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

While recent semantic segmentation methods achieve impressive results [6, 17, 18, 46], they require very deep networks and their architectures tend to focus on high-resolution and large-scale datasets and to rely on pre-trained backbones. For instance, state-of-the-art models, such as Deeplab [5, 6], PSPnet [46] and RefineNet [17], use a ResNet101 [15] as their backbone. This results in high GPU memory usage and inference time, and makes them less than ideal for operation in power-limited environments where real-time performance is nevertheless required, such as when segmenting hands using the onboard resources of an Augmented Reality headset. This has been addressed by architectures such as the ICNet [45] at the cost of a substantial performance drop. Perhaps even more importantly, training very deep networks usually requires either massive amounts of training data or image statistics close to that of ImageNet [10], which may not be appropriate in fields such as biomedical image segmentation where the more compact U-Net architecture remains prevalent [33].

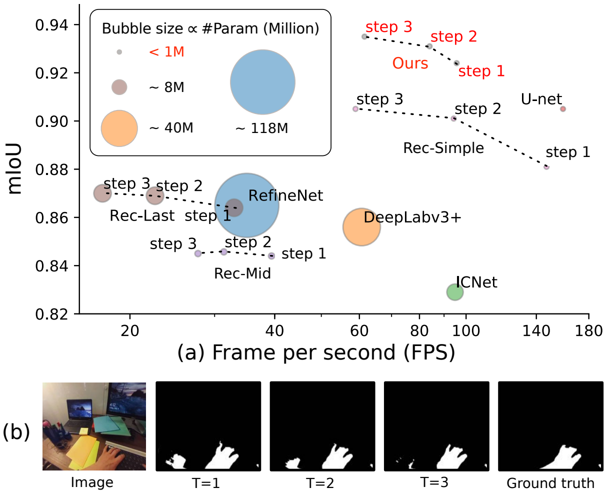
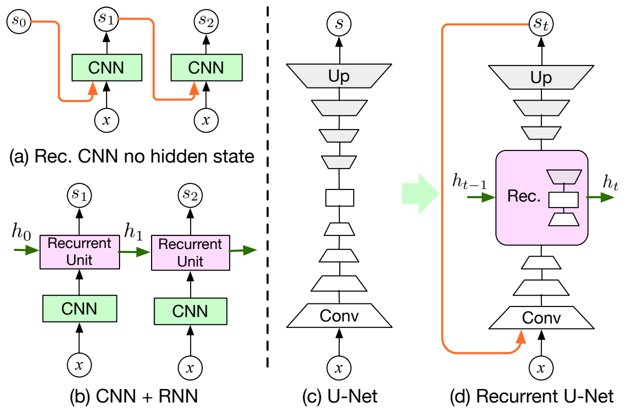


Figure 1: **Speed vs accuracy.** Each circle represents the performance of a model in terms frames-per-second and mIoU accuracy on our Keyboard Hand Dataset using a Titan X (Pascal) GPU. The radius of each circle denotes the models' number of parameters. For our recurrent approach, we plot these numbers after 1, 2, and 3 iterations, and we show the corresponding segmentations in the bottom row. The performance of our approach is plotted in red and the other acronyms are defined in Section 4.2. ICNet [45] is slightly faster than us but at the cost of a significant accuracy drop, whereas RefineNet [17] and DeepLab [6] are both slower and less accurate on this dataset, presumably because there are not enough training samples to learn their many parameters.

In this paper, we argue that these state-of-the-art methods do not naturally generalize to resource-constrained situations and 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 current state of the art on 5 hand-segmentation datasets, one of which is showcased in Fig. 1, and a retina vessel segmentation one. With only 0.3 million parameters, our model is much smaller than the ResNet101-based DeepLabv3+ [6] and RefineNet [17], with 40 and 118 million weights, respectively. This helps explain why we can outperform state-of-the-art networks on speclalized tasks: The pre-trained ImageNet features are not necessarily the best and training sets are not quite as large as CityScapes [9]. As a result, the large networks tend to overfit and do not perform as well as compact models trained from scratch.

The standard U-Net takes the image as input, processes it, and directly returns an output. By contrast, our recurrent

[*] Equal contributions.



(a) Rec. CNN no hidden state

(b) CNN + RNN

(c) U-Net

(d) Recurrent U-Net

Figure 2: **Recurrent segmentation.** **(a)** The simple strategy of [21, 24] consists of concatenating the previous segmentation mask s_{t-1} to the image x, and recurrently feeding this to the network. **(b)** For sequence segmentation, to account for the network's internal state, one can instead combine the CNN with a standard recurrent unit as in [41]. Here, we build upon the U-Net architecture of [33] **(c)**, and propose to build a recurrent unit over several of its layers, as shown in **(d)**. This allows us to propagate higher-level information through the recurrence, and, in conjunction with a recurrence on the segmentation mask, outperforms the two simpler recurrent architectures **(a)** and **(b)**.

architecture iteratively refines both the segmentation mask and the network's internal state. This mimics human perception as in the influential AutoContext paper [39]: When we observe a scene, our eyes undergo saccadic movements, and we accumulate knowledge about the scene and continuously refine our perception [29]. To this end we retain the overall structure of the U-Net, but build a recurrent unit over some of its inner layers for internal state update. By contrast with the simple CNN+RNN architecture of Fig. 2(b), often used for video or volumetric segmentation [41, 27, 3], this enables the network to keep track of and to iteratively update more than just a single-layer internal state. This gives us the flexibility to choose the portion of the internal state that we exploit for recursion purposes and to explore variations of our scheme.

