Star-Shapes and Convex Shape representation by DNNs.

Xue-Cheng Tai

Department of Mathematics, Hong Kong Baptist University

In collaboration with: Jun Liu (Beijing Normal U.), Shousheng Luo (Henan U.), Xiangyue Wang, (Beijing Normal U.),

July 2, 2020

Classic architectures for Semantic Image Segmentation

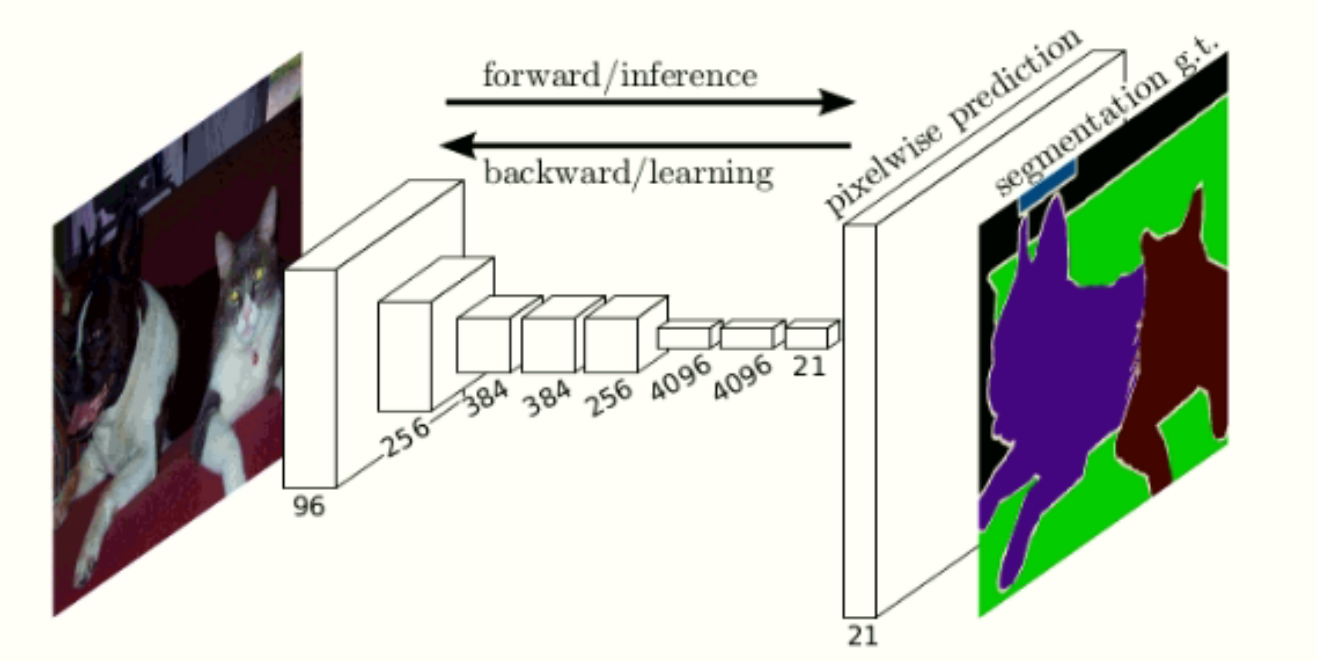


Figure 1: Architecture of FCN [Long J, Shelhamer E, Darrell T.]. Fully convolutional networks can efficiently learn to make dense predictions for per-pixel tasks like semantic segmentation.

Examples with and without convex shape priori

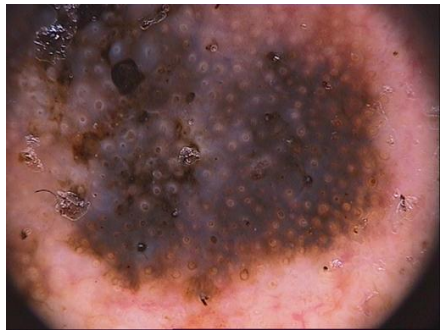
(a) Without convexity

Examples with and without convex shape priori

(c) Without convexity

(d) With convexity

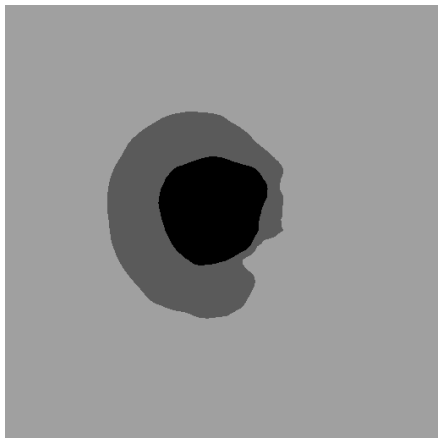
Lack of geometry shape prior:



(p) Input

(q) DeepLabV3+

(r) Input



(s) DeepLabV3+

General Neural Network for Semantic Image Segmentation

- Let $v \in \mathcal{R}^{N_1 N_2}$ be a column vector by stacking the columns of an image with size $N_1 \times N_2$. Taking v as an input of a pixel-wise segmentation neural network. Mathematically, this network can be written as a parameterized nonlinear operator $\mathcal{N}_{\Theta}$ defined by

$$v^K = \mathcal{N}_{\Theta}(v).$$

The output v^K of the network is given by some recursive connections

$$\begin{cases} v^0 = & v, \\ v^k = & \mathcal{A}^k \circ \mathcal{T}_{\Theta^{k-1}}(v^{k-1}), k = 1, \cdots, K. \end{cases} \tag{5}$$

Here $\mathcal{A}^k$ is an activation function of the k-th layer, $\mathcal{T}_{\Theta^{k-1}}$ is convolution operator defined as $\mathcal{T}_{\Theta^{k-1}}(v) = \mathcal{W}^{k-1}v + b^{k-1}$. The parameter set $\Theta = \{\Theta^k = (\mathcal{W}^k, b^k) | k = 0, \ldots, K-1\}$.

General Neural Network for Semantic Image Segmentation

- The training process is to learn the parameter set Θ by giving some images $\mathcal{V} = (v_1, v_2, \cdots, v_N) \in \mathbb{R}^{N_1 N_2 \times N}$ and their C classes ground truth segmentation $\mathcal{U} = stack(\mathcal{U}_1, \mathcal{U}_2, \cdots, \mathcal{U}_N) \in \{0,1\}^{N_1 N_2 \times C \times N}$ with $\mathcal{U}_n \in \{0,1\}^{N_1 N_2 \times C}$ to minimize a loss functional $\mathcal{L}(\mathcal{N}_\Theta(\mathcal{V}), \mathcal{U})$, namely

$$\Theta^* = \arg\min_{\Theta} \ \mathcal{L}(\mathcal{N}_\Theta(\mathcal{V}), \mathcal{U}).$$

In many references, the loss function are set as the cross entropy which is given by

$$\mathcal{L}(\mathcal{N}_\Theta(\mathcal{V}), \mathcal{U}) = -\sum_{n=1}^{N} < \mathcal{U}_n, \log \mathcal{N}_\Theta(v_n) > .$$

