LUND UNIVERSITY

# Data-Efficient Learning of Semantic Segmentation

Nilsson, David

2022

Data-Efficient Learning of Semantic Segmentation

# Data-Efficient Learning of Semantic Segmentation

by David Nilsson


LUND
UNIVERSITY

PhD Thesis
Thesis advisors: Prof. Cristian Sminchisescu, Dr. Carl Olsson
Faculty opponent: Prof. Atsuto Maki, KTH Royal Institute of Technology, Stockholm

To be presented, with the permission of the Faculty of Engineering at Lund University, for public criticism in the lecture hall MH:Hörmander at the Centre for Mathematical Sciences, Sölvegatan 18, Lund on Monday, the 13th of June 2022 at 10:15.

Organization
**LUND UNIVERSITY**

Centre for Mathematical Sciences, Sölvegatan 18, Lund
Box 118
SE–221 00 LUND
Sweden

Document name
**Doctoral thesis**

Date of presentation
2022-06-13

Sponsoring organization

Author(s)
David Nilsson

Title and subtitle
Data-Efficient Learning of Semantic Segmentation

Abstract

Semantic segmentation is a fundamental problem in visual perception with a wide range of applications ranging from robotics to autonomous vehicles, and recent approaches based on deep learning have achieved excellent performance. However, to train such systems there is in general a need for very large datasets of annotated images. In this thesis we investigate and propose methods and setups for which it is possible to use unlabelled data to increase the performance or to use limited application specific data to reduce the need for large datasets when learning semantic segmentation.

In the first paper we study semantic video segmentation. We present a deep end-to-end trainable model that uses propagated labelling information in unlabelled frames in addition to sparsely labelled frames to predict semantic segmentation. Extensive experiments on the CityScapes and CamVid datasets show that the model can improve accuracy and temporal consistency by using extra unlabelled video frames in training and testing.

In the second, third and fourth paper we study active learning for semantic segmentation in an embodied context where navigation is part of the problem. A navigable agent should explore a building and query for the labelling of informative views that increase the visual perception of the agent. In the second paper we introduce the embodied visual active learning problem, and propose and evaluate a range of methods from heuristic baselines to a fully trainable agent using reinforcement learning (RL) on the Matterport3D dataset. We show that the learned agent outperforms several comparable pre-specified baselines. In the third paper we study the embodied visual active learning problem in a lifelong setup, where the visual learning spans the exploration of multiple buildings, and the learning in one scene should influence the active learning in the next e.g. by not annotating already accurately segmented object classes. We introduce new methodology to encourage global exploration of scenes, via an RL-formulation that combines local navigation with global exploration by frontier exploration. We show that the RL-agent can learn adaptable behaviour such as annotating less frequently when it already has explored a number of buildings. Finally we study the embodied visual active learning problem with region-based active learning in the fourth paper. Instead of querying for annotations for a whole image, an agent can query for annotations of just parts of images, and we show that it is significantly more labelling efficient to just annotate regions in the image instead of the full images.

Key words
semantic segmentation, embodied learning, active learning, semantic video segmentation, computer vision, deep learning

Classification system and/or index terms (if any)

Supplementary bibliographical information

Language
English

ISSN and key title
1404-0034

ISBN
978-91-8039-283-9 (print)
978-91-8039-284-6 (pdf)

Recipient's notes

Number of pages
147

Price

Security classification

# Data-Efficient Learning of Semantic Segmentation

by David Nilsson


LUND
UNIVERSITY

**Funding information:** This work was supported in part by the European Research Council Consolidator grant SEED, CNCS-UEFISCDI PN-III-P4-ID-PCE-2016-0535 and PCCF-2016-0180, the EU Horizon 2020 Grant DE-ENIGMA, Swedish Foundation for Strategic Research (SSF) Smart Systems Program, as well as the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation.

Centre for Mathematical Sciences
Lund University
Box 118
SE-221 00 Lund
Sweden

www.maths.lth.se

Doctoral Thesis in Mathematical Sciences 2022:5
ISSN: 1404-0034

ISBN: 978-91-8039-283-9 (print)
ISBN: 978-91-8039-284-6 (pdf)
LUTFMA-1076-2022



Printed in Sweden by Media-Tryck, Lund University, Lund 2022

# Abstract

Semantic segmentation is a fundamental problem in visual perception with a wide range of applications ranging from robotics to autonomous vehicles, and recent approaches based on deep learning have achieved excellent performance. However, to train such systems there is in general a need for very large datasets of annotated images. In this thesis we investigate and propose methods and setups for which it is possible to use unlabelled data to increase the performance or to use limited application specific data to reduce the need for large datasets when learning semantic segmentation.

In the first paper we study semantic video segmentation. We present a deep end-to-end trainable model that uses propagated labelling information in unlabelled frames in addition to sparsely labelled frames to predict semantic segmentation. Extensive experiments on the CityScapes and CamVid datasets show that the model can improve accuracy and temporal consistency by using extra unlabelled video frames in training and testing.

In the second, third and fourth paper we study active learning for semantic segmentation in an embodied context where navigation is part of the problem. A navigable agent should explore a building and query for the labelling of informative views that increase the visual perception of the agent. In the second paper we introduce the embodied visual active learning problem, and propose and evaluate a range of methods from heuristic baselines to a fully trainable agent using reinforcement learning (RL) on the Matterport3D dataset. We show that the learned agent outperforms several comparable pre-specified baselines. In the third paper we study the embodied visual active learning problem in a lifelong setup, where the visual learning spans the exploration of multiple buildings, and the learning in one scene should influence the active learning in the next e.g. by not annotating already accurately segmented object classes. We introduce new methodology to encourage global exploration of scenes, via an RL-formulation that combines local navigation with global exploration by frontier exploration. We show that the RL-agent can learn adaptable behaviour such as annotating less frequently when it already has explored a number of buildings. Finally we study the embodied visual active learning problem with region-based active learning in the fourth paper. Instead of querying for annotations for a whole image, an agent can query for annotations of just parts of images, and we show that it is significantly more labelling efficient to just annotate regions in the image instead of the full images.

# List of publications

This thesis is based on the following publications:

I **Semantic Video Segmentation by Gated Recurrent Flow Propagation**

**David Nilsson**, Cristian Sminchisescu
*Computer Vision and Pattern Recognition (CVPR)*, Salt Lake City, USA, 2018.

**Author contributions:** DN and CS jointly came up with the main idea and wrote the paper. DN implemented everything and ran all experiments.

II **Embodied Visual Active Learning for Semantic Segmentation**

**David Nilsson**, Aleksis Pirinen, Erik Gärtner, Cristian Sminchisescu
*Association for the Advancement of Artificial Intelligence (AAAI)*, Virtual conference, 2021.

**Author contributions:** DN and CS jointly came up with the problem setup. DN wrote most of the code, but with significant help from AP and EG. DN ran most experiments. DN, EG and AP wrote most of the paper.

III **Embodied Learning for Lifelong Visual Perception**

**David Nilsson**, Aleksis Pirinen, Erik Gärtner, Cristian Sminchisescu
*arXiv preprint arXiv:2112.14084*

**Author contributions:** The authors jointly came up with the high level ideas. DN wrote most of the code specific for this paper and ran all experiments. DN and AP wrote the paper.

IV **Online Learning of Semantic Segmentation via Embodied Region-Based Active Learning**
**David Nilsson**, Erik Gärtner, Aleksis Pirinen, Cristian Sminchisescu
Manuscript in preparation.

**Author contributions:** The high level ideas came up during discussions among the authors of paper 2 and 3. DN wrote all of the code specific for this paper and ran all experiments. DN wrote most of the paper with feedback and edits from AP and EG.

All papers are reproduced with permission of their respective publishers.

# Acknowledgements

My first thanks goes to my supervisor Cristian Sminchisescu. He has guided me through this process and I am thankful for all the valuable advice, feedback and encouragement I have gotten over the years. I am also grateful for having been given the opportunity to do an internship with his group at Google Research in Zürich.

I also would like to thank my co-authors Aleksis Pirinen and Erik Gärtner. They were both very helpful and supportive during the second half of my PhD when we collaborated, and without them this thesis would not have been what it is.

I also had the great pleasure of working with the other closest members of the research group, namely Maria Priisalu, Ted Kronvall, Henning Petzka and Martin Trimmel.

The colleagues at the Centre for Mathematical Sciences at LTH also deserve thanks for providing a good work environment. A special thanks goes to my assistant supervisor Carl Olsson.

Last but certainly not least, I want to thank my parents Kristina and Bengt for always being supportive and believing in me.