We demonstrate the benefits of our recurrent U-Net on several tasks, including hand segmentation, retina vessel segmentation and road segmentation. Our approach consistently outperforms earlier and simpler approaches to recursive segmentation [21, 27, 41]. For retina vessel segmentation, it also outperforms the state-of-the-art method of [19] on the DRIVE [38] dataset, and for hand segmentation, the state-of-the-art RefinetNet-based method of [40] on several modern benchmarks [11, 4, 40]. As these publicly available hand segmentation datasets are relatively small, with at most 4.8K annotated images, we demonstrate the scalability of our approach, along with its applicability in a keyboard typing scenario, by introducing a larger dataset containing 12.5K annotated images. It is the one we used to produce the results shown in Fig. 1. Both the dateset and code are available at https://github.com/WeiWangTrento/Recurrent-U-Net.

Our contribution is therefore an effective recurrent approach to semantic segmentation that can operate in environments where the amount of training data and computational power are limited. It does not require more memory than the standard U-Net thanks to parameter sharing and does not require training datasets as large as other state-of-the-art networks do. It is practical for real-time application, reaching 55 frames-per-second (fps) to segment 230×306 images on an NVIDIA TITAN X with 12G memory. Furthermore, as shown in Fig. 1, we can trade some accuracy for speed by reducing the number of iterations. Finally, while we focus on resource-constrained applications, our model can easily be made competitive on standard benchmarks such as Cityscapes by modifying its backbone architecture. We will show that replacing the U-Net encoder by a VGG16 backbone yields performance numbers comparable to the state of the art on this dataset.

2. Related Work

Compact Semantic Segmentation Models. State-of-the-art semantic segmentation techniques [6, 17, 18, 46] rely on very deep networks, which makes them ill-suited in resource-constrained scenarios, such as real-time applications and when there are only limited amounts of training data. In such cases, more compact networks are preferable. Such networks fall under two main categories.

The first group features encoder-decoder architectures [33, 28, 2, 31, 25, 35, 12]. Among those, U-Net [33] has demonstrated its effectiveness and versatility on many tasks, in particular for biomedical image analysis where it remains a favorite. For example, a U-net like architecture was recently used to implement the flood-filling networks of [16] and to segment densely interwoven neurons and neurites in teravoxel-scale 3D electron-microscopy image stacks. This work took advantage of the immense amount of computing power that Google can muster but, even then, it is unlikely that this could have been accomplished with much heavier architectures.

The second type involves multi-branch structures [26, 44, 45] to fuse low-level and high-level features at different resolutions. These require careful design to balance speed against performance. By contrast, the U-Net relies on simpler skip connections and, thus, does not require a specific design, which has greatly contributed to its popularity.

Recurrent Networks for Segmentation. The idea of recurrent segmentation predates the deep learning era and was first proposed in AutoContext [39], and recurrent random forest [36]. It has inspired many recent approaches, including several that rely on deep networks. For example, in [21], the segmentation mask produced by a modified U-Net was passed back as input to it along with the original image, which resulted in a progressive refinement of the segmentation mask. Fig. 2(a) illustrates this approach. A similar one was followed in the earlier work of [24], where the resolution of the input image patch varied across the iterations of the refinement process.

Instead of including the entire network in the recursive procedure, a standard recurrent unit can be added at the output of the segmentation network, as shown in Fig. 2(b). This was done in [32] to iteratively produce individual segmentation masks for scene objects. In principle, such a convolutional recurrent unit [3, 27, 41] could also be applied for iterative segmentation of a single object and we will evaluate this approach in our experiments. We depart from this strategy by introducing gated recurrent units that encompass several U-Net layers. Furthermore, we leverage the previous segmentation results as input, not just the same image at every iteration.

Iterative refinement has also been used for pose estimation [30, 42, 22]. The resulting methods all involve consecutive modules to refine the predictions with a loss function evaluated on the output of each module, which makes them similar in spirit to the model depicted by Fig. 2(a). Unlike in our approach, these methods do not share the parameters across the consecutive modules, thus requiring more parameters and moving away from our aim to obtain a compact network. Furthermore, they do not involve RNN-inspired memory units to track the internal hidden state.

3. Method

We now introduce our novel recurrent semantic segmentation architecture. To this end, we first discuss the overall structure of our framework, and then provide the details of the recurrent unit it relies on. Finally, we briefly discuss the training strategy for our approach.

3.1. Recurrent U-Net

We rely on the U-Net architecture of [33] as backbone to our approach. As shown in Fig. 3(a), the U-Net has an encoder-decoder structure, with skip connections between the corresponding encoding and decoding layers that allow the network to retain low-level features for the final prediction. Our goal being to operate in resource-constrained environments, we want to keep the model relatively simple. We therefore rely on a U-Net design where the first convolutional unit has 8 feature channels, and, following the original U-Net strategy, the channel number doubles after every pooling layer in the encoder. The decoder relies on transposed convolutions to increase the model's representation power compared to bilinear interpolation. We use group-normalization [43] in all convolutional layers since we usually rely on very small batch sizes.

Our contributions are to integrate recursions on 1) the predicted segmentation mask s and 2) multiple internal states of the network. The former can be achieved by simply concatenating, at each recurrent iteration t, the previous segmentation mask s_{t-1} to the input image, and passing the resulting concatenated tensor through the network. For the latter, we propose to replace a subset of the encoding and decoding layers of the U-Net with a recurrent unit. Below, we first formalize this unit, and then discuss two variants of its internal mechanism.

