan-Vese with \textsuperscript{n} iterations, and 'CVNN - \textsuperscript{n}it' stands for our proposed method with \textsuperscript{n} iterations. Note Dice of the initialization fed into CV and CVNN for liver is 90.43.

Liver results are based on 4-fold cross validation. For Weizmann horse dataset [Borenstein and Ullman, 2002] 200 images reserved for testing and 128 of images which had aspect ratio close to 3x4 resized to 180x240 and used for training, testing images are not resized.
CVNN-UNet

Replacing the CNN with a U-Net
3D CVNN-UNet

Figure: The 3D UNet architecture with 2 convolution blocks, each convolution block has 2 convolution layers.

▶ Each convolution layer is followed by an exponential linear unit, ELU, activation.
▶ Has only $\sim 353,000$ parameters to be optimized.
▶ Needs low-resolution inputs due memory issues
▶ The Chan-Vese update takes place in 3D, not in 2D.
The CVNN iterations

$$\varphi^{t+1} = \varphi^t + \eta \delta_{\epsilon}(\varphi^t)(g(\varphi^t, \beta) + (l - \mu_o)^2 - (l - \mu_i)^2)$$

(16)
3D Applications: Data

Same as [Gibson et al., 2018]:

- A multi-organ segmentation dataset.
- 90 CT scans: 43 from the TCIA Pancreas-CT dataset ([Clark et al., 2013]) and 47 from the BTCV dataset ([Landman et al., 2015])
- Provided reference segmentations for all 90 CT scans and up to 14 abdominal organs
- Some of the BTCV patients had metastatic liver cancer or other forms of abdominal pathologies.
3D Data Preprocessing

We have

- resized the CT scans so that each CT scan in itself is isotropic (has the same resolution in all three directions)
- the axial dimensions are $512 \times 512$.

For the isotropic input size is $512 \times 512 \times 4k$

- resized to $256 \times 256 \times 2k$ (medium-resolution)
- resized to $128 \times 128 \times k$ (low-resolution)

Different resolution levels of the same CT scan gives us the flexibility to experiment with both computationally efficient and expensive 3D CVNN methods.
3D CVNN-UNet Results

Figure: Segmentation example of the 3D CVNN-UNet on a CT scan from the BTCV dataset.
3D CVNN with medium-resolution input

We used the same CVNN architecture that we used for 2D except each convolution kernel is $3 \times 3 \times 3$, with a total number of 592 parameters.

- Took the segmentation maps obtained from 4 iterations of 3D CVNN-UNet, and upsampled them 2x
- Used these upsampled segmentations as new detection maps and populated probability maps given newly derived detection maps using the probability map algorithm mentioned earlier.
There are three main purposes for this step;

▶ To further improve the accuracy given the new detection and probability maps.
▶ The low-resolution segmentation would look coarse when upsampled. We aim to obtain finer segmentations for the upsampled input.
▶ To show that when we combine 3D CVNN-UNet and 3D CVNN, we can achieve high accuracy for medium resolution input while reducing computation complexity.
Figure: Diagram of the Deep Chan-Vese 3D approach.
## Our Approach vs. State of the Art