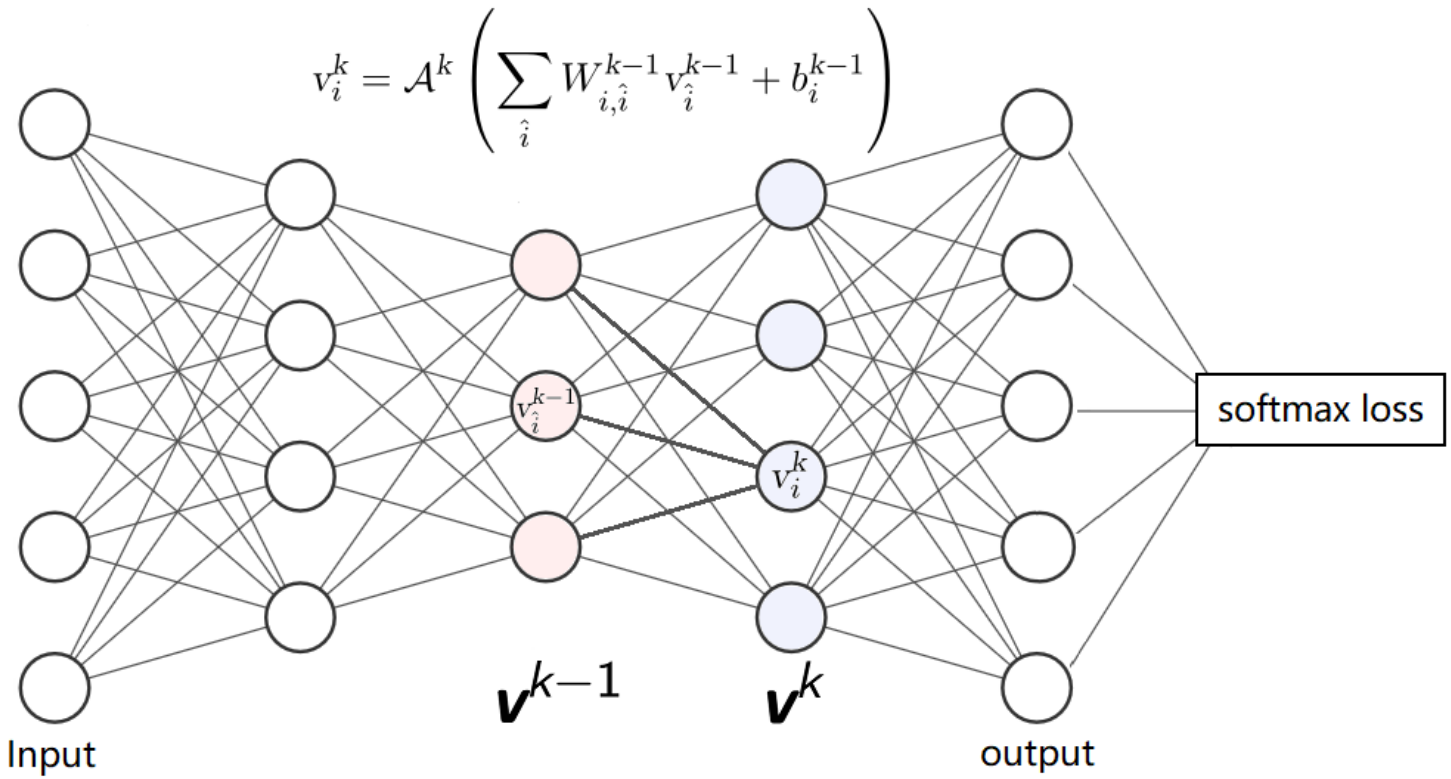
General Neural Network for Semantic Image Segmentation

- The algorithm of learning is a gradient descent method:

$$(\mathbf{\Theta}^k)^{step} = (\mathbf{\Theta}^k)^{step-1} - \tau_\Theta \frac{\delta \mathcal{L}}{\delta \mathbf{\Theta}^k}\Big|_{\mathbf{\Theta}^k=(\mathbf{\Theta}^k)^{step-1}}, \quad k = 0, \cdots, K-1,$$

  where $step = 1, 2, \cdots$ is the iteration number and τ_Θ is a time step or so called learning rate. $\frac{\delta \mathcal{L}}{\delta \mathbf{\Theta}^k}$ can be calculated by backpropagation technique using chain rule.

Networks


$v_i^k = \mathcal{A}^k\left(\sum_{\hat{i}} W_{i,\hat{i}}^{k-1} v_{\hat{i}}^{k-1} + b_i^{k-1}\right)$
$v_{\hat{i}}^{k-1}$
v_i^k
softmax loss
v^{k-1}
v^k
Input
output

Variational explanation of softmax

- Given a vector $\boldsymbol{o} = (o_1, o_2, \ldots, o_I) \in \mathbb{R}^I$, the standard (unit) softmax function $\mathcal{S} : \mathbb{R}^I \to \mathbb{R}^I$ is defined by the formula:

$$\mathcal{S}(\boldsymbol{o})_i = \frac{e^{o_i}}{\sum_{j=1}^{n} e^{o_j}}, i = 1, \ldots, I. \tag{8}$$

Variational explanation of softmax

- In segmentation case, softmax could be derived from a minimization problem. When given $\boldsymbol{o} \in \mathbb{R}^{I \times N_1 N_2}$ as the input, I is the number of classes, $N_1 \times N_2$ is the image size, we want to find a corresponding output $\boldsymbol{u} \in \mathbb{R}^{I \times N_1 N_2}$ such that $\boldsymbol{u}$ is the minimizer of the following problem:

$$\begin{aligned} &\min - < \boldsymbol{u}, \boldsymbol{o} > + < \boldsymbol{u}, \log \boldsymbol{u} >, \\ &s.t.\ \boldsymbol{u} \in \mathcal{C}. \end{aligned} \tag{9}$$

$$\mathcal{C} = \{\boldsymbol{u} |\ \boldsymbol{u}_{ip} \in [0,1],\ \sum_{i=1}^{I} \boldsymbol{u}_{ip} = 1, \forall p = 1, \ldots, N_1 N_2\}.$$

Variational explanation of softmax

- Let $\boldsymbol{u} = (\boldsymbol{u}_1, \ldots, \boldsymbol{u}_I), \boldsymbol{u}_i \in \mathbb{R}^{N_1 N_2}$ for $i = 1, 2, \ldots I$. Some simple calculations can show that the minimizer of the above problem is:

$$\hat{\boldsymbol{u}}_i^* = \frac{\exp(\boldsymbol{o}_i)}{\sum_{j=1}^{C} \exp(\boldsymbol{o}_j)}, i = 1, \ldots, I. \tag{10}$$

$\hat{\boldsymbol{u}}_i^*$ is the i-th class probability map of the input image. One can easily see that this is just the commonly used softmax activation function, i.e.

$$\hat{\boldsymbol{u}}^* = \mathcal{S}(\boldsymbol{o}) \tag{11}$$

However, this function doesn't have any spatial regularization. Prediction of each pixel is independent of other pixels.

Spatial regularization through activation function

Our idea is to add spatial regularization through activation functions.

Proposed Regularization Neural Network

- Inspired by the soft-max variational problem, we proposed the following regularized softmax:

$$\tilde{u}(\boldsymbol{o}) = \arg\min_{\boldsymbol{u}\in\mathcal{C}}\{- < \boldsymbol{u}, \boldsymbol{o} > + < \boldsymbol{u}, \log \boldsymbol{u} > +\lambda \mathrm{TV}(\boldsymbol{u})\}, \tag{12}$$

where the item $\lambda TV(\boldsymbol{u})$ is defined by

$$\mathrm{TV}(\boldsymbol{u}) = \sum_{i=1}^{I} \int_{\Omega} |\nabla \boldsymbol{u}_i(x)| dx = \sup_{(\xi_1,\ldots,\xi_i)\in\mathbb{B}} \left\{ \sum_{i=1}^{I} \int_{\Omega} \boldsymbol{u}_i(x) div \boldsymbol{\xi}_i(x) dx \right\}$$

$\mathrm{TV}(\boldsymbol{u})$ is the total variation of the vector image $\boldsymbol{u}$.

$$\mathbb{B} = \{(\xi_1, \ldots, \xi_I) |\ \|(\xi_i)\|_{l^2} \leq 1, \forall i = 1, \ldots, I\}.$$

Proposed Regularization Neural Network

- the first problem in (12) can be easily solved by primal-dual method:

$$(\tilde{u}, \boldsymbol{\eta}) = \arg\min_{\boldsymbol{u}\in\mathcal{C}} \max_{\boldsymbol{\xi}\in\mathbb{B}} \{ - < \boldsymbol{u}, \boldsymbol{o} > + < \boldsymbol{u}, \log \boldsymbol{u} > + \lambda < \boldsymbol{u}, div\boldsymbol{\xi} > \}$$

