Hand Segmentation with Recurrent U-Net

Speaker: Wang Wei

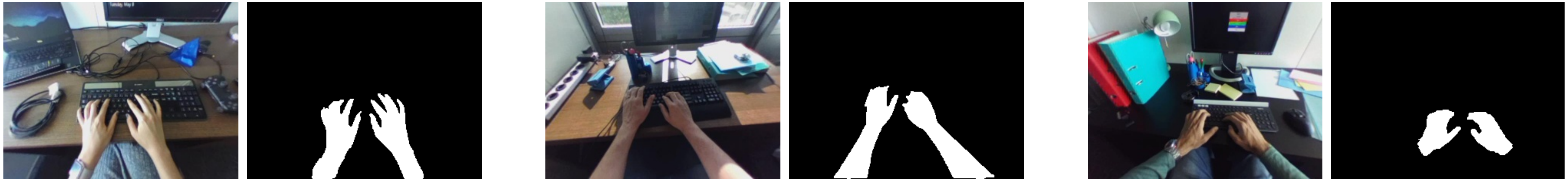
CVLAB EPFL

04 October, 2019

Motivation

Augmented Reality for Wearable Camera

Highlight Hands for better User Experience

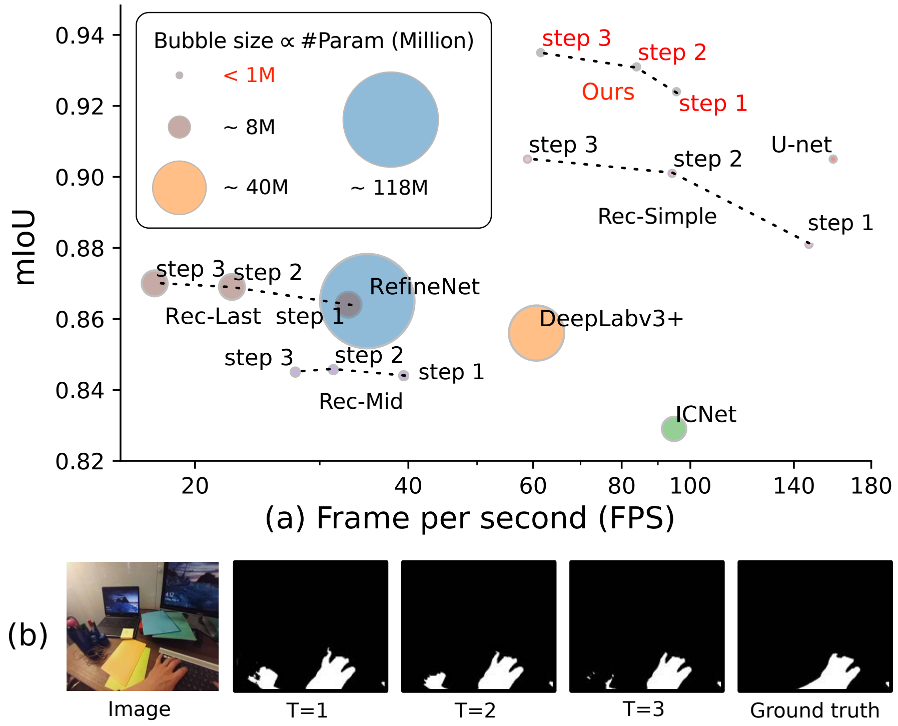


Challenges:

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		# Images			
Dataset	Width	Height	Train	Val.	Test	Total
KBH (Ours)	230×	306	2300	2300	7936	12536

(a) Environment setup

Parameters	Amount	Details
Desk	3	White, Brown, Black
Desk position	3	-
Keyboard	9	-
Lighting	8	3 sources on/off
Objects on desk	3	3 different objects