To formalize our recurrent unit, let us consider the process at iteration t of the recurrence. At this point, the network takes as input an image x concatenated with the previously-predicted segmentation mask s_{t-1}. Let us then denote by e_t^ℓ the activations of the ℓ^{th} encoding layer, and by d_t^ℓ those of the corresponding decoding layer. Our recurrent unit takes as input e_t^ℓ, together with its own previous hidden tensor h_{t-1}, and outputs the corresponding activations d_t^ℓ, along with the new hidden tensor h_t. Note that, to mimic the computation of the U-Net, we use multiple encoding and decoding layers within the recurrent unit.

In practice, one can choose the specific level ℓ at which the recurrent unit kicks in. In Fig. 3 (b), we illustrate the whole process for $\ell = 3$. When $\ell = 0$, the entire U-Net is included in the recurrent unit, which then takes the concatenation of the segmentation mask and the image as input. Note that, for $\ell = 4$, the recurrent unit still contains several layers because the central portion of the U-Net in Fig. 3(a) corresponds to a convolutional *block*. In our experiments, we evaluate two different structures for the recurrent units, which we discuss below.

3.2. Dual-gated Recurrent Unit

As a first recurrent architecture, we draw inspiration from the Gated Recurrent Unit (GRU) [8]. As noted above, however, our recurrent unit replaces multiple encoding and decoding layers of the segmentation network. We therefore modify the equations accordingly, but preserve the underlying motivation of GRUs. Our architecture is shown in Fig. 3(c).

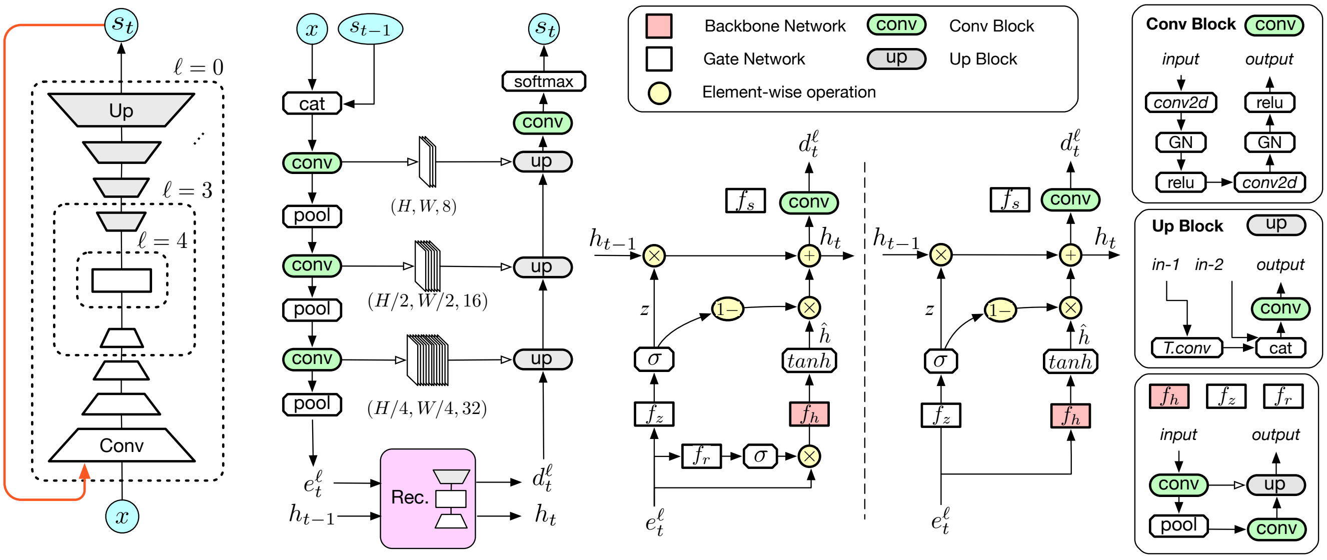
Specifically, at iteration t, given the activations e_t^ℓ and the previous hidden state h_{t-1}, we aim to produce a candidate update $\hat{h}$ for the hidden state and combine it with the previous one according to how reliable the different elements of this previous hidden state tensor are. To determine this reliability, we use an update gate defined by a tensor

$$z = \sigma(f_z(e_t^\ell)) \; , \tag{1}$$

where $f_z(\cdot)$ denotes an encoder-decoder network with the same architecture as the portion of the U-Net that we replace with our recurrent unit.

Similarly, we obtain the candidate update as

$$\hat{h} = tanh(f_h(r \odot e_t^\ell)) \; , \tag{2}$$



(a) Options Sketch

(b) R-UNet ($\ell = 3$)

(c) DRU

(d) SRU

Figure 3: **Recurrent UNet (R-UNet). (a)** As illustrated in Fig. 2(d), our model incorporates several encoding and decoding layers in a recurrent unit. The choice of which layers to englobe is defined by the parameter ℓ. **(b)** For $\ell = 3$, the recurrence occurs after the third pooling layer in the U-Net encoder. The output of the recurrent unit is then passed through three decoding up-convolution blocks. We design two different recurrent units, the Dual-gated Recurrent Unit (DRU) **(c)** and the Single-gated Recurrent Unit (SRU) **(d)**. They differ by the fact that the first one has an additional reset gate acting on its input. See the main text for more detail.

where $f_h(\cdot)$ is a network with the same architecture as $f_z(\cdot)$, but a separate set of parameters, $\odot$ denotes the element-wise product, and r is a reset tensor allowing us to mask parts of the input used to compute $\hat{h}$. It is computed as

$$r = \sigma(f_r(e_t^\ell)) \ , \tag{3}$$

where $f_r(\cdot)$ is again a network with the same encoder-decoder architecture as before.