# Popular Summary

Computer vision is about understanding and interpreting images. While it is often obvious to a human observing an image whether there is a human or car present, mathematically it is very complicated to replicate. An image is often represented as a grid of pixels with intensities ranging from 0 (black) to 255 (white), and to go from such a representation to understanding which objects are present is highly non-trivial. In recent years there have been drastic improvements in computer vision, in large part due to deep learning, which is a term for methods that use a neural network to *learn* to recognize objects in images. To train such a model, we need algorithms to adapt the model to the data, but the main ingredient is to have a lot of data. If such a model is to be trained to detect cars in an image, the typical requirement is that there are lots of images available where the cars are marked in every image, so that the model can learn from all the data. Annotating such images requires a lot of time and manual labor, and it can for some problems take up to 1.5 hours to annotate a single image. It is thus interesting and important to develop methods to reduce the dependency on large datasets, which can yield large savings in terms of both time and labor.

An example of where there is ample data, but annotating everything is highly redundant, is video data. Consecutive frames in a video often overlap significantly, so to ask a person to annotate every single frame of a video is probably not a good investment. It should be possible to annotate frames sparsely and still be able to train deep learning models on video data. Such an approach, to use all available data from sparsely labeled videos, is a part of this thesis. By carefully adapting the model to consider multiple adjacent frames, it is possible to learn recognition for video without annotating all frames, by using the rich temporal information present.

To train systems for image recognition, there is a need for a lot of manually annotated data, as shown on the right. This thesis concerns how to use data efficiently, so that less data needs to be annotated.

Another problem studied in this thesis is that of selecting which data to annotate in order to reduce the overall annotation effort. Since annotating images is costly and time-consuming, it is important that each annotated image adds some value. When training a model for image recognition it is important that the data is diverse and that not all examples are similar. It is studied how a room should be explored to see and collect the data to annotate, in order to learn accurate perception of the objects present in the scene. The problem here is not only to utilize the collected data, but to explore and navigate to gather the data itself. Several methods that automatically solve this problem are presented and compared, of which some are fully trainable via reinforcement learning. These methods have to explore and navigate, and for specific views query for annotations for all the objects.

# Populärvetenskaplig sammanfattning

Datorseende handlar om att tolka och förstå bilder. Det är oftast uppenbart för en människa som ser en bild om den exempelvis innehåller en människa eller en bil, men matematiskt är det mycket komplicerat. I en dator representeras en bild oftast som ett rutnät av pixlar med ljusintensiteter representerade mellan 0 (svart) och 255 (vit), och att gå från en sådan representation till att förstå vilka objekt som finns i bilden är icke-trivialt. Metoder för datorseende har de senaste åren förbättrats drastiskt. Detta har skett med hjälp av djupinlärning (deep learning), som är ett samlingsnamn för metoder som använder så kallade neurala nätverk som *lär sig* att känna igen objekt i bilder. För att få en sådan modell att lära sig krävs dels algoritmer för att anpassa modellen till data, men huvudingrediensen är ofta mycket data. Om en sådan modell ska tränas för att detektera bilar i bilder så krävs det väldigt många bilder där just bilen är markerad i varje bild för att få modellen att fungera bra. Detta kräver mycker tidskrävande manuellt arbete för att markera och ringa in det korrekta svaret i alla bilder, vilket för vissa problem kan ta så mycket som 1,5 timmar per bild. Det är alltså intressant och viktigt att utveckla metoder för att minska beroendet av mycket annoterad data, vilket kan ge stora tid- och kostnadsbesparingar.

Ett exempel där det finns mycket data men där det är överflödigt att annotera allt är video. Efterföljande bilder i en video har ofta stort överlapp så att be en person att annotera varje bild är troligen inte en bra investering. Det borde alltså gå att bara annotera ett fåtal bilder i en video och ändå kunna träna modeller för bildigenkänning. Just en sådan metod för att använda och utnyttja all data som finns i video, där det mesta inte är annoterat, är ett bidrag i denna avhandling. Genom att anpassa modellen så att den tar hänsyn till flera bilder i rad går det att effektivt lära sig bildigenkänning för video trots att alla bilder inte är annoterade.

För att träna system för bildigenkänning krävs stora mängder manuellt annoterad data så som till höger. Denna avhandling handlar om att använda data effektivt, så att en mindre mängd data behöver annoteras.

Ett annat problem som studeras i denna avhandling är vilken data som ska annoteras i syfte att minska den totala annoteringsbördan. Eftersom annotering är kostsamt och tidskrävande är det viktigt att det som annoteras tillför något. Ska man träna en modell för bildigenkänning vill man att datan ska vara varierande och inte att alla exempel är likadana. Det studeras hur ett rum ska utforskas och vilka delar och vyer i rummet som ska annoteras för att en modell ska lära sig känna igen alla objekt som finns i rummet. Problemet är inte bara att effektivt behandla den insamlade datan, utan även att utforska och navigera för att se och samla in själva datan. Det presenteras automatiska metoder för att navigera och för vissa särskilt relevanta vyer be en annoterare om att markera alla objekt i bilden, varav vissa lär sig via så kallad förstärkningsinlärning.

# Contents

# Chapter 1

# Introduction

## 1 Overview

This thesis covers the learning of semantic segmentation, which is the problem of assigning every pixel in an image to a semantic label such as e.g. wall, road, chair or car, as shown in fig. 1.1. It is a difficult fundamental problem in visual perception since it is not only required to recognize all objects in a scene and to label them, but also to accurately predict all boundaries. Furthermore, the viewing directions from which the objects are seen, as well as the object sizes in the images, can vary greatly for different instances of an object class. Methods for semantic segmentation are highly useful for scene understanding in applications and setups such as driver assistance, home assistant robots or drones.

In recent years, deep learning has proven very successful for a range of computer vision tasks including image classification [44, 32], semantic segmentation [16], and object detection [65]. Deep learning is a term used for neural networks that contain many layers stacked after one another, and for image data convolutional layers are commonly used in what are called convolutional neural networks (CNNs). For deep learning to work well, there is in general a need for very large annotated datasets, and the data collection and labelling requires extensive efforts. As an example, the ImageNet dataset [22] contains more than a million labelled images with one label per image. CityScapes [21] is a dataset consisting of street-view images containing $5000$ fully pixel-wise labelled images for semantic segmentation, where a single image is of size $1024 \times 2048$ and on average takes $1.5$ hours to annotate.

Since labelling is very expensive and time-consuming, a relevant research direction is to examine situations and methods for which it is possible to achieve high performance using less annotated data. One such approach is by using additional inputs, e.g. in the form of video. Since consecutive frames are highly overlapping, it is possible to only sparsely label such data