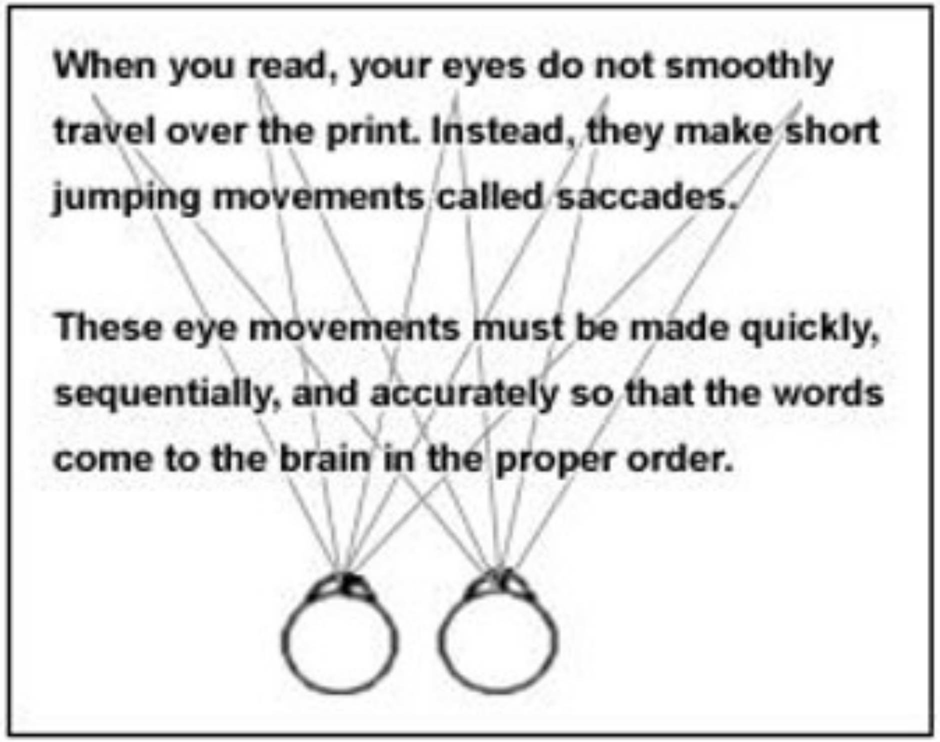
(b) Attributes

Attribute	#IDs
Bracelet	10
Watch	14
Brown-skin	2
Tatoo	1
Nail-polish	1
Ring(s)	6

Intuition

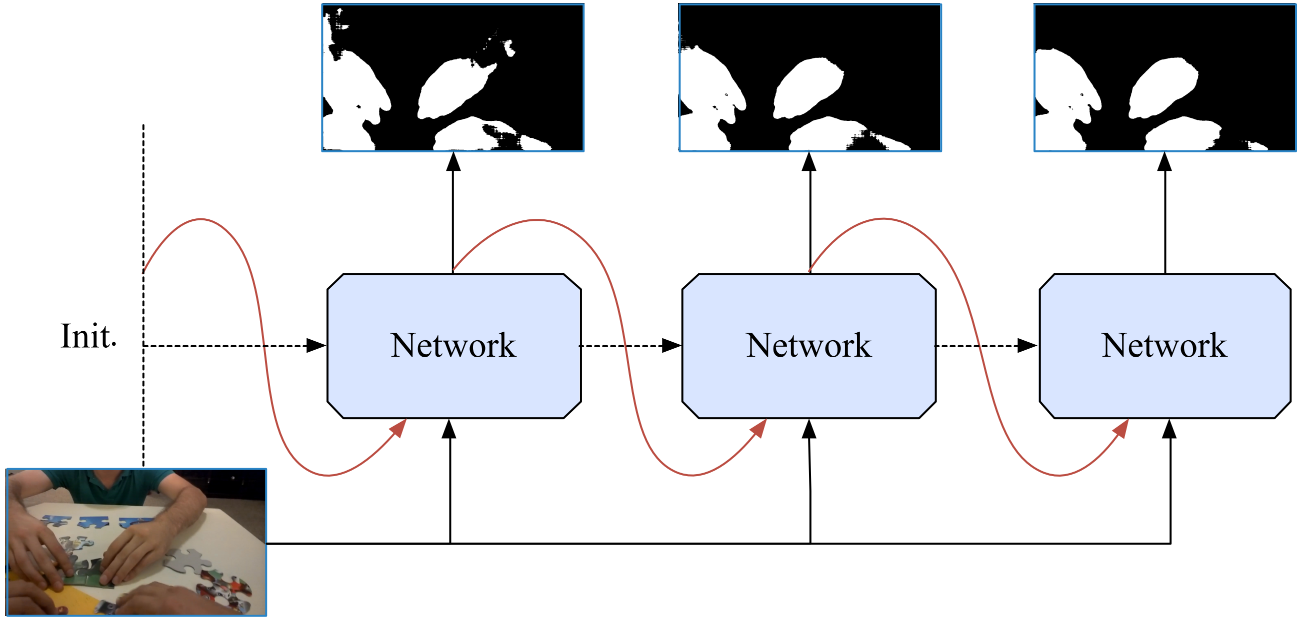
Mimic human Saccadic eye Movement

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.