Given these different tensors, the new hidden state is computed as

$$h_t = z \odot h_{t-1} + (1 - z) \odot \hat{h} \ . \tag{4}$$

Finally, we predict the output of the recurrent unit, which corresponds to the activations of the ℓ^{th} decoding layer as

$$d_t^\ell = f_s(h_t) \ , \tag{5}$$

where, as shown in Fig. 3(c), $f_s(\cdot)$ is a simple convolutional block. Since it relies on two gates, r and z, we dub this recurrent architecture Dual-gated Recurrent Unit (DRU). One main difference with GRUs is the fact that we use multi-layer encoder-decoder networks in the inner operations instead of simple linear layers. Furthermore, in contrast to GRUs, we do not directly make use of the hidden state h_{t-1} in these inner computations. This allows us not to have to increase the number of channels in the encoding and decoding layers compared to the original U-Net. Nevertheless, the hidden state is indirectly employed, since, via the recursion, e_t^ℓ depends on d_{t-1}^ℓ, which is computed from h_{t-1}.

3.3. Single-Gated Recurrent Unit

As evidenced by our experiments, the DRU described above is effective at iteratively refining a segmentation. However, it suffers from the drawback that it incorporates three encoder-decoder networks, which may become memory-intensive depending on the choice of ℓ. To decrease this cost, we therefore introduce a simplified recurrent unit, which relies on a single gate, thus dubbed Single-gated Recurrent Unit (SRU).

Specifically, as illustrated in Fig. 3(d), our SRU has a structure similar to that of the DRU, but without the reset tensor r. As such, the equations remain mostly the same as above, with the exception of the candidate hidden state, which we now express as

$$\hat{h} = tanh(f_h(e_t^\ell)) \ . \tag{6}$$

This simple modification allows us to remove one of the encoder-decoder networks from the recurrent unit, which, as shown by our results, comes at very little loss in segmentation accuracy.

3.4. Training

To train our recurrent U-Net, we use the cross-entropy loss. More specifically, we introduce supervision at each iteration of the recurrence. To this end, we write our overall loss as

$$L = \sum_{t=1}^{N} w_t L_t, \tag{7}$$

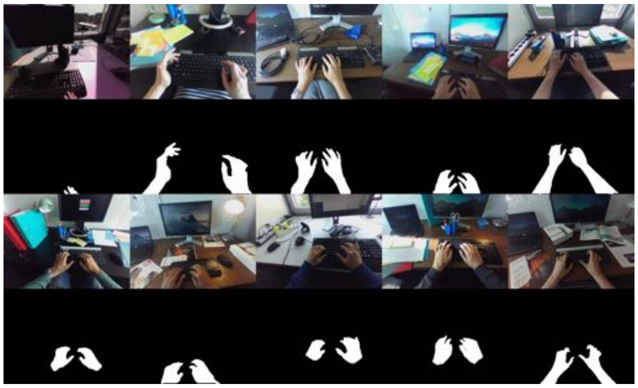


Figure 4: **Keyboard Hand (KBH) dataset.** Sample images featuring diverse environmental and lighting conditions, along with associated ground-truth segmentations.

where N represents the number of recursions, set to 3 in this paper, and L_t denotes the cross-entropy loss at iteration t, which is weighted by w_t.

$$w_t = \alpha^{N-t}. \tag{8}$$

The weight, by setting $\alpha \leq 1$, increases monotonically with the iterations. In our experiments, we either set $\alpha = 1$, so that all iterations have equal importance, or $\alpha = 0.4$, thus encoding the intuition that we seek to put more emphasis on the final prediction. A study of the influence of α is provided in supplementary material, where we also discuss our training protocol in detail.

4. Experiments

We compare the two versions of our Recurrent U-Net against the state of the art on several tasks including hand segmentation, retina vessel segmentation and road delineation. The hyper-parameters of our models were obtained by validation, as discussed in the supplementary material. We further demonstrate that the core idea behind our idea also applies to non-resource-constrained scenarios, such as Cityscapes, by increasing the size of the U-Net encoder.

4.1. Datasets.

Hands. We report the performance of our approach on standard hand-segmentation benchmarks, such as GTEA [11], EYTH [40], EgoHand [4], and HOF [40]. These, however, are relatively small, with at most 4,800 images in total, as can be seen in Table 1. To evaluate our approach on a larger dataset, we therefore acquired our own. Because this work was initially motivated by an augmented virtuality project whose goal is to allow someone to type on a keyboard while wearing a head-mounted display, we asked 50 people to type on 9 keyboards while wearing an HTC Vive [1]. To make this easier, we created a mixed-reality application to allow the users to see both the camera view and a virtual browser showing the text being typed. To ensure diversity, we varied the keyboard types, lighting conditions, desk colors, and objects lying on them, as can be seen in Fig. 4. We provide additional details in Table 2.

	Resolution		# Images			
Dataset	Width	Height	Train	Val.	Test	Total
KBH (Ours)	230	306	2300	2300	7936	12536
EYTH [40]	216	384	774	258	258	1290
HOF [40]	216	384	198	40	62	300
EgoHand [4]	720	1280	3600	400	800	4800
GTEA[11]	405	720	367	92	204	663

Table 1: Hand-segmentation benchmark datasets.

(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

Table 2: Properties of our new KBH dataset.

We then recorded 161 hand sequences with the device's camera. We split them as 20/ 20/ 60% for train/ validation/ test to set up a challenging scenario in which the training data is not overabundant and to test the scalability and generalizability of the trained models. We guaranteed that the same person never appears in more than one of these splits by using people's IDs during partitioning. In other words, our splits resulted in three groups of 30, 30, and 101 separate videos, respectively. We annotated about the same number of frames in each one of the videos, resulting in a total of 12,536 annotated frames.