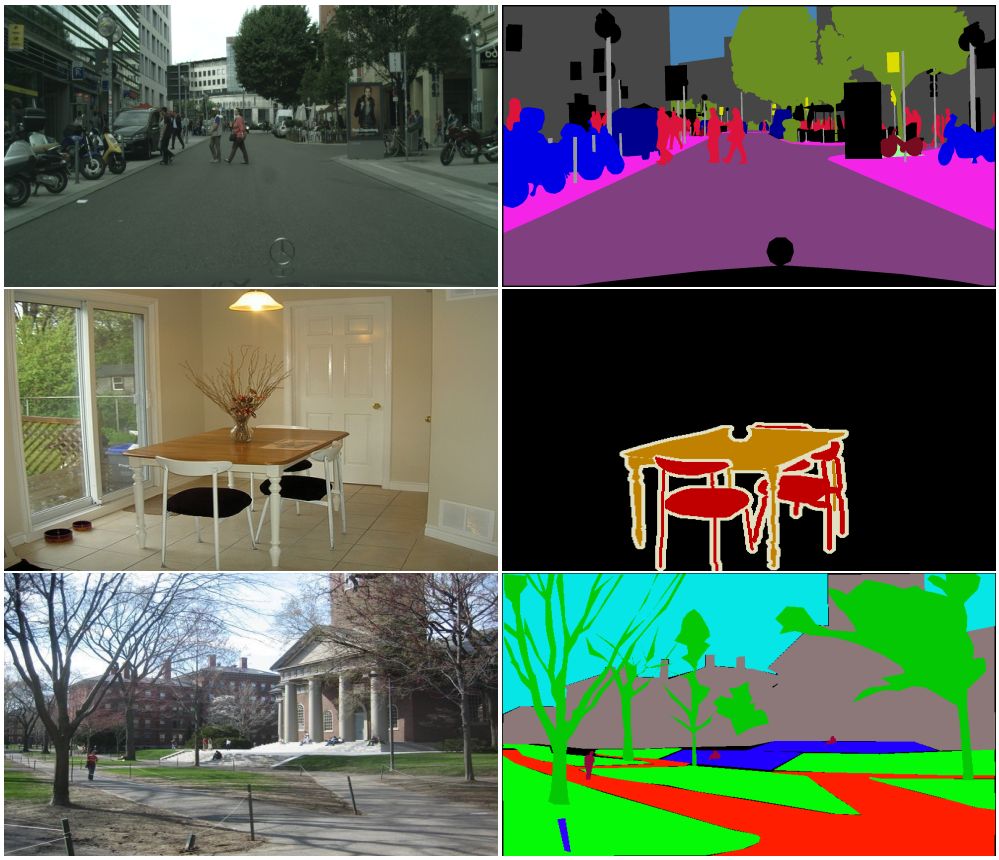

**Figure 1.1:** Three images and their corresponding semantic segmentations. Each pixel is classified as one specific class. The images are from three popular datasets showcasing the wide range of applications of semantic segmentation. From top to bottom: CityScapes [21], Pascal VOC [25] and ADE20K [91]. Each dataset has its own specific set of labels, and the colors correspond to semantic classes such as e.g. road, pedestrian and building in the first image. All pixels in an image do not necessarily belong to the given set of labels, as shown by the black pixels which are not assigned to any label.

and still obtain accurate performance. Learning with limited supervision is often referred to as weakly supervised learning. In weakly supervised learning, the annotations are noisy or of a lower granularity than the task at hand or just a subset of the data is labelled. Examples are bounding box supervision for segmentation problems [60], single-image supervision for video problems [10, 55, 28], or image-level supervision for object detection [59].

Another setup where limited task-specific data might suffice is if the perception task has a limited scope, e.g. as in the case where a robot explores and performs tasks in a single building. In such a case it is not necessary to obtain labels for all kinds of buildings, and instead it might be sufficient to just obtain a few, carefully selected, labels from the building in question. The question then is exactly how these images and labels should be selected to obtain as accurate perception as possible. How to select which data to annotate out of a large collection of

unlabelled data is the *active learning* problem [71].

In paper 1 [55] we study semantic video segmentation, and show how a semantic segmentation network trained to segment single images can be transformed to handle video streams. We show how to incorporate the temporal context in a principled manner and experimentally show increased segmentation accuracy and more temporally consistent and less flickering segmentation results. We develop an end-to-end trainable methodology which aligns the segmentation masks via optical flow, learns how to combine segmentations from different frames, and only uses sparsely annotated frames in a weakly supervised manner.

In paper 2 [56] we study active learning for semantic segmentation in an embodied context where navigation is part of the problem. The views to consider for annotations are not given, and an agent must actively explore to obtain them. We further task the agent to select which views that should be annotated. The agent should navigate and annotate in a manner which gives the best data to use for learning semantic segmentation of the scene, in a cost-efficient way requesting few annotations. We compare multiple methods ranging from pre-specified baselines to a fully trainable agent using reinforcement learning (RL) on the simulated photo-realistic Matterport3D [12] environment. This paper studies a problem setup that in many ways is a relaxation of the assumptions in paper 1. We do not assume a given video where a fixed frame is annotated, but instead an agent has to actively navigate to collect the video, and the agent must also select which frames that should be annotated in the video.

In paper 3 [57] we study lifelong learning for semantic segmentation in an embodied context. Here we extend paper 2 in two main directions. We enforce systematic exploration of a whole scene instead of just exploring locally for short episodes, by explicitly modelling exploration and incorporating frontier exploration into our RL task formulation. Furthermore we study the task in a lifelong manner where the agent can retain knowledge in one exploration episode to another. We show how the earlier learned perception affects an agent both in terms of exploration and active learning. Specifically, the agents do not request annotations as frequently when multiple scenes have already been explored.

In paper 4 we consider region-based embodied visual active learning. We consider how to use more fine-grained and hence more labelling efficient annotations than full images. Specifically, we task an agent exploring an environment to not just select which image to annotate, but also which region in the current image. We show that using region-based active learning is significantly more labelling efficient than using active learning with full images.

In conclusion, the common theme is that we consider setups and methods for which to reduce the labelling cost or using the annotated data more efficiently in order to learn semantic segmentation. In the paper on semantic video segmentation this was done by efficiently using unlabelled frames of a video. For the papers on embodied visual active learning, it was done by carefully considering what to annotate. These works all contribute towards data-efficient learning of semantic segmentation.

This introduction is organized as follows. We first give an introduction to semantic segmentation and describe modern approaches to the problem using convolutional neural networks. This is relevant for all papers. We then give an overview of tools for video analysis such as optical flow and RNNs. This is highly relevant for paper 1, and there is also relevance for paper 2, 3 och 4. Finally, we give an overview of embodied learning and reinforcement learning, which is relevant for paper 2, 3 and 4.

# 2 Semantic Segmentation

Semantic segmentation is the problem of assigning every pixel of an image to a specific label. Several example images are shown in fig. 1.1, where the set of classes include e.g. road, person or tree. All pixels belonging to a specific object need to be correctly detected, and the boundaries need to be sharp to delineate the object from the background and other objects. In contrast to image classification where one label per image should be predicted, this task is more complex since there can be multiple objects per image at varying scales, and the exact boundaries should also be predicted.

In fig. 1.1 we show several examples of semantic segmentation using images from three different datasets with diverse applications in mind. The first is from CityScapes [21] and is taken from a camera mounted on a car. Applications of this are driver assistance, and as a component for self-driving cars. The image from Pascal VOC shows an indoor scene with detections of chair and table, which would be useful for e.g. an indoor assistant robot. The bottom row is an image from ADE20K showing an aerial view of a scene. This could be useful for e.g. a drone delivering a package.

In recent years, there has been much work on semantic segmentation, and approaches based on convolutional neural networks (CNNs) have proven very successful [16, 89]. We will now turn to a brief description of CNNs and then describe how to use them for semantic segmentation and briefly review the state-of-the-art.

## 2.1 Introduction to Convolutional Neural Networks

In modern computer vision, convolutional neural networks (CNNs) are ubiquitous, and they are extensively used in all the papers of this thesis. CNNs are widely used when processing images. The basic idea is that multiple convolutional layers are stacked as in fig. 1.4. An image is used as input, and we transform the features in multiple steps using convolutional layers, where each convolution typically is followed by batch normalization and nonlinearities, e.g. relu. When training a CNN all the parameters, including the filter weights of all convolutional layers, are learned using e.g. gradient descent. The use of CNNs for com-

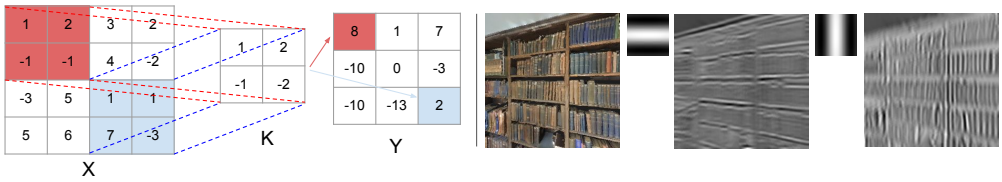


**Figure 1.2:** Illustration of computations in a convolutional layer. One the left we compute the scalar product between patches in the input $X$ and the kernel $K$. In this example the input patch marked red corresponds to the output $1 \cdot 1 + 2 \cdot 2 + (-1) \cdot (-1) + (-1) \cdot (-2) = 8$ and similarly for the one marked blue it is $1 \cdot 1 + 1 \cdot 2 + 7 \cdot (-1) + (-3) \cdot (-2) = 2$. On the right we show an image and the image convolved with the two filters shown, and we see that in the first case horizontal edges are detected and in the second that vertical edges are detected.

puter vision problems was popularized starting in 2012 when Krizhevsky et al. [44] showed that CNNs far outperformed earlier approaches for image classification on the large-scale dataset ImageNet [22]. It remains popular to use pre-trained networks on the ImageNet dataset so that parameters are initialized for image classification rather than randomly, and two examples are the widely used VGG [74] and ResNet [32] architectures. To go from image classification to other tasks such as semantic segmentation, it is needed to adapt the CNN architecture, and we will describe that in §2.2.