<table>
<thead>
<tr>
<th>Architecture</th>
<th>x-val vols</th>
<th>Dice</th>
<th>95% Hauss/Segm.</th>
<th>Test Dice</th>
<th>Dist (mm)</th>
<th>Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEEDS + JLF (Wang et al ‘12)</td>
<td>144</td>
<td>9</td>
<td>90</td>
<td>94</td>
<td>2.1</td>
<td>6.2</td>
</tr>
<tr>
<td>VoxResNet (Chen et al. ‘16)</td>
<td>144</td>
<td>9</td>
<td>90</td>
<td>95</td>
<td>2.0</td>
<td>5.2</td>
</tr>
<tr>
<td>VNet (Milletari et al ‘16)</td>
<td>144</td>
<td>9</td>
<td>90</td>
<td>94</td>
<td>2.2</td>
<td>6.4</td>
</tr>
<tr>
<td>DenseVNet (Gibson et al. ‘18)</td>
<td>144</td>
<td>9</td>
<td>90</td>
<td>96</td>
<td>1.6</td>
<td>4.9</td>
</tr>
<tr>
<td>ObeliskNet (Heinrich et al ‘19)</td>
<td>144</td>
<td>4</td>
<td>43</td>
<td>95.4[^1]</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>SETR (Zheng et al ‘21)</td>
<td>96</td>
<td>5</td>
<td>30</td>
<td>95.4</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>CoTr (Xie et al ‘21)</td>
<td>96</td>
<td>5</td>
<td>30</td>
<td>96.3</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>UNETR (Hatamizadeh et al ‘22)</td>
<td>96</td>
<td>5</td>
<td>30</td>
<td>97.1[^2]</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>nnU-Net (Isensee et. al ‘20)</td>
<td>128</td>
<td>1</td>
<td>13</td>
<td>96.4[^3]</td>
<td>1.7</td>
<td>-</td>
</tr>
<tr>
<td>DISSM (Raju et al ‘22)</td>
<td>-</td>
<td>1</td>
<td>13</td>
<td>96.5[^4]</td>
<td>1.1</td>
<td>-</td>
</tr>
<tr>
<td>3D CVNN-UNet (ours)</td>
<td>128</td>
<td>4</td>
<td>90</td>
<td>95.6</td>
<td>1.67</td>
<td>4.42</td>
</tr>
<tr>
<td>Deep Chan-Vese 3D (ours)</td>
<td>256</td>
<td>4</td>
<td>90</td>
<td>95.2</td>
<td>1.49</td>
<td>4.40</td>
</tr>
</tbody>
</table>

Table: Comparison with the state of the art for liver segmentation. The 9-fold cross-validation results are from [Gibson et al., 2018], the 5-fold results from Hatamizadeh et. al. 2022, and the 1-fold from Raju et. al 2022.
Ablation Study

Questions?
- What is the contribution of U-Net vs CNN?
- What is the contribution of 3D vs 2D?

<table>
<thead>
<tr>
<th></th>
<th>3D U-Net</th>
<th>$\varphi^0$</th>
<th>1-it</th>
<th>2-it</th>
<th>3-it</th>
<th>4-it</th>
</tr>
</thead>
<tbody>
<tr>
<td>2D CVNN</td>
<td>-</td>
<td>-</td>
<td>87.58</td>
<td>92.75</td>
<td>93.66</td>
<td>93.63</td>
</tr>
<tr>
<td>2D CVNN-UNet</td>
<td>-</td>
<td>+</td>
<td>87.58</td>
<td>92.61</td>
<td>93.72</td>
<td>93.63</td>
</tr>
<tr>
<td>3D CVNN</td>
<td>+</td>
<td>-</td>
<td>87.58</td>
<td>88.29</td>
<td>90.23</td>
<td>91.43</td>
</tr>
<tr>
<td>3D CVNN-UNet</td>
<td>+</td>
<td>+</td>
<td>87.58</td>
<td>92.83</td>
<td>94.41</td>
<td>95.09</td>
</tr>
</tbody>
</table>

Table: Ablation results comparing 2D vs. 3D approaches and CNN vs UNet. The results are shown as average Dice scores obtained with 4-fold cross-validation.
Discussion

The Chan-Vese NN

- Reduced cost of computation.
- Outperforms Chan-Vese by CVNN both in terms of speed and accuracy.
- Competitive with the state of the art models, using a small number of parameters.
Looking Deeper at the CVNN

The CVNN Model:

\[ \varphi^{t+1} = \varphi^t + \eta \delta_\epsilon(\varphi^t)(g(\varphi^t, \beta) + (I - \mu_o)^2 - (I - \mu_i)^2) \]

Motivation

▶ The CNN or U-Net \( g(\varphi^t, \beta) \) helps update a level set function \( \varphi \).
▶ What matters is only the zero level set \( \varphi^{-1}(0) = \) a shape.
▶ The input \( \varphi^0 \) is a corrupted (noisy) shape.
▶ The CNN is trained to recover from many noisy shapes.
▶ How well can it work in one step, without the data term?
▶ What kind of possible corruptions=noises are there?
▶ We call this task Shape Denoising.
The shapes are represented as binary images of a certain size, e.g. $128 \times 128$.

