Investigating Semantic Segmentation Models to Assist Visually Impaired People

Michael Villamizar, Olivier Canévet, and Jean-Marc Odobez

Idiap Research Institute, Switzerland

Abstract. In this work, we address the semantic segmentation task with the purpose of allowing visually impaired people to comprehend their environments. To this end, we study and leverage convolutional networks trained on public automotive datasets and a new egocentric infrared dataset collected in urban areas. Domain adaptation, efficiency and segmentation accuracy are the focus of our study.

1 Introduction

Recent advances in deep learning have led to significant progress in fields like computer vision and natural language processing, resulting in highly effective algorithms. Today, deep networks are widely used in computer vision for tasks like object recognition and image segmentation [3,11]. These successes are largely due to the use of massive datasets and complex network models.

However, deploying these networks on small devices is challenging because they need to be lightweight, efficient, and low in complexity to operate with limited resources. While some models, like MobileNets [7], offer efficient solutions, their performance is often limited. Additionally, it is also common for many networks to be available in different variants with varying model sizes [3, 8, 10].

In this article, we focus on the 2D semantic image segmentation task [2, 4]. Specifically, our goal is to develop a perception system for visually impaired individuals, enhancing their understanding of their surroundings through a sound-based HCI interface. This system aims to increase their independence in activities such as navigating outdoors. While this work focuses on semantic segmentation, a crucial component of this challenging endeavor, ongoing and future research will extend to obstacle detection and 3D scene analysis using depth data.

In our problem, we face two main problems related to domain adaptation. The first one is that, due to both goals to allow the system to work with limited illuminations and data bandwidth limitations on the commercial device, the acquired images are mainly built from infrared images, which differs from the data available in traditional semantic segmentation datasets. Secondly, our application required data acquired at breast levels and from walking people, which contrasts with available datasets which are more oriented towards automotive data shot from cars rather than from pedestrian areas.

Hence, to address our problem, we have followed the following strategy. We investigate the use of several state-of-the-art deep networks known for their performance and efficiency. We concentrated our efforts on fully convolutional networks due to their ease of implementation on embedded devices and their

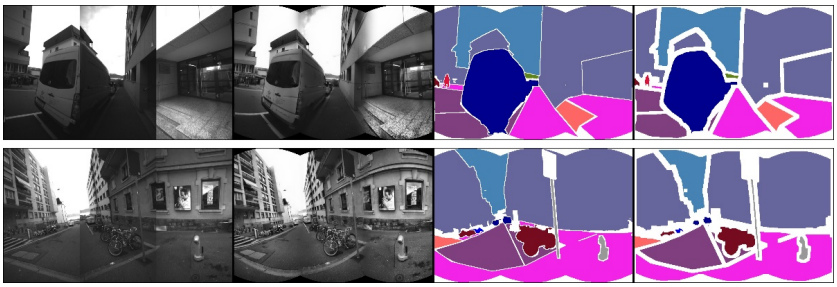

Fig. 1: Our dataset. First column: infrared camera images collected by the device and stacked horizontally. Second column: panoramic images resulting from cylindrical projection and multi-band stitching. Third column: fine semantic segmentation annotations (13 classes). Fourth column: coarse annotations. Annotations in white are not used during training and evaluation.

efficiency with high-resolution images [8, 10]. To train these models, we first leverage existing datasets (Cityscapes [5] and UrbanSyn [6]) to benchmark the different models and pretrain them.

In a second step, a large dataset of domain data was collected by visually impaired people while they were walking indoors and outdoors in urban settings. This egocentric dataset was conceived for capturing typical situations that they face when navigating outdoor (large variability in sidewalk diversity and geometry, street objects, etc). The dataset was then annotated relatively crudely with different semantic categories that are pertinent to users and for the task, such as ground, sidewalk, stairs, pedestrian crossing, etc.

2 Datasets

This section presents the three datasets that we used to train and assess the network architectures. The first two datasets are the Cityscapes dataset [5] and the UrbanSyn dataset [6]. They are public datasets that we use to assess and pretrain the networks. The last one is our dataset. It comprises panoramic images in urban settings, and is used to finetune the network for our task and need.

Cityscapes dataset [5]: it is an egocentric and driving-view image dataset with high-resolution images (2048x1024 pixels). This dataset has two sets of images. The first one has 5000 images with fine annotations, and the other has 20000 images with coarse ones. Each image has a semantic segmentation annotation (masks) using 19 semantic classes, such as cars or buildings. Commonly, the fine annotations are used for training and evaluation, while the coarse annotations are for pretraining or extending the training set. Specifically, for fine annotations, 2975, 500, and 1525 images are used for training, validation, and testing.