Proposed Regularization Neural Network

- We can use the following primal-dual gradient algorithm to find the solution for

$$\min_{\boldsymbol{u}\in\mathcal{C}} \max_{\boldsymbol{\xi}\in\mathbb{B}} - < \boldsymbol{u}, \boldsymbol{o} > + < \boldsymbol{u}, \log \boldsymbol{u} > + \lambda < \boldsymbol{u}, div\boldsymbol{\xi} >, \tag{14}$$

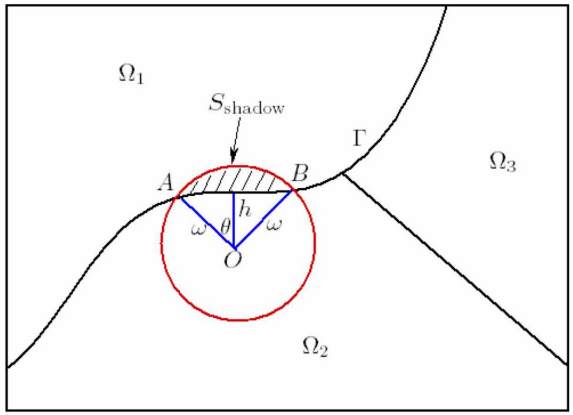
  in an iterative way:

$$\begin{cases} \boldsymbol{\xi}^{t+1} & = \boldsymbol{\xi}^{t} - \tau\lambda\nabla\boldsymbol{u}^{t}, \\ \boldsymbol{\eta}^{t+1} & = \mathcal{P}_{\mathbb{B}}(\boldsymbol{\xi}^{t+1}), \\ \boldsymbol{u}^{t+1} & = \mathcal{S}(\boldsymbol{o} - \lambda div(\boldsymbol{\eta}^{t+1})), \end{cases} \tag{15}$$

  where $\mathcal{S}$ is the softmax operator, t is the iteration number and τ is a time step, $\mathcal{P}_{\mathbb{B}}$ is a projection operator onto the convex set $\mathbb{B}$.
- Notation: The item $\tau\lambda$ in Equation (15) could be seen as a scaled step size, we set it to a fixed constant in all computation.

Spatial regularization with TD (Threshold Dynamics)

TD regularization



- MBO (Threshold Dynamics (TD). For a binary u_i:

$$\sum |\partial\Omega_i| = \sum TV(u_i) \approx \sum \sqrt{\frac{\pi}{\sigma}} \sum_{\hat{i}=1,\hat{i}\neq i}^{C} \int_\Omega u_i(x)(k_\sigma * u_{\hat{i}})(x)dx := \mathcal{R}(u)$$

k_σ is the Gaussian kernel with width σ or an indicator function of the ball of radius σ.

Entropic & spatial regularization

- Regularized and geometry prior based softmax

$$\mathcal{A} = \arg \min_{u \in \mathcal{C} \bigcap \mathcal{P}} \left\{ \underbrace{< -o, u > + \varepsilon \overbrace{< u, \log u >}^{\mathcal{H}(u)}}_{\mathcal{F}(u;o)} + \mathcal{R}(u). \right\}$$

- $\mathcal{C}$—-segmentation condition.
- $\mathcal{P}$—-geometry shape constraint or volume constraint.
- $\mathcal{F}$—-dual formulation of smooth max function.
- $\mathcal{H}$—- negative entropic regularization (for smooth backpropagation).
- $\mathcal{R}$—-spatial regularization (for smoothness segmentation boundaries).

Application 1 : STD-softmax (Soft Threshold Dynamics softmax)

- Taking $\mathcal{R}$ to be the threshold dynamics (TD) regularization, we get soft threshold dynamics (STD)

$$\tilde{\boldsymbol{u}} = \arg\min_{\boldsymbol{u}\in\mathcal{C}} \left\{ \underbrace{< -\boldsymbol{o}, \boldsymbol{u} > + \varepsilon < \boldsymbol{u}, \log \boldsymbol{u} >}_{\mathcal{F}(\boldsymbol{u})} + \mathcal{R}(\boldsymbol{u}). \right\}$$

- $\mathcal{R}(\boldsymbol{u}) = \langle e\boldsymbol{u}, k_\sigma * (1-\boldsymbol{u}) \rangle$, where e is an edge detector weighting function and k is a Gaussian kernel.

Stable and fast algorithm for unrolling

- $\mathcal{F}$ is convex, $\mathcal{R}$ is concave.
- Efficient algorithm (DCA, difference of convex algorithm)

$$\boldsymbol{u}^{t_1+1} = \arg\min_{\boldsymbol{u}\in\mathcal{C}} \left\{ \mathcal{F}(\boldsymbol{u};\boldsymbol{o}) + \mathcal{R}(\boldsymbol{u}^{t_1}) + \left\langle \boldsymbol{p}^{t_1}, \boldsymbol{u} - \boldsymbol{u}^{t_1} \right\rangle \right\}.$$

- $\boldsymbol{p}^{t_1} = \lambda((k_\sigma * (1 - \boldsymbol{u}^{t_1}))e - k_\sigma * (e\boldsymbol{u}^{t_1})) \in \partial\mathcal{R}(\boldsymbol{u}^{t_1})$.
- Regularized softmax solution

$$u_i^{t_1+1}(x) = \frac{e^{\frac{o_i(x) - p_i^{t_1}(x)}{\varepsilon}}}{\sum_{\hat{i}=1}^{I} e^{\frac{o_{\hat{i}}(x) - p_{\hat{i}}^{t_1}(x)}{\varepsilon}}} = \text{softmax}_\varepsilon(\boldsymbol{o} - \boldsymbol{p}^{t_1}).$$

STD-softmax

Algorithm 1: STD-softmax

Input: The feature $\boldsymbol{o}$

Output: Soft segmentation function $\boldsymbol{u}$.

Initialization: $\mathbf{u}^0 = \mathcal{S}(\boldsymbol{o})$.

for $t_1 = 0, 1, 2, \cdots$ **do**

1. compute the solution of (7) by STD-softmax

$$\boldsymbol{u}^{t_1+1} = \mathcal{S}\left(\frac{\boldsymbol{o} - \boldsymbol{p}^{t_1}}{\varepsilon}\right).$$

2. Convergence check. If it is converged, end the algorithm.

end

return Segmentation function $\boldsymbol{u}$.

Proposition

(Energy decay). Let $\boldsymbol{u}^{t_1}$ be the t_1-th iteration of STD-softmax algorithm, then we have $\mathcal{F}(\boldsymbol{u}^{t_1+1}) + \mathcal{R}(\boldsymbol{u}^{t_1+1}) \leq \mathcal{F}(\boldsymbol{u}^{t_1}) + \mathcal{R}(\boldsymbol{u}^{t_1})$.

Proposed STD DCNN segmentation

- Proposed STD softmax (Liu,Wang and Tai 2020))[1]:

$$\begin{cases} \boldsymbol{o}^t = & \mathcal{T}_{\Theta^{t-1}}(\boldsymbol{v}^{t-1}, \boldsymbol{v}^{t-2}, \cdots, \boldsymbol{v}^0), \\ \boldsymbol{v}^t = & \arg\min_{\boldsymbol{u}\in\mathcal{C}} \left\{ \mathcal{F}(\boldsymbol{u}; \boldsymbol{o}^t) + \mathcal{R}(\boldsymbol{u}) \right\}, t = 1, \cdots, T. \end{cases}$$

- Superiority : spatial prior can be kept both in back and forward propagation.

Results on PASCAL VOC 2012 dataSet.

