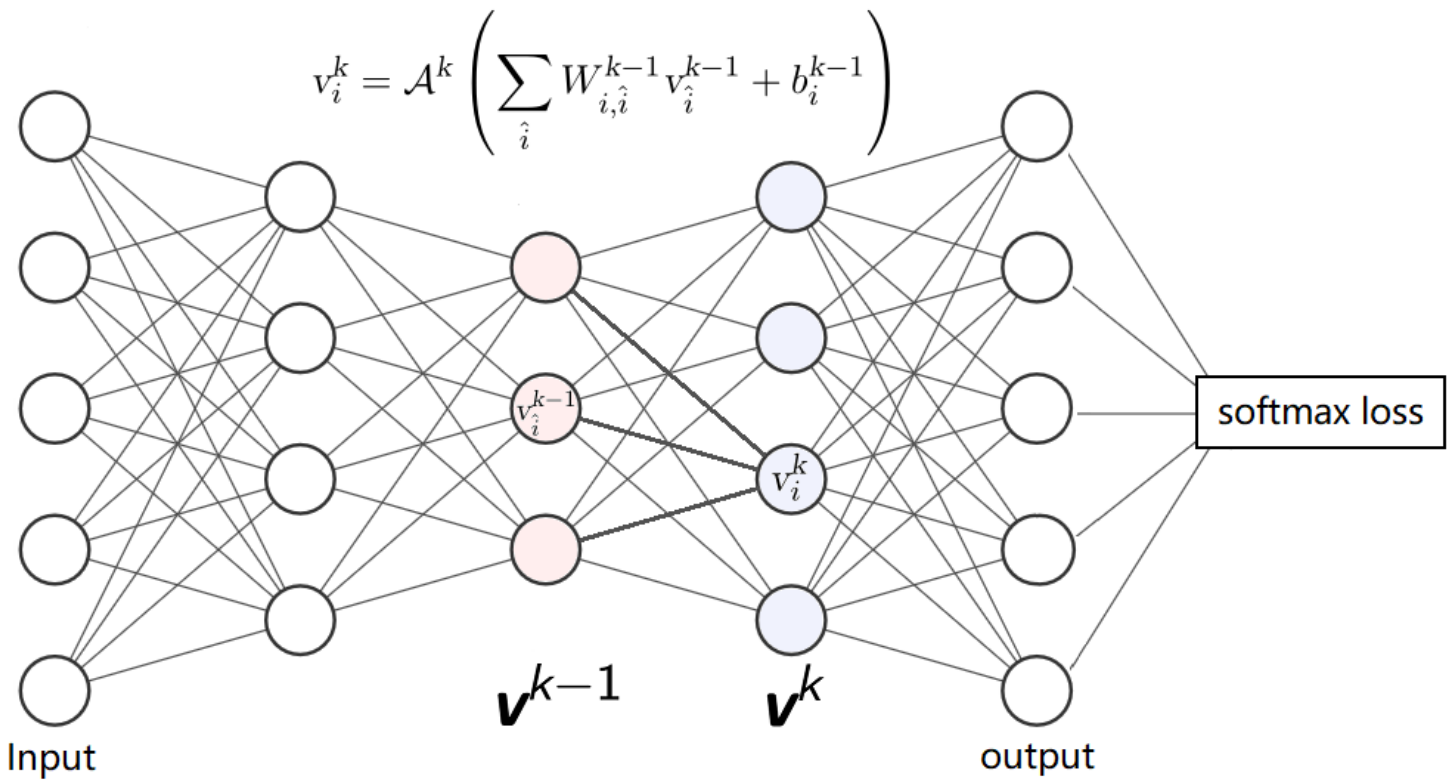
Mathematical Deep Neural Networks and Applications

Xue-Cheng Tai

Norwegian Research Centre, Bergen, Norway.

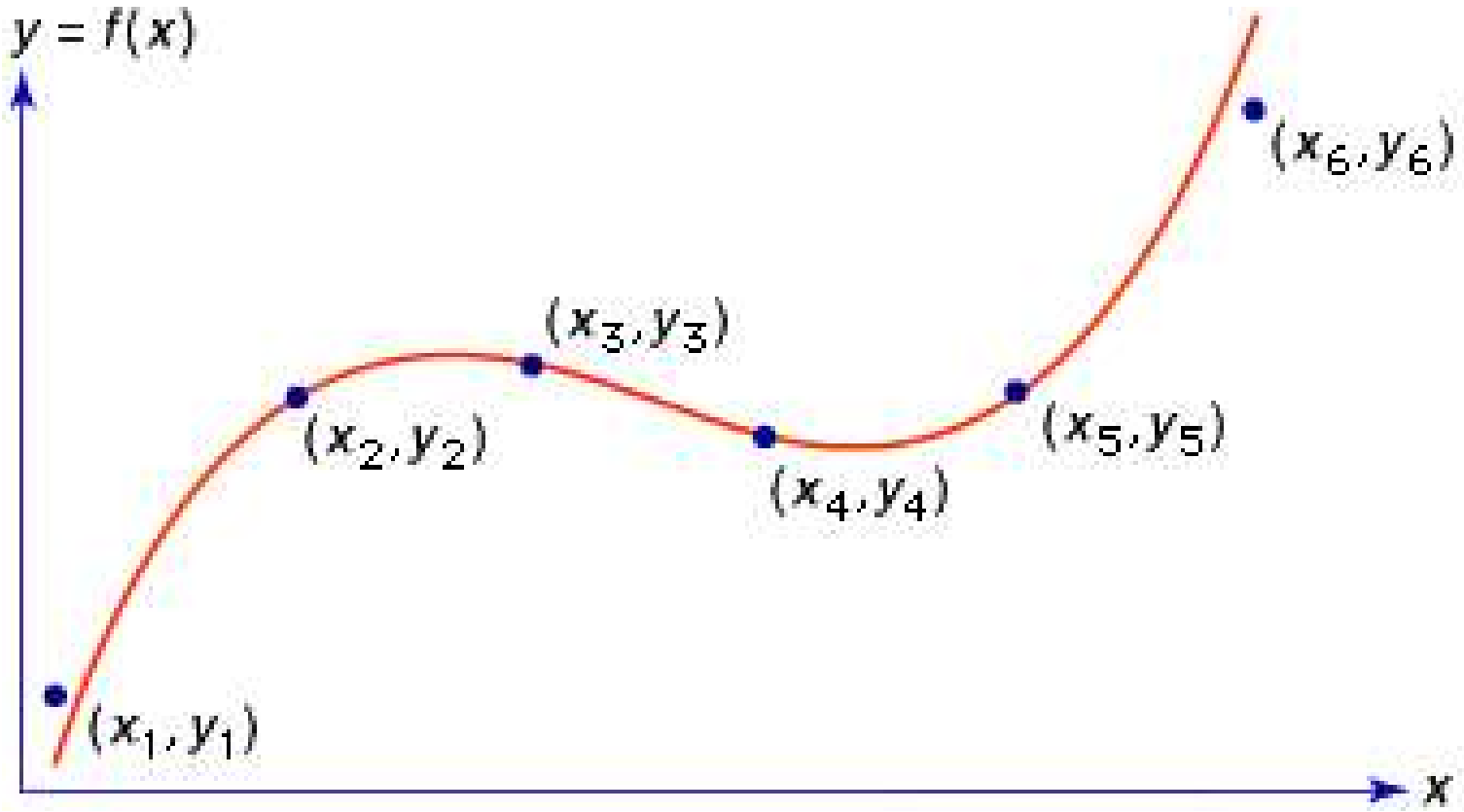
June 10, 2026

Neural Networks in Machine Learning



Deep neural networks have achieved great success. Commonly used neural networks are as illustrated above. It produces excellent results in many cases. It can also fail in some situations. Why?

Learning is an interpolation problem: mathematical explanations

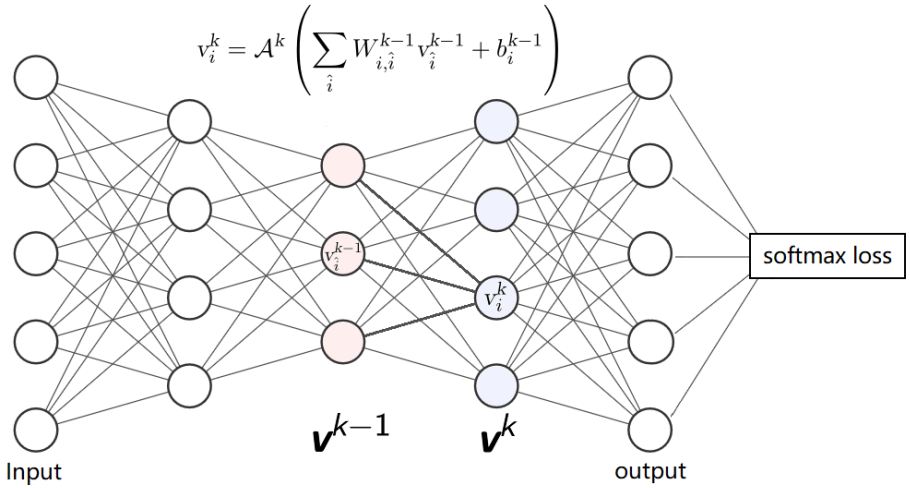


DCNNs: Why it works?

Reference:

Le Cun Y, et al. Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11):2278–2324, 1998.

DNNs: why it works?



$$f(\boldsymbol{x},\theta) = \mathcal{A}^K \circ (\boldsymbol{W}_{K-1} \circ (\cdots \mathcal{A}^0 \circ (\boldsymbol{W}_0 \boldsymbol{x} + \mathbf{b}_0) + \mathbf{b}_1) \cdots),$$

$$\theta = (\boldsymbol{W}_0, \boldsymbol{W}_1, \cdots, \boldsymbol{W}_K, \boldsymbol{b}_0, \boldsymbol{b}_1, \cdots, \boldsymbol{b}_K).$$

$\mathcal{A}^k$ is a scalar function.

- $\mathcal{A}^k(x) = \max(x, 0)$, the ReLU (rectified linear units) function
- $\mathcal{A}^k(x) = (1 + e^{-x})^{-1}$, the "sigmoid function".
- $\mathcal{A}^k(x) = cos(x)$

"$\circ$" means acting on each components, the $\boldsymbol{W}$'s are matrices.

DNNs: why it works?

Difference between linear and nonlinear approximations:

- Linear: $f(\boldsymbol{x}) \approx f_m(\boldsymbol{x}) = \sum_{k=1}^{m} a_{\boldsymbol{k}} cos(2\pi \boldsymbol{k} \cdot x)$, $\boldsymbol{x} \in [0,1]^d$

$$\max_f \|f - f_m\|_2 \geq C(f) m^{-1/d}$$

  "Curse of dimensionality": number of parameters needed goes up exponentially fast as a function of the accuracy requirement.

- Nonlinear: $f(\boldsymbol{x}) \approx f_m(\boldsymbol{x}) = \sum^{m} a_{\boldsymbol{k}} cos(2\pi \boldsymbol{b}_k \cdot x)$, $\boldsymbol{x} \in [0,1]^d$
  (two-layer neural network, with $\cos(x)$ as the activation function).

$$\max_f \|f - f_m\|_2 \leq C(f) m^{-1/2}$$

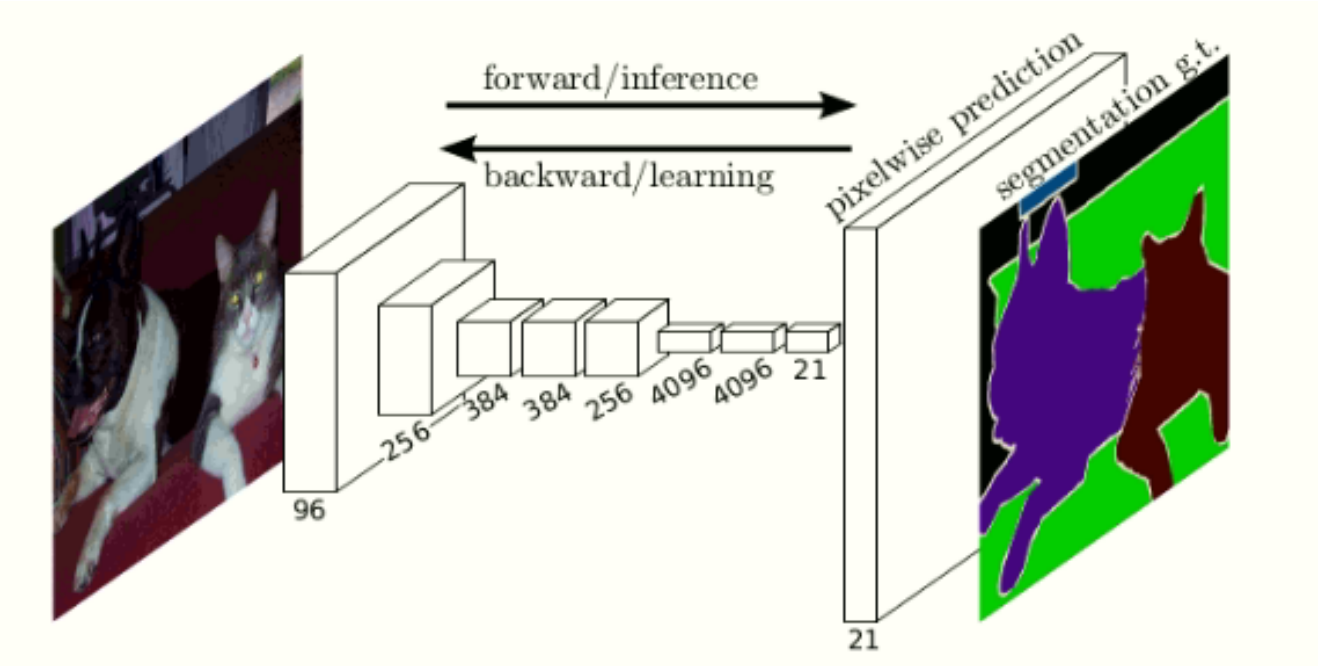
  This is the best one can hope for.

