Hand Segmentation with Recurrent U-Net

Speaker: Wang Wei

CVLAB EPFL

04 October, 2019
Motivation

**Augmented Reality** for Wearable Camera
Highlight Hands for better User Experience

Challenges:
1. Limited Resources (no powerful GPUs).
2. Process images in real time.
Recurrent U-Net for Resource-Constrained Segmentation

Wei Wang* Kaicheng Yu* Joachim Hugonot Pascal Fua
Mathieu Salzmann
CVLab, EPFL, 1015 Lausanne
{first.last}@epfl.ch

Abstract

State-of-the-art segmentation methods rely on very deep networks that are not always easy to train without very large training datasets and tend to be relatively slow to run on standard GPUs. In this paper, we introduce a novel recurrent U-Net architecture that preserves the compactness of the original U-Net [33], while substantially increasing its performance to the point where it outperforms the state of the art on several benchmarks. We will demonstrate its effectiveness for several tasks, including hand segmentation, retina vessel segmentation, and road segmentation. We also introduce a large-scale dataset for hand segmentation.

1. Introduction
Hand Segmentation

Dataset Collection - **KBH** (Keyboard Hand Dataset)

### Resolution and Image Count

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Resolution</th>
<th>Train</th>
<th>Val.</th>
<th>Test</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>KBH (Ours)</td>
<td>230×306</td>
<td>2300</td>
<td>2300</td>
<td>7936</td>
<td>12536</td>
</tr>
</tbody>
</table>

### Environment Setup

- **Parameters**
  - Desk: 3 (White, Brown, Black)
  - Desk position: 3
  - Keyboard: 9
  - Lighting: 8 (3 sources on/off)
  - Objects on desk: 3 (3 different objects)

### Attributes

- **Attribute**
  - Bracelet: 10
  - Watch: 14
  - Brown-skin: 2
  - Tatoo: 1
  - Nail-polish: 1
  - Ring(s): 6
Intuition

Mimic human Saccadic eye Movement

When we observe a scene, our eyes undergo saccadic movements [Neuroscience. 2nd edition], and we accumulate knowledge about the scene and continuously refine our perception.
Intuition

Recursive Refinement
Model Overview

**Ours-DRU vs Others**

1. RGB Image
2. Ours-DRU
3. Rec-Middle
4. Rec-Last
5. Rec. Simple
6. U-Net-G
Model Backbone: U-Net

• Resource Constrained tasks:
  • Limited computing resource 😐 e.g., VR Camera.
  • Limited training data 😞 e.g., Biomedical Images.

• Main stream fast segmentation models:
  • Multi-branch based ones;
    • Complex
    • Careful Design
  • **U-Net** based ones;
    • Compact (lower risk of overfitting)
    • Simple (do not require careful design)
Model Details

Recurrent U-Net: DRU & SRU

(a) Options Sketch
(b) R-UNet ($\ell = 3$)
(c) DRU
(d) SRU
Experiment

• Segmentation Tasks
  • Retina Vessel
  • Hand
  • Road

Recursive Refinement
Train with 3 Recurrences
Test with 3 Recurrences

Image  T = 1  T = 2  T = 3  Ground Truth
Experiment

More Results of Hand
Study of Hyper-Parameters & Other Architectures

• Hyperparameters:
  • 1. Weight of loss at each recurrence.
  • 2. Number of recurrences

• New Architectures:
  • 1. VGG16 as Encoder
  • 2. ResNet50 as Encoder
Hyper-parameter: The impact of $\alpha$ in loss.

Network Init

$L_1 \rightarrow L_2 \rightarrow L_3$

Mean-IOU

$\text{Validating } \alpha \text{ for DRU}(l=4)$.

$\text{EYTH validation set}$

$\alpha=0.4$ has the best performance.

$\begin{align*}
L &= \sum_{t=1}^{N} w_t L_t \\
\text{where } w_t &= \alpha^{N-t} \quad (0 < \alpha \leq 1)
\end{align*}$
Hyper-parameter: The impact of Rec. Number

<table>
<thead>
<tr>
<th>Rec No</th>
<th>R: 3</th>
<th>R: 6</th>
<th>R: 9</th>
<th>R: 12</th>
</tr>
</thead>
<tbody>
<tr>
<td>[h,w]=[1,0.4]</td>
<td>0.8383</td>
<td>0.8406</td>
<td><strong>0.8420</strong></td>
<td>0.8361</td>
</tr>
</tbody>
</table>

**EYTH validation set**

Same Recurrence Number in the training and validation stage.

9 Recurrences lead to the best performance.

**Question:** Can we have different Recurrence Number for each image in the training and validation stage?
Compare with more baselines w.r.t Accuracy, Speed, Size.