Retina Vessels. We used the popular DRIVE dataset [38]. It contains 40 retina images used for making clinical diagnoses, among which 33 do not show any sign of diabetic retinopathy and 7 show signs of mild early diabetic retinopathy. The images have been divided into a training and a test set with 20 images for each set.

Roads. We used the Massachusetts Roads dataset [20]. It is one of the largest publicly available collections of aerial road images, containing both urban and rural neighborhoods, with many different kinds of roads ranging from small paths to highways. The data is split into 1108 training and 49 test images, one of which is shown in Fig. 6.

Urban landscapes. We employed the recent Cityscapes dataset. It is a very challenging dataset with high-resolution 1024×2048 images. It has 5,000 finely annotated images which are split into training/validation/test sets with 2975/500/1525 images. 30 classes are annotated, and 19 of them are used in training and testing.

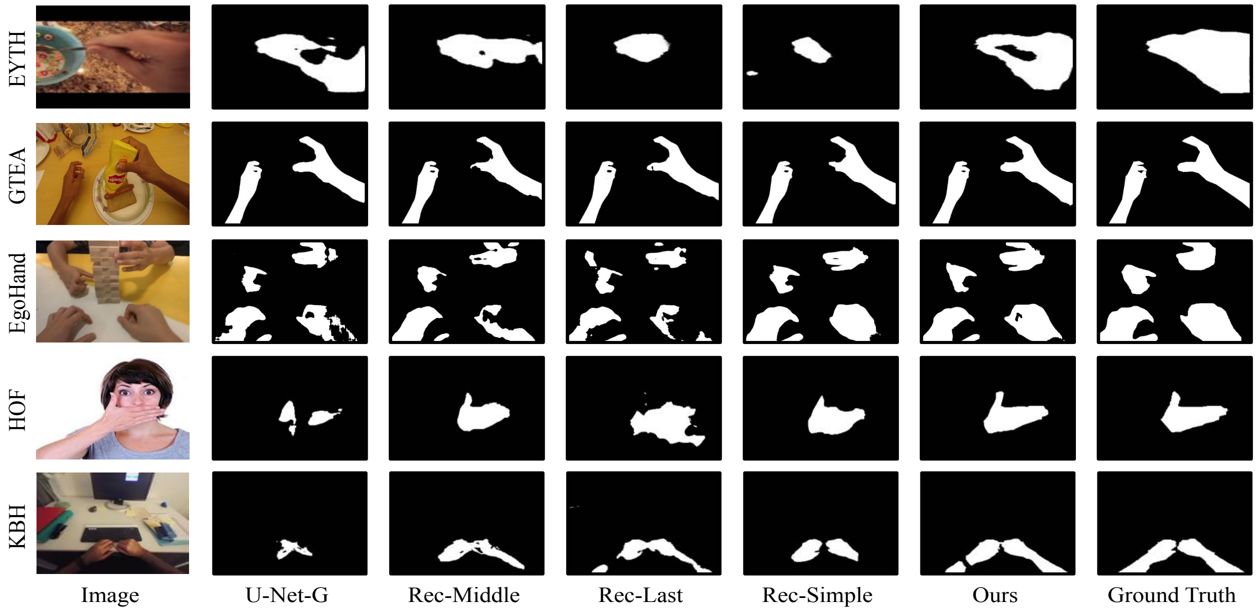


Figure 5: **Example predictions on hand segmentation datasets.** Note that our method yields accurate segmentations in diverse conditions, such as with hands close to the camera, multiple hands, hands over other skin regions, and low contrast images in our KBH dataset. By contrast, the baselines all fail in at least one of these scenarios. Interestingly, our method sometimes yields a seemingly a more accurate segmentation than the ground-truth ones. For example, in our EYTH result at the top, the gap between the thumb and index finger is correctly found whereas it is missing from the ground truth. Likewise, for KBH at the bottom, the watch band is correctly identified as not being part of the arm even though it is labeled as such in the ground truth.

4.2. Experimental Setup

Baselines. We refer to the versions of our approach that rely on the dual gated unit of Section 3.2 and the single gated unit of Section 3.3 as *Ours-SRU* and *Ours-DRU*, respectively, with, e.g., *Ours-SRU*(3) denoting the case where $\ell = 3$ in Fig. 3. We compare them against the state-of-the-art model for each task, i.e., *RefineNet* [40] for hand segmentation, [19] for retina vessel segmentation and [21] for road delineation, the general purpose DeepLab V3+ [6], the real-time ICNet [45], and the following baselines.

- ***U-Net*-B** and ***U-Net*-G** [33]. We treat our U-Net backbone by itself as a baseline. *U-Net*-B uses batch-normalization and *U-Net*-G group-normalization. For a fair comparison, they, *Ours-SRU*, *Ours-DRU*, and the recurrent baselines introduced below all use the same parameter settings.
- ***Rec-Last***. It has been proposed to add a recurrent unit after a convolutional segmentation network to process sequential data, such as video [27]. The corresponding *U-Net*-based architecture can be directly applied to segmentation by inputing the same image at all time steps, as shown in Fig. 2(b). The output then evolves as the hidden state is updated.
- ***Rec-Middle***. Similarly, the recurrent unit can replace the bottleneck between the U-Net encoder and decoder, instead of being added at the end of the network. This has been demonstrated to handle volumetric data [41]. Here we test it for segmentation. The hidden state then is of the same size as the inner feature backbone, that is, 128 in our experimental setup.
- ***Rec-Simple*** [21]. We perform a recursive refinement process, that is, we concatenate the segmentation mask with the input image and feed it into the network. Note that the original method of [21] relies on a VGG-19 pre-trained on ImageNet [37], which is far larger than our *U-Net*. To make the comparison fair, we therefore implement this baseline with the same U-Net backbone as in our approach.