Outline

Background

- Deep Convolutional Neural Networks (DCNNs)
  - achieve outstanding performance in a series of image processing problems
  - need large scale datasets to learn effective features
- Total Variation(TV)
  - shows good performance when handling minimizing problems in image restoration
  - can preserve discontinuity

Classic architectures for Semantic Image Segmentation



Semantic Image Segmentation: UNet

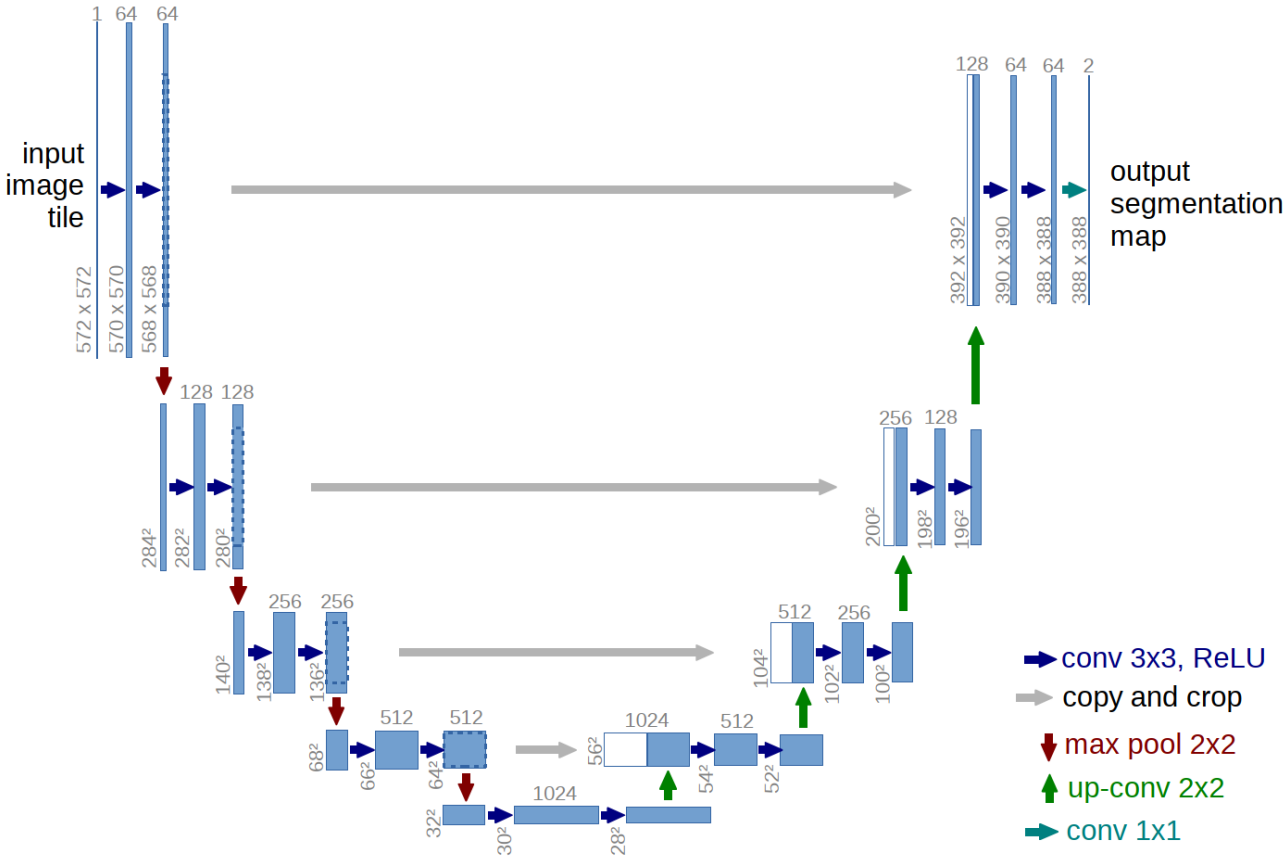
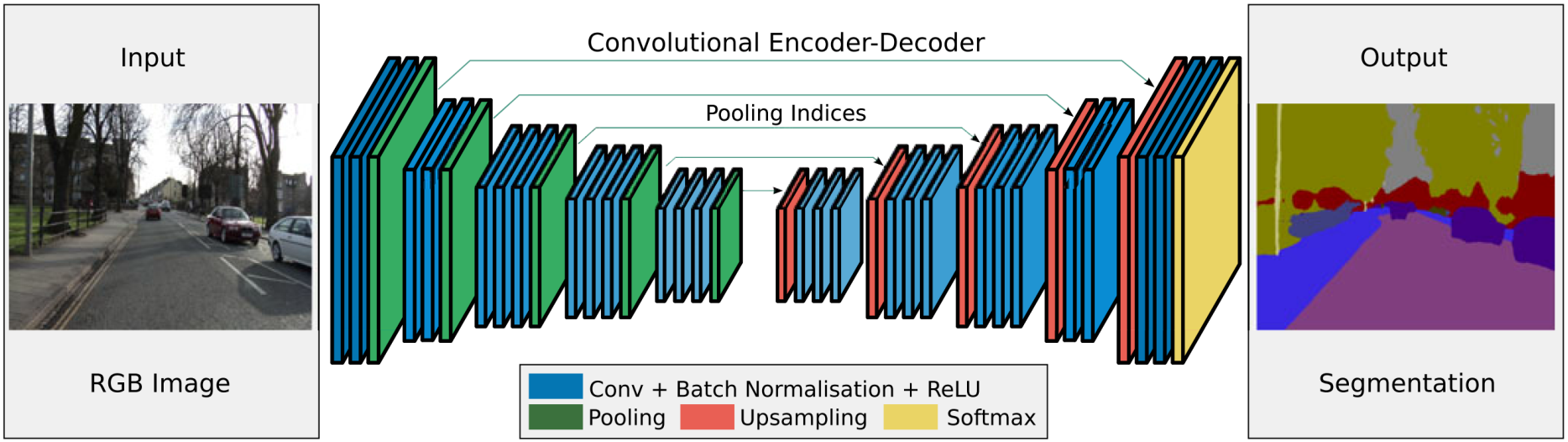


Figure 1: Architecture of Unet [Ronneberger O, Fischer P, Brox T.]. Each blue box corresponds to a multi-channel feature map. The number of channels is denoted on top of the box. The x-y-size is provided at the lower left edge of the box. White boxes represent copied feature maps. The arrows denote the different operations.

Semantic Image Segmentation: Segnet

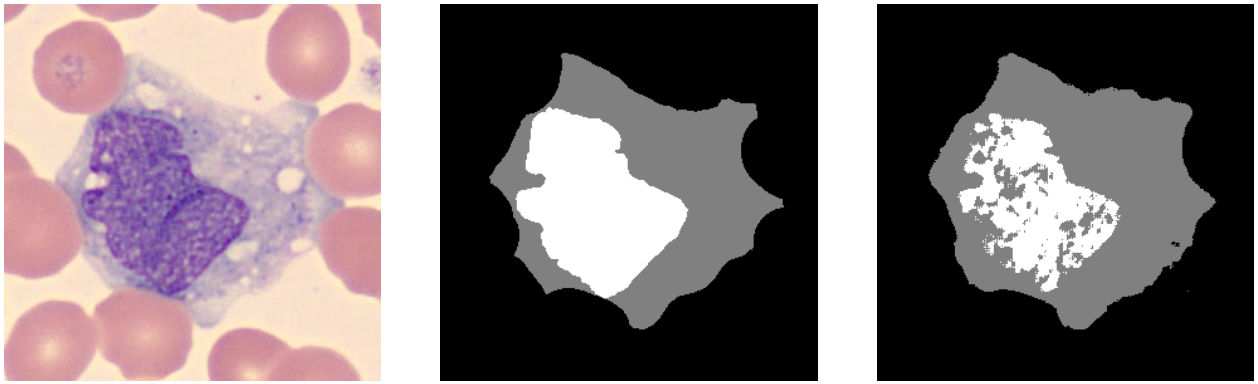


Drawbacks of Current Architectures

(a) Cell Image (b) Ground Truth (c) Unet

Figure 2: An example of segmentation results by performing the original Unet and our method on a white blood cell image. Figure 2(b) is ground truth, the gray part is cell sap, the white part is nucleus. In Figure 7(c), we can see there are coarse edges and isolated points.

Drawbacks of Current Architectures



(a) Cell Image (b) Ground Truth (c) Unet

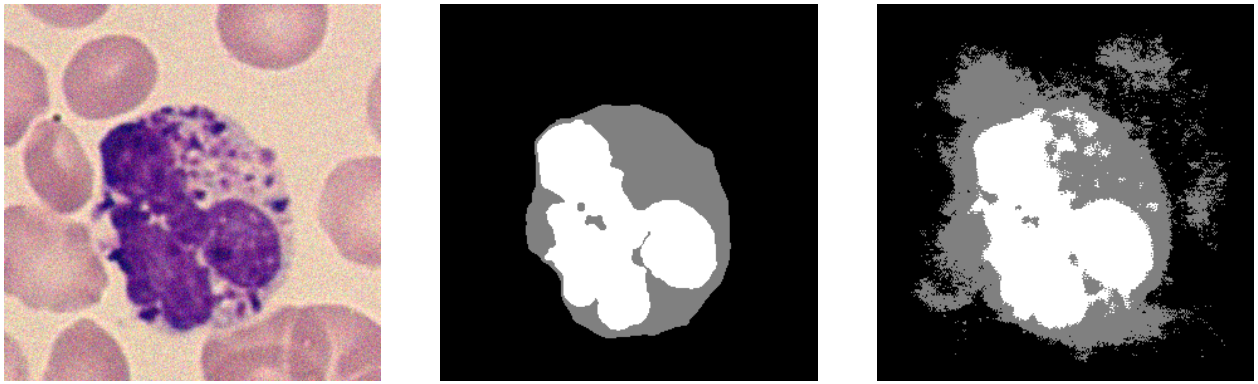
Figure 3: Tests with added noise.

Drawbacks of Current Architectures

(a) Cell Image (b) Ground Truth (c) Unet

Figure 4: Tests with added noise.

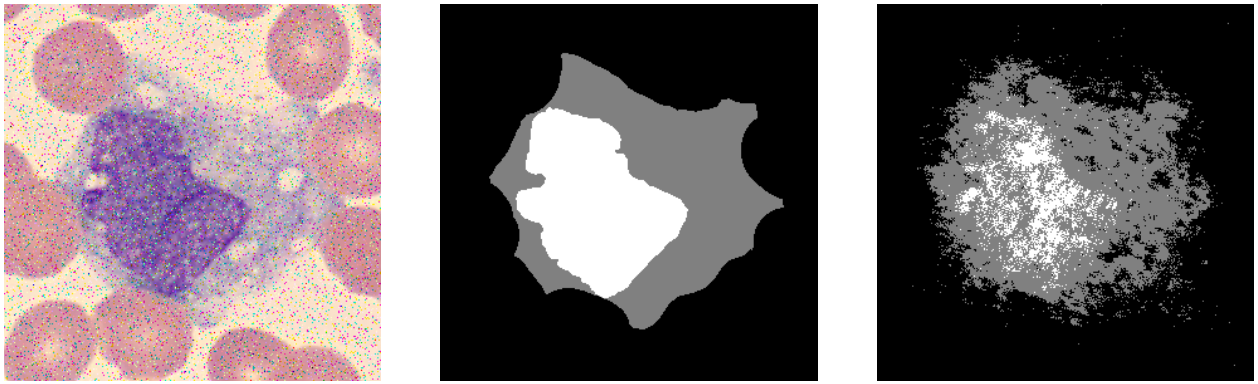
Drawbacks of Current Architectures



(a) Cell image (b) Ground Truth (c) Unet

Figure 5: Tests with added noise.