UrbanSyn dataset [6]: it is a synthetic dataset of 7539 images with a resolution of 2048×1024 that was created artificially following the Cityscapes format. Each image has a precise semantic annotation using the Cityscapes classes. Also, it has depth data and bounding box annotations. In this work, we only use the color images and the semantic annotations to enlarge the pretraining data.

Our dataset: it corresponds to 3012 panoramic and infrared images extracted from 51 videos recorded in the cities of Lausanne and Barcelona (around 60 frames were sampled and labeled per video). Some examples are shown in Fig. 1.

The images were collected using three Realsense cameras placed on a harness that was worn by the user (around shoulders) during walks in some urban areas like metro stations, parks, and streets. These camera images (infrared images) were then processed and stitched to obtain the panoramas (1254x848 pixels). Specifically, cylindrical projection was applied to each camera image, and then the resulting images were stitched using multiband blending (Laplacian pyramid). The collected data also includes depth images; however, they are not used in this work and will be part of the future work for 3D analysis.

The dataset images were annotated manually for 13 semantic categories relevant to our task: road, sidewalk, pedestrian cross, building, stairs, car, bicycle, person, dog, street object, vegetation, ceiling, and sky. Unlike other datasets, pedestrian cross and stairs were annotated since they are important to inform users of safe and potential risky paths. Dogs were also labeled since some users can be accompanied by guide dogs. Similar to [5], coarse annotations were also computed by applying a morphological operator (erosion) to the manual-annotated segmentation masks (fine annotations), see Fig. 1. They allow to measure the precision of the semantic segmentation results. The degree of erosion was proportional to the size of the segments, being 3 pixels and 31 pixels the minimum and maximum value, respectively.

For training and evaluation, the images were split according to video IDs such that images from one video can only be in either the training set or the evaluation set. Specifically, 41, 4, and 6 videos are used for training, validation, and testing, which corresponds to 2340, 113, and 559 images, respectively.

3 Networks

Since the objective is to have an efficient segmentation system that can run in a small processing device, we explore different network architectures in the state of the art that have stood out by their good results and efficiency. Particularly, we consider the Mask2Former [3] and DeepLabV3+ [2] networks, and two customized networks based on EfficientNet [10] and ConvNeXt [8]. Mask2Former is a Transformer network, whereas the rest are fully convolutional ones.

Mask2Former [3]: Masked-attention Mask Transformer (Mask2Former) is a recent network architecture that can perform any image segmentation task. It consists of a pixel encoder (backbone), a pixel decoder, and a transformer decoder. The backbone is the MaskFormer network [4], while the decoder is a multi-scale deformable attention Transformer (MSDeformAttn [11]). The transformer decoder processes object queries and uses masked attention to extract localized features by constraining cross-attention within predicted mask regions. To deal with small objects, it uses multi-scale features from the pixel decoder into the transformer decoder. The output segmentation predictions (masks) are decoded from per-pixel embeddings with object queries. In our experiments, we evaluate three Mask2Former models: the *large*, the *small*, and the *tiny*.

DeepLabV3+ [2]: this network extends DeepLabV3 [1] by adding an efficient decoder to refine segmentation predictions. The encoder is the DeepLabV3 which consists of stacks of convolutional layers followed by an Atrous Spatial Pyramid

Pooling (ASPP) to extract features at arbitrary resolution and capture larger context in the input image. The decoder is simple and effective, comprising convolutional layers, skip connections, and upsampling operations until the final prediction is obtained at 1/4 of the input image resolution. In this work, the final prediction is upsampled (4x) by applying bilinear interpolation in order that it gets the image resolution. We use the DeepLabV3 model with the ResNet101.

E□cientUNet: similar to DeepLabV3+, this architecture is a U-Net [9] consisting of an encoder and a decoder with skip connections between them. As encoder, we use EfficientNet [10] which provides a family of models (B0 to B7), each with a particular combination of efficiency and accuracy. They were designed by uniformly scaling the models in all dimensions (depth, width, and resolution). We selected the B1 and B5 models for our experiments. In the decoder, convolutional layers and bilinear upsampling operations are used to compute features at different resolutions. These features are added with features from the encoder through skip connections. The output prediction has the input image resolution.