Scaling Up using Pretrained Deep Networks as Encoder While our goal is resource-constrained segmentation, our method extends to the general setting. In this case, to further boost its performance, we replace the U-Net encoder with a pretrained VGG-16 backbone. This process is explained in the supplementary material. We refer to the corresponding models as U-Net-VGG16 and DRU-VGG16.

Metrics. We report the mean intersection over union (mIoU), mean recall (mRec) and mean precision (mPrec).

4.3. Comparison to the State of the Art

We now compare the two versions of our approach to the state of the art and to the baselines introduced above on the tasks of hand segmentation, retina vessel segmentation and road delineation. We split the methods into the light ones and the heavy ones. The light models contain fewer parameters and are trained from scratch, whereas the heavy ones use a pretrained deep model as backbone.

	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.

Hands. As discussed in Section 4.1, we tested our approach using 4 publicly available datasets and our own large-scale one. We compare it against the baselines in Table 3 quantitatively and in Fig. 5 qualitatively.

Overall, among the light models, the recurrent methods usually outperform the one-shot ones, *i.e*, ICNet [45] and *U-Net*. Besides, among the recurrent ones, *Ours-DRU*(4) and *Ours-SRU*(0) clearly dominate with *Ours-DRU*(4) usually outperforming *Ours-SRU*(0) by a small margin. Note that, even though *Ours-DRU*(4) as depicted by Fig. 3(a) looks superficially similar to *Rec-Middle*, they are quite different because *Ours-DRU* takes the segmentation mask as input and relies on our new DRU gate, as discussed at the end of Section 3.1 and in Section 3.2. To confirm this, we evaluated a simplified version of *Ours-DRU*(4) in which we removed the segmentation mask from the input. The validation mIOU on EYTH decreased from 0.836 to 0.826 but remained better than that of *Rec-Middle* which is 0.814.

Note that *Ours-DRU*(4) is better than the heavy *RefineNet* model on 4 out of the 5 datasets, despite *RefineNet* representing the current state of the art. The exception is HOF, and we believe that this can be attributed to HOF being the smallest dataset, with only 198 training images. Under such conditions, *RefineNet* strongly benefits from exploiting a ResNet-101 backbone that was pre-trained on PASCAL person parts [7], instead of training from scratch as we do. This intuition is confirmed by looking at the results of our DRU-VGG16 model, which, by using a pretrained deep backbone, yields the overall best performance.

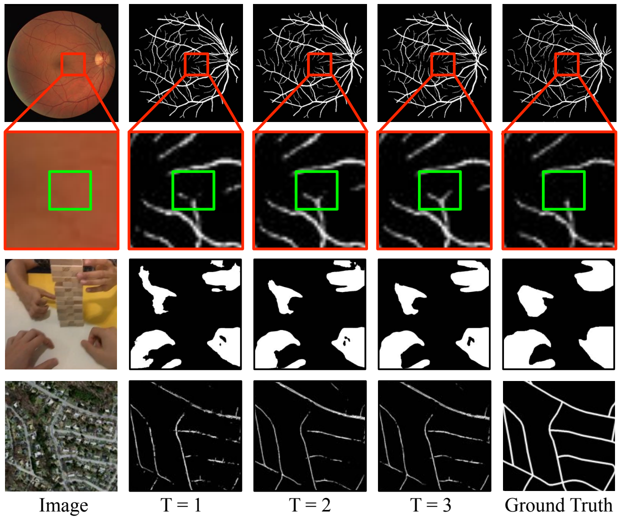


Figure 6: **Recursive refinement.** Retina, hand and road images; segmentation results after 1, 2, and 3 iterations; ground truth. Note the progressive refinement and the holes of the vessels, hands and roads being filled recursively. It is worth pointing out that even the tiny vessel branches in the retina which are ignored by the human annotators could be correctly segmented by our algorithm. Better viewed in color and zoom in.

Model Performance, Size and Speed. Table 3 shows that DRU-VGG16 outperforms *Ours-DRU*, e.g., by 0.02 mIoU points on KBH. This, however, comes at a cost. To be precise, DRU-VGG16 has 41.38M parameters. This is 100 times larger than *Ours-DRU*(4), which has only 0.36M pa-

	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**

Table 4: **Retina vessel segmentation results.**

rameters. Moreover, DRU-VGG16 runs only at 18 fps, while *Ours-DRU*(4) reaches 61 fps. This makes DRU-VGG16, and the other heavy models, ill-suited to embedded systems, such as a VR camera, while *Ours-DRU* can more easily be exploited in resource-constrained environments.

Retina Vessels. We report our results in Table 4. Our DRU yields the best mIOU, mPrec and mF1 scores. Interestingly, on this dataset, it even outperforms the larger DRU-VGG16 and DeepLab V3+, which performs comparatively poorly on this task. This, we believe, is due to the availability of only limited data, which leads to overfitting for such a very deep network. Note also that retina images significantly differ from the ImageNet ones, thus reducing the impact of relying on pretrained backbones. On this dataset, [19] constitutes the state of the art, reporting an F1 score on the vessel class only of 0.822. According to this metric, *Ours-DRU*(4) achieves 0.92, thus significantly outperforming the state of the art.