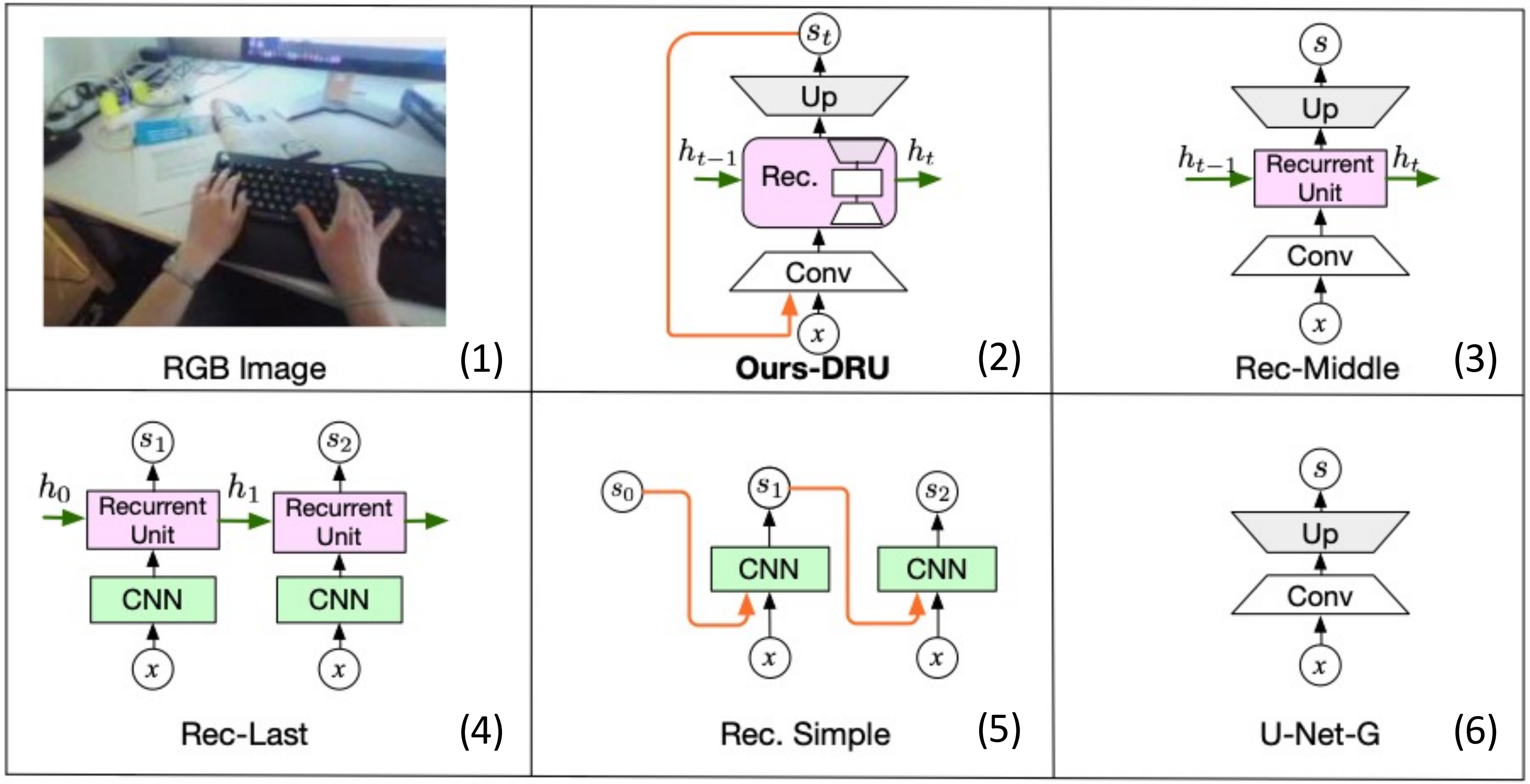
Intuition

Recursive Refinement



Model Overview

Ours-DRU vs Others



Model Backbone: U-Net

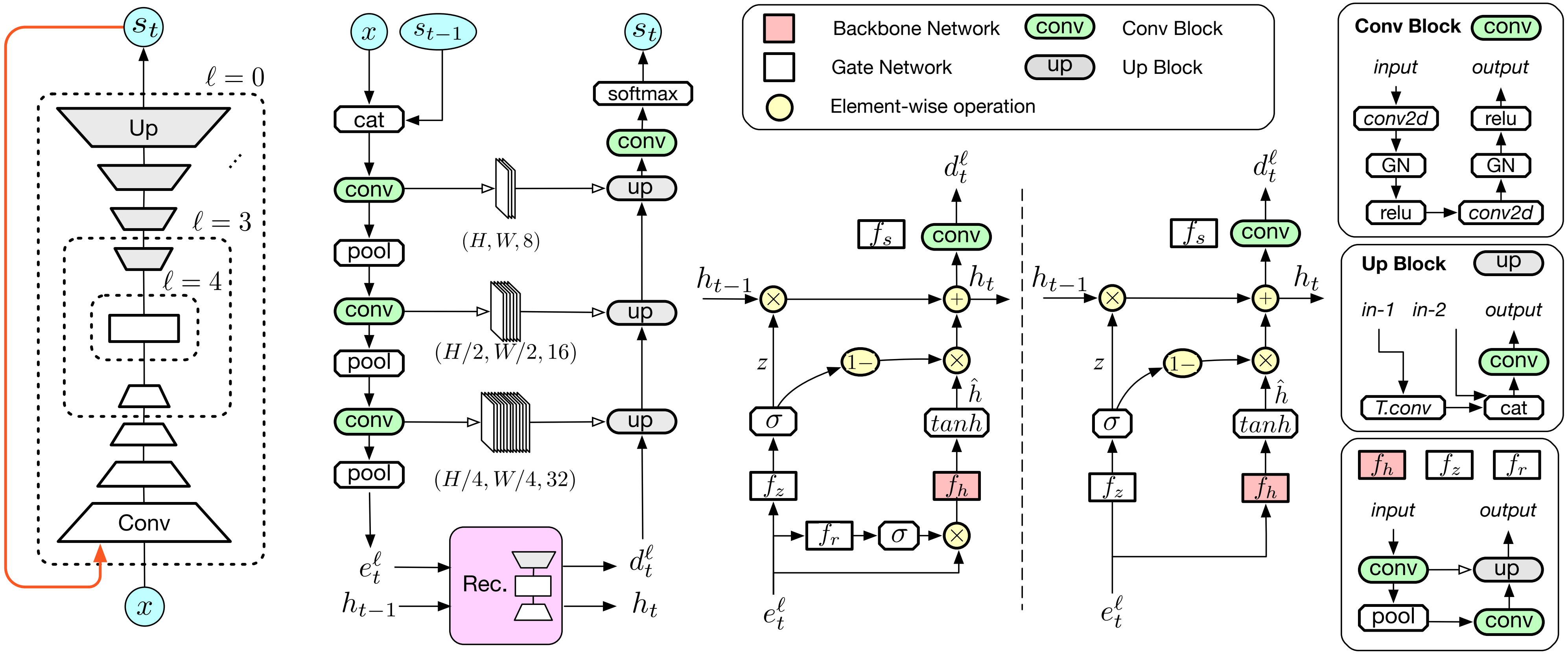
- Resource Constrained tasks:
  - Limited computing resource ☹ e.g., VR Camera.
  - Limited training data ☹ e.g., Biomedical Images.