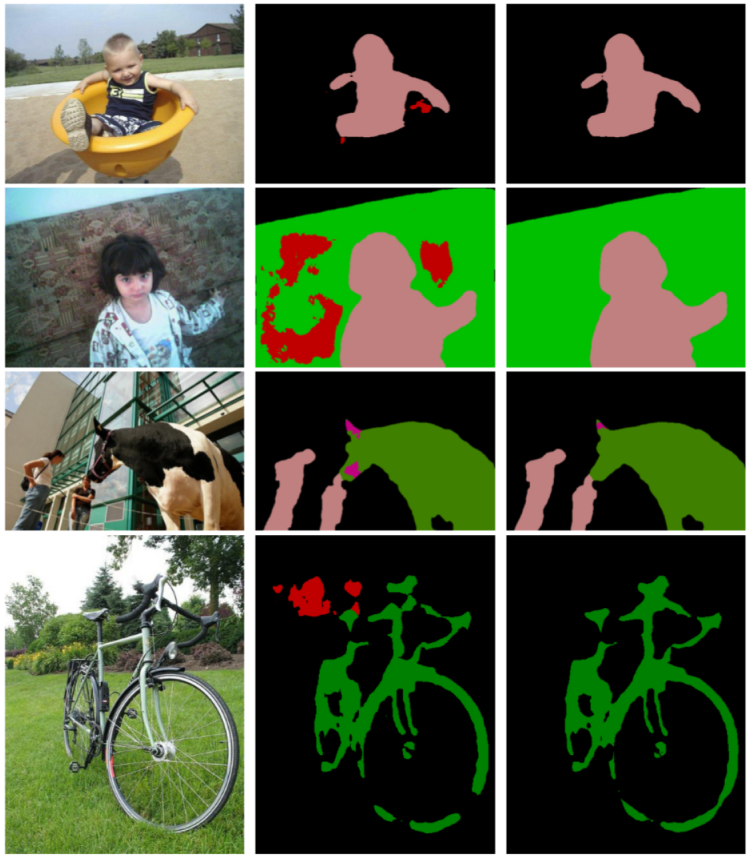


Figure 5: Visual effects of the DeepLabV3+ and proposed STD-DeeplabV3+ on PASCAL VOC 2012 test set.

Star-shape prior (SS)

Star-shape prior (SS)

Application 3: SS (Star shape)-STD softmax

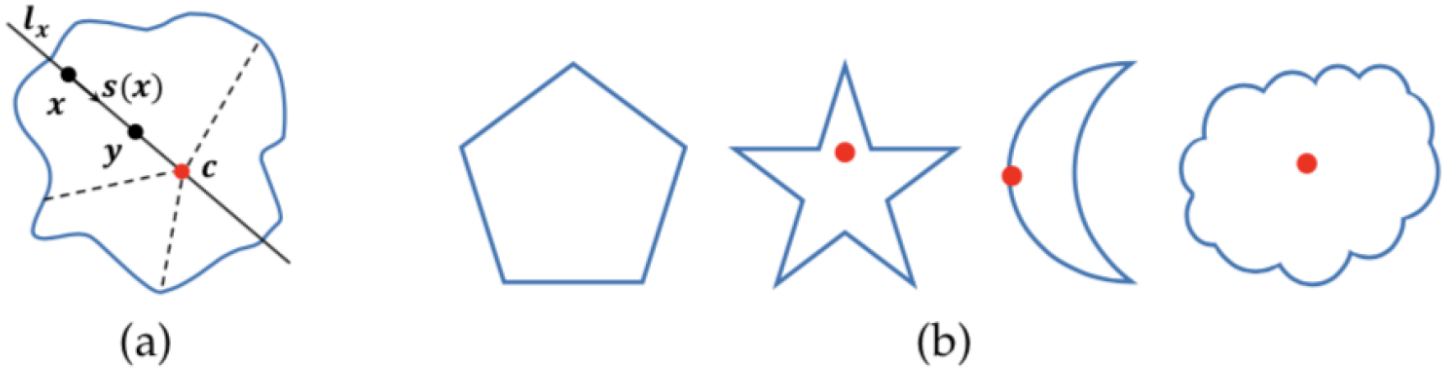


Figure 10: Star shape objects with given centers

If $u : \Omega \to \{0, 1\}$ is the indicator function of a region, then this region is star-shape iff

$$\nabla u(x) \cdot (x - c) \geq 0, \quad \forall x \in \Omega.$$

Here c is center (red point). We will denote $\boldsymbol{s}(x) = x - c$.
References: Veksler et al (2008), Yuan et al (2012).

Proposed star-shape soft threshold dynamics

- Star-shape softmax: if we want the ith class segmented object to be star-shape, then we need $u_i \in \mathcal{P}$ and solve:

$$\widetilde{\boldsymbol{u}} = \arg \min_{\boldsymbol{u} \in \mathcal{C} \cap u_i \in \mathcal{P}} \{\mathcal{F}(\boldsymbol{u}; \boldsymbol{o}) + \mathcal{R}(\boldsymbol{u})\},$$

$$\mathcal{P} = \{\boldsymbol{u} : \langle \nabla u_i(x), \boldsymbol{s}(x) \rangle \geq 0\}.$$

- Dual problem in terms of KKT condition

$$(\widetilde{\boldsymbol{u}}, \widetilde{q}) = \arg \min_{\boldsymbol{u} \in \mathcal{C}} \max_{q \geqslant 0} \{\mathcal{F}(\boldsymbol{u}; \boldsymbol{o}) + \mathcal{R}(\boldsymbol{u}) - \langle q, \boldsymbol{s} \cdot \nabla u_i \rangle\}.$$

- Algorithm

$$\begin{cases} q^{t_1+1} = & \max\{q^t - \tau_q \boldsymbol{s} \cdot \nabla u_i^t, 0\}. \\ \boldsymbol{u}^{t_1+1} = & \arg \min_{\boldsymbol{u} \in \mathcal{C}} \left\{\mathcal{F}(\boldsymbol{u}; \boldsymbol{o}) + \hat{\mathcal{R}}(\boldsymbol{u}) + \langle div(q^{t_1+1} \boldsymbol{s}), u_i \rangle\right\}, \end{cases}$$

SS-STD softmax segmentation algorithm

Algorithm 3: SS-STD softmax

Input: The feature $\boldsymbol{o}$, and a center c of star-shape.
Output: Soft segmentation function $\boldsymbol{u}$.
Initialization: $\mathbf{u}^0 = \mathcal{S}(\boldsymbol{o})$. Calculating the star-shape vector field $\boldsymbol{s}$ according to c.
for $t_1 = 0, 1, 2, \cdots$ **do**

1.update dual variable for the i-th star-shape region

$$q^{t_1+1} = \max\{q^{t_1} - \tau_q \boldsymbol{s} \cdot \nabla u_i^{t_1}, 0\}.$$

2. compute the solution of (7) by SS-STD softmax

$$\boldsymbol{u}_{\hat{i}}^{t_1+1} = \mathcal{S}\left(\frac{\boldsymbol{o}_{\hat{i}} - \boldsymbol{p}_{\hat{i}}^{t_1} - \delta_{\hat{i},i} div(q^{t_1+1}\boldsymbol{s})}{\varepsilon}\right), \ \hat{i} = 1, \cdots$$

3. Convergence check. If it is converged, end the algorithm.

end
return Segmentation function $\boldsymbol{u}$.