Roads. Our results on road segmentation are provided in Table 5. We also outperform all the baselines by a clear margin on this task, with or without ImageNet pretraining. In particular, Ours-DRU(4) yields an mIoU 8 percentage point (pp) higher than U-Net-G, and DRU-VGG16 5pp higher than U-Net-VGG16. This verifies that our recurrent strategy helps. Furthermore, Ours-DRU(4) also achieves a better performance than DeepLab V3+ and U-Net-VGG16. Note that, here, we also report two additional metrics: Precision-recall breaking point (P/R) and F1-score. The cutting threshold for all metrics is set to 0.5 except for P/R. For this experiment, we did not report the results of *U-Net*-B because *U-Net*-G is consistently better.

Note that a P/R value of 0.778 has been reported on this dataset in [21]. However, this required using an additional topology-aware loss and a *U-Net* much larger than ours, that is, based on 3 layers of a VGG19 pre-trained on ImageNet. Rec-Simple duplicates the approach of [21] without the topology-aware loss and with the same *U-Net* as Ours-DRU. Their mIoU of 0.723, inferior to ours of 0.757, shows our approach to recursion to be beneficial.

	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**

Table 5: **Road segmentation results.**

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

Table 6: **Cityscapes Validation Set with Resolution** 1024×2048**.** ×2 indicates that we doubled the number of channels in the U-Net backbone. Note that, for our method, we do not use multi-scaling or horizontal flips during inference.

Urban landscapes. The segmentation results on the Cityscapes validation set are shown in Table 6. Note that *Ours-DRU* is consistently better than U-Net-G and than the best recurrent baseline, *i.e.*, Rec-Last. Furthermore, doubling the number of channels of the U-Net backbone increases accuracy, and so does using a pretrained VGG-16 as encoder. Ultimately, our DRU-VGG16 model yields comparable accuracy with the state-of-the-art DeepLab V3 one, despite its use of a ResNet101 backbone.

5. Conclusion

We have introduced a novel recurrent *U-Net* architecture that preserves the compactness of the original one, while substantially increasing its performance. At its heart is the fact that the recurrent units encompass several encoding and decoding layers of the segmentation network. In the supplementary material we demonstrate it running in real-time on a virtual reality device. We also introduced a new hand segmentation dataset that is larger than existing ones.

In future work, we will extend our approach of recurrent unit to other backbones than *U-Net* and to multi-scale recurrent architectures.

Acknowledgements

This work was supported in part by the Swiss Innovation Agency (Innosuisse) and by the Swiss National Science Foundation.

References

[1] HTC Vive Virtual Reality Toolkit. https://www.vive.com/. 5

[2] V. Badrinarayanan, A. Kendall, and R. Cipolla. Segnet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation. *arXiv Preprint*, 2015. 2

[3] Nicolas Ballas, Li Yao, Chris Pal, and Aaron Courville. Delving deeper into convolutional networks for learning video representations. *International Conference on Learning Representations*, 2016. 2, 3

[4] Sven Bambach, Stefan Lee, David J Crandall, and Chen Yu. Lending a hand: Detecting hands and recognizing activities in complex egocentric interactions. In *International Conference on Computer Vision*, pages 1949–1957, 2015. 2, 5, 7

[5] L. Chen, G. Papandreou, F. Schroff, and H. Adam. Rethinking Atrous Convolution for Semantic Image Segmentation. *arXiv Preprint*, abs/1706.05587, 2017. 1, 8

[6] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation. *arXiv Preprint*, abs/1802.02611, 2018. 1, 2, 6, 7, 8, 12

[7] Xianjie Chen, Roozbeh Mottaghi, Xiaobai Liu, Sanja Fidler, Raquel Urtasun, and Alan Yuille. Detect what you can: Detecting and representing objects using holistic models and body parts. In *Conference on Computer Vision and Pattern Recognition*, 2014. 7

[8] Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv Preprint*, 2014. 3

[9] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Conference on Computer Vision and Pattern Recognition*, 2016. 1

[10] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A Large-Scale Hierarchical Image Database. In *Conference on Computer Vision and Pattern Recognition*, 2009. 1

[11] Alireza Fathi, Ali Farhadi, and James M Rehg. Understanding egocentric activities. In *International Conference on Computer Vision*, pages 407–414, 2011. 2, 5, 7

[12] Damien Fourure, Rémi Emonet, Elisa Fromont, Damien Muselet, Alain Tremeau, and Christian Wolf. Residual conv-deconv grid network for semantic segmentation. In *British Machine Vision Conference*, 2017. 2

[13] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. 12

[14] K. He, X. Zhang, R. Ren, and J. Sun. Delving Deep into Rectifiers: Surpassing Human-Level Performance on Imagenet Classification. In *International Conference on Computer Vision*, 2015. 12

[15] K. He, X. Zhang, S. Ren, and J. Sun. Deep Residual Learning for Image Recognition. In *Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016. 1

[16] Michał Januszewski, Jörgen Kornfeld, Peter H Li, Art Pope, Tim Blakely, Larry Lindsey, Jeremy Maitin-Shepard, Mike Tyka, Winfried Denk, and Viren Jain. High-precision automated reconstruction of neurons with flood-filling networks. *Nature methods*, 15(8):605, 2018. 2

[17] G. Lin, A. Milan, C. Shen, and I. Reid. Refinenet: Multi-path refinement networks for high-resolution semantic segmentation. In *Conference on Computer Vision and Pattern Recognition*, 2017. 1, 2

[18] J. Long, E. Shelhamer, and T. Darrell. Fully Convolutional Networks for Semantic Segmentation. In *Conference on Computer Vision and Pattern Recognition*, 2015. 1, 2