(a) Frame per second (FPS)

(b) Image

Hand segmentation

Road segmentation
Other Architectures with Powerful Encoders

(a) DRU-VGG16

(b) DRU-ResNet50
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>mIOU</td>
<td>mRec</td>
<td>mPrec</td>
<td>mIOU</td>
<td>mRec</td>
</tr>
<tr>
<td><strong>No pre-train</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ICNet [45]</td>
<td>0.731</td>
<td>0.915</td>
<td>0.764</td>
<td>0.898</td>
<td>0.971</td>
</tr>
<tr>
<td>U-Net-B [33]</td>
<td>0.803</td>
<td>0.912</td>
<td>0.830</td>
<td>0.950</td>
<td>0.973</td>
</tr>
<tr>
<td>U-Net-G</td>
<td>0.837</td>
<td>0.928</td>
<td>0.883</td>
<td>0.952</td>
<td>0.977</td>
</tr>
<tr>
<td>Rec-Middle [27]</td>
<td>0.827</td>
<td>0.920</td>
<td>0.877</td>
<td>0.924</td>
<td><strong>0.979</strong></td>
</tr>
<tr>
<td>Rec-Last [41]</td>
<td>0.838</td>
<td>0.920</td>
<td>0.894</td>
<td>0.957</td>
<td>0.975</td>
</tr>
<tr>
<td>Rec-Simple [21]</td>
<td>0.827</td>
<td>0.918</td>
<td>0.864</td>
<td>0.952</td>
<td>0.975</td>
</tr>
<tr>
<td><strong>Light</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Ours at layer (</strong>$\ell$)**</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ours-SRU(0)</td>
<td>0.844</td>
<td>0.924</td>
<td>0.890</td>
<td><strong>0.960</strong></td>
<td>0.976</td>
</tr>
<tr>
<td>Ours-SRU(3)</td>
<td>0.845</td>
<td><strong>0.931</strong></td>
<td>0.891</td>
<td>0.956</td>
<td>0.977</td>
</tr>
<tr>
<td>Ours-DRU(4)</td>
<td><strong>0.849</strong></td>
<td>0.926</td>
<td><strong>0.900</strong></td>
<td>0.958</td>
<td>0.978</td>
</tr>
<tr>
<td><strong>Heavy</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>With pretrain</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RefineNet [40]</td>
<td>0.688</td>
<td>0.776</td>
<td>0.853</td>
<td>0.821</td>
<td>0.869</td>
</tr>
<tr>
<td>Deeplab V3+ [6]</td>
<td>0.757</td>
<td>0.819</td>
<td>0.875</td>
<td>0.907</td>
<td>0.928</td>
</tr>
<tr>
<td>U-Net-VGG16</td>
<td>0.879</td>
<td>0.945</td>
<td>0.921</td>
<td>0.961</td>
<td>0.978</td>
</tr>
<tr>
<td>U-Net-ResNet50</td>
<td>0.893</td>
<td>0.942</td>
<td>0.939</td>
<td>0.959</td>
<td>0.978</td>
</tr>
<tr>
<td><strong>DRA-VGG16</strong></td>
<td>0.897</td>
<td>0.946</td>
<td>0.940</td>
<td><strong>0.964</strong></td>
<td><strong>0.981</strong></td>
</tr>
<tr>
<td><strong>DRA-ResNet50</strong></td>
<td><strong>0.902</strong></td>
<td><strong>0.947</strong></td>
<td><strong>0.945</strong></td>
<td>0.959</td>
<td>0.980</td>
</tr>
</tbody>
</table>

Table 3: **Comparing against the state of the art.** According to the mIOU, *Ours-DRA(4)* performs best on average, with *Ours-SRU(0)* a close second. Generally speaking all recurrent methods do better than *RefineNet*, which represents the state of the art, on all datasets except HOF. We attribute this to HOF being too small for optimal performance without pre-training, as in *RefineNet*. This is confirmed by looking at *DRA-VGG16*, which yields the overall best results by relying on a pretrained deep backbone.
# Experiment