Drawbacks of Current Architectures

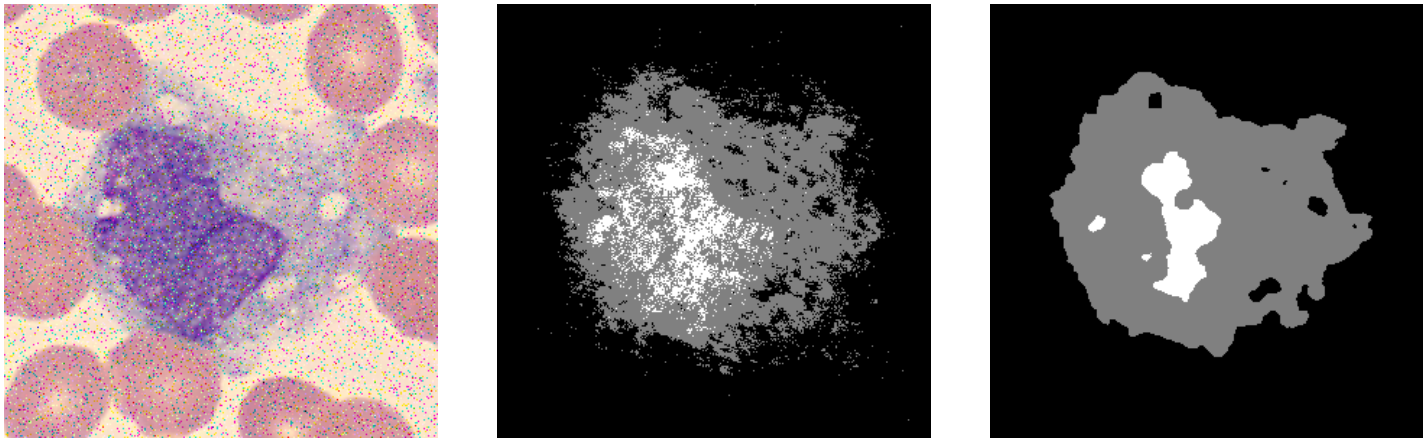


(a) Cell image (b) Ground Truth (c) Unet

Figure 6: Tests with added noise.

Drawback of DCNN segmentation

Lack of spatial regularization (smoothness):

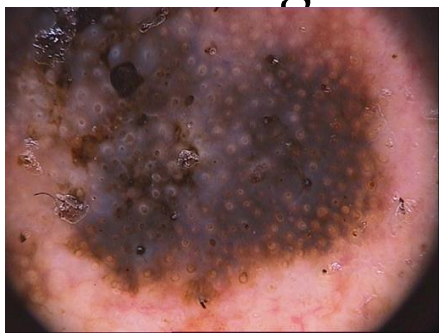


(a) Input (b) U-Net (c) U-Net+TVL[1] (Post-processing, undesirable results)

Figure 7: Tests with added noise

Drawback of DCNN segmentation

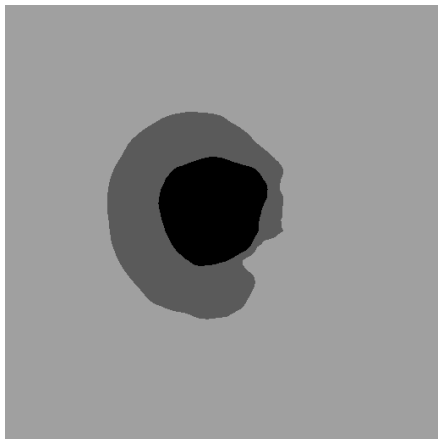
Lack of geometry shape prior:



(a) Input

(b) DeepLabV3+

(c) Input



(d) DeepLabV3+

Examples with and without convex shape priori

(e) Without convexity

(f) With convexity

Drawbacks of Current Architectures

- For small training set, they could not handle details well. What's more, the performance of deep and complex CNNs will be greatly reduced.
- They have no explicit spatial regularization.
- Coarse edges and isolated points often appear in the segmentation results provided by CNNs

How to add spatial regularization?

- **How to add spatial regularization to CNNs?**
- Approaches 1: Add smoothness regularization item to loss function.
- Approaches 2: Use CRFs and similar techniques fro post-processing.
- **Our methods add spatial regularization directly to the CNN layers and give no extra difficulty to the computations. It gives very effective spatial regularization**

How to add spatial regularization?

- Approaches 1: Add smoothness regularization item to loss function.
  For example, Amy Zhao et al.[Zhao A, Balakrishnan G, Durand F, Guttag JV, Dalca AV.] define a smoothness regularization function based on the atlas segmentation map:

$$\mathcal{L}_{smooth}(c, \psi) = \int_{\Omega} (1 - c(x)) |\nabla \psi(x)| dx \tag{1}$$

  where ψ is the network output, c is a binary image of anatomical boundaries computed from the atlas segmentation labels l_x, and ∇ is the spatial gradient operator. Intuitively, this term discourages dramatic intensity changes within the same anatomical region.

How to add spatial regularization?

In the total appearance transform model loss $\mathcal{L}_a$, they use mean squared error for the image similarity loss $\mathcal{L}_{sim} = ||\tilde{y} - y||$. They balance the similarity loss with the regularization term $\mathcal{L}_{smooth}$:

$$\mathcal{L}_a = \mathcal{L}_{sim} + \lambda_\alpha \mathcal{L}_{smooth} \tag{2}$$

where λ_α is a hyperparameter.

How to add spatial regularization?

- Approaches 2: Use CRFs(conditional random fields) as post-processing techniques.
  PSPNet, Deeplab and many other networks employ fully connected CRF model[P. Krähenbühl and V. Koltun.] to refine their segmentation results. The model employs the energy function:

$$E(x) = \sum_i \theta_i(x_i) + \sum_{ij} \theta_{ij}(x_i, x_j) \tag{3}$$

  where x is the label assignment for pixels. They use as unary potential $\theta_i(x_i) = -\log P(x_i)$, where $P(x_i)$ is the label assignment probability at pixel i as computed by a DCNN.

How to add spatial regularization?

The pairwise potential has a form that allows for efficient inference while using a fully-connected graph, i.e. when connecting all pairs of image pixels, p_i, p_j. In particular, they use the following expression:

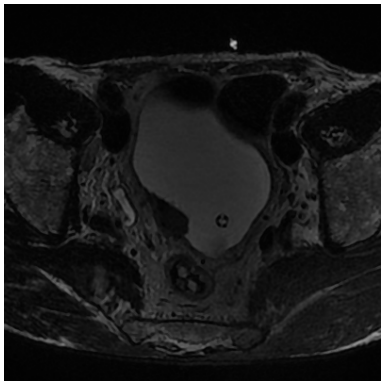
$$\begin{aligned}\theta_{ij}(x_i, x_j) = \quad & \mu(x_i, x_j)[w_1 \exp(-\frac{||p_i - p_j||^2}{2\sigma_\alpha^2} - \frac{||I_i - I_j||^2}{2\sigma_\beta^2}) \\ & + w_2 \exp(-\frac{||p_i - p_j||^2}{2\sigma_\gamma^2})]\end{aligned} \tag{4}$$

where $\mu(x_i, x_j) = 1$ if $x_i \neq x_j$, and zero otherwise, which, as in the Potts model, means that only nodes with distinct labels are penalized.

How to add spatial regularization?

The remaining expression uses two Gaussian kernels in different feature spaces; the first, bilateral kernel depends on both pixel positions (denoted as p) and RGB color (denoted as I), and the second kernel only depends on pixel positions. The hyper parameters $\sigma_\alpha, \sigma_\beta, \sigma_\gamma$ control the scale of Gaussian kernels. The first kernel forces pixels with similar color and position to have similar labels, while the second kernel only considers spatial proximity when enforcing smoothness.

Drawbacks of post processing techniques

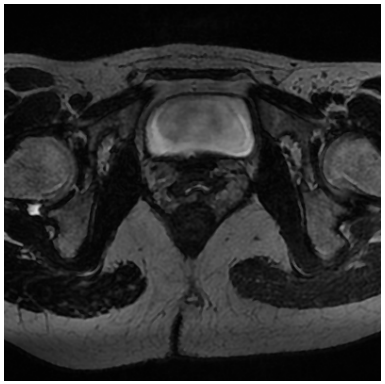


(g) Input image

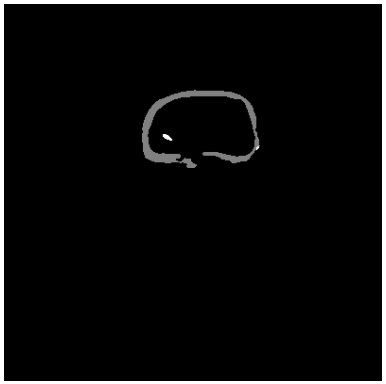
(h) UNet

(i) Unet+CRF

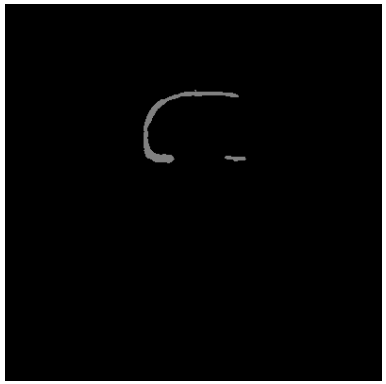
Drawbacks of post processing techniques



(j) Input image



(k) UNet



(l) Unet+CRF

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

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

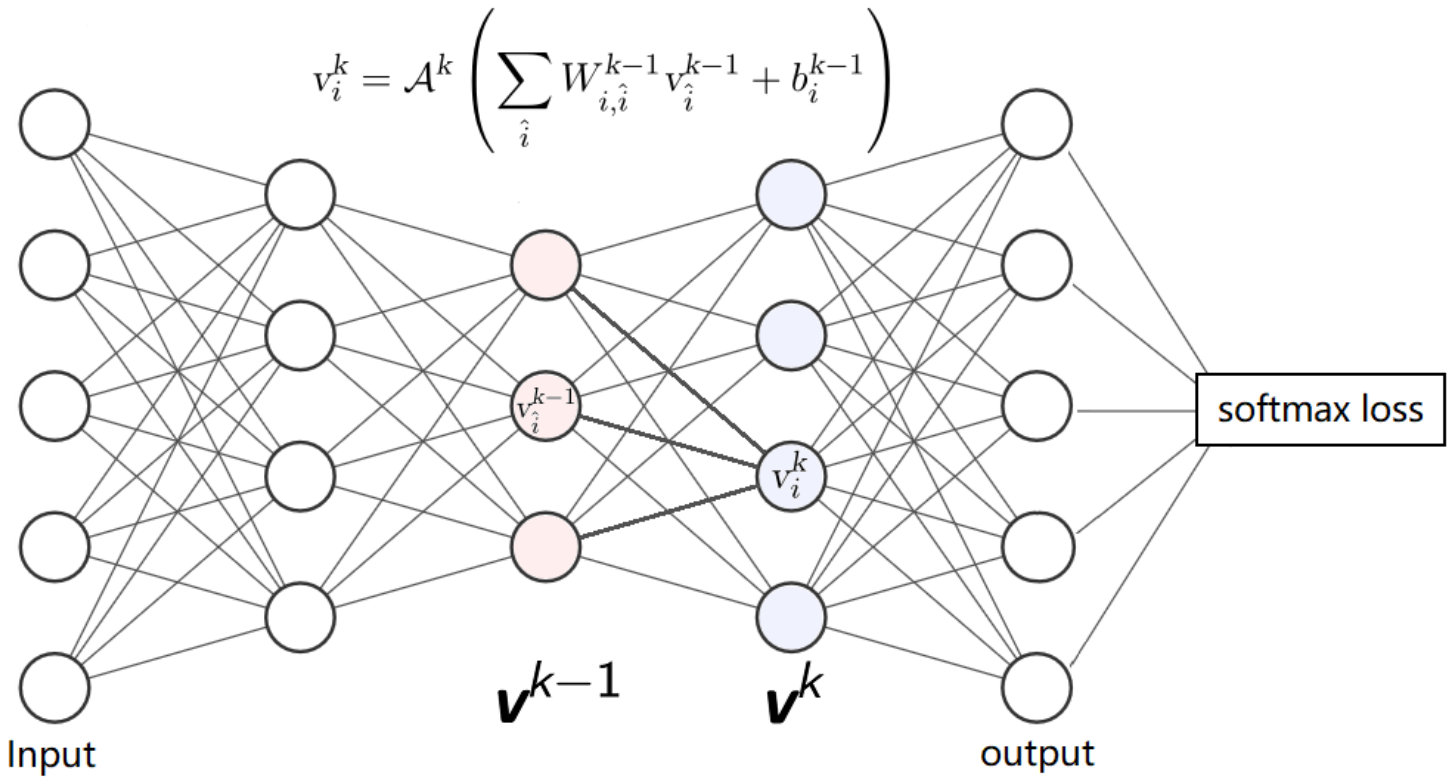
General Neural Network for Semantic Image Segmentation