- 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

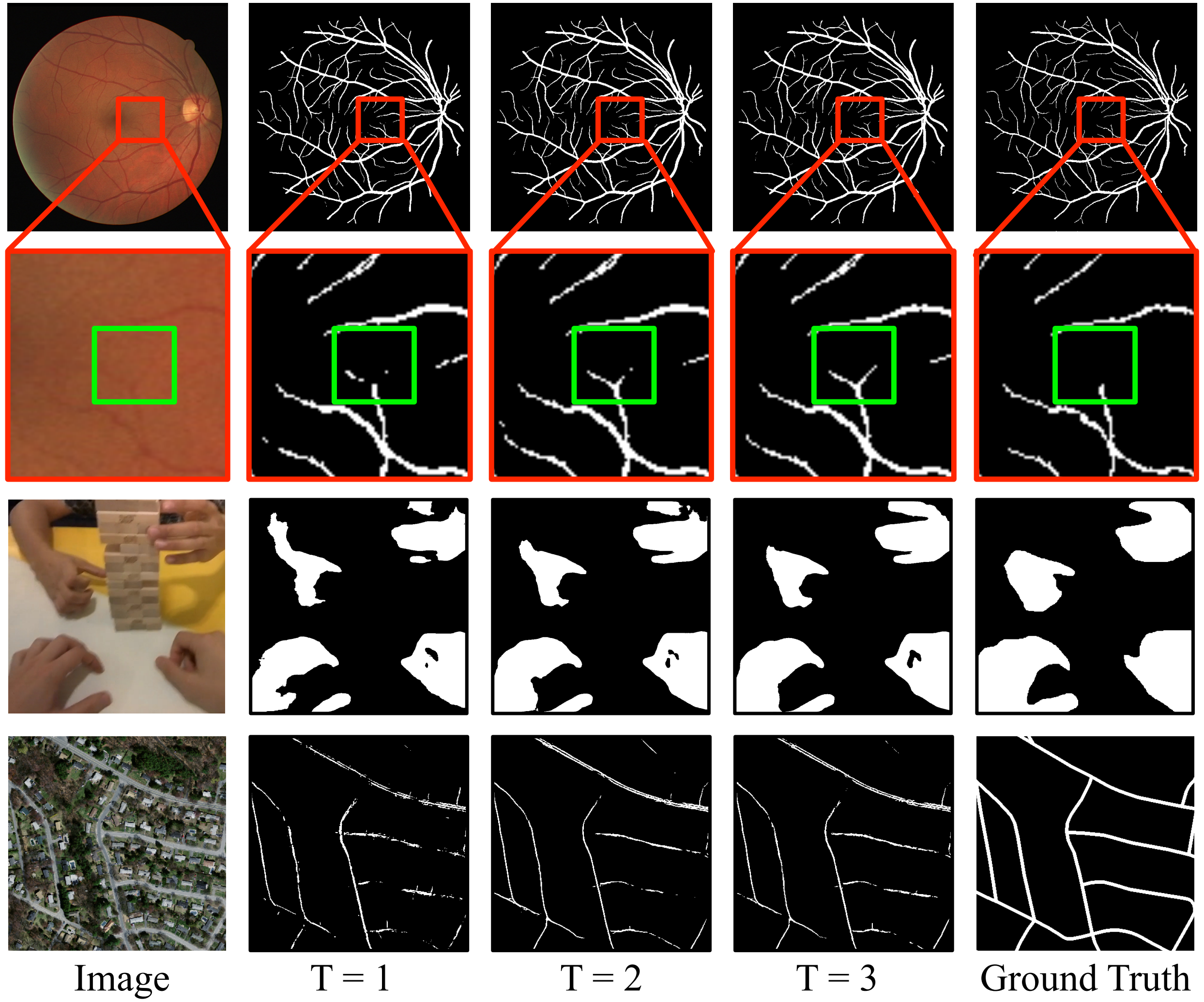
Experiment

- Segmentation Tasks
  - Retina Vessel
  - Hand
  - Road

Recursive Refinement

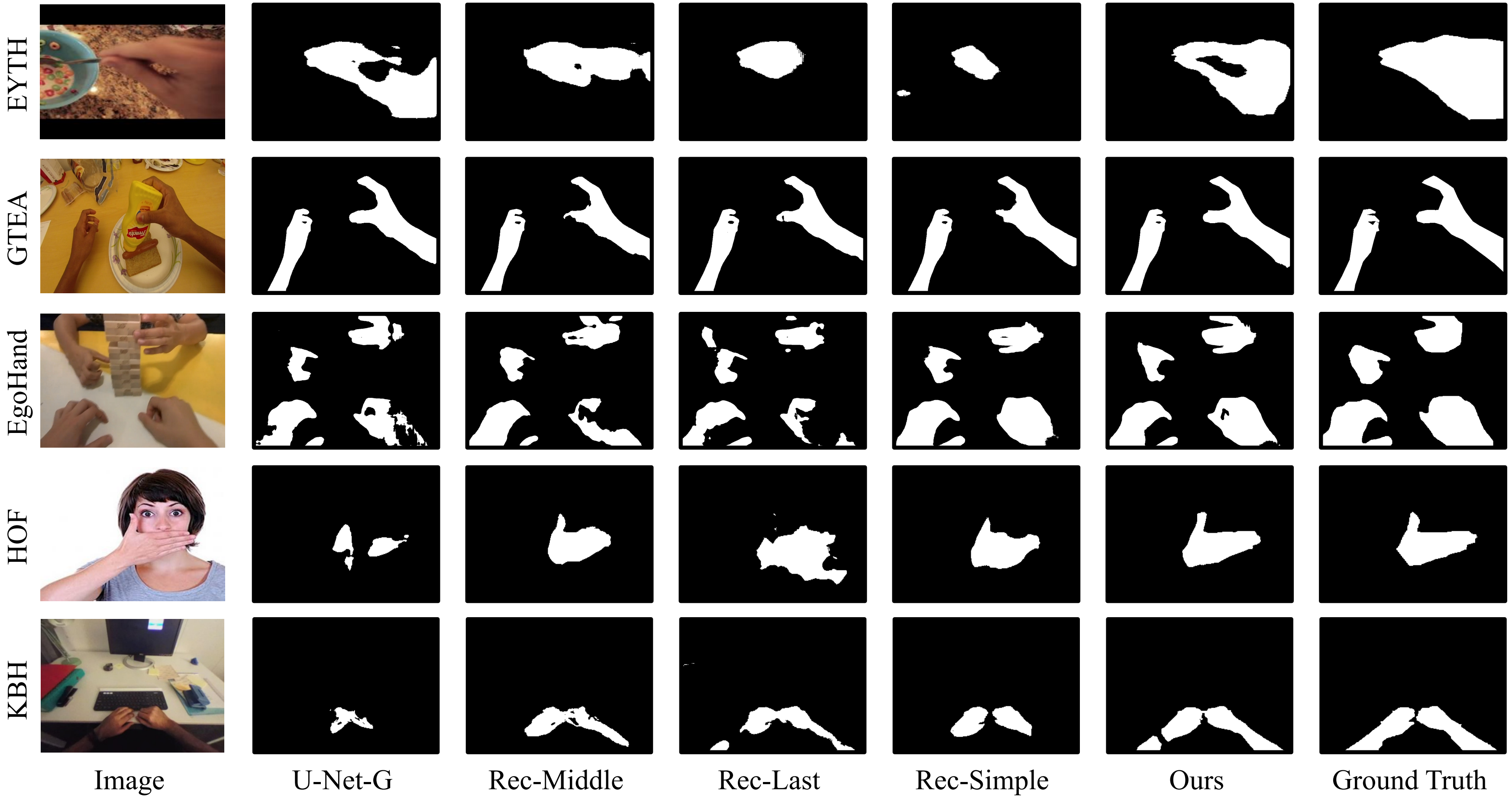
Train with 3 Recurrences

Test with 3 Recurrences



Experiment

More Results of Hand

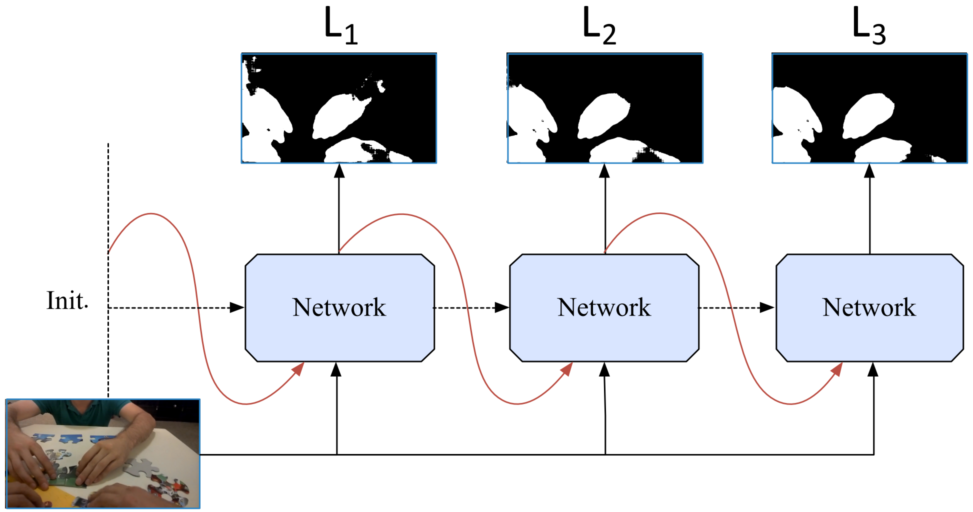


Study of Hyper-Parameters & Other Architectures