We will now describe the convolutional layer. The convolutional operator or layer, defined here on an image or feature map $X(i, j, c)$, where $i$ and $j$ are height and width indices and $c$ the channel, convoluted with a kernel $K(i, j, c)$ is defined as

$$Y(i, j) = \sum_{m,n,c} K(m, n, c)X(i + m, j + n, c) \tag{1.1}$$

Note that we use different signs than traditionally for convolutions (not $X(i-m, j-n, \cdot)$) and the feature channel indexed by c follows a different convention, following standard practice for CNNs [31]. We showed the formula for a convolution with a single kernel $K$. In practice multiple kernels are used, producing an output map of size $(H, W, C)$ where $C$ is the number of kernels, and $(H, W)$ is the spatial output size. We show the computations and the convolution of an image in fig. 1.2. All parameters of the kernel $K$ need to be trained.

In fig. 1.3 we show the receptive fields of every output of the convolutional layer as well as variations to the standard convolution, illustrated here in just one dimension. The extension to 2d image data is straight-forward by applying similar modifications as in the 1d case to the height and width dimensions independently. When padding the input, we simply append extra elements at the start and end of the input. This is commonly used to keep the shape of the output the same as the input. If we have a $3 \times 3$ convolution with padding 1 in both dimensions the output will have size $(H, W)$ if the input has size $(H, W)$. Strides are commonly used to reduce the spatial size. If we have stride 2, the convolution window will not shift one position at a time as a standard convolution, but instead two positions at a time which has the effect of spatially downsampling the output. With dilations the kernels are

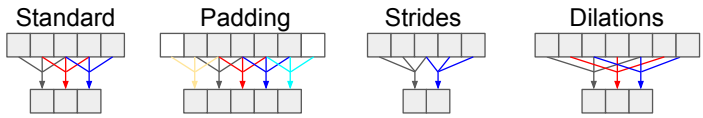


**Figure 1.3:** Illustration of which input elements are used for each output for a 1d convolution with a filter of size 3, and how padding, strides and dilations alter a standard convolution. For the standard convolution an input of length 5 gets an output of length 3. Padding means that we append extra elements at the start and end of the input (often with zeros). This is useful to keep the length of the output the same as the length of the input. With strides, here stride 2, the first output uses the inputs 1, 2 and 3, while the second output uses the inputs 3, 4 and 5, so the receptive field shifts 2 positions instead of 1 here, effectively downsampling the output. With dilated convolutions, here dilation 2, the first output uses the inputs 1, 3 and 5 instead of 1, 2 and 3 as in a standard convolution.

expanded to use spaced input patches rather than consecutive, which can be used to increase the receptive field, so that inputs further away from the input affect the output. In general, for a spatial input of size $(H, W)$, a convolutional layer with padding $p$, kernel size $k \times k$, stride $s$ and dilation $d$ will result in the output size

$$\left( \left\lfloor \frac{H + 2p - d(k-1) - 1}{s} \right\rfloor + 1, \left\lfloor \frac{W + 2p - d(k-1) - 1}{s} \right\rfloor + 1 \right) \tag{1.2}$$

The convolutional operator has several properties that makes it suitable for image data which are listed below. In the calculations we assume that the input $X$ has size $(M, N, C)$ and the output $Y$ has size $(M, N, C')$, and that the convolutional kernel has spatial size $k \times k$. We assume suitable input padding to keep the spatial dimensions, and do not assume that there are any bias parameters.

- Spatial invariance. The kernels are spatially invariant, meaning that the convolutions are computed with the same filters in the whole image. The assumption is that a given kernel is useful for the whole image, and not just particular regions. This is illustrated in fig. 1.2 where filters detecting edges are useful for the whole image.
- Sparse computations. When computing $Y(i, j)$ we only consider parts of the input around $X(i, j)$ where the kernel is non-zero, and ignore the rest of the input. This requires $MNk^2CC'$ element-wise multiplications instead of $M^2N^2CC'$ for a fully connected layer, and $k^2$ is typically much smaller than $MN$.
- Few trainable weights. A convolutional kernel has far fewer weights than a fully connected layer from $X$ to $Y$ would have. The number of parameters are $k^2CC'$ instead of $M^2N^2CC'$, which is significantly less since $k^2$ is much less than $M^2N^2$.

In the top part of fig. 1.4 we show a simple example of what a CNN for image classification can look like. It begins with the input image of size $(H, W, 3)$ and this is transformed by multiple convolutional layers, until finally arriving at the score over image categories, which is a vector with the length being the number of categories. The predicted label is obtained

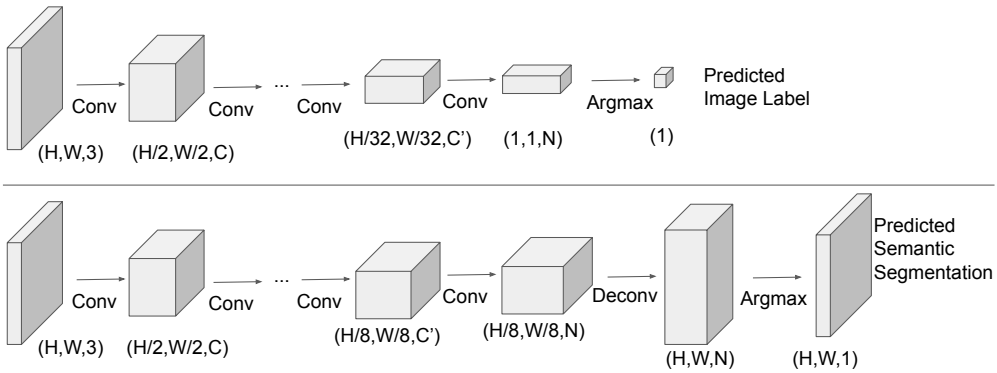


**Figure 1.4:** Schematic illustration of how multiple convolutions are stacked for image classification (top) and semantic segmentation (bottom). The input in both cases is an RGB-image of size $(H, W, 3)$. The output for image classification is a single number corresponding to the predicted class, while for semantic segmentation, there is one prediction per pixel. The second last layer is a distribution over all $N$ classes, for image classification for the whole image, while for semantic segmentation it is per pixel. Selecting the class with the highest probability ($\arg\max$) will give the predictions. The intermediate layers are often of higher resolution for dense prediction tasks such as semantic segmentation, as illustrated here where the lowest resolution is $1/8$ for semantic segmentation and $1/32$ for image classification.

by selecting the category with the highest score. All parameters of all layers of the CNN are trained, and the learned features of layers close to the input are typically composed of low level structures such as edges while successive layers learn increasingly abstract concepts, see Zeiler and Fergus [88].

## 2.2 From Image Classification to Dense Predictions

The standard way to learn semantic segmentation is to start with a CNN trained for image classification [32, 37, 74] and transform the architecture to not just predict one label per image, but one label per pixel, and then refine the whole network end-to-end with a semantic segmentation loss. We will now describe the modifications required to get dense predictions, starting with the representation of the output, the modifications applied to not spatially downsample too much and lose fine details, and how to modify the loss function. These modifications are widely used in successful approaches [16, 89]. We show a schematic overview of the modifications in fig. 1.4 of how to keep all parameters of the pre-trained networks for image classification, but adapt the architecture for dense predictions. There are mainly two modifications, namely how to increase the resolution of the intermediate layers, and how to represent the output.

In general, to get a dense prediction for semantic segmentation, we require that the last layer has shape $(H, W, N)$, where $N$ is the number of classes and $H$ and $W$ are the height and


**Figure 1.5:** How to transform a fully connected (dense) layer to a convolutional layer. The approach if using dense layers (left) is to first remove the spatial dimensions by flattening the tensor to just one dimension and then apply a fully connected layer. To make it fully convolutional (right) the weights of the fully connected layer are simply encoded in the weights of the convolutional layer instead, and the output gets a spatial dimension of size $(1, 1)$.

width of the input image, respectively. If we apply softmax[1] per spatial location we get a probability distribution over the labels per pixel. In practice, if we have any layer in a neural network of size $(H', W', C)$ we can always add one or more convolutional or deconvolutional layer to get a map of size $(H, W, N)$, and transforming with softmax will give class probabilities. It is common to use deconvolutional layers at the very end of the CNN to go from e.g. $(H/8, W/8, N)$ or $(H/16, W/16, N)$ to $(H, W, N)$, and initialize the parameters as bilinear upsampling. Such an upsampling is schematically shown in fig. 1.4.