The focus of our study is to study shape denoising - the task of recovering a shape corrupted by noise.

Similar to Post-DAE [Larrazabal et al., 2020] but different focus.

We will introduce different types of shape noise.

We will evaluate and compare the performance of seven shape modeling methods for shape denoising.
The Shape Denoising Problem

Shape denoising is the process of removing the noise from a shape, with the goal of obtaining a shape as close to the original shape as possible.

(a) Original shape  (b) Noisy shape  (c) Denoised shape

Figure: Shape denoising example. The noisy shape (b) has been obtained from the original shape (a) by a noise inducing process. A shape denoising method is used to obtain the denoised shape (c).
Shape Alignment

- We used binary images of size $128 \times 128$.
- All binary images were aligned to have the objects centered and of approximately the same size.

Figure: Two alignment examples from the Weizmann Horse Dataset.
Six types of noise

original, aligned, salt and pepper, circle

real image, occlusion, detection image, thr. probability
Salt and Pepper Noise

The salt and pepper noise is obtained by flipping each pixel to its opposite value with a probability $p$.

Figure: Salt and pepper noise with different levels $p$. 

(a) $p = 0.01$  (b) $p = 0.05$  (c) $p = 0.10$  (d) $p = 0.15$
Circle Noise

The circle noise is obtained by adding semicircles or punching holes at random locations on the boundary between the foreground and the background.

(a) $r = 1$  (b) $r = 3$  (c) $r = 6$  (d) $r = 10$

**Figure:** Circle noise with different levels.
Real Image Noise

- Noisy backgrounds are obtained by thresholding real images using various thresholds
- The shape background pixels are replaced with the noisy background

Figure: Examples of real image noise.
Occlusion Noise

The shape foreground pixels are occluded using thresholded real images.

Figure: Examples of occlusion noise.
Detection Image Noise

Detection image noise is obtained as the segmentation of an object from a color or grayscale image using a trained CNN (convolutional neural network).

(a) object shape  (b) color image  (c) detection image noise

**Figure:** Example of detection image noise in the Weizmann Horse Dataset.
Thresholded Probability Noise

Uses a color image $I$ and a binary image $M$ representing the object in the image.

- Denote $C_1 = \{(x, y), M(x, y) = 1\}$ as the foreground region and $C_0 = \{(x, y), M(x, y) = 0\}$ as the background region.

- $k$-means clustering is used to partition the image into $k$ clusters, obtaining cluster indices for all pixels $L \in \{1, 2, ..., k\}^N$.

- For cluster $i \in \{1, 2, ..., k\}$, obtain the number of pixels that belong to foreground or background:

  \begin{align}
  N_{i1} &= |\{(x, y)|L(x, y) = i \land (x, y) \in C_1\}|, \\
  N_{i0} &= |\{(x, y)|L(x, y) = i \land (x, y) \in C_0\}|. 
  \end{align}

- Then the probability map of color image $I$ can be computed as follows:

  \[ P(x, y) = \frac{N_{j1}}{N_{j1} + N_{j0}}, \text{ where } j = L(x, y). \]
Thresholded Probability Noise

Binary noisy shapes are obtained by thresholding the probability map $P$. 

![Object shape, color image, prob map with thresholds]

- Object shape
- Color image
- Prob map

- Threshold 0.04
- Threshold 0.5
- Threshold 0.98
Clean Image Datasets

The **Weizmann Horse dataset** [Borenstein et al., 2004]:
- Contains 327 horse images and their corresponding mask images.
- 159 images were randomly selected as the training set $S_{\text{train}}^{\text{clean}}$ and the other 168 images as the test set $S_{\text{test}}^{\text{clean}}$.