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

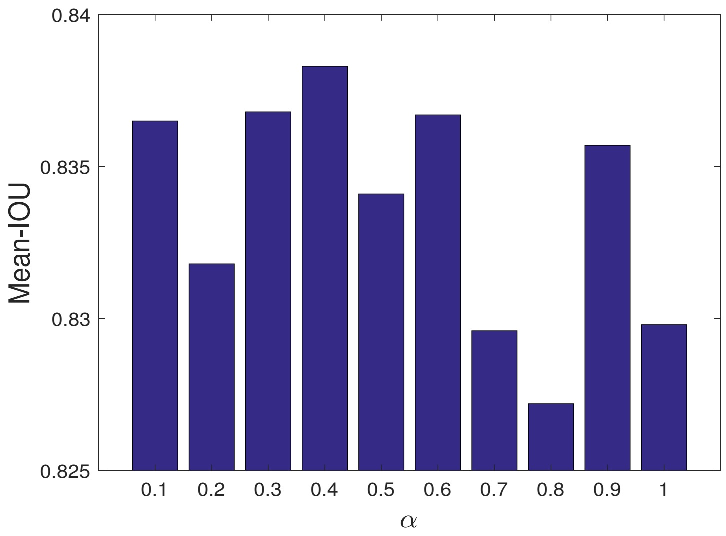
- New Architectures:
  - 1. VGG16 as Encoder
  - 2. ResNet50 as Encoder

Hyper-parameter: The impact of α in loss.



$$L = \sum_{t=1}^{N} w_t L_t$$

$$w_t = \alpha^{N-t}$$ $(0<\alpha \leq 1)$



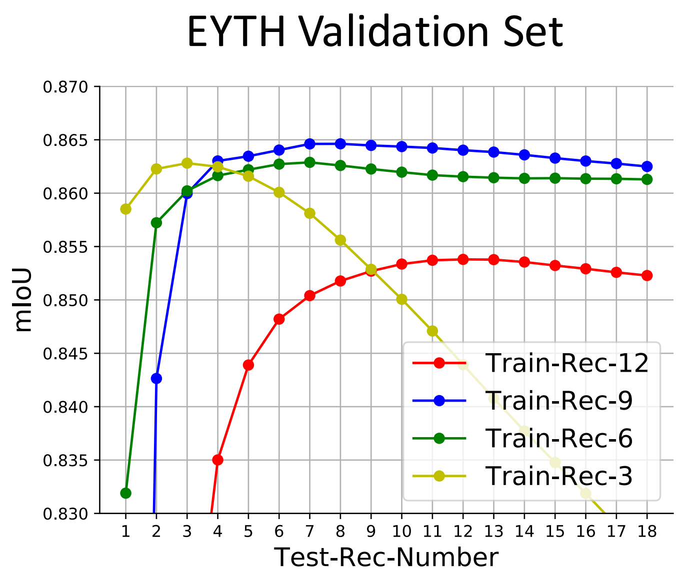
Validating α for DRU(l=4).

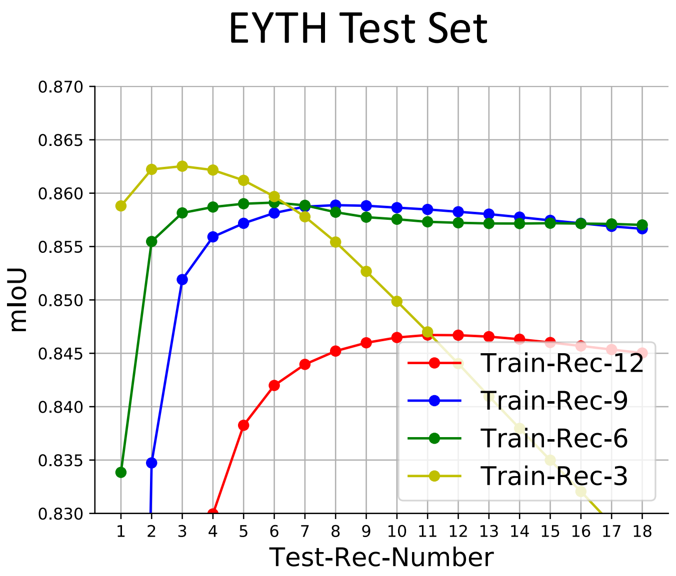
EYTH validation set

α=0.4 has the best performance.