It is common to have fully connected layers in networks for image classification, just before the predictions. Such layers can easily be transformed to convolutional layers, while still keeping the computations the same. In fig. 1.5 we show such a transformation. In the original network, some spatial map $(H, W, C)$ is flattened to a one-dimensional vector of length $HWC$, and it is followed by a fully connected (dense) layer with $C'$ output channels. The parameters of the dense layer can be encoded in a $H \times W$-convolution by simply reshaping (and maybe permuting the channels) the weights of the dense layer (matrix of e.g. size $HWC \times C'$) to weights of a convolutional layer (shape e.g. $H \times W \times C \times C'$ depending on convention). After this transformation, the input can be any spatial size, whereas for the dense layer it must be exactly $(H, W, C)$. The result is a fully convolutional network [49], and in practice for any pre-trained network for image classification with dense layers, those layers are transformed to convolutional layers in this manner when we want dense predictions. If the pre-trained network for image classification does not use fully connected layers, but instead use global average pooling prior to the classification, we simply remove the pooling layer.

If we have a pre-trained network with downsampling at fixed points, we can easily control and change the output resolution via dilated convolutions. We simply skip the downsampling (commonly implemented either by max-pooling or strides in the convolutional layers) and change convolutions that follow to be dilated. With this transformation, the dilated convolutions have the same receptive field in the higher resolution input as the original convolution had in the lower resolution input. Specifically, if the original convolution was computed with input size $(H, W)$, then if we want to apply the same convolution to an upsampled input of size $(sH, sW)$ we can use the same convolutional weights, but with the dilation $s$. We show

[1] The softmax function of a vector $(x_1, \ldots, x_n)$ is defined by $y_i = \frac{\exp(x_i)}{\sum_j \exp(x_j)}$ for $i = 1, \ldots, n$ and define a probability distribution since $y_i \geq 0$ and $\sum_i y_i = 1$.

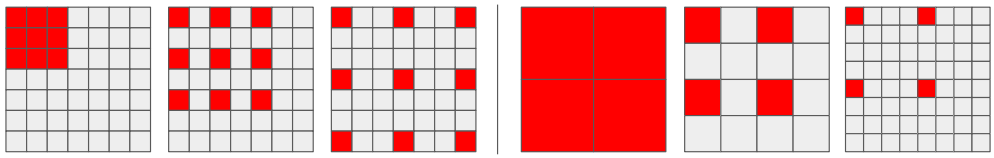

**Figure 1.6:** Dilated convolutions. On the left we show the receptive fields for the top-left output element of a $3 \times 3$ convolution with dilation rate 1 (a standard convolution), 2 and 3. On the right we show a $2 \times 2$ convolution and an input at three different resolutions, and we assume that the top-left pixels correspond. We see that if we double the resolution and double the dilation rate, the convolution is computed using equally spaced input elements in all three resolutions.

the basic idea behind dilated convolutions in fig. 1.6, where we show a normal $3 \times 3$ convolution and the same convolution with dilation rates 2 and 3, and how dilations and input resolutions are related. We show how to change convolutional layers with strides 2 to convolutional layers with stride 1, but adding dilations in fig. 1.7. Note that all the convolutional weights are initialized from the pre-training, and for the second convolutional layer initially defined on input of size $(H/2, W/2)$, it needs to have dilation 2 if the input is of size $(H, W)$. An example is given in fig. 1.4 where the network for image classification has a lowest resolution of $(H/32, W/32)$ but the corresponding layer has size $(H/8, W/8)$ when changed to predict semantic segmentation, to not loose fine spatial details. Although not central for this presentation, we can mention that the spatial size does not necessarily go from $(H, W)$ to $(H/2, W/2)$ in all architectures, and it depends on the implementation and which convention is used. As an example, if we use $3 \times 3$ convolutions with stride 2 and padding 1 and $H$ and $W$ are odd, the output size is $((H + 1)/2, (W + 1)/2)$.

The standard loss function for image classification is the cross entropy of the ground truth and estimated class probabilities given by the CNN. If we let $y = (y_1, y_2, \dots y_n)$ be the output of the CNN, and let $y' = (y'_1, y'_2, \dots, y'_n)$ be the ground truth, one-hot encoded, so that $y'_c = 1$ for the correct label $c$ and $y'_i = 0$ for $i \neq c$, then the cross-entropy is

$$L(y', y) = -\sum_k y'_k \log y_k = -\log y_c \tag{1.3}$$

The last equality follows from the one-hot encoding and the last expression is the negative log likelihood. If we consider semantic segmentation as one classification problem per pixel we can easily extend the cross entropy loss for image classification to semantic segmentation. Let $y'_{ij} = (y'_{ij1}, y'_{ij2}, \dots, y'_{ijn})$ be the ground truth of pixel $(i, j)$, where the ground truth is one-hot encoded and let $y_{ij} = (y_{ij1}, y_{ij2}, \dots, y_{ijn})$ be the estimated class probabilities by the CNN. We define the segmentation loss as

$$L(y', y) = -\sum_{ijk} y'_{ijk} \log y_{ijk} \tag{1.4}$$

where the only difference to (1.3) is that we sum over all pixels.

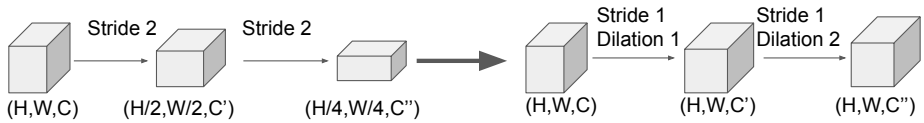


**Figure 1.7:** We show the basic principle for transforming a pre-trained network with fixed subsampling (using strides or pooling) for dense predictions by increasing the resolution of intermediate layers, and still keeping all weights for all layers. We remove the strides and add dilations to increase the resolution of the intermediate feature maps. The kernel weights of all convolutional filters are the same as in the pre-trained version.

## 2.3 Evaluating Semantic Segmentation

To evaluate semantic segmentation, it is common to use class balanced metrics such as mIoU instead of global accuracy. It is very common that the semantic classes are unbalanced, with one example being indoor scenes where the classes wall, ceiling and floor might make up a very large fraction of the pixels, while coffee mug might be a label occupying very few images and pixels. Another example is street view scenes where much of the images might consist of roads or sidewalks, with pedestrians occupying significantly fewer pixels. In the latter example, a classifier correctly segmenting road and sidewalk while misclassifying all pedestrians will get a high accuracy since most pixels will be correctly labelled, but it will probably not be useful in practice. Instead of global accuracy a metric that takes all classes explicitly into account will be more relevant for semantic segmentation. The most commonly used metric is mIoU, which is defined as follows. If we let $S_i$ denote the set of pixels predicted as class $i$ and $T_i$ the set of pixels where the ground truth is class $i$, then the IoU (intersection over union) of class $i$ is defined as $\text{IoU}_i = |S_i \cap T_i|/|S_i \cup T_i|$. To get the aggregate segmentation metric, the mean IoU is defined as

$$\text{mIoU} = \frac{1}{N}\sum_i \text{IoU}_i = \frac{1}{N}\sum_i \frac{|S_i \cap T_i|}{|S_i \cup T_i|} \tag{1.5}$$

where $N$ is the number of classes. This metric gives equal weights for the IoUs of all individual classes, and a high value requires accurate segmentations of also relatively uncommon classes, in contrast to e.g. global accuracy. We show an example in fig. 1.8.

The mIoU is sometimes defined in an equivalent way using the confusion matrix. If we let $c_{ij}$ be the number of pixels predicted as class $i$ where the ground truth is $j$, it is easy to see that

$$\text{IoU}_i = \frac{c_{ii}}{\sum_k c_{ik} + \sum_k c_{ki} - c_{ii}} \tag{1.6}$$

and

$$\text{mIoU} = \frac{1}{N}\sum_i \frac{c_{ii}}{\sum_k c_{ik} + \sum_k c_{ki} - c_{ii}} \tag{1.7}$$

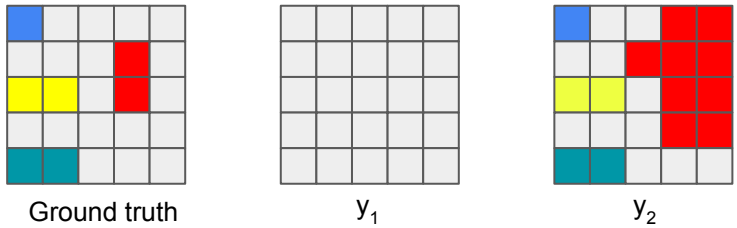


**Figure 1.8:** Illustration of the difference between using global accuracy and mIoU. There are five semantic classes, including the background. Looking qualitatively, we expect a metric judging segmentation quality to rank $y_2$ higher than $y_1$ since all classes are detected in $y_2$ while all pixels are labelled as background in $y_1$. However, the accuracy is $18/25$ for both $y_1$ and $y_2$ while the mIoUs are $\frac{1}{5}(\frac{18}{25} + \frac{0}{1} + \frac{0}{2} + \frac{0}{2} + \frac{0}{2}) \approx 0.14$ for $y_1$ and $\frac{1}{5}(\frac{11}{18} + \frac{1}{1} + \frac{2}{2} + \frac{2}{2} + \frac{2}{9}) \approx 0.77$ for $y_2$. We see that $y_1$ has the same accuracy as $y_2$ despite not detecting four classes at all, but the mIoU is significantly higher for $y_2$ where all objects were detected. This example illustrates that global accuracy is not necessarily indicative of segmentation quality, and other metrics such as mIoU are often more suitable.