[19] Kevis-Kokitsi Maninis, Jordi Pont-Tuset, Pablo Arbeláez, and Luc Van Gool. Deep retinal image understanding. In *Conference on Medical Image Computing and Computer Assisted Intervention*, 2016. 2, 6, 8

[20] V. Mnih. *Machine Learning for Aerial Image Labeling*. PhD thesis, University of Toronto, 2013. 5

[21] A. Mosinska, P. Marquez-Neila, M. Kozinski, and P. Fua. Beyond the Pixel-Wise Loss for Topology-Aware Delineation. In *Conference on Computer Vision and Pattern Recognition*, 2018. 2, 6, 7, 8

[22] Alejandro Newell, Kaiyu Yang, and Jia Deng. Stacked hourglass networks for human pose estimation. In *European Conference on Computer Vision*, 2016. 3

[23] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. Devito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic Differentiation in Pytorch. In *Advances in Neural Information Processing Systems*, 2017. 12

[24] P.O. Pinheiro and R. Collobert. Recurrent Neural Networks for Scene Labelling. In *International Conference on Machine Learning*, 2014. 2, 3

[25] Tobias Pohlen, Alexander Hermans, Markus Mathias, and Bastian Leibe. Full-resolution residual networks for semantic segmentation in street scenes. In *Conference on Computer Vision and Pattern Recognition*, 2017. 2

[26] Rudra PK Poudel, Ujwal Bonde, Stephan Liwicki, and Christopher Zach. Contextnet: Exploring context and detail for semantic segmentation in real-time. *British Machine Vision Conference*, 2018. 2

[27] Rudra PK Poudel, Pablo Lamata, and Giovanni Montana. Recurrent fully convolutional neural networks for multi-slice mri cardiac segmentation. In *Reconstruction, Segmentation, and Analysis of Medical Images*, pages 83–94. Springer, 2016. 2, 3, 6, 7, 8

[28] Rudra PK Poudel, Stephan Liwicki, and Roberto Cipolla. Fast-scnn: Fast semantic segmentation network. *arXiv preprint arXiv:1902.04502*, 2019. 2

[29] Dale Purves, George J Augustine, David Fitzpatrick, Lawrence C Katz, Anthony-Samuel LaMantia, James O McNamara, and S Mark. Williams. Types of eye movements and their functions. In *Neuroscience*, 2011. 2

[30] Varun Ramakrishna, Daniel Munoz, Martial Hebert, James Andrew Bagnell, and Yaser Sheikh. Pose machines: Articulated pose estimation via inference machines. In *European Conference on Computer Vision*, 2014. 3

[31] Eduardo Romera, José M Alvarez, Luis M Bergasa, and Roberto Arroyo. Erfnet: Efficient residual factorized convnet for real-time semantic segmentation. *IEEE Transactions on Intelligent Transportation Systems*, 19(1):263–272, 2018. 2

[32] Bernardino Romera-Paredes and Philip Hilaire Sean Torr. Recurrent instance segmentation. In *European Conference on Computer Vision*, pages 312–329, 2016. 3

[33] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Conference on Medical Image Computing and Computer Assisted Intervention*, 2015. 1, 2, 3, 6, 7, 8

[34] C. Rother, V. Kolmogorov, and A. Blake. "GrabCut" - Interactive Foreground Extraction Using Iterated Graph Cuts. In *ACM SIGGRAPH*, pages 309–314, 2004. 12

[35] Shreyas Saxena and Jakob Verbeek. Convolutional neural fabrics. In *Advances in Neural Information Processing Systems*, 2016. 2

[36] Jamie Shotton, Matthew Johnson, and Roberto Cipolla. Semantic texton forests for image categorization and segmentation. In *Conference on Computer Vision and Pattern Recognition*, 2008. 2

[37] K Simonyan and A. Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *International Conference on Learning Representations*, 2015. 6

[38] J.J. Staal, M.D. Abramoff, M. Niemeijer, M.A. Viergever, and B. van Ginneken. Ridge based vessel segmentation in color images of the retina. *IEEE Transactions on Medical Imaging*, 23(4):501–509, 2004. 2, 5

[39] Z. Tu and X. Bai. Auto-Context and Its Applications to High-Level Vision Tasks and 3D Brain Image Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2009. 2

[40] Aisha Urooj and Ali Borji. Analysis of hand segmentation in the wild. In *Conference on Computer Vision and Pattern Recognition*, 2018. 2, 5, 6, 7

[41] Sepehr Valipour, Mennatullah Siam, Martin Jagersand, and Nilanjan Ray. Recurrent fully convolutional networks for video segmentation. In *IEEE Winter Conference on Applications of Computer Vision*, 2017. 2, 3, 6, 7, 8

[42] Shih-En Wei, Varun Ramakrishna, Takeo Kanade, and Yaser Sheikh. Convolutional pose machines. In *Conference on Computer Vision and Pattern Recognition*, 2016. 3

[43] Yuxin Wu and Kaiming He. Group normalization. In *European Conference on Computer Vision*, 2018. 3

[44] Changqian Yu, Jingbo Wang, Chao Peng, Changxin Gao, Gang Yu, and Nong Sang. Bisenet: Bilateral segmentation network for real-time semantic segmentation. In *European Conference on Computer Vision*, 2018. 2

[45] Hengshuang Zhao, Xiaojuan Qi, Xiaoyong Shen, Jianping Shi, and Jiaya Jia. Icnet for real-time semantic segmentation on high-resolution images. In *European Conference on Computer Vision*, pages 405–420, 2018. 1, 2, 6, 7, 8, 12

[46] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia. Pyramid Scene Parsing Network. In *Conference on Computer Vision and Pattern Recognition*, 2017. 1, 2