Hyper-parameter: The impact of Rec. Number

Rec No	R: 3	R: 6	R: 9	R: 12
[h,w]=[1,0.4]	0.8383	0.8406	**0.8420**	0.8361





EYTH validation set

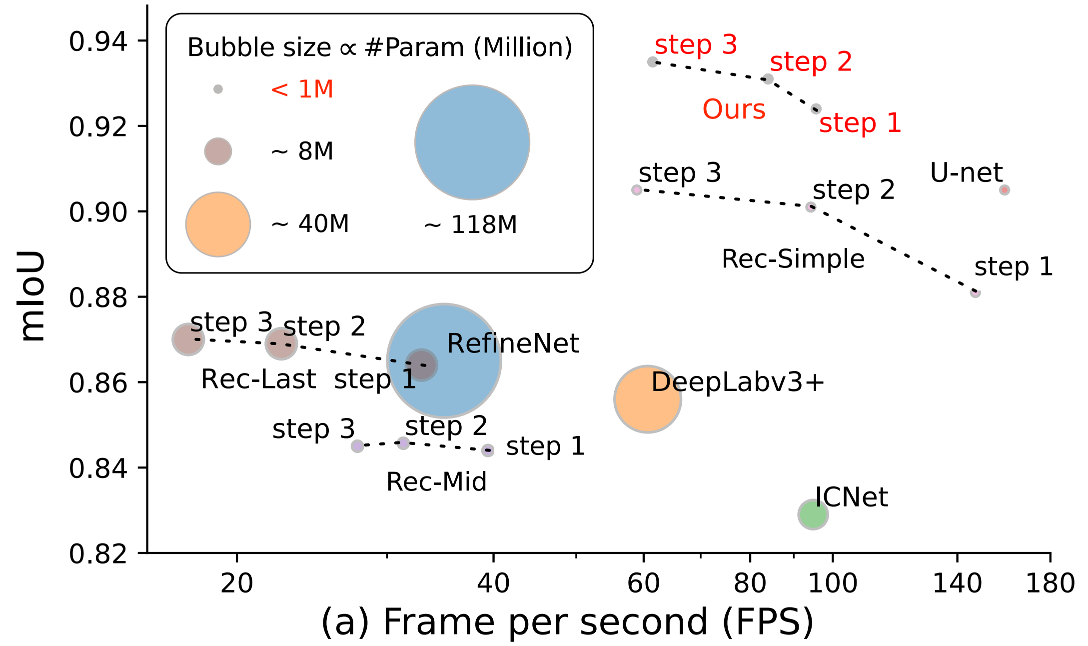
Same Recurrence Number
in the **training** and **validation** stage.

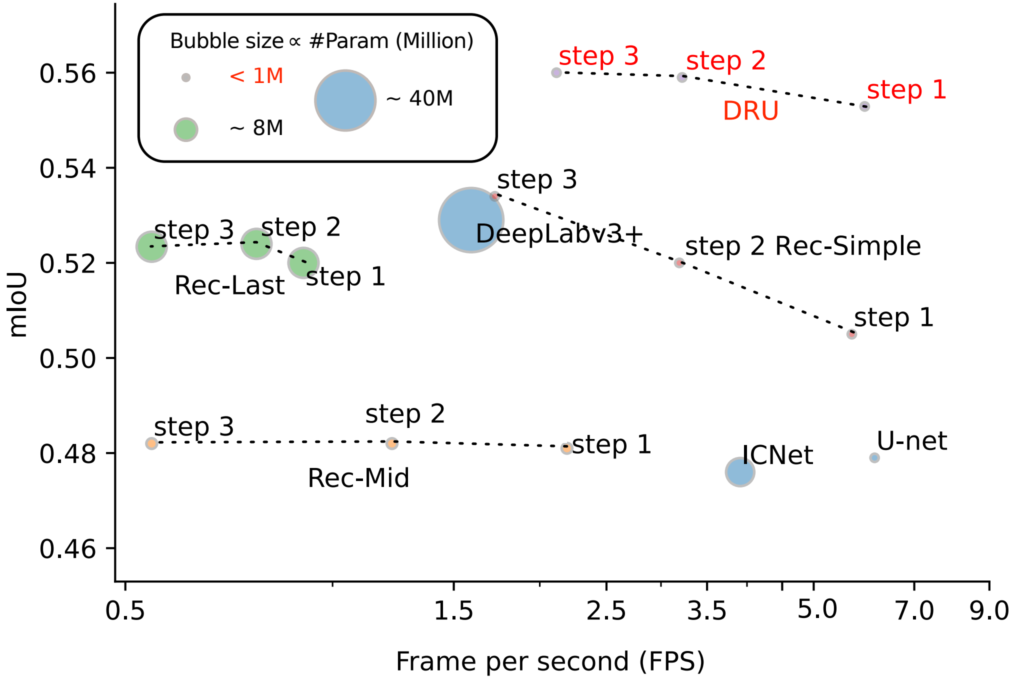
9 Recurrences lead to the best performance.

Question: Can we have **different**
Recurrence Number for each image
in the **training** and **validation** stage?

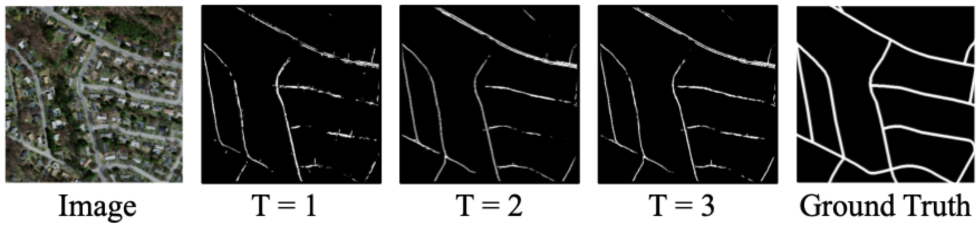
Compare with more baselines w.r.t

Accuracy, Speed, Size.