## 2.4 Review of the State-of-the-Art

Adapting CNNs pre-trained on ImageNet for semantic segmentation was pioneered by Long et al. [49], in which a pre-trained CNN for image classification was made fully convolutional. The sub-sampling of the pre-trained networks were kept, but multiple skip connections from multiple parts of the network with varying resolutions were added to the output layer to recover fine spatial details. In Farabet et al. [26] a deep convolutional network is applied with multiple resolutions of the input image, and then the predictions are merged.

Another line of work use a so called encoder-decoder structure where the features of the lowest resolution gradually are upsampled in a decoding module typically consisting of multiple layers per resolution, as schematically illustrated in fig. 1.9. For instance, the encoder can be a pre-trained CNN with output resolution $1/32$, and the decoder starts by upsampling the output of the encoder to $1/16$ and then apply a few convolutional layers, and then upsample to $1/8$ and apply a few more convolutional layers, and so on. For these methods it is often not necessary to increase the resolution of the intermediate features of the pre-trained CNN. Noh et al. [58] use unpooling layers to during upsampling recover the boundaries by keeping track of which indices that were used in the corresponding max-pooling layers. A similar upsampling method is proposed in SegNet [3]. Another approach is to in addition to upsampling have skip connections between layers in the encoder and decoder of similar spatial sizes, as proposed in U-Net [67], thus not losing track of fine spatial details. In RefineNet [48] the decoder module is further refined using residual connections [32]. In Ghiasi and Fowlkes [30] the features of the skip connections are masked to only propagate features close to object boundaries to recover fine spatial details by predicting the boundaries using the features of the decoder.

Methods to incorporate and aggregate features of varying scales are often implemented by

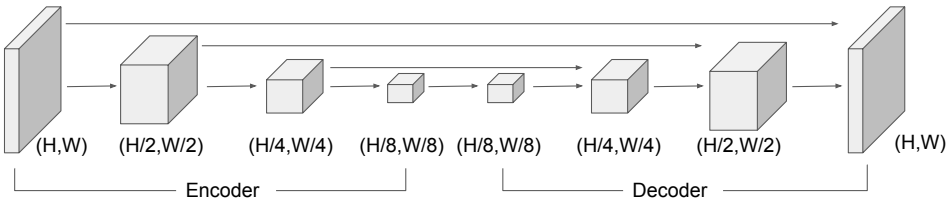


**Figure 1.9:** Schematic illustration of encoder-decoder networks. The input is gradually transformed to lower resolutions in the encoder and then transformed back in the decoder, typically increasing or decreasing the resolution by a factor 2 at each step. The connections on the top are skip connections which are often used for dense prediction tasks to recover fine spatial details which may have been lost during the downsampling in the encoder. The encoder is often a network pre-trained on ImageNet while all layers of the decoder are initialized randomly.

cascading special network architectures following the feature extraction (schematically, just before or after the deconv layer in fig. 1.4). In DeepLab [18, 17] an ASPP-module is introduced to handle multiple scales. Prior to the output layer, a layer concatenating multiple parallel convolutions with varying dilations, e.g. four filters with dilation rates 6, 12, 18 and 24, is used to handle multiple scales. A similar methodology was proposed by Yu and Koltun [87]. In earlier versions of DeepLab [16] the output was refined using CRF post-processing. Specifically, in the fully connected CRF of Krähenbühl and Koltun [43] all pixels are connected pairwise via Gaussian kernels, and it allows for efficient approximate inference. Such CRF inference was also done end-to-end by Zheng et al. [90] by implementing all inference operations as neural network layers and training the system jointly. In PSPNet [89] a method of pooling at varying scales is employed in contrast to ASPP by not using dilations but instead global pooling operations. For instance, when using a 4 level pyramid, an input of is average pooled into e.g. the sizes $1 \times 1$ (global average pooling), $2 \times 2$, $3 \times 3$ and $6 \times 6$. Following these representations is a convolutional filter to reduce the dimensionality, and then all representations are upsampled and concatenated with the input tensor, thus adding global context.

# 3 Video Analysis

In this section we give a brief introduction to tools for video analysis, and especially methods relevant to semantic video segmentation. This is especially relevant for paper 1, but is also used in paper 2, 3 and 4 to allow the propagation of labels from annotated frames to unlabelled frames. We will cover optical flow which is the displacements between frames, followed by recurrent neural networks which can handle sequential data, and finally we will discuss the semantic video segmentation problem and state-of-the-art approaches to it.

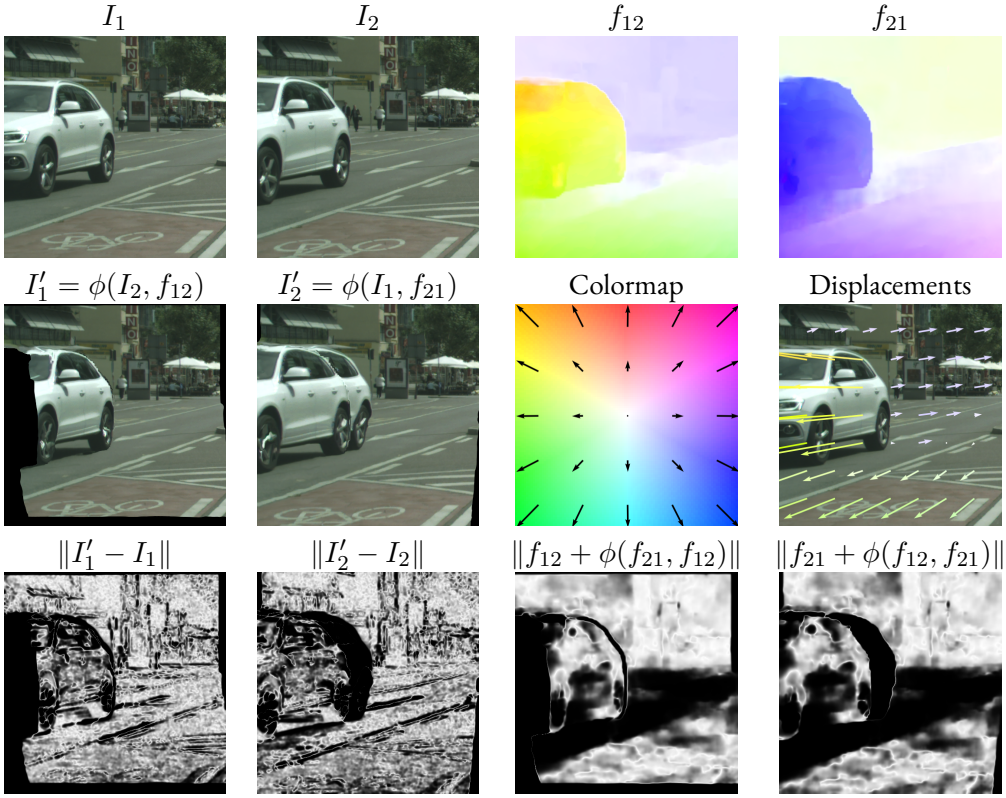


**Figure 1.10:** Illustration of optical flow. We show two images, where a car has moved between the frames and the camera pose has changed slightly. The pixel displacements $f_{12}$ and $f_{21}$ are color-coded according to the colormap shown. Each color in $f_{12}$ and $f_{21}$ represents a displacement vector, and we also show the displacements $f_{12}$ overlayed on $I_1$, but magnified for illustrative purposes. If we warp the images we see two types of occlusions. The first is that the front of the car is visible in $I_1$ but not in $I_2$ and the second is that the object visible behind the car in $I_2$ are not visible in $I_1$. We also show two different confidence estimates of the optical flow, where white pixels have a high confidence value and black a low confidence. We can see that the occluded regions have low confidence as expected.