- 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

General Neural Network for Semantic Image Segmentation

- Denote $\boldsymbol{o}^k = \mathcal{T}_{\boldsymbol{\Theta}^{k-1}}(\boldsymbol{v}^{k-1})$, then (5) becomes

$$\begin{cases} \boldsymbol{v}^k = & \mathcal{A}^k(\boldsymbol{o}^k), \\ \boldsymbol{o}^k = & \mathcal{T}_{\boldsymbol{\Theta}^{k-1}}(\boldsymbol{v}^{k-1}), \end{cases} \quad where\ k = 1, \cdots, K. \tag{6}$$

- Let us write $\Delta^k = \frac{\delta \mathcal{L}}{\delta o^k}$, then

$$\begin{aligned} \Delta^k = & \frac{\delta v^k}{\delta o^k}\frac{\delta o^{k+1}}{\delta v^k}\frac{\delta \mathcal{L}}{\delta o^{k+1}} = diag((\mathcal{A}^k)'(o^k))(\mathcal{W}^k)^T\Delta^{k+1}, \\ \frac{\delta \mathcal{L}}{\delta \boldsymbol{\Theta}^k} = & \frac{\delta o^{k+1}}{\delta \boldsymbol{\Theta}^k}\frac{\delta \mathcal{L}}{\delta o^{k+1}} = \frac{\delta o^{k+1}}{\delta \boldsymbol{\Theta}^k}\Delta^{k+1}, \\ k = & \ 0, 1, \cdots, K-1. \end{aligned} \tag{7}$$

Total Variation

- The total variation (TV) is proposed to produce piece-wise constants cartoon restorations in ROF model[Rudin, L.I., Osher, S., Fatemi, E, 1992]. TV can be wrote as:

$$\mathrm{TV}(u) = \int_\Omega |\nabla u(x)| dx,$$

where Ω is a bounded subset of $\mathbb{R}^2$.It has a dual formulation as

$$\mathrm{TV}(u) = \sup_{\boldsymbol{\xi} \in \mathbb{B}} \left\{ \int_\Omega u(x) div \boldsymbol{\xi}(x) dx \right\},$$

where $\mathbb{B} = \{\xi \in C_c^1(\Omega; \mathbb{R}^2) \mid ||\boldsymbol{\xi}||_\infty = \max_{x \in \Omega} \{||\boldsymbol{\xi}(x)||_2\} \leq 1\}$.

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 TV (total variation) 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:

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

- If the dual variable $\boldsymbol{\eta}$ is known, then we have:

$$v_{ji} = \quad \tilde{\mathcal{A}}(o)_{ji} = \frac{\exp(o_{ji} - \lambda(div\boldsymbol{\eta}_j)_i)}{\sum_{c=1}^{C} \exp(o_{ci} - \lambda(div\boldsymbol{\eta}_c)_i)}. \tag{13}$$

where $j = 1, \ldots, C, i = 1, \ldots, N_1 N_2$.

Proposed Regularization Neural Network

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

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

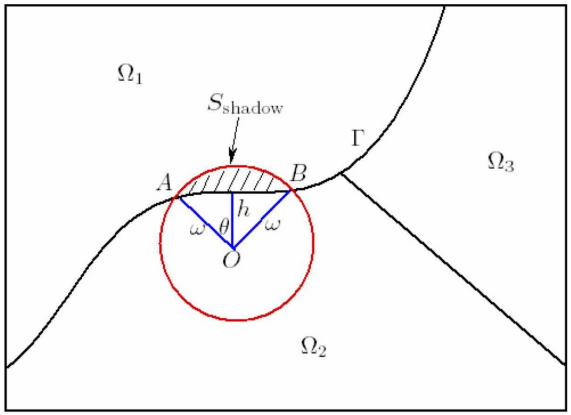
  in an iterative way:

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

  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

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

- $\mathcal{C}$—-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)

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

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

Output: Soft segmentation function u.

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

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

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

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

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

end

return Segmentation function u.

Proposition

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

Proposed STD DCNN segmentation

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

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

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

[1] Jun Liu, Xiangyue Wang, and Xue-Cheng Tai. "Deep convolutional neural networks with spatial regularization, volume and star-shape priori for image segmentation". arXiv:2002.03989. 2020.

Results on PASCAL VOC 2012 dataSet.

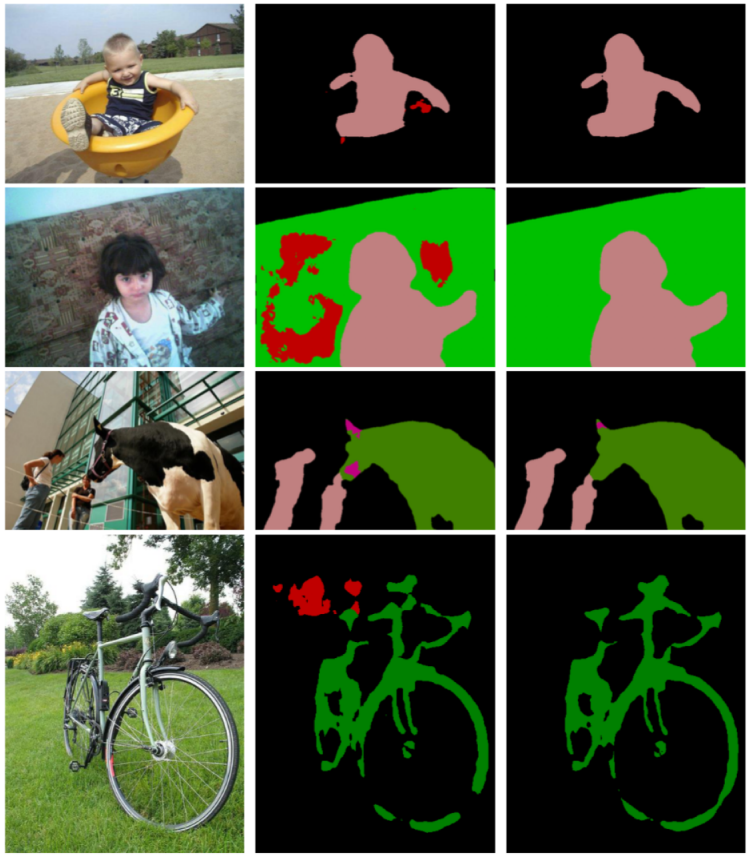


Figure 8: Visual effects of the DeepLabV3+ and proposed STD-DeeplabV3+ on PASCAL VOC 2012 test set.

White Blood Cell Dataset2

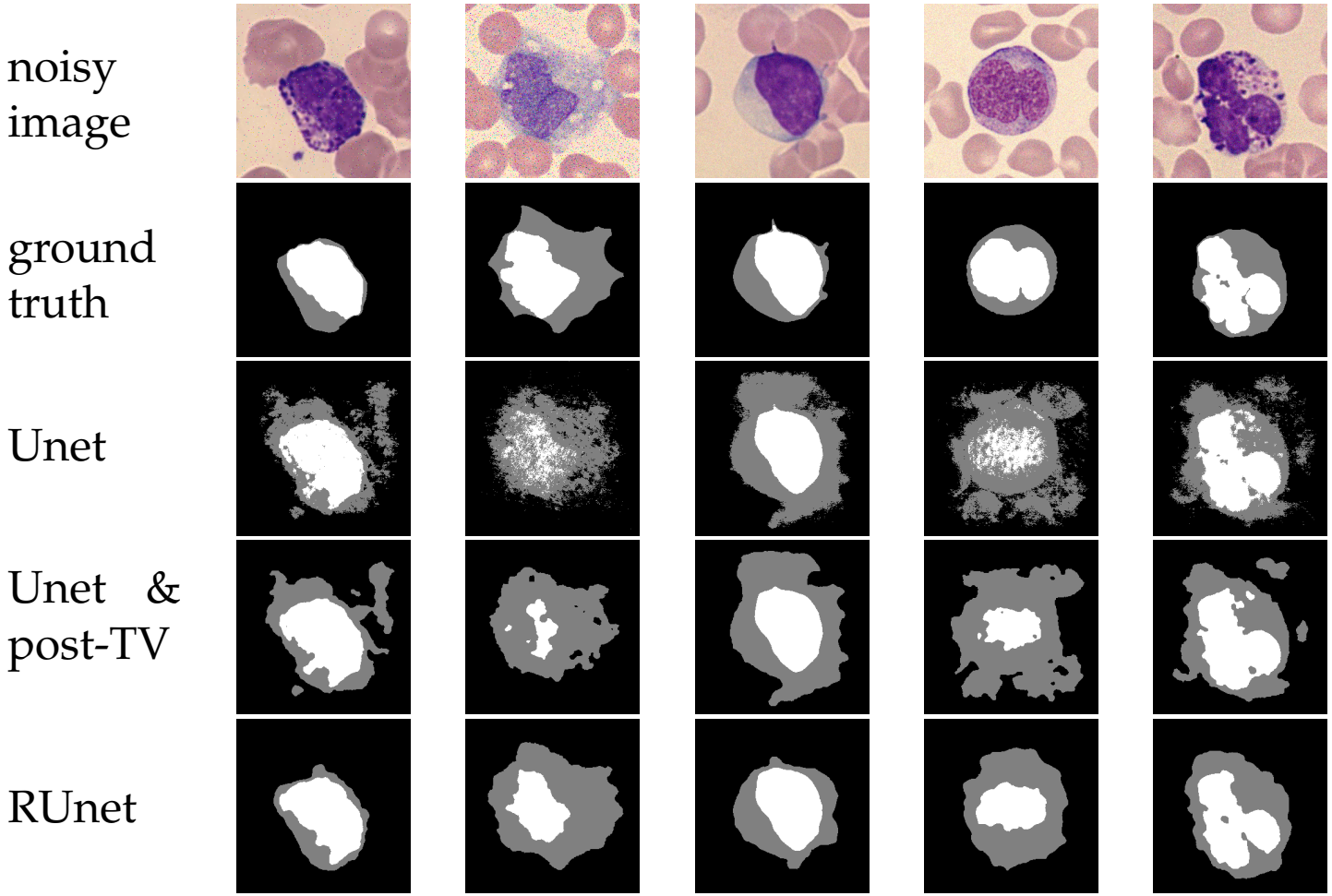


Figure 9: Segmentation results predicted by Unet and RUnet trained on noisy dataset. Noise type from left to right: small level salt and pepper(s&p) noise, large level s&p noise, small level gaussian noise, the following two are medium level gaussian noise.

White Blood Cell Dataset2

	Method	clean	gaussain		pepper	salt
noise level			0.05	0.09	0.01	0.01
mIoU	Unet1	89.79	73.94	67.67	80.68	77.75
	RUnet1	**90.15**	**79.25**	**74.20**	**85.46**	**84.07**
Accuracy	Unet1	97.04	86.43	80.33	92.55	90.00
	RUnet1	**97.13**	**90.54**	**86.55**	**94.87**	**96.26**
RE	Unet1	1.82	7.80	13.60	5.99	4.39
	RUnet1	**1.30**	**1.48**	**1.56**	**1.35**	**1.32**

Table 1: Results of Unet, trained on clean dataset and test on noise data.

As we cannot always obtain very clean images in practice, we want to know how the segmentation result will change when encountering small perturbation. In table 1, Unet, RUnet are both trained on clean data but tested on data with different noise.

	Method	clean	gaussain		pepper	salt
noise level			0.05	0.09	0.01	0.01
mIoU	Unet2	90.57	89.49	85.87	88.20	88.84
	RUnet2	**91.86**	**90.24**	**88.37**	**89.85**	**91.22**
Accuracy	Unet2	97.25	96.89	95.77	96.54	96.77
	RUnet2	**97.64**	**97.00**	**96.38**	**96.96**	**97.37**
RE	Unet2	1.79	2.02	2.87	2.38	2.19
	RUnet2	**1.32**	**1.35**	**1.34**	**1.32**	**1.34**

Table 2: Results of Unet2 and RUnet2 trained on noisy dataset.