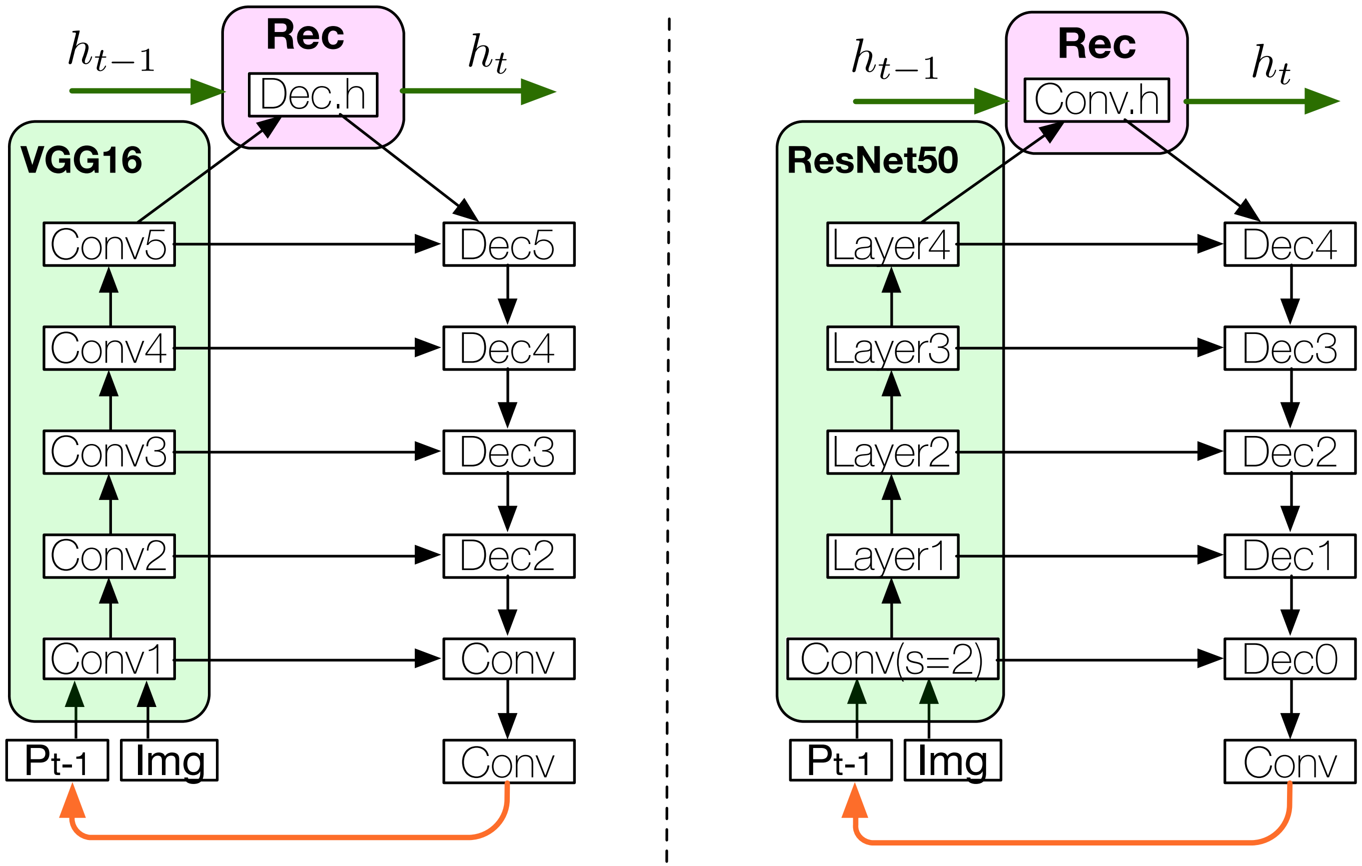


Hand segmentation



Road segmentation

Other Architectures with Powerful Encoders



(a) DRU-VGG16 (b) DRU-ResNet50

Experiment

	Model	EYTH [40]			GTEA [11]			EgoHand [4]			HOF [40]			KBH		
		mIOU	mRec	mPrec	mIOU	mRec	mPrec	mIOU	mRec	mPrec	mIOU	mRec	mPrec	mIOU	mRec	mPrec
Light	*No pre-train*															
	ICNet [45]	0.731	0.915	0.764	0.898	0.971	0.922	0.872	**0.925**	0.931	0.580	0.801	0.628	0.829	0.925	0.876
	U-Net-B [33]	0.803	0.912	0.830	0.950	0.973	0.975	0.815	0.869	0.876	0.694	**0.867**	0.778	0.870	0.943	0.911
	U-Net-G	0.837	0.928	0.883	0.952	0.977	0.980	0.837	0.895	0.899	0.621	0.741	0.712	0.905	0.949	0.948
	Rec-Middle [27]	0.827	0.920	0.877	0.924	**0.979**	0.976	0.828	0.894	0.905	0.654	0.733	**0.796**	0.845	0.924	0.898
	Rec-Last [41]	0.838	0.920	0.894	0.957	0.975	0.980	0.831	0.906	0.897	0.674	0.807	0.752	0.870	0.930	0.924
	Rec-Simple [21]	0.827	0.918	0.864	0.952	0.975	0.976	0.858	0.909	0.931	0.693	0.833	0.704	0.905	0.951	0.944
	Ours at layer (ℓ)															
	Ours-SRU(0)	0.844	0.924	0.890	**0.960**	0.976	0.981	0.862	0.913	0.932	**0.712**	0.844	0.764	0.930	0.968	0.957
	Ours-SRU(3)	0.845	**0.931**	0.891	0.956	0.977	**0.982**	0.864	0.913	0.933	0.699	0.864	0.773	0.921	0.964	0.951
	Ours-DRU(4)	**0.849**	0.926	**0.900**	0.958	0.978	0.977	**0.873**	0.924	**0.935**	0.709	0.866	0.774	**0.935**	**0.980**	**0.970**
Heavy	*With pretrain*															
	RefineNet [40]	0.688	0.776	0.853	0.821	0.869	0.928	0.814	0.919	0.879	0.766	0.882	0.859	0.865	0.954	0.921
	Deeplab V3+ [6]	0.757	0.819	0.875	0.907	0.928	0.976	0.870	0.909	**0.958**	0.722	0.822	0.816	0.856	0.901	0.935
	U-Net-VGG16	0.879	0.945	0.921	0.961	0.978	0.981	0.879	0.916	0.951	0.849	0.937	0.893	0.946	0.971	0.972
	U-Net-ResNet50	0.893	0.942	0.939	0.959	0.978	0.980	0.900	0.936	0.954	0.867	0.949	0.904	0.948	0.973	0.972
	DRU-VGG16	0.897	0.946	0.940	**0.964**	**0.981**	**0.982**	0.892	0.925	**0.958**	0.863	0.948	**0.901**	0.954	0.973	**0.979**
	DRU-ResNet50	**0.902**	**0.947**	**0.945**	0.959	0.980	0.978	**0.898**	**0.937**	0.952	**0.889**	**0.948**	**0.930**	**0.957**	**0.978**	0.977

Table 3: **Comparing against the state of the art.** According to the mIOU, *Ours-DRU*(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 DRU-VGG16, which yields the overall best results by relying on a pretrained deep backbone.