## 3.1 Optical Flow

Optical flow is a measurement of the pixel-wise displacements between frames. It is widely used for video analysis in e.g. action recognition [73], video object segmentation [19, 82] and semantic video segmentation [55, 28]. If we have two consecutive frames in a video, $I_1$ and $I_2$, the optical flow of the pixel $(i, j)$ in $I_1$ is defined as the displacement to the corresponding pixel $(i', j')$ in $I_2$. We define the optical flow $f$ by $f_{ij} = (f_{ij}^y, f_{ij}^x) = (i' - i, j' - j)$. So if we want to track $(i, j)$ in $I_1$ the corresponding pixel is $(i + f_{ij}^y, j + f_{ij}^x)$ in $I_2$. The optical flow is very fine-grained, since displacements are computed for every single pixel.

In fig. 1.10 we show an image pair $(I_1, I_2)$. The optical flow $f_{12}$ is computed from $I_1$ to $I_2$ while $f_{21}$ is computed from $I_2$ to $I_1$. The optical flow is commonly encoded by a colormap,

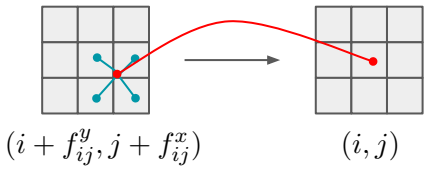


**Figure 1.11:** We describe the receptive fields and how to propagate values for the warping operation defined in (1.8). The output $y_{ij}$ is computed by copying the features in the corresponding position in $x$, but shifted by the optical flow. We want the features of $x$ at the position $(i + f_{ij}^y, j + f_{ij}^x)$. Since these typically are not integer coordinates, we interpolate the closest points by means of a kernel, illustrated by the four closest points in the figure. Note that the optical flow is computed in the reverse direction (from $y$ to $x$) that we propagate.

and the coloring of displacements is shown in the figure. We furthermore show the warpings, $I_1'$ and $I_2'$ where the images $I_1$ and $I_2$ are propagated via the computed optical flows to align with the other image. We can see that while most of the warpings are aligned with the target images, there are a few places where it is incorrect. In $I_1'$ we note that the left and bottom parts of the image are not correct, since these parts of $I_1$ are not visible in $I_2$ and hence these parts can not be propagated from $I_2$. We also note that the back of the car is incorrect in $I_2'$, and there is a doubling of the back of the car. This is since the objects right of the car in $I_2$ are not visible in $I_1$ and while the pole is correct, the parts left of the pole in $I_1$ is the car which then appears double in $I_2'$. On the last row we show two different ways of measuring the confidences of the optical flow, which are later described in more detail, but for now we just note that the regions we mentioned are marked with low confidences by both estimates.

In fig. 1.10 we showed the images warped to align with the other images by means of the computed optical flow. We will now describe this mapping mathematically and describe how it can be used as a neural network layer. Any spatial map can be propagated, e.g. RGB values, semantic segmentations or any general feature map. We define the warping from a grid $x_{ij}$ to $y_{ij}$ given by an optical flow $f$ as

$$y_{ij} = \sum_{m,n} x_{mn} k(i + f_{ij}^y - m, j + f_{ij}^x - n) \tag{1.8}$$

where $k(x, y)$ is a kernel and $f_{ij}$ is the optical flow from $y$ to $x$ which is the reverse direction in which we propagate. The optical flow encodes how to fetch values from $x$ to $y$ rather than how to propagate values from $x$ to $y$, i.e. $y_{ij}$ should fetch from $(i + f_{ij}^y, j + f_{ij}^x)$ in $x$. These coordinates are typically not integer valued, so we need to interpolate. We will use the bilinear kernel $k(x, y) = \max(0, 1 - |x|) \max(0, 1 - |y|)$. We describe the receptive field and give more intuition for why the optical flow is computed in the reverse direction in fig. 1.11. We denote the mapping from $x$ to $y$ using the flow $f$ as $y = \phi(x, f)$. In practice, $x$ can have multiple channels, and we simply warp the channels independently of each other.

Note that (1.8) is differentiable. If we have $dL/dy$ we can easily compute $dL/dx$ and $dL/df$ using the chain rule, allowing backpropagation to both the input $x$ and the optical flow $f$. Any kernel $k$ will make backpropagation to $x$ possible, while backpropagating to $f$ requires

that $k$ is differentiable. The sum over all pixels $\{m, n\}$ will only have at most 4 non-zero terms if we use the bilinear kernel and all parts of the warping are differentiable[2], so we can include it as a layer in a neural network and run backpropagation through the layer.

In many applications of optical flow, there is a necessity to assess the confidence of the displacements. There are two straight-forward ways which we will describe here, the photo-consistency and forward-backward consistency, and they are also illustrated in fig. 1.10. If we let $I_1$ and $I_2$ denote the images, and $I_2' = \phi(I_1, f_{21})$ be the warping of the first image to the second, the photo-consistency measure of confidence is then $\|I_2 - I_2'\|$, with the norm applied per spatial location, when aligned with $I_2$. In a similar way it is $\|I_1 - I_1'\|$, where $I_1' = \phi(I_2, f_{12})$ if it is aligned with $I_1$. In the example figure the occluded regions are correctly of low confidence, but we also note that the estimate is fairly noisy.

The other commonly used measure is to compare the forward $f_{12}$ and backward flow $f_{21}$. If the optical flow is accurate in both directions, then if we track a pixel $(i, j)$ in $I_1$ to $(i', j')$ in $I_2$, we can expect the optical flow in the reverse direction to go from $(i', j')$ in $I_2$ to $(i, j)$ in $I_1$. We can formalize the pixel-wise confidence as $\|f_{12} + \phi(f_{21}, f_{12})\|$, where the first term is the forward flow and the second term is the backward flow warped to aligned with $I_1$. Note the plus sign since we expect the backward displacement to be the negative forward displacement. The forward-backward consistency requires that we compute optical flow in two directions, while the photo-consistency measure $\|I_2 - I_2'\|$ only requires optical flow in one direction, so the run-time will be higher if we compute forward-backward consistency. We show an example in fig. 1.10, where we note that the occluded regions are of low confidence, but there are also large regions (e.g. the road) that are marked with low confidence. The estimate is less noisy than when comparing intensity differences, which is expected since the optical flow maps contain significantly less edges than the RGB images.

In recent years approaches based on CNNs to compute optical flow have proven very successful, with examples being Flownet [23], Flownet2 [38] and PWC-Net [76], all of which are used in the papers of this thesis. Prior to deep learning based methods, a common approach was to use variational optimization and possibly to combine it with combinatorial matching [85, 8, 66].

## 3.2 Recurrent Neural Networks

When processing videos, we are given a sequence of consecutive images instead of just a single image. To process such time series we need special neural network structures, and what we will describe here are recurrent neural networks (RNNs). In paper 1 we use recurrent net-

[2]Technically the bilinear kernel $k(x, y) = \max(0, 1 - |x|) \max(0, 1 - |y|)$ is not differentiable if $x \in \{-1, 0, 1\}$ and $-1 \leq y \leq 1$ or if $y \in \{-1, 0, 1\}$ and $-1 \leq x \leq 1$, but this is not an issue. It is similar to relu activation function defined by $x \mapsto \max(x, 0)$ which is not differentiable at $x = 0$ but used nevertheless.

works to process videos for semantic segmentation, and in paper 2, 3 and 4 we use recurrent neural networks in the policy networks trained with reinforcement learning.

The setup of RNNs is that there is a sequence of inputs $\{x_t\}$ that are transformed into a hidden state $\{h_t\}$ and for each time step there is also an output $\{y_t\}$. The hidden state $h_t$ is computed one time step at a time via a neural network as $h_t = f_\theta(x_t, h_{t-1})$, and can encode information that is relevant for the task over multiple time-steps. The choice of $f$ depends on the problem, and two commonly used versions are the gated recurrent unit (GRU) [20] and long short-term memory (LSTM) [34]. The parameters $\theta$ do *not* depend on the time $t$, since the RNNs should be able to operate on variable length sequences, and also that it is often assumed that the RNN performs the same task at all times, but only with different inputs and hidden states.