The **Caltech-UCSD Birds 200 dataset** [Welinder et al., 2010] contains photos of 200 bird species.
- We use 417 images of seven Flycatcher species in our experiments.
- 207 images were randomly selected as the training set $S_{\text{train}}^{\text{clean}}$ and the other 210 images as the test set $S_{\text{test}}^{\text{clean}}$. 
In our experiments, the criterion we use to estimate the performance of modeling against noises is Intersection over Union (IoU), also known as the Jaccard Index.

\[ \text{IOU}(A, B) = \frac{|A \cap B|}{|A \cup B|}. \]

**Figure:** Computation of \( \text{IOU}(A, B) = \frac{|A \cap B|}{|A \cup B|} \).
Noisy Image Datasets

We denote the set \( \{ S_{\text{salt test}}^{\text{test}}, S_{\text{circle test}}^{\text{test}}, S_{\text{real test}}^{\text{test}}, S_{\text{occlusion test}}^{\text{test}}, S_{\text{detection test}}^{\text{test}}, S_{\text{probability test}}^{\text{test}} \} \) as \( S_{\text{all test}} \).

<table>
<thead>
<tr>
<th>Dataset</th>
<th>IoU (begin)</th>
<th>0.5-0.6</th>
<th>0.6-0.7</th>
<th>0.7-0.8</th>
<th>0.8-0.9</th>
<th>0.9-1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Horse</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>( S_{\text{salt test}} )</td>
<td>1000</td>
<td>1000</td>
<td>1000</td>
<td>1000</td>
<td>1000</td>
</tr>
<tr>
<td></td>
<td>( S_{\text{circle test}} )</td>
<td>77</td>
<td>1000</td>
<td>1000</td>
<td>1000</td>
<td>1000</td>
</tr>
<tr>
<td></td>
<td>( S_{\text{real test}} )</td>
<td>1000</td>
<td>1000</td>
<td>1000</td>
<td>1000</td>
<td>1000</td>
</tr>
<tr>
<td></td>
<td>( S_{\text{occlusion test}} )</td>
<td>479</td>
<td>622</td>
<td>690</td>
<td>938</td>
<td>1779</td>
</tr>
<tr>
<td></td>
<td>( S_{\text{detection test}} )</td>
<td>3</td>
<td>7</td>
<td>14</td>
<td>77</td>
<td>60</td>
</tr>
<tr>
<td></td>
<td>( S_{\text{probability test}} )</td>
<td>298</td>
<td>673</td>
<td>1095</td>
<td>1345</td>
<td>525</td>
</tr>
<tr>
<td>Bird</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>( S_{\text{salt test}} )</td>
<td>1000</td>
<td>1000</td>
<td>1000</td>
<td>1000</td>
<td>1000</td>
</tr>
<tr>
<td></td>
<td>( S_{\text{circle test}} )</td>
<td>416</td>
<td>1000</td>
<td>1000</td>
<td>1000</td>
<td>1000</td>
</tr>
<tr>
<td></td>
<td>( S_{\text{real test}} )</td>
<td>1000</td>
<td>1000</td>
<td>1000</td>
<td>1000</td>
<td>1000</td>
</tr>
<tr>
<td></td>
<td>( S_{\text{occlusion test}} )</td>
<td>435</td>
<td>556</td>
<td>809</td>
<td>972</td>
<td>1000</td>
</tr>
<tr>
<td></td>
<td>( S_{\text{detection test}} )</td>
<td>22</td>
<td>30</td>
<td>44</td>
<td>56</td>
<td>15</td>
</tr>
<tr>
<td></td>
<td>( S_{\text{probability test}} )</td>
<td>843</td>
<td>962</td>
<td>934</td>
<td>519</td>
<td>70</td>
</tr>
</tbody>
</table>