In table 2, Unet, RUnet are both trained on noise data but tested on data with different noise.

White Blood Cell Dataset2

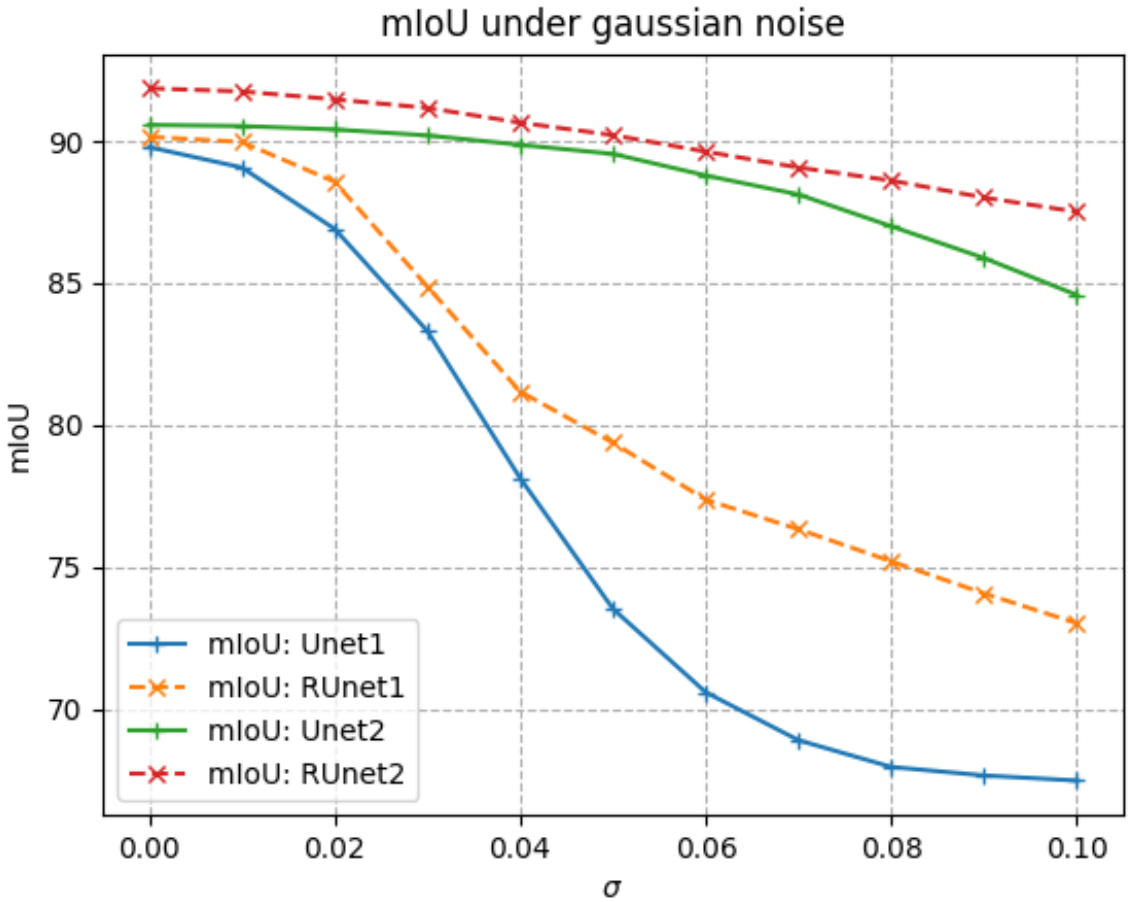


Figure 10: Unet1 and RUnet1 are trained on clean WBC dataset, Unet2 and RUnet2 are trained on noisy WBC dataset. We add gaussian noise with zero mean, standard deviation σ from 0.01 to 0.1 to WBC testing dataset.

Volume Preservation (VP)

VP-STD (Volume preserving Soft Threshold Dynamics) softmax

- Volume preserving (VP) prior by modifying $\mathcal{C}$ as

$$\mathcal{C}_{\boldsymbol{V}} = \{\boldsymbol{u} \in \mathcal{C} : \int_{\Omega} u_i(x)dx = V_i\}.$$

- $\boldsymbol{V} = (V_1, V_2, \cdots, V_I)$ is a given volume.
- VP-STD model

$$\widetilde{\boldsymbol{u}} = \arg\min_{\boldsymbol{u}\in\mathcal{C}_{\boldsymbol{V}}}\{\mathcal{F}(\boldsymbol{u};\boldsymbol{o}) + \mathcal{R}(\boldsymbol{u})\}.$$

- Related iteration

$$\boldsymbol{u}^{t_1+1} = \arg\min_{\boldsymbol{u}\in\mathcal{C}_{\boldsymbol{V}}}\{\mathcal{F}(\boldsymbol{u};\boldsymbol{o}) + \underbrace{\langle \boldsymbol{u}, \boldsymbol{p}^{t_1}\rangle}_{:=\hat{\mathcal{R}}(\boldsymbol{u})}\}.$$

Dual VP-STD softmax segmentation algorithm

Algorithm 2: VP-STD softmax

Input: The feature $\boldsymbol{o}$ and volume $\boldsymbol{V}$.

Output: Volume preserving soft segmentation function $\boldsymbol{u}$.

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

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

1. calculate the volume preserving dual variable by (10), i.e.

$$\boldsymbol{q}^{t_1+1} = \boldsymbol{q}^{t_1} + \varepsilon \left(\log \boldsymbol{V} - \log \langle \mathcal{S}(\frac{\boldsymbol{o} - \boldsymbol{p}^{t_1} + \boldsymbol{q}^{t_1}}{\varepsilon}), \mathbf{1} \rangle \right).$$

2. compute the solution of (8) by VP-STD softmax

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

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

end

return Segmentation function $\boldsymbol{u}$.

Network architecture

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

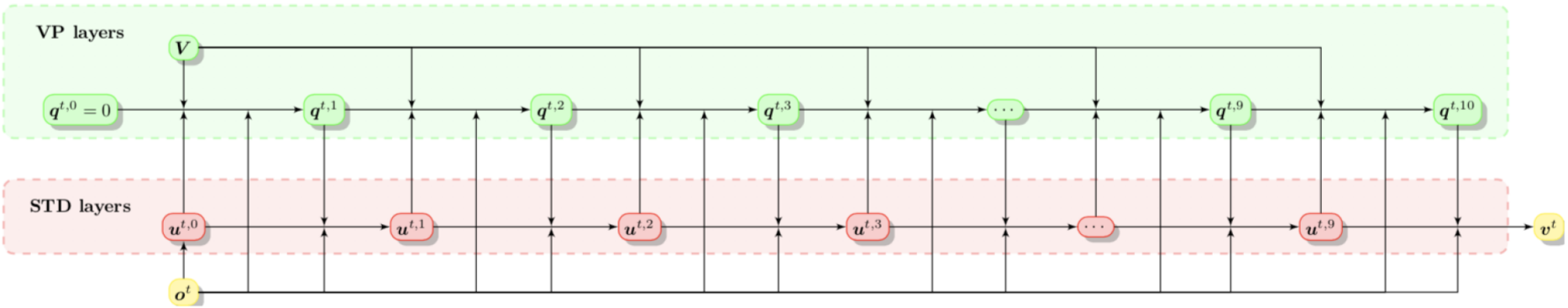
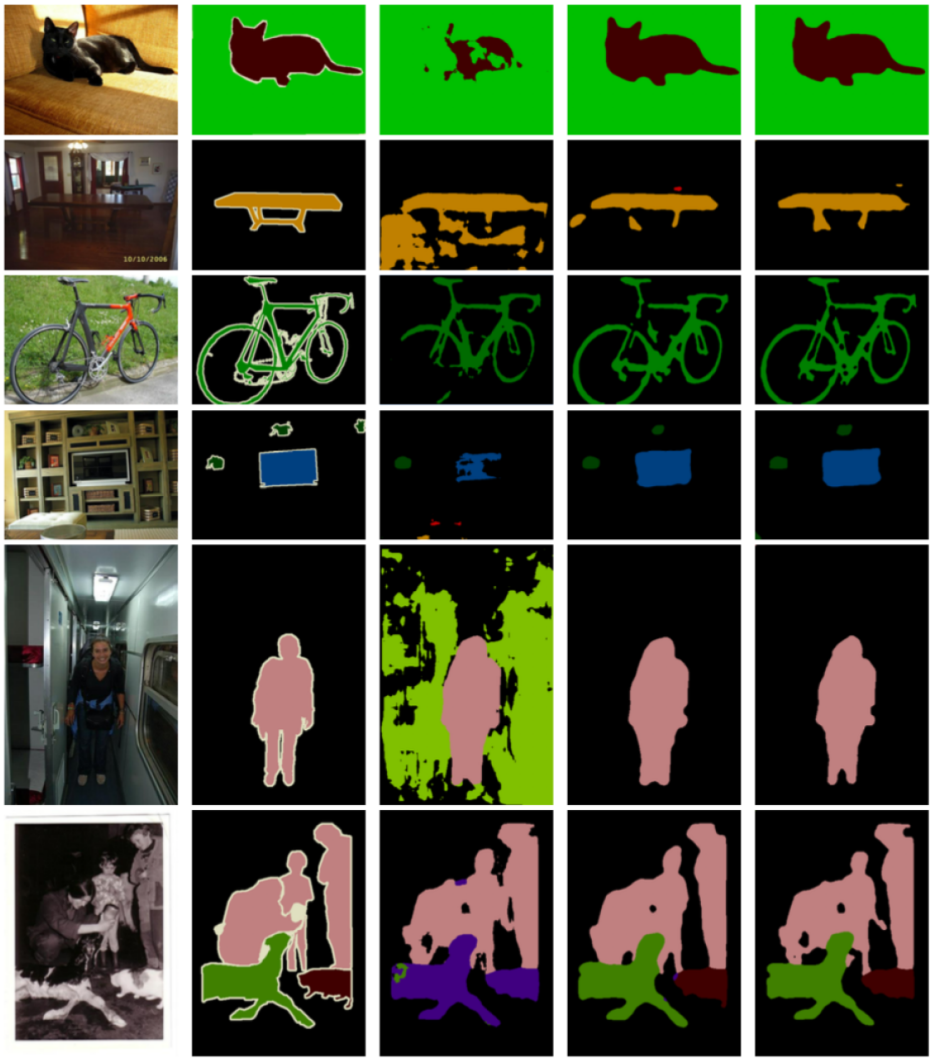


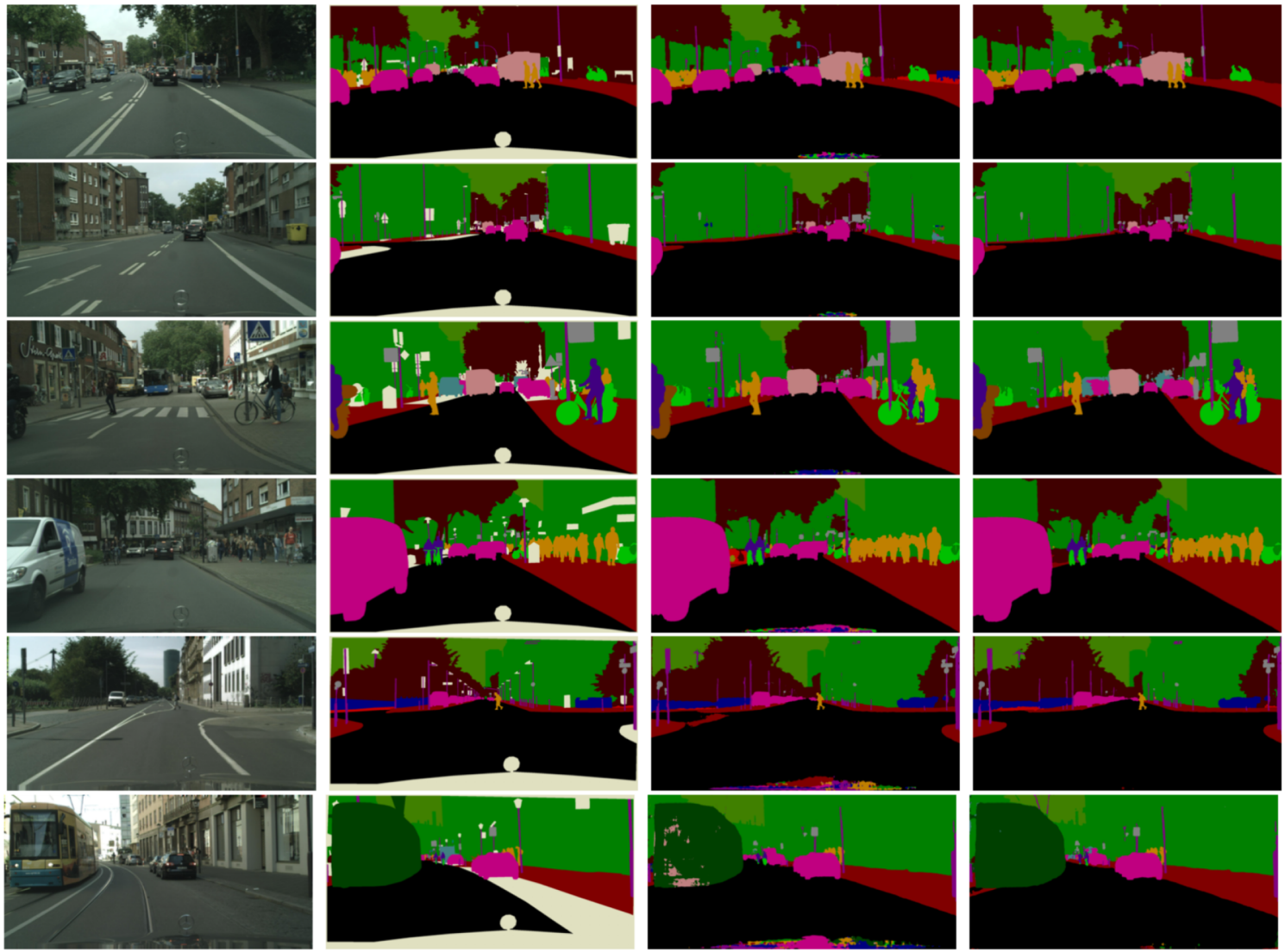
Figure 11: VP-STD blocks for DCNN. This architecture is constructed according to the VP-STD algorithm. STD block is a special case of this architecture by dropping all the VP layers.