Experiment

Retina Vessel, Road, Cityscapes Segmentation

Retina

	Models	mIOU	mRec	mPrec	mF1
Light	ICNet [45]	0.618	0.796	0.690	0.739
	U-Net-G [33]	0.800	0.897	0.868	0.882
	Rec-Middle [27]	0.818	**0.903**	0.886	0.894
	Rec-Simple [21]	0.814	0.898	0.885	0.892
	Rec-Last [41]	0.819	0.900	0.890	0.895
	Ours-DRU(4)	**0.821**	0.902	**0.891**	**0.896**
Heavy	DeepLab V3+ [6]	0.756	0.875	0.828	0.851
	U-Net-VGG16	0.804	**0.910**	0.862	0.886
	DRU-VGG16	**0.817**	0.905	**0.883**	**0.894**

Road

	Models	mIOU	mRec	mPrec	P/R	mF1
Light	ICNet [45]	0.476	0.626	0.500	0.513	0.656
	U-Net-G [33]	0.479	0.639	0.502	0.642	0.563
	Rec-Middle [27]	0.494	0.767	0.518	0.660	0.574
	Rec-Simple [21]	0.534	0.802	0.559	0.723	0.659
	Rec-Last [41]	0.526	0.786	0.551	0.730	0.648
	Ours-DRU(4)	**0.560**	**0.865**	**0.583**	**0.757**	**0.691**
Heavy	Deeplab V3+ [6]	0.529	0.763	0.555	0.710	0.643
	U-Net-VGG16	0.521	0.836	0.544	0.745	0.659
	DRU-VGG16	**0.571**	**0.862**	**0.595**	**0.761**	**0.704**

Cityscapes

Model	mIoU	Model	mIoU
ICNet[45]	0.695	DeepLab V3 [5]	0.778
U-Net-G	0.429	U-Net-G $\times 2$	0.476
Rec-Last	0.502	Rec-Last $\times 2$	0.521
DRU(4)	0.532	DRU(4) $\times 2$	0.627
		DRU-VGG16	0.761
		DRU-VGG16	0.775

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 Wang[1], Zheng Dang[2], Yinlin Hu[1], Pascal Fua[1], and Mathieu Salzmann[1]

[1]CVLab, EPFL, CH-1015 Lausanne, Switzerland {firstname.lastname}@epfl.ch
[2]Xi'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 its impossible to exploit the power of ED to the full.

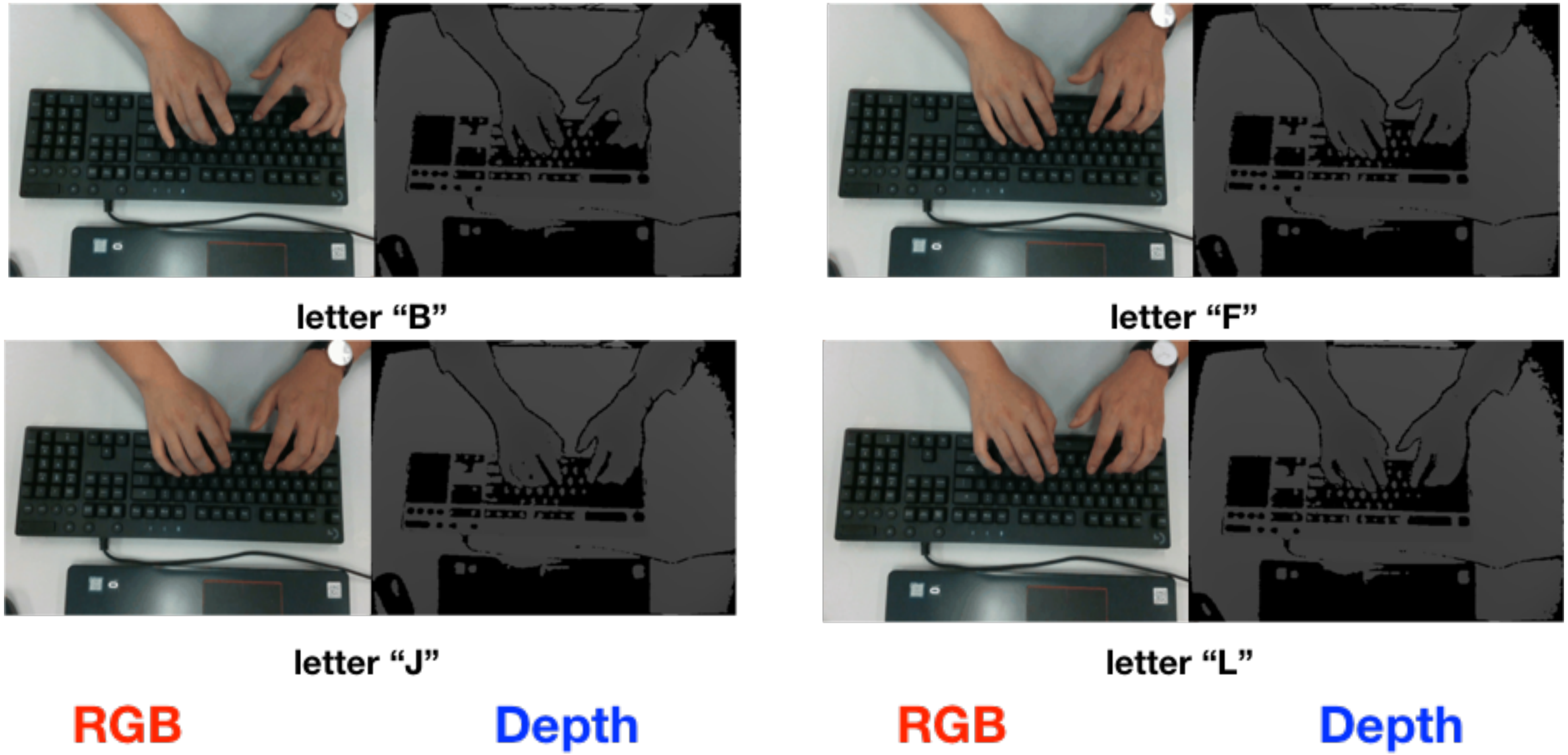
In this paper, we introduce a numerically stable and differentiable approach to leveraging eigenvectors in deep networks. It can handle large matrices without requiring to split them. We demonstrate the better robustness of our approach over standard ED and PI for ZCA whitening, an alternative to batch 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 ☺