Table: Number of noisy test images in each noise category.
Shape Denoising Methods Evaluated

The following methods have been evaluated for shape modeling and denoising:

- Active Shape Model (ASM) [Cootes et al., 1995]
- Deep Boltzmann Machine (DBM) [Salakhutdinov and Hinton, 2009]
- Centered Convolutional DBM (CDBM) [Yang et al., 2021]
- Energy Based Model (EBM) [Pang et al., 2020]
- U-Net [Ronneberger et al., 2015]
- DeepLab V3+ [Chen et al., 2018]
- Masked Autoencoder (MAE) [He et al., 2021]
Masked Autoencoder (MAE)

The masked autoencoder (MAE) [He et al., 2021]
- Scalable self-supervised learner
- Trained to fill-in missing parts of an image

Figure: MAE architecture [He et al., 2021]
Salt and Pepper Noise Example

Figure: Example of results of different methods on a shape perturbed by salt and pepper noise.
Salt and Pepper Noise Results

- Performance comparison of all methods against salt and pepper noise

Figure: Performance on salt and pepper noise data $S_{test}^{salt}$. 
Circle Noise Example

Figure: Example of results of different methods on a shape perturbed by circle noise.
Circle Noise Results

- Performance comparison of all methods against circle noise

**Figure:** Performance on circle noise data $S_{test}^{circle}$. 
Real Image Noise Example

Figure: Example of results of different methods on a shape perturbed by real image noise.
Real Image Noise Results

- Performance comparison of all methods against real image noise

**Figure:** Performance on real image noise data $S_{real}^{test}$. 
Occlusion Noise Example

Figure: Example of results of different methods on a shape perturbed by occlusion noise.
Occlusion Noise Results

- Performance comparison of all methods against occlusion noise

Figure: Performance on real image noise data $S_{occlusion test}$. 
Detection Image Noise Example

Figure: Example of results of different methods on a shape perturbed by detection image noise.
Detection Image Noise Results

- Performance comparison of all methods against detection image noise

![Line charts showing performance comparison for different methods on real image noise data for Horses and Birds.](image)

**Figure:** Performance on real image noise data $S_{test}^{detection}$. 
Thresholded Probability Noise Example

Figure: Example of results of different methods on a shape perturbed by thresholded probability noise.
Thresholded Probability Noise Results

- Performance comparison of each methods against thresholded probability noise

**Figure:** Performance on thresholded probability noise data $S_{test}^{probability}$. 

Horses

Birds
Summary and Discussion

Comparing the methods:

▶ Experiments reveal that MAE and U-Net are the best shape denoising methods we evaluated for all six types of noise.
▶ DeepLabv3+ is the third best shape denoising method for the six noise types in most situations.
▶ EBM outperforms CDBM on all six noise types, especially when dealing with real image noise.

Comparing the noise types:

▶ The salt and pepper noise is the easiest to deal with, followed by real image noise.
▶ Circle noise and occlusion noise are more challenging than the above two, especially when the noise level is high.
▶ The most challenging noises among these six are the thresholded probability noise and detection image noise.
Conclusions

CVNN - a method for object segmentation

- Generalizes the Chan-Vese level set method, replacing the length-based regularization with a trained CNN shape model
- The CNN is trained end-to-end by backpropagation
- Fast and competitive with state of the art 3D liver segmentation methods
- Multiple types of initializations are used to avoid overfitting
- The initializations lead us to the shape denoising problem

A study of methods for shape denoising

- Shapes are represented as binary images
- Shapes are aligned by translation and scaling
- We introduced six types of shape noise
- We evaluated seven shape modeling/denoising methods on these types of noise
Future Work

Future work:
▶ Study shape denoising in the wild, where the shapes are not aligned
▶ Study methods trained on aligned shapes vs. methods trained on unaligned shapes
▶ Come back to CVNN, apply what we learned to object segmentation
References I


References IV


In Artificial intelligence and statistics, pages 448–455. PMLR.