The network architecture of STD -DeepLabV3+

$$\begin{cases} \boldsymbol{o}^t = & \mathcal{T}_{\boldsymbol{\Theta}^{t-1}}(\boldsymbol{v}^{t-1}, \boldsymbol{v}^{t-2}, \cdots, \boldsymbol{v}^0), \\ \boldsymbol{v}^t = & \underset{\boldsymbol{u}\in\mathcal{C}, u_i\in\mathcal{P}}{\arg\min} \left\{ \mathcal{F}(\boldsymbol{u}; \boldsymbol{o}^t) + \mathcal{R}(\boldsymbol{u}) \right\}, t = 1, \cdots, T. \end{cases}$$

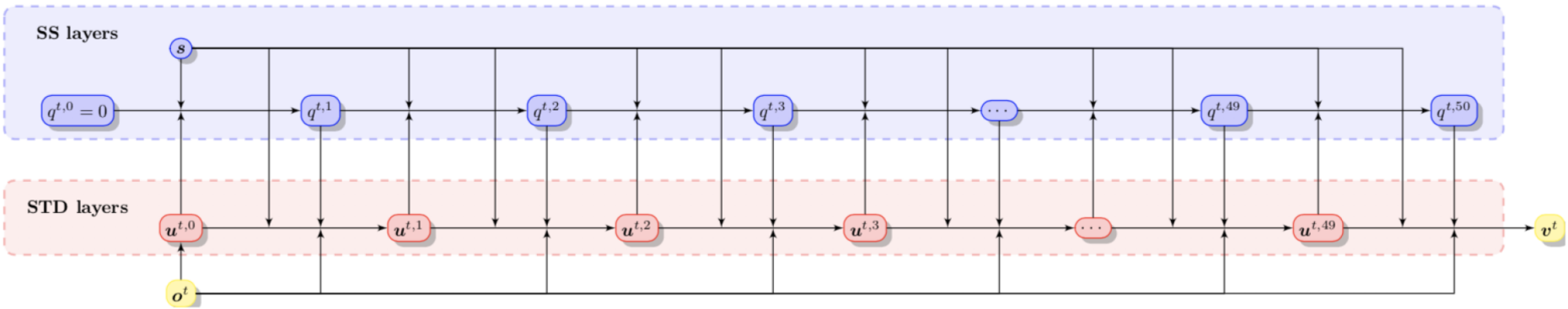


Figure 11: SS-STD block for DCNN. This architecture is constructed according to the SS-STD algorithm.

Comparison

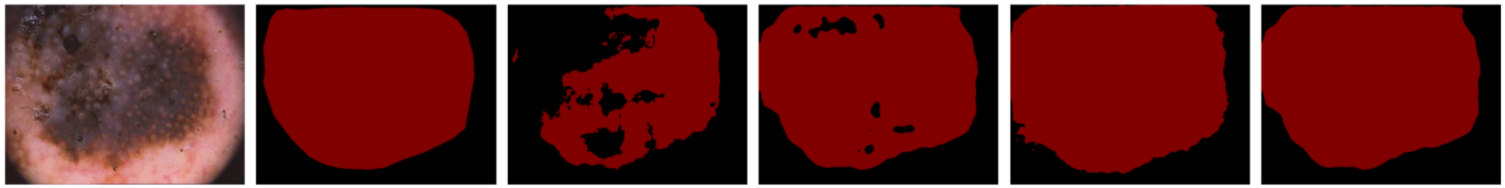
Testing of Algorithm 3 as a segmentation tool:

(a) Feature o (b) Softmax (c) STD (d) SS-STD (e) SS-STD

Figure 12: Segmentation results by softmax, proposed STD and SS-STD softmax.

Comparison

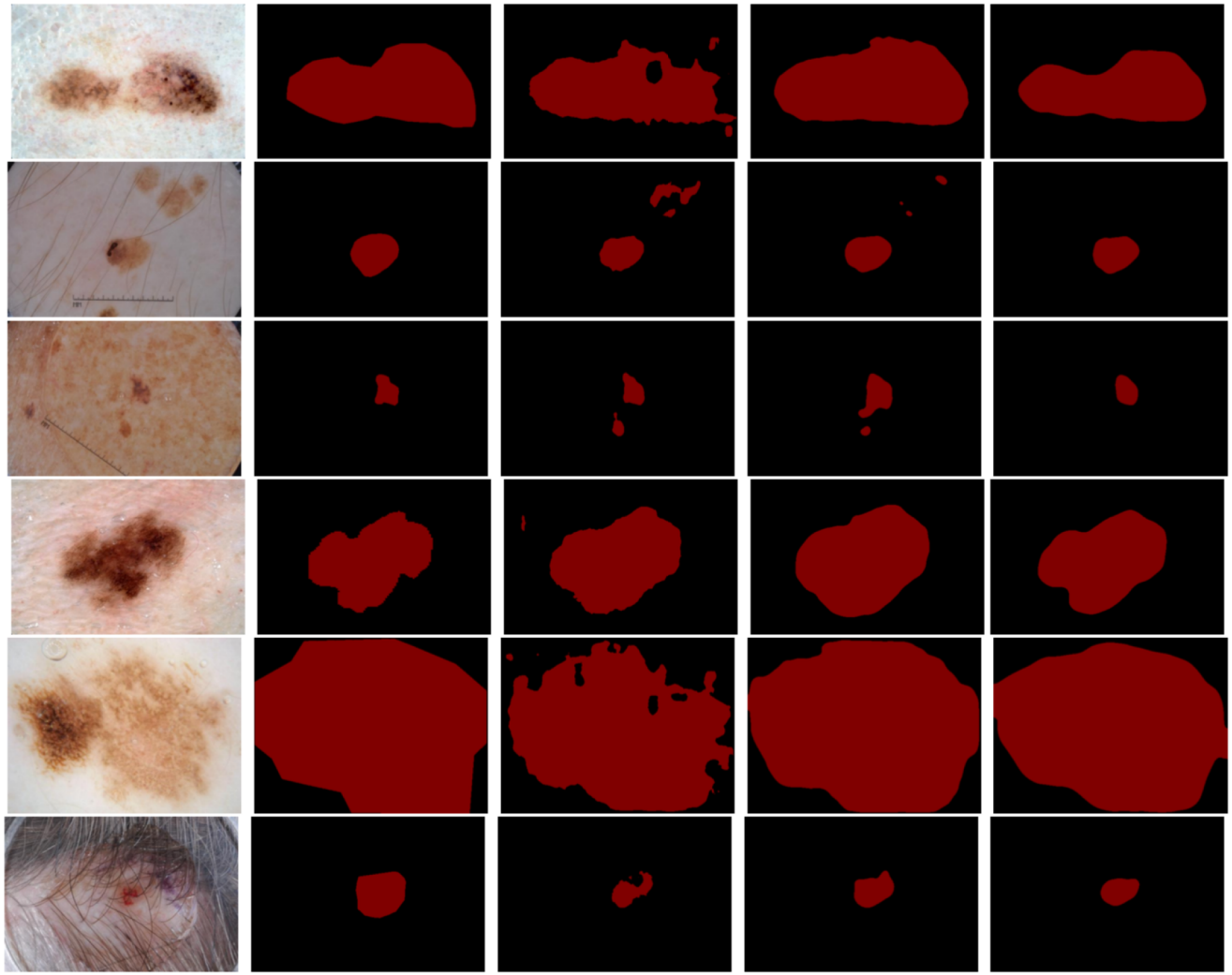
Testing when Algorithm 3 is integrated into the CNN network for:



(a) image (b) ground truth (c) DeepLabV3+ [38] (without spatial priori) (d) with regularization (e) with volume preserving & regularization (f) with star-shape & regularization

Figure 13: An example of without and with the proposed spatial priori for DeepLabV3+ on ISIC2018 validation set.

Results on ISIC2018 validation set.