## Retina Vessel, Road, Cityscapes Segmentation

<table>
<thead>
<tr>
<th>Models</th>
<th>Retina</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>mIOU</td>
<td>mRec</td>
<td>mPrec</td>
<td>mF1</td>
<td></td>
</tr>
<tr>
<td>ICNet [45]</td>
<td>0.618</td>
<td>0.796</td>
<td>0.690</td>
<td>0.739</td>
<td></td>
</tr>
<tr>
<td>U-Net-G [33]</td>
<td>0.800</td>
<td>0.897</td>
<td>0.868</td>
<td>0.882</td>
<td></td>
</tr>
<tr>
<td>Rec-Middle [27]</td>
<td>0.818</td>
<td><strong>0.903</strong></td>
<td>0.886</td>
<td>0.894</td>
<td></td>
</tr>
<tr>
<td>Rec-Simple [21]</td>
<td>0.814</td>
<td>0.898</td>
<td>0.885</td>
<td>0.892</td>
<td></td>
</tr>
<tr>
<td>Rec-Last [41]</td>
<td>0.819</td>
<td>0.900</td>
<td>0.890</td>
<td>0.895</td>
<td></td>
</tr>
<tr>
<td>Ours-DRU(4)</td>
<td><strong>0.821</strong></td>
<td>0.902</td>
<td><strong>0.891</strong></td>
<td><strong>0.896</strong></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Models</th>
<th>Road</th>
<th></th>
<th></th>
<th>P/R</th>
<th>mF1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>mIOU</td>
<td>mRec</td>
<td>mPrec</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ICNet [45]</td>
<td>0.476</td>
<td>0.626</td>
<td>0.500</td>
<td>0.513</td>
<td>0.656</td>
</tr>
<tr>
<td>U-Net-G [33]</td>
<td>0.479</td>
<td>0.639</td>
<td>0.502</td>
<td>0.642</td>
<td>0.563</td>
</tr>
<tr>
<td>Rec-Middle [27]</td>
<td>0.494</td>
<td>0.767</td>
<td>0.518</td>
<td>0.660</td>
<td>0.574</td>
</tr>
<tr>
<td>Rec-Simple [21]</td>
<td>0.534</td>
<td>0.802</td>
<td>0.559</td>
<td>0.723</td>
<td>0.659</td>
</tr>
<tr>
<td>Rec-Last [41]</td>
<td>0.526</td>
<td>0.786</td>
<td>0.551</td>
<td>0.730</td>
<td>0.648</td>
</tr>
<tr>
<td>Ours-DRU(4)</td>
<td><strong>0.560</strong></td>
<td><strong>0.865</strong></td>
<td><strong>0.583</strong></td>
<td><strong>0.757</strong></td>
<td><strong>0.691</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Models</th>
<th>Cityscapes</th>
<th></th>
<th></th>
<th>mIOU</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>mIOU</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ICNet[45]</td>
<td>0.695</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>U-Net-G</td>
<td>0.429</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rec-Last</td>
<td>0.502</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DRU(4)</td>
<td>0.532</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DeepLab V3 [5]</td>
<td>0.778</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>U-Net-G × 2</td>
<td>0.476</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rec-Last × 2</td>
<td>0.521</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DRU(4) × 2</td>
<td>0.627</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DRU-VGG16</td>
<td>0.761</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DRU-VGG16</td>
<td>0.775</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Summary

• Recurrent U-Net refines predictions step by step.
• It is friendly to embedded systems with
  • less parameters,
  • high speed,
  • lower risk of overfitting.
• It is easy to scale up for unconstrained settings with more powerful encoders.
• Two ongoing works:
  • 1. Improve performance: Remove the noise and redundancy in deep networks.
  • 2. Virtual keyboard typing.
Ongoing Work 1

- Problem:
  - Big Network has more parameters, but brings noise & redundancy.

- Solution:
  - Build normalization layers (ZCA/PCA normalization) to Remove Noise & Redundancy (correlation).

- Challenges:
  - Need a numerically stable eigendecomposition layer in deep networks.

- Our work:
  - Use SVD in the forward pass.
  - Use Power Iteration in the backward pass (it has bounded gradients).

---

Backpropagation-Friendly Eigendecomposition

Wei Wang¹, Zheng Dang², Yinlin Hu¹, Pascal Fua¹, and Mathieu Salzmann¹

¹CVLab, EPFL, CH-1015 Lausanne, Switzerland {firstname.lastname}@epfl.ch
²Xi’an Jiaotong University, China {dangzheng713@stu.xjtu.edu.cn}

Abstract

Eigendecomposition (ED) is widely used in deep networks. However, the backpropagation of its results tends to be numerically unstable, whether using ED directly or approximating it with the Power Iteration method, particularly when dealing with large matrices. While this can be mitigated by partitioning the data in small and arbitrary groups, doing so has no theoretical basis and makes it impossible to exploit the power of ED to the full.

In this paper, we introduce a numerically stable and differentiable approach to leveraging eigenvectors in deep networks. It can handle large matrices without requiring to split them. We demonstrate the better robustness of our approach over standard ED and PI for ZCA whitening, an alternative to batch normalization, and for PCA denoising, which we introduce as a new normalization strategy for deep networks, aiming to further denoise the network’s features.

Partially Done!

1. Can not compute full rank eigenvectors.
2. Forward pass is slow for large matrices whose dim>=128.
Ongoing Work 2

• Virtual Keyboard Typing.
Thanks 😊