ConvUNeXt: it is similar to EfficientUNet, but we use ConvNeXt [8] as encoder and use light ConvNeXt blocks in the decoder block. ConvNeXt was proposed to modernize traditional convolutional networks (CNNs) by rethinking their design and have shown competitive performance with Transformers in terms of accuracy and complexity. In this work, and to seek efficiency, we consider two light ConvNeXt models in the encoder: the *tiny* and the *small* models. As in DeepLabV3, we added an ASPP block in the bottleneck of the U-Net to capture larger context in the input image, which is beneficial in segmentation tasks.

4 Experiments

Networks: for the encoder, we employ models pretrained in the ImageNet dataset to train DeepLabV3+, EfficientUNet, and ConvUNeXt. For the decoder, the convolutional blocks are trained from scratch by initiating their weights randomly. In this paper, Mask2Former models are utilized for comparison. They are not trained, and we use the available models pretrained on Cityscapes.

Training: all networks are trained using a batch size of 4 and for at least 30 epochs. Data augmentation is applied during training: random horizontal flipping, Gaussian blurring, random photometric changes (contrast, brightness), random image scaling (between 25% and 100%), and random cropping with a size of half of the input image (e.g 627x424 pixels for our dataset). We use the DeepLab loss that only backpropagates the errors associated to the pixels with the top-k crossentropy loss (hard example mining).

Testing: during inference, we operate on the whole image resolution.

Evaluation metric: we use the standard mIoU (mean Intersection-over-Union) to evaluate the network models for semantic segmentation.

Pretraining experiments: we train the networks (except Mask2Former) on Cityscapes using the fine annotations (2975 images). Once trained, they are evaluated in the validation set (500 images). Tab. 1 reports their semantic segmentation performance using mIoU for fine and coarse ground-truth annotations.

Network	# Parameters	Extra data	mIoU (fine)	mIoU (coarse)	FPS
Mask2Former (large)	215		82.9	88.4	1.5
Mask2Former (small)	68		82.2	88.0	2.5
Mask2Former (tiny)	47		81.8	87.2	2.9
DeepLabV3+	58		75.6	80.9	6.0
		✓	77.6	85.8	6.0
EfficientUNet (B5)	31		77.7	84.6	6.5
		✓	78.7	86.4	6.5
EfficientUNet (B1)	8		72.6	80.1	6.9
		✓	76.0	85.6	6.9
ConvUNeXt (small)	74		80.3	86.2	4.3
		✓	81.3	88.9	4.3
ConvUNeXt (tiny)	42		79.4	86.4	5.7
		✓	81.1	88.4	5.7

Table 1: Segmentation rates on the Cityscapes validation set. Parameters in Millions.

Network	From scratch		Pretrained		
	fine	coarse	fine	coarse	FPS
DeepLabV3+	74.2	77.8	81.2	84.5	12.9
EfficientUNet (B5)	75.6	79.3	82.9	86.1	13.9
EfficientUNet (B1)	71.2	74.4	71.8	74.3	17.8
ConvUNeXt (small)	82.7	86.6	84.6	87.7	9.6
ConvUNeXt (tiny)	81.4	84.5	85.0	88.5	12.8

Table 2: Performance rates (mIoU) on the test set of our dataset.

The table also shows the rates for the pretrained Mask2Former models, the efficiency (measured as frames per second in a RTX3090 GPU), and the number of parameters of each model. We also report the results for when the networks are trained with extra data, including Cityscapes data with coarse annotations, and the UrbanSyn data. In that case, the networks are trained with 30000 images.

In general, we see that using abundant data improves the accuracy of every network, especially the rates for ground-truth coarse annotations since most of the extended training data has coarse labels.

Mask2Former obtains the best performance using the large and small models, at the expense of the lowest speed and increased complexity which is not suitable for our embedded device. The lightest model, EfficientUNet (B1), performs on par with DeepLabV3+ despite being seven times smaller. Interestingly, EfficientUNet (B5) performs better and faster than DeepLabV3+. ConvUNeXt provides a good tradeoff between efficiency and accuracy. The performance of the tiny and small variants is comparable, with the former being quicker and smaller. Interestingly, the tiny model is competitive against the Mask2Former tiny model while being almost two times faster. These networks were designed to perform on par with Transformers [8], which is demonstrated here.

Finetuning experiments: the networks are finetuned in our dataset using the models pretrained above, with the extra data, and with the same training settings. In addition, we compare against training the networks from scratch (not pretraining). The results are shown in Tab. 2. It is clear that using pretraining greatly increases segmentation accuracy; this is most likely because the dataset is challenging and small, making feature learning from scratch more difficult.

Similarly to the results on Cityscapes, ConvUNeXt achieves the best accuracy scores, followed by EfficientUNet (B5) and DeepLabV3+. Surprisingly, the