(a) Images (b) Ground Truth (c) DeepLabV3+ [38] (d) STD (e) SS-STD

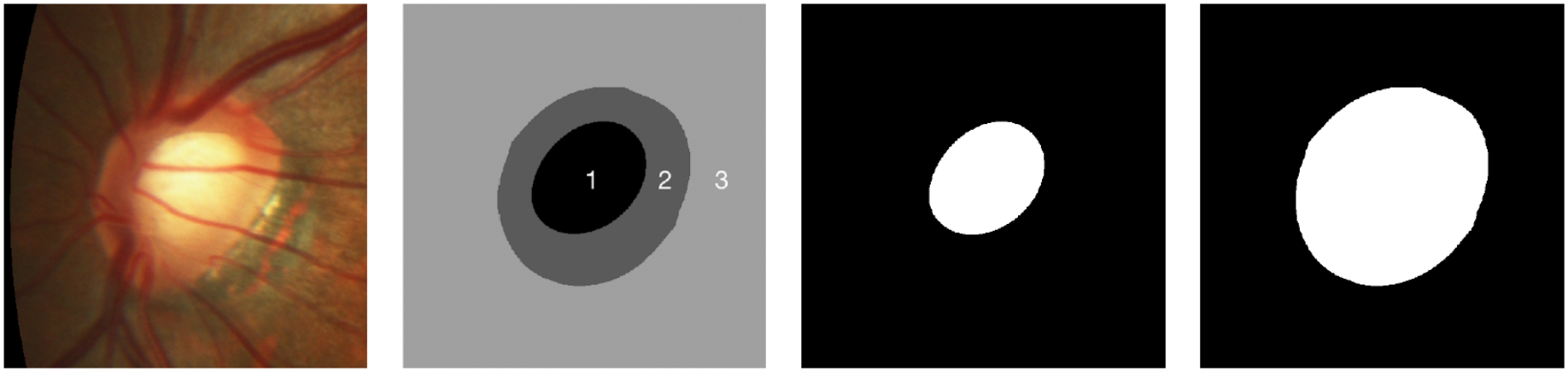
Performance on ISIC2018 validation set

	Methods	mIoU
Baseline	DeepLabV3+ [38]	89.77
Ours	STD	91.02
	VP-STD	92.46
	SS-STD	91.57

Convex shape (CS) prior

Convex shape (CS) prior

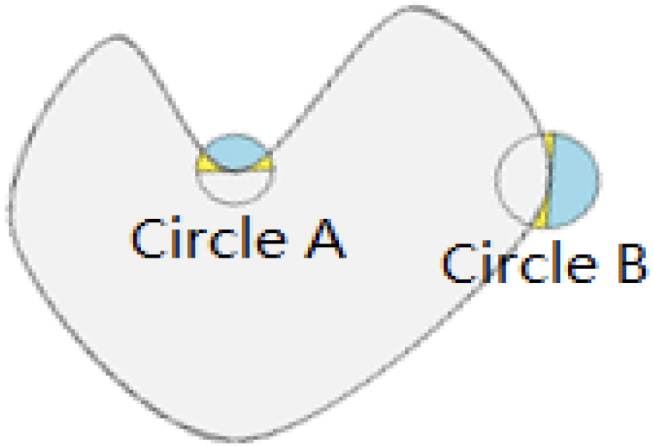
Convex shape prior



(a) image (b) label $l(x)$ (c) sublevel set u_1 (d) sublevel set u_2

Figure 14: The cup and disc areas in retinal images with sublevel set functions u1 and u2 representation can be both convex.

Conditions for convex shapes with binary representation



- Convex shape condition(Luo,Tai,Wang,2019)[4]:

$$g_r(x) = \begin{cases} \frac{1}{|\mathbb{B}_r|}, & x \in \mathbb{B}_r \subset \Omega, \\ 0, & \text{else.} \end{cases}$$

- If $u \in \mathcal{C}_{CS}$ with $\mathcal{C}_{CS}$ being defined as

$$\mathcal{C}_{CS} = \{u : \ (1-u(x))(g_r * (1-2u))(x) \geqslant 0, \forall\, \mathbb{B}_r \subset \Omega, \forall x \in \Omega\},$$

then the connected components of $\mathbb{D}$ are all convex.

CS-STD (Convex Shape Soft Threshold Dynamics) sigmoid

- Convex shape (CS) soft thresholding dynamics (STD):

$$\widetilde{u} = \underset{u\in[0,1]\bigcap \mathcal{C}_{CS}}{\arg\min} \{\underbrace{\langle -o, u\rangle + \varepsilon(u\ln u + (1-u)\ln(1-u))}_{:=\mathcal{F}(u;o)} + \underbrace{\langle eu, k*(1-u)\rangle}_{:=\mathcal{R}(u)}\}.$$

- $\mathcal{C}_{CS}$ is a quadratic constraint for convex shape.

Algorithm

- Apply DCA

$$u^{t_1+1} = \underset{u\in[0,1]\bigcap \mathcal{C}_{CS}}{\arg\min} \left\{ \mathcal{F}(u;o) + \mathcal{R}(u^{t_1}) + \langle p^{t_1}, u - u^{t_1} \rangle \right\}.$$

- $p^{t_1} = (k * (1 - u^{t_1}))e - k * (eu^{t_1}) \in \partial\mathcal{R}(u^{t_1})$
- No closed-form solution.

Our new algorithm

- Pseudo projection algorithm

$$\begin{cases} u^{t_1+\frac{1}{2}} &= \arg\min_{u} \left\{ \mathcal{F}(u;o) + \lambda \langle p^{t_1}, u \rangle \right\}, \\ u^{t_1+1} &= \text{Proj}_{[0,1]\bigcap \mathcal{C}_{CS}}(u^{t_1+\frac{1}{2}}). \end{cases}$$

- The first subproblem

$$u^{t_1+\frac{1}{2}} = \frac{1}{1+e^{\frac{-o+\lambda p^{t_1}}{\varepsilon}}} = \mathcal{S}(\frac{o-\lambda p^{t_1}}{\varepsilon}).$$

- The second subproblem has sigmoid solution can be solved by a simple pseudo projection.

$$u^{t_1+1} = \arg\min_{u \in [0,1]\bigcap \mathcal{C}_{CS}} ||u - u^{t_1+\frac{1}{2}}||^2.$$

Algorithms

Algorithm 4: $\text{Proj}_{[0,1]\bigcap \mathbb{C}}$ for convex shapes

Input: $u^{t_1+\frac{1}{2}}$. Different sphere radius $\boldsymbol{r} = (r_0, r_1, r_2, r_3, r_4)$.

Initialization: $u^0 = u^{t_1+\frac{1}{2}}$

for $t_2 = 0, 1, \cdots$ **do**

1. Set $r = r_{\text{mod}(t_2,5)}$.
2. Find the active set

$$\mathbb{A} = \{x : (1 - u^{t_2}(x))(g_r * (1 - 2u^{t_2}))(x) < 0\}$$

2. Update

$$u^{t_2+1}(x) = \begin{cases} 1, & x \in \mathbb{A}, \\ u^{t_2}(x), & x \notin \mathbb{A}. \end{cases}$$

3. Convergence check. If it is converged, end the algorithm.

end

Output: Segmentation function $u^{t_1+1} = u^{t_2+1}$ with convex shape.

Algorithm 5: CS-STD sigmoid activation function

Input: The feature o.

Initialization: $u^0 = \mathcal{S}(o)$.

for $t_1 = 0, 1, 2, \cdots$ **do**

1. Compute the solution of the first subproblem in by regularized STD sigmoid.
2. Calculate the pseudo projection $u^{t_1+1} = \text{Proj}_{[0,1]\bigcap \mathbb{C}}(u^{t_1+\frac{1}{2}})$ by Algorithm 4.
3. Convergence check. If it is converged, end the algorithm.

end

Output: Segmentation function u with convex shape prior.

New CS-STD sigmoid block for DCNN

$$
\begin{cases}
\begin{cases} \boldsymbol{o}^t = & \mathcal{T}_{\boldsymbol{\Theta}^{t-1}}(\boldsymbol{v}^{t-1}, \boldsymbol{v}^{t-2}, \cdots, \boldsymbol{v}^0), t = 1, \cdots, T, \\ \boldsymbol{v}^t = & \mathcal{A}^t(\boldsymbol{o}^t), t = 1, \cdots, T-1, \end{cases} \\
\boldsymbol{v}^T = \underset{\boldsymbol{u} \in [0,1] \bigcap \mathcal{C}_{CS}}{\arg\min} \{\mathcal{F}(\boldsymbol{u}; \boldsymbol{o}^T) + \lambda \mathcal{R}(\boldsymbol{u})\}.
\end{cases}
$$

Here $\mathcal{F}$ is the dual representation with sublevel set functions.