Results on PASCAL VOC 2012 dataSet.



(a) Images (b) Ground Truth (c) DeepLabV3+ (d) STD (e) VP-STD

Numerical Results on CITYSCAPE validation set.



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

Results on CITYSCAPE validation set

	road	s.walk	build.	wall	fence	pole	t-light	t-sign	veg	terrain	sky	person	rider	car	truck	bus	train	motor	bike	mIoU
DeepLabV3+ [38]	98.14	84.71	92.69	57.26	62.19	**65.11**	68.41	**78.78**	92.66	63.39	95.30	82.14	62.77	**95.31**	85.32	89.07	80.91	64.51	77.26	78.73
Ours(VP-STD)	**98.43**	**87.68**	**93.35**	**71.40**	**73.58**	65.08	**68.70**	78.38	**92.79**	**72.59**	**95.56**	**82.29**	**67.43**	95.20	**89.89**	**91.89**	**86.56**	**70.73**	**77.88**	**82.07**

Figure 12: Performance on CITYSCAPE validation set.

Star-shape prior (SS)

Star-shape prior (SS)

Application 3: SS (Star shape)-STD softmax

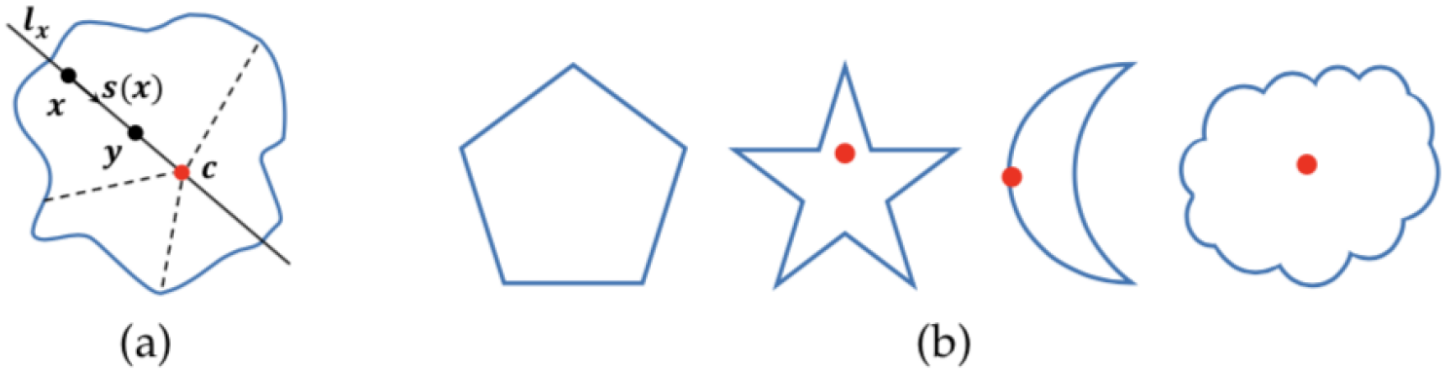


Figure 13: Star shape objects with given centers

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).

Star shape prior

- If $u : \Omega \to \{0, 1\}$, discrete energy (Veksler[2],2008) for star shape

$$\mathcal{P}_0(u(x), u(y)) = \begin{cases} 0, & if\ u(y) = u(x), \\ +\infty, & if\ u(y) = 0\ and\ u(x) = 1, \\ \beta \geqslant 0, & if\ u(y) = 1\ and\ u(x) = 0. \end{cases}$$

- After continuation and simplification ($\beta = 0$), the star-shape continuous energy:

$$\mathcal{P}(u) = \begin{cases} 0, & \langle \nabla u(x), \boldsymbol{s}(x) \rangle \geq 0, \\ +\infty, & \langle \nabla u(x), \boldsymbol{s}(x) \rangle < 0. \end{cases}$$

- We get the geometry constraint

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

- $\mathcal{P}$ is the indicative function of convex set $\mathcal{P}$.

[2] Olga Veksler. "Star shape prior for graph-cut image segmentation". In: *Computer Vision -10th European Conference on Computer Vision, Marseille, France, Proceedings, Part III.*. Springer. 2008, pp. 454–467.

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:

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

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

- Dual problem in terms of KKT condition

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

- 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+

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

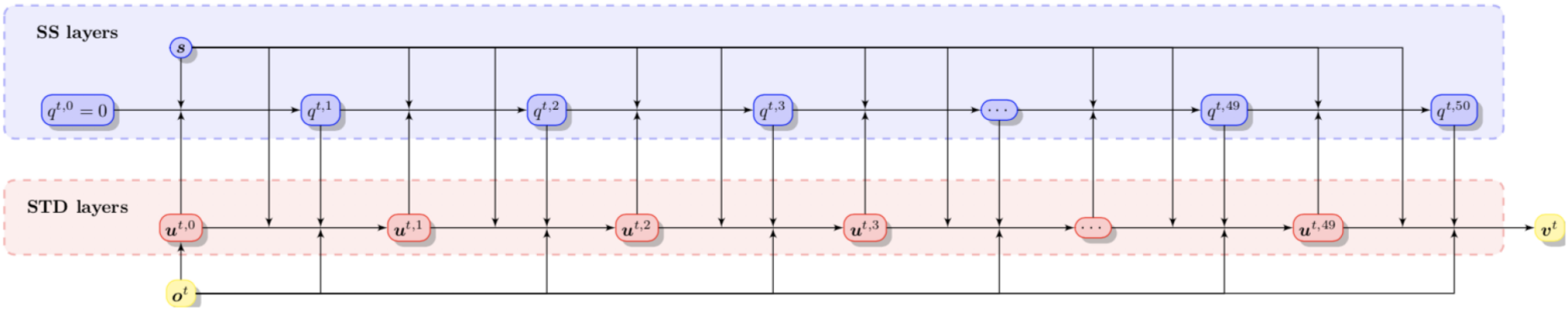


Figure 14: SS-STD block for DCNN. This architecture is constructed according to the SS-STD algorithm.

Comparison

Testing of Algorithm 3 as a segmentation tool:

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

Figure 15: Segmentation results by softmax, proposed STD and SS-STD softmax.

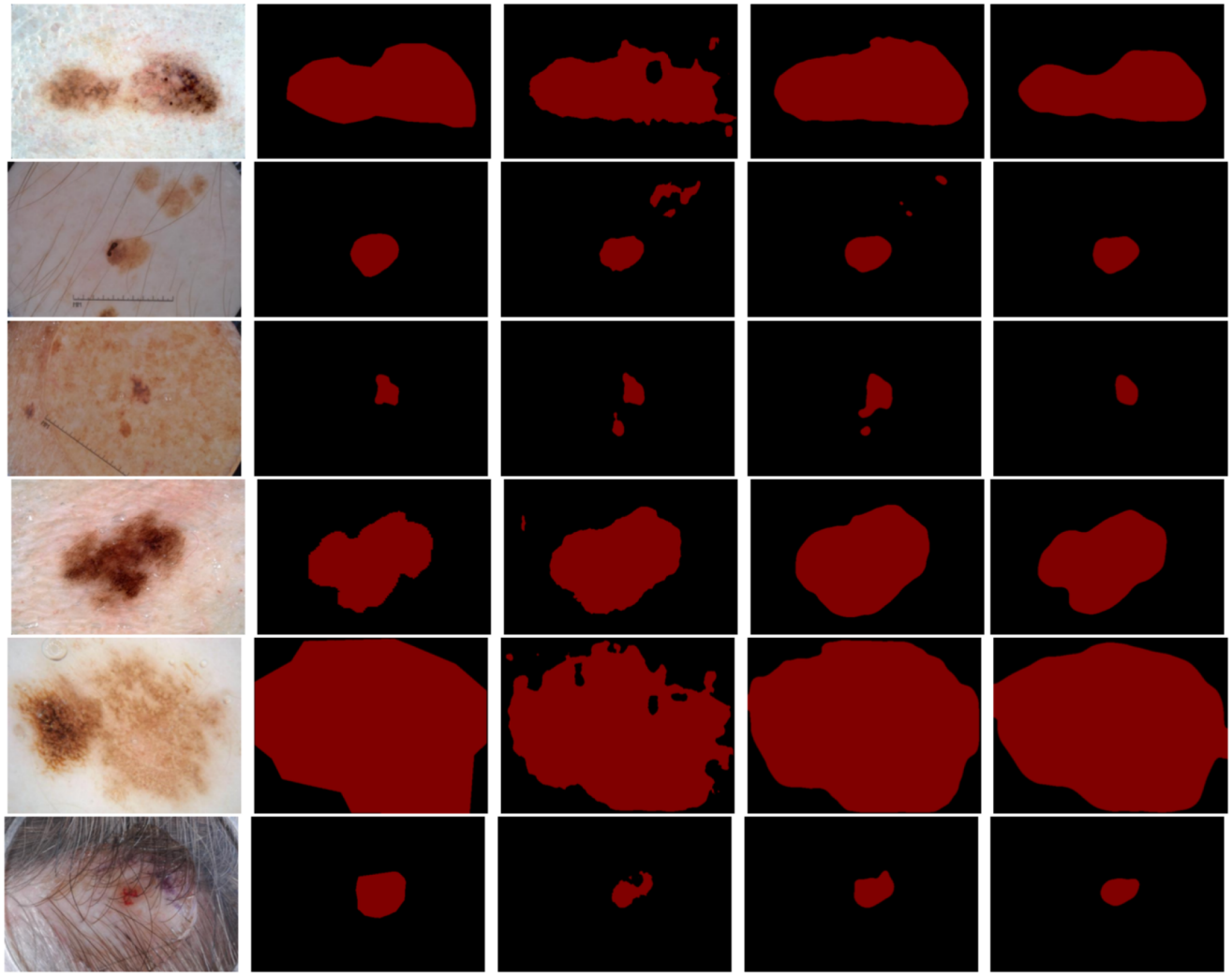
Comparison

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

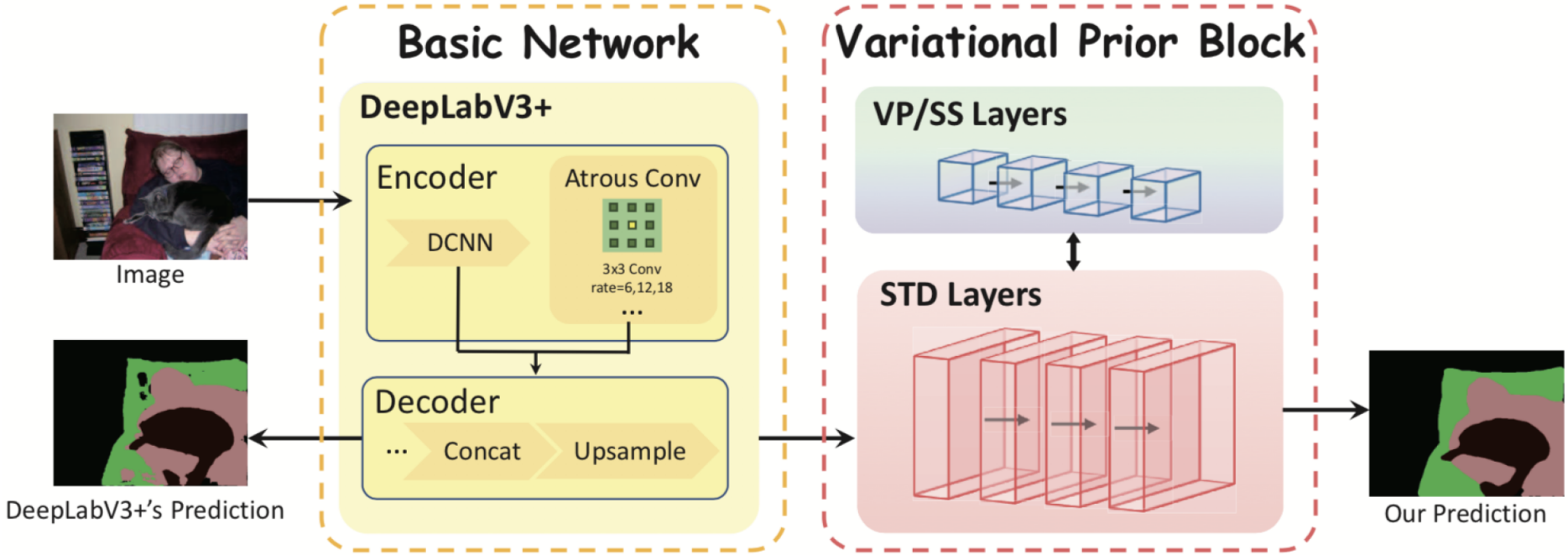
Figure 16: An example of without and with the proposed spatial priori for DeepLabV3+ on ISIC2018 validation set.