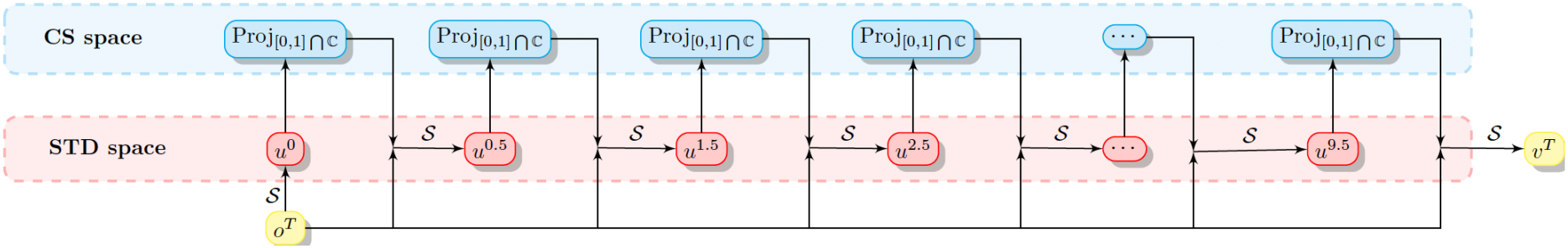
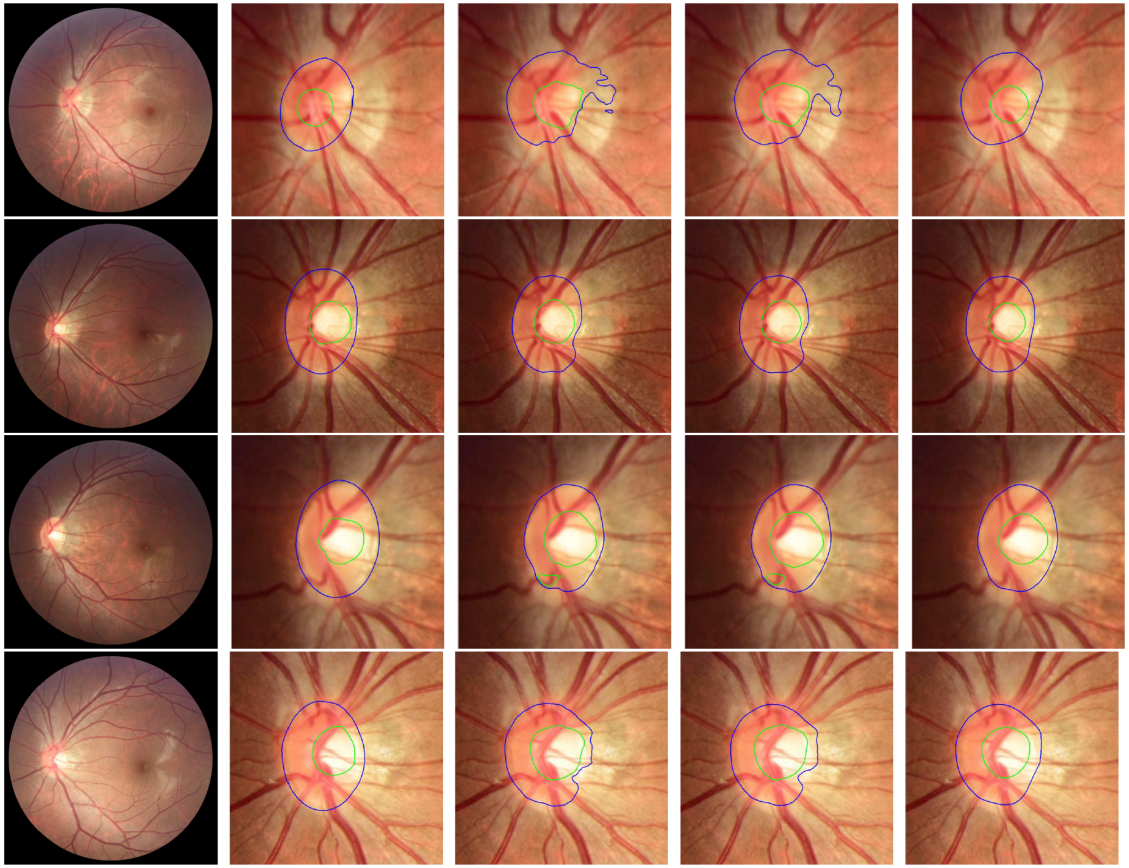


Figure 15: CS-STD block for DCNN. This architecture is constructed according to the CS-STD algorithm.

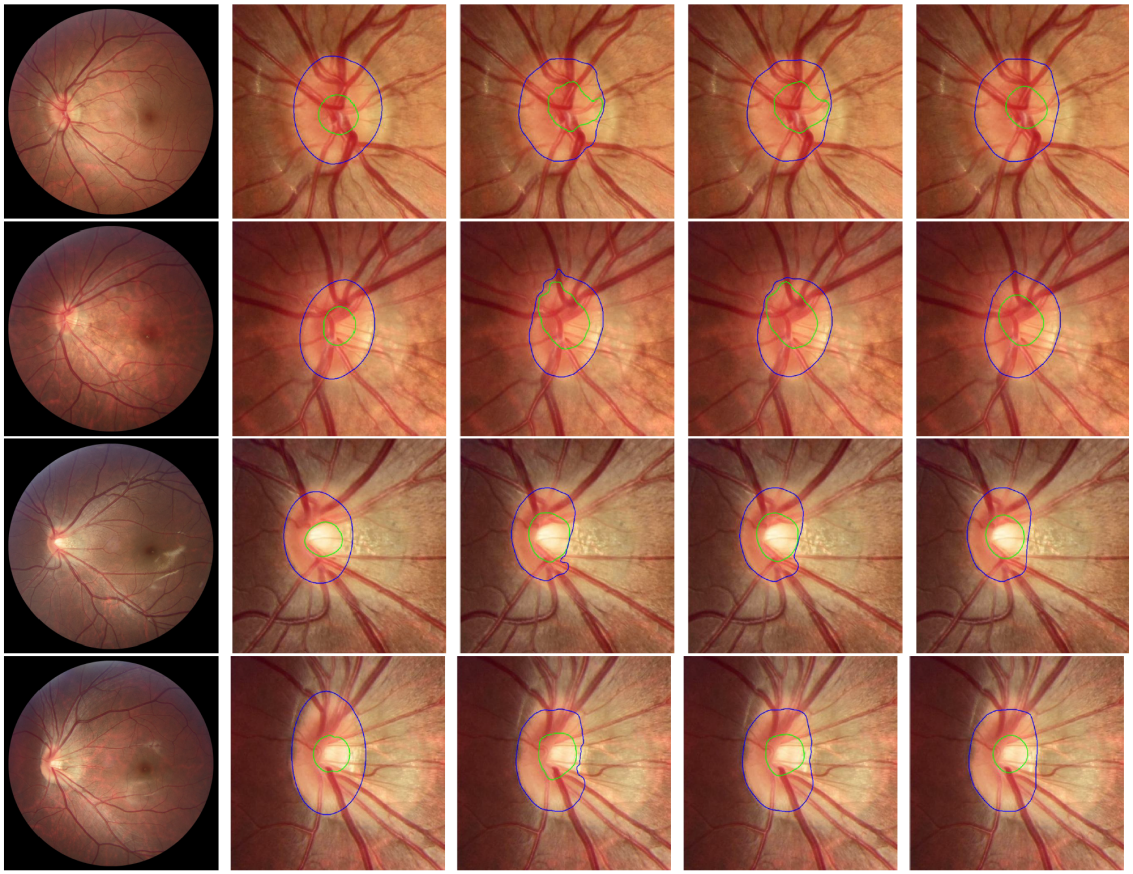
Visual quality



(a) Images (b) Ground truth (c) DeeplabV3+ (d) STD (e) CS-STD

Figure 16: Visual quality of the sigmoid, STD-sigmoid, CS-STD-sigmoid on Refuge test set.