Results on ISIC2018 validation set.



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

The network architecture of STD based DeepLabV3+



Performance on ISIC2018 validation set

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

Application 4: CS-STD sigmoid

- Similar idea, we can get the new variational interpretation for other common activation functions.
- Sigmoid $\mathcal{S}(o)(x) = \frac{1}{1+e^{-o(x)}}$:

$$\mathcal{S} = \arg\min_{u\in[0,1]} \{- < o, u > +\varepsilon(u \ln u + (1-u)\ln(1-u))\}.$$

- ReLU $\max\{o, 0\}$:

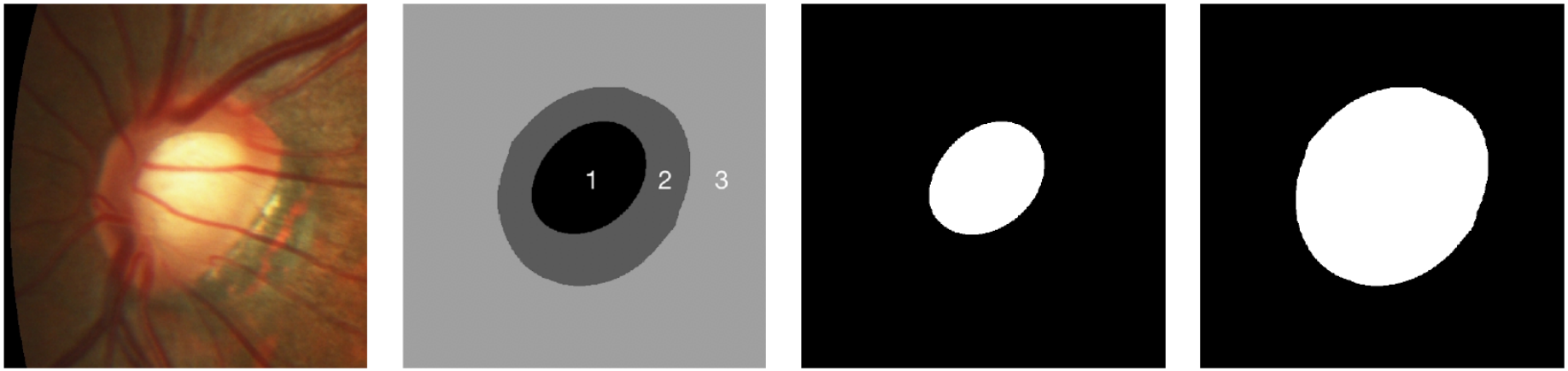
$$\text{ReLU} = \arg\min_{u\geqslant 0} \left\{ ||u-o||^2 \right\}.$$

- We will show how to use binary sigmoid to segment multi convex objects.

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

Multi-label segmentation

$$\min_{l} \left\{ \int_{\Omega} o(v(x), l(x)) dx + \lambda \int_{\Omega} |\nabla l(x)| dx \right\},$$

- $l : \Omega \to \{1, \cdots, L\}$: label function
- $l(x)$: the pixel located at x belongs to $l(x)$-th class
- $o(v(x), l(x))$ is a feature of $l(x)$-th class for a given image $v(x)$

Sublevel set functions

- γ-sublevel set functions u

$$u(x,\gamma) = \begin{cases} 1, & l(x) \leqslant \gamma, \\ 0, & l(x) > \gamma, \end{cases}$$

- $l(x) = l_{max} - \int_{l_{min}}^{l_{max}} u(x,\gamma)\mathrm{d}\gamma.$
- Convex energy (Pock[3] et.al.)

$$\min_{u,\partial_\gamma u \geqslant 0} \left\{ \int_{\mathbb{R}} \int_{\Omega} o(v(x),\gamma)\partial_\gamma u(x,\gamma) dx \mathrm{d}\gamma + \lambda \int_{\mathbb{R}} \int_{\Omega} |\nabla_x u(x,\gamma)| dx \mathrm{d}\gamma \right\}.$$

[3] T. Pock et al. "A convex formulation of continuous multi-label problems". In: *In European Conference on Computer Vision 2008*. 2008, pp. 792–805.

Convexity

- $l(x) \in \{1, 2, \cdots, L\}$,
  $u_\gamma = u(x, \gamma), o_\gamma = o(v(x), \gamma)$,
  $u_L = 1$,
  Dual representation

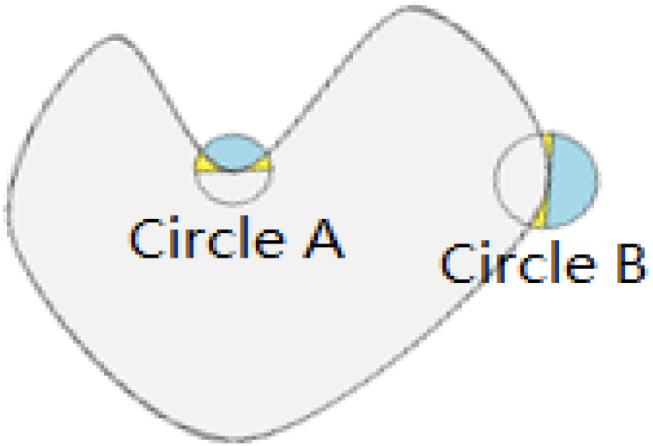
$$\min_{\boldsymbol{u} \in \mathcal{C}_{cs}} \left\{ \underbrace{\sum_{\gamma=1}^{L-1} \int_\Omega (o_\gamma - o_{\gamma+1})\, u_\gamma dx}_{\mathcal{F}} + \underbrace{\lambda \sum_{\gamma=1}^{L-1} \int_\Omega |\nabla u_\gamma| dx}_{\mathcal{R}} \right\}.$$

- $\mathcal{C}_{cs}$ is a relaxed sublevel set function set

$$\mathcal{C}_{cs} = \{\boldsymbol{u} = (u_1, \cdots, u_{L-1}) : \ 0 \leqslant u_1(x) \leqslant \cdots \leqslant u_{L-1}(x) \leqslant 1\}.$$

- $l(x) = L - \sum_{\gamma=1}^{L-1} u_\gamma(x)$.

Conditions for convex shapes with binary representation



- Convex shape condition(Luo,Tai,Wang,2019)

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

- 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.

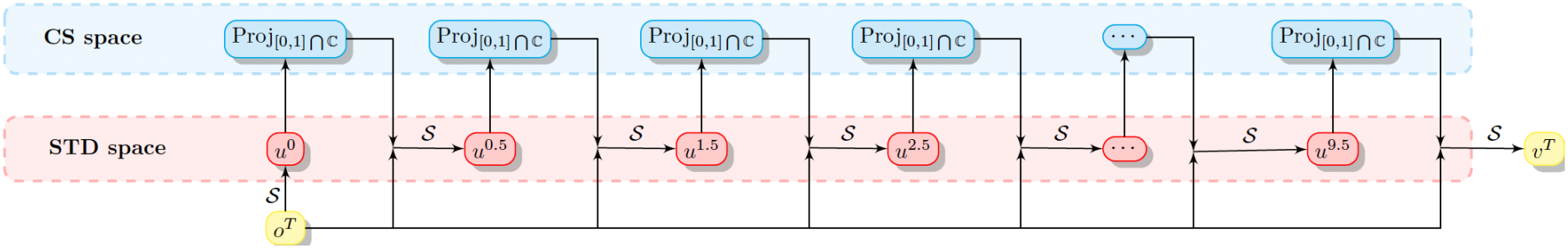
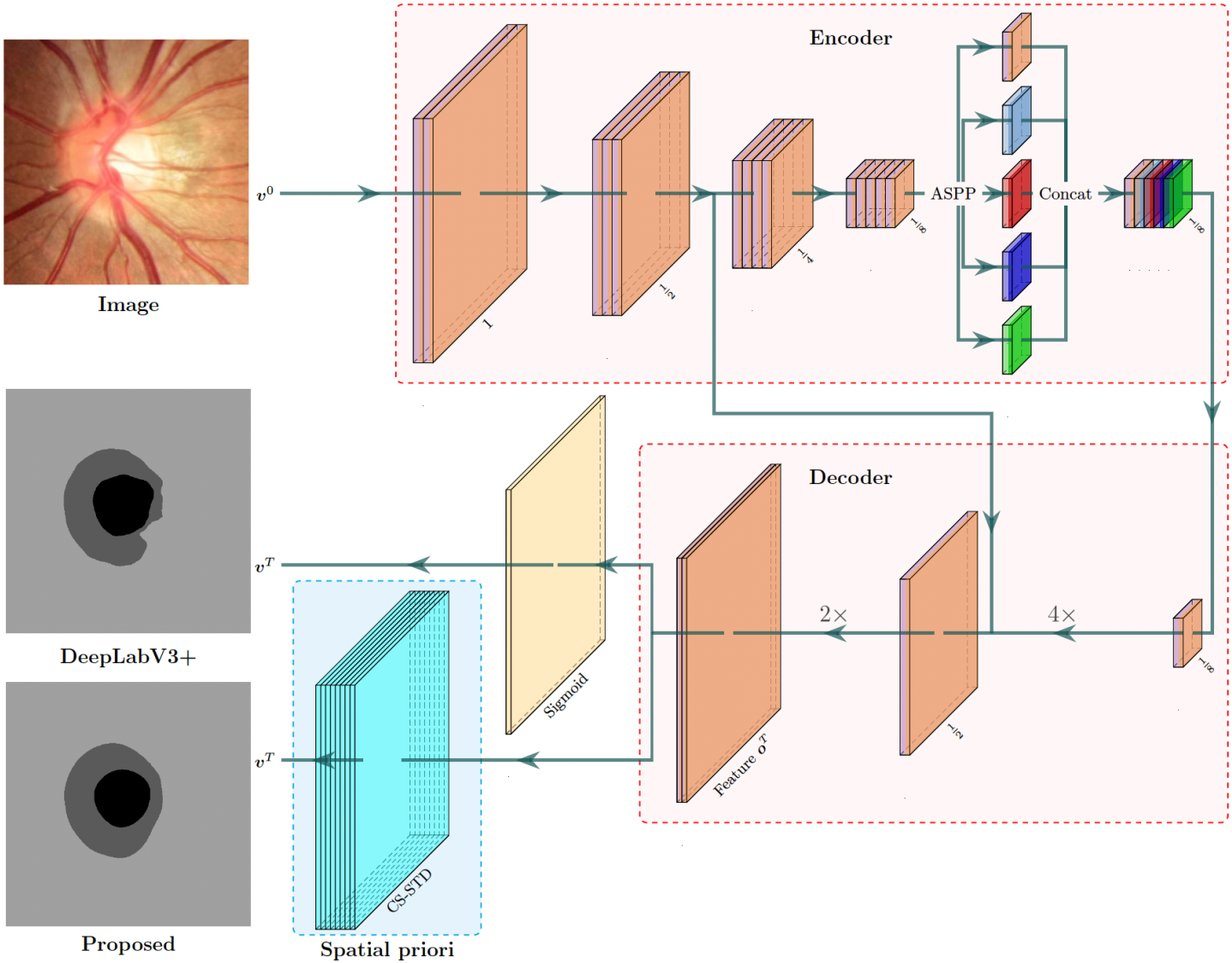


Figure 18: CS-STD block for DCNN. This architecture is constructed according to the CS-STD algorithm.

The architecture of the CS-STD based DeeplabV3+ segmentation network



Visual quality

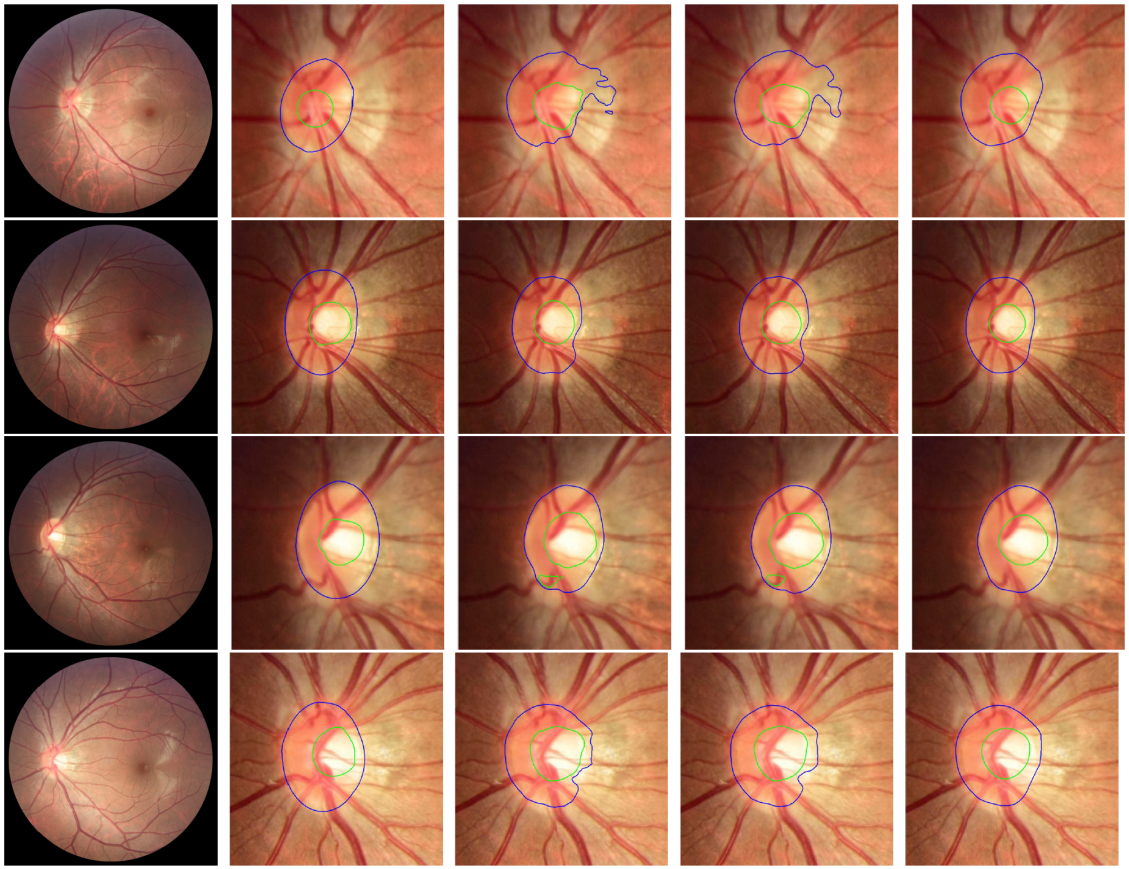
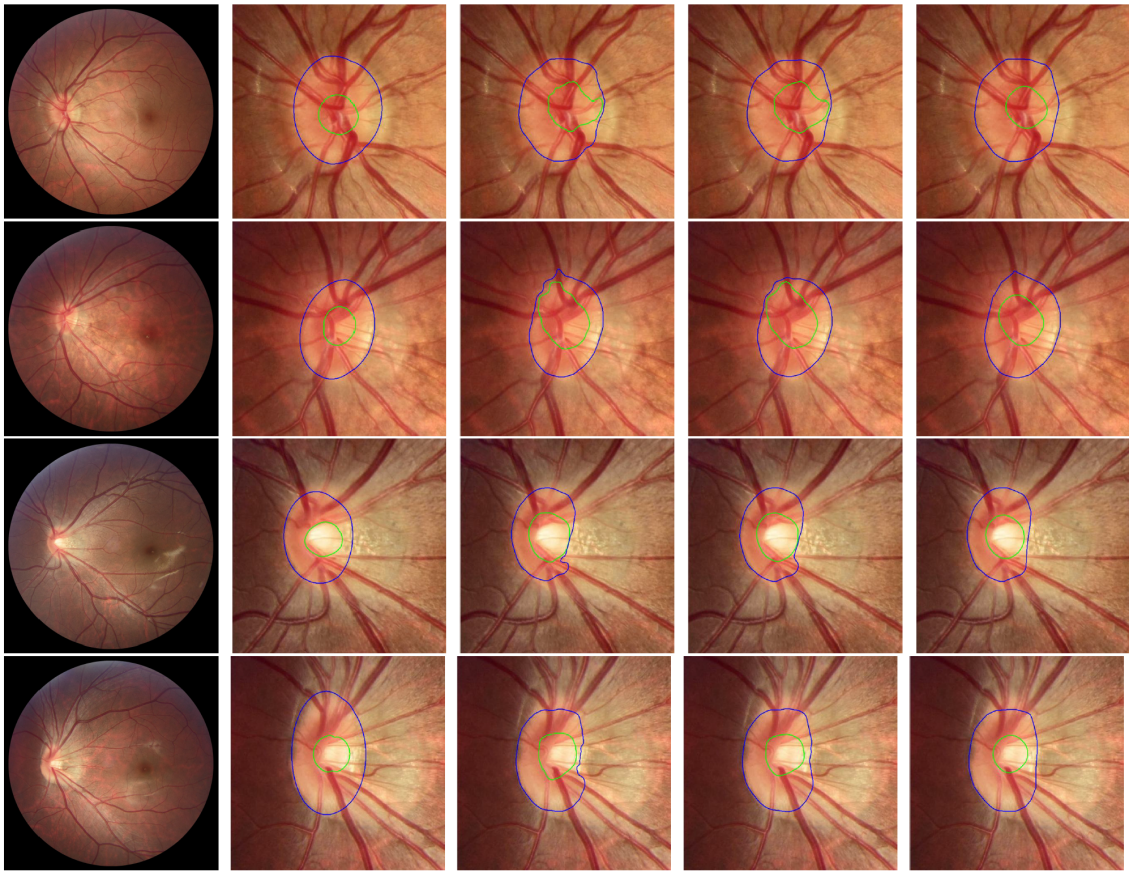


Figure 19: Visual quality on Refuge test set.

Visual quality



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

Figure 20: Visual quality on Refuge validation set.

Generalization ability

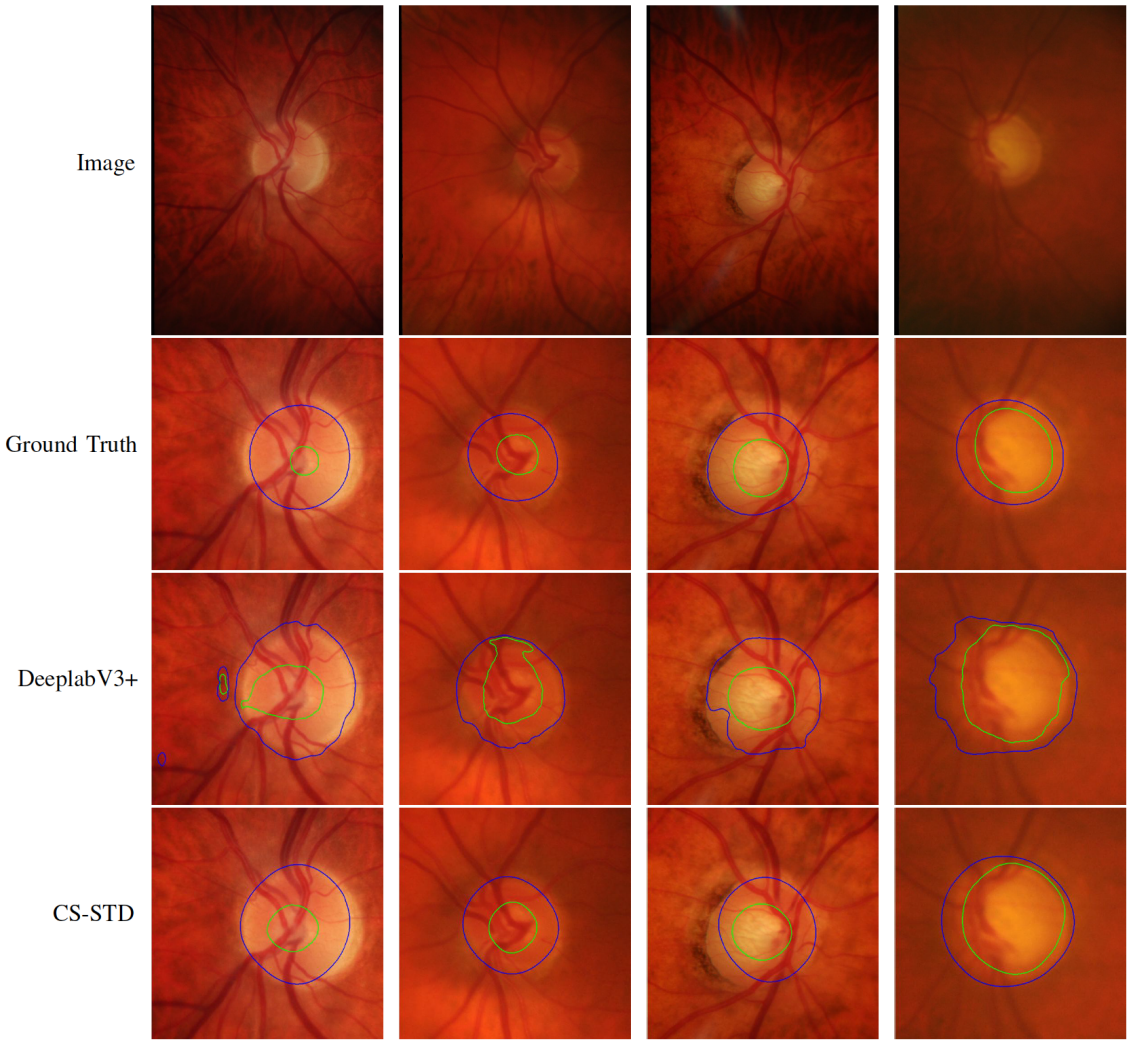


Figure 21: Visual quality of training on Refuge train set and predicting on RIM-ONE-r3 test set.

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 22: 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 23: 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

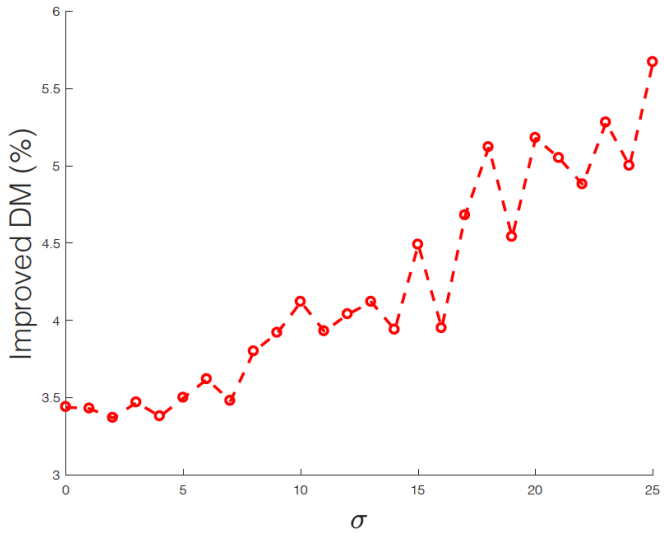


Figure 24: The improved DM values for cup regions in the Refuge test sets (400 images) between DeeplabV3+ and the proposed CS-STD under different levels of noise with standard deviation.

Conclusion and discussion

- A general method to integrate variational segmentation models into DCNN architecture.
- Spatial regularization, volume, star shape, convexity prior for DCNN such as popular deeplabV3+.
- Model prior can guide the DCNN to learn important feature, and can improve the generalization ability of DCNN.
- The idea can be extended to many DCNN applications in which the data has special knowledge.

Thank you!

Homepage: `https://www.norceresearch.no/personer/xue-cheng-tai/`