Visual quality



(a) Images (b) Ground truth (c) DeeplabV3+ (d) STD (e) CS-STD

Figure 17: Visual quality of the sigmoid, STD-sigmoid, CS-STD-sigmoid on Refuge validation set.

Generalization ability

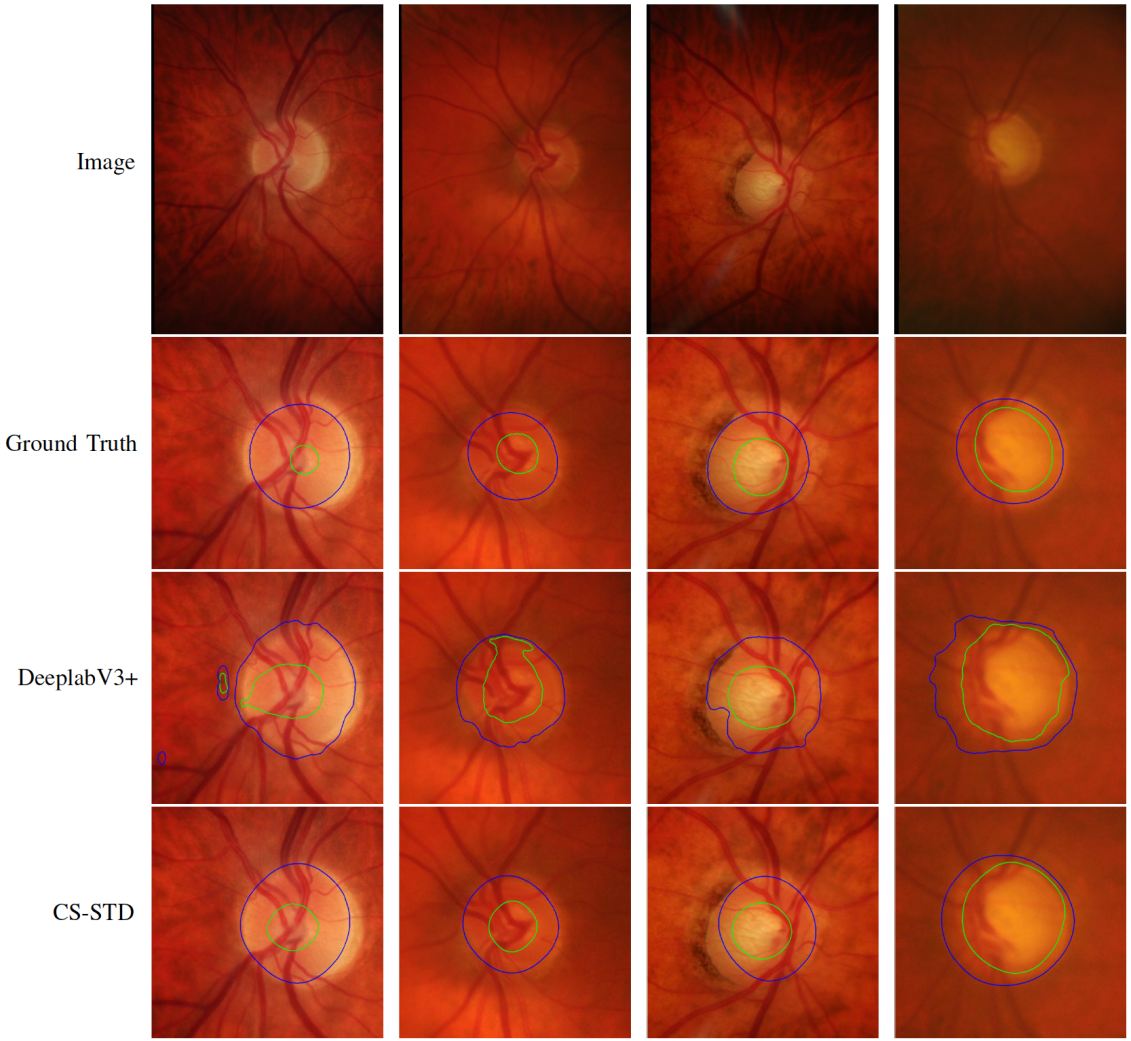


Figure 18: Visual quality of training on Refuge train set and predicting on RIM-ONE-r3 test set.

Comparison

	methods	val. set disc	val. set cup	test set disc	test set cup
Existing	STD [29]	95.1	86.7	95.2	85.0
	pOSAL-seg [27]	93.2	86.9	-	-
Baseline	DeeplabV3+ [20]	95.0	86.4	95.1	84.3
Proposed	CS-STD	**95.1**	**88.3**	**95.2**	**87.7**

Figure 19: DM (dice measure) values of different methods for Refuge validation and test sets.

	methods	disc	cup
Existing	TD-GAN [35]	85.3	72.8
	Hoffman *et al.* [36]	85.2	75.5
	Javanmardi *et al.* [37]	85.3	77.9
	pOSAL [27]	86.5	78.9
Baseline	DeeplabV3+ [20]	85.4	70.9
Proposed	CS-STD	**92.2**	**80.7**

Figure 20: DM values of different methods for training on Refuge train set and predicting on RIM-ONE-r3 test set.

Numerical Results of CS-STD

Noise levels	σ								
	0	1	2	3	4	5	6	7	8
DeepLabV3+ [20]	84.3	84.2	84.1	83.8	83.6	83.1	82.6	82.4	81.7
CS-STD	87.7	87.6	87.5	87.3	86.9	86.6	86.2	85.9	85.5

Noise levels	σ								
	9	10	11	12	13	14	15	16	17
DeepLabV3+ [20]	81.2	80.5	80.1	79.5	79.1	78.7	77.8	77.4	76.6
CS-SCT	85.1	84.6	84.0	83.5	83.2	82.7	82.3	81.3	81.2

Noise levels	σ							
	18	19	20	21	22	23	24	25
DeepLabV3+ [20]	76.1	75.3	74.6	73.9	72.8	71.6	70.5	69.0
CS-STD	81.2	79.9	79.8	78.9	77.7	76.9	75.5	74.7

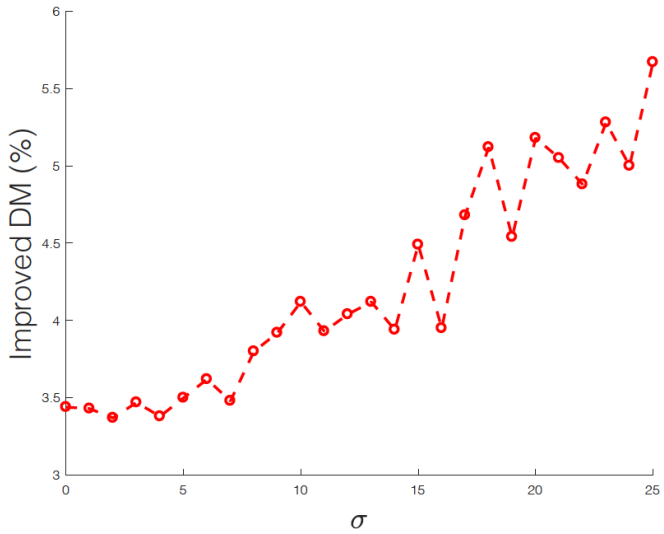


Figure 21: The improved DM values for cup regions in the Refuge test sets (400 images) between DeeplabV3+ and the proposed CS-STD under different levels of noise with standard deviation.

Thank you!

Homepage:
`http://www.math.hkbu.edu.hk/~xuechengtai/HKBU.html`

Thank you!

Homepage:
`http://www.math.hkbu.edu.hk/~xuechengtai/HKBU.html`
Phd Students and postdocs are welcome !!