We here show the equations for a GRU. Note that this describes a mapping of the form $h_t = f_\theta(x_t, h_{t-1})$. We assume that $x_t$ and $h_t$ are vectors for all $t$.

$$\begin{aligned} z_t &= \sigma(W_{xz}x_t + W_{hz}h_{t-1} + b_z) \\ r_t &= \sigma(W_{xr}x_t + W_{hr}h_{t-1} + b_r) \\ \hat{h}_t &= \tanh(W_{xh}x_t + W_{hh}(r_t \odot h_{t-1}) + b_h) \\ h_t &= (1 - z_t) \odot h_{t-1} + z_t \odot \hat{h}_t \end{aligned} \tag{1.9}$$

where $\odot$ is element-wise multiplication, and $W$ and $b$ are parameters, so the set of parameters is $\theta = \{W_{xz}, W_{hz}, b_z, W_{xr}, W_{hr}, b_r, W_{xh}, W_{hh}, b_h\}$. The sigmoid function is defined as $\sigma(x) = 1/(1 + \exp(-x))$. The output $y_t$ depends on the problem and a simple choice is e.g. $y_t = \text{softmax}(W_{hy}h_t + b_y)$ for a classification problem. To handle spatial data we just change all fully connected layers to convolutional layers. The problem with video data though, is that due to motion, the spatial coordinates $(i, j)$ in frame $t$ do not correspond to the same coordinates $(i, j)$ in frame $t + 1$. This can however be mitigated by computing the optical flow and warping at appropriate places, which will be done in paper 1.

The concept of gating is central for both an LSTM and a GRU. If we have a vector $x$ a gating in this context is a transformation $x \odot g$ where all elements of $g$ are between 0 and 1, and we multiply $x$ and $g$ element-wise. In LSTMs and GRUs the gating functions are learned, and they are typically implemented by sigmoid functions. In the equations for the GRU (1.9), $z_t$ and $r_t$ are used as gating functions when computing $\hat{h}_t$ and $h_t$, and they are implemented by the sigmoid function and depend on the inputs $x_t$ and $h_{t-1}$.

## 3.3 Semantic Video Segmentation

We described semantic segmentation in section §2, and we will now describe semantic video segmentation, where the task is to semantically segment a video. Instead of just segmenting a

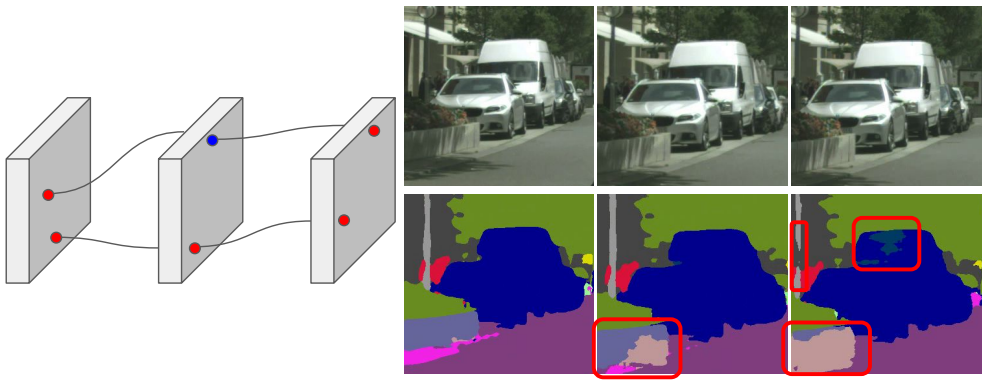

**Figure 1.12:** Illustration of flickering in the context of semantic segmentation. On the left we show three consecutive frames and two tracks, computed e.g. by optical flow, and the semantic segmentation, here shown as blue or red. The top track is not consistent since the labelling changes along the track while the bottom track is consistent since the labelling is the same throughout the whole track. On the right we show three consecutive frames of a video along with semantic segmentations and mark with red boxes several regions that are not temporally consistent with the preceding frame.

single image, the input is a whole video with multiple frames, and a label should be predicted for every pixel of every frame. A naive method for semantic video segmentation is to simply use a model for image semantic segmentation and process each frame independently. In paper I we show how to incorporate temporal data in a principled manner to improve over this per-frame baseline, both in terms of segmentation accuracy and temporal consistency.

When using the simple per-frame baseline, the temporal continuity of the input is entirely ignored. An object that is hard to perceive in one frame can in principle be detected in a preceding or succeeding frame where it is easier to detect, and then be propagated temporally to the uncertain frame. Thus it should be possible to improve the accuracy of a video segmentation system by using the temporal consistencies inherently available in the video data. Another issue with predicting semantic segmentation for a video independently per frame is that it often introduces flickering. If we track the same pixel for multiple frames, the labelling can vary for different frames. We show an example in fig. 1.12. To evaluate the flickering quantitatively, we can use methods of pixel tracking and count consistency along the tracks [45, 77]. Specifically, if we have a track of pixels $\{x_k\}_{k=t}^{T}$ from frame $t$ to $T$, we look at the labelling of the pixels along the track. If all pixels are of the same label we say that the track is consistent, and if there are two points along a track with different labels the track is not consistent. The temporal consistency metric is then the fraction of tracks which are consistent, computed from a large set of tracks covering as much of the video as possible. In paper I we use a method that tracks pixels using optical flow and ends the tracks if the optical flow is uncertain [77], as measured by the forward-backward criteria (see §3.1). Note that it is trivial to get $100\%$ temporal consistency by always predicting the same label for the whole image. Temporal consistency computed in this way should thus not be measured in isolation but be complemented by e.g. mIoU.

There is a large body of work on semantic video segmentation, targeting different aspects of the problem. Common for most methods is that they are based on pre-trained image semantic segmentation systems (see §2.4), and extend or alter these to use temporal data. The main trade-off is between speed and accuracy. By exploiting the temporal consistency inherent in video data it is possible to on one hand increase the accuracy at the expense of run-time and on the other hand to increase the speed at the expense of accuracy. The first line of work concerns how to make the semantic segmentation more accurate and temporally consistent than per-frame processing. This is often approached using highly accurate optical flow and warping features or segmentations between time-frames, e.g. in an end-to-end trainable architecture [55, 28] or by post-processing using 3d point clouds [27, 7] or CRFs [45, 46, 81, 75]. The CRFs are usually constructed so that the unary terms are computed by a CNN for image semantic segmentation. The pairwise terms can be connected temporally e.g. along the optical flow, by comparing deep features or by using 3d geometry. While these approaches typically improve the segmentation accuracy, the run-time is often higher than per-frame processing. There are also some recent approaches that use attention mechanisms instead of optical flow or CRFs [61, 36].

Another line of work is to significantly speed up the video processing [47, 72, 39, 50, 35]. Since consecutive frames are highly redundant, it is common to process different frames with different levels of granularity. Specifically, it is common to use a deep CNN for image semantic segmentation only for sparse frames, and for the frames in between use a light-weight module possibly combined with motion via optical flow. These approaches are typically faster but less accurate than per-frame baselines.

# 4 Embodied Learning

In embodied learning we study how to learn by interacting with an environment. The hypothesis is that by acting, and observing the results of the actions, an agent can learn intelligent behavior. This is in contrast to the learning in the previous sections where data was supplied as is, and the learning was done with static data that was labelled. The term embodied here means that the agent operating has a body, which can mean that it loosely resembles that of a human body in the sense that it moves around and that its perception is mainly based on visual, egocentric observations of its environment.

The scope of embodied learning as defined above is very large, with significant parts of the field of robotics fitting the description. We will here greatly limit the scope of the review to only cover visual navigation problems, with an emphasis towards trainable methods, since that is relevant for paper 2, 3 and 4 in this thesis.

Visual navigation broadly concerns how to move around in an environment to solve a given

**Figure 1.13:** Example from the Matterport3D dataset. It is possible to use simulated environments created from real-world scans in an interactive manner to study navigation problems. There are multiple modalities available, and here we show a map, image, semantic segmentation and depth.

task given only visual observations. One aspect of the problem which greatly varies depending on the approach and the application is how to map the environment. To navigate efficiently it is useful to infer the 2d floor plan using the available sensors such as e.g. RGB, RGBD or LIDAR data, and use that for planning [80]. The exhaustive approach when only visual sensors are available is to create a full 3d reconstruction of an environment using SLAM [54, 9], where a euclidean reconstruction is used. It is also possible to use less granular representations such as sparsely computed landmarks, in a topological map [68, 6, 29]. There are also learning-based methods that do not use a map at all and directly learn the mapping from observations to actions using e.g. reinforcement learning [86]. There are multiple works comparing learned methods to mapping based methods for point goal navigation [52, 42, 64], however the variations in problem setup and range of methods to consider makes it difficult to draw general conclusions. It can vary whether it is evaluated in simulation or in the real world, and the availability and noise levels of input modalities such as depth and pose can vary in different setups. Some works argue that learning based methods outperform mapping based ones given enough training [51], while others reach the opposite conclusion [52, 42].

In paper 2, 3 and 4 we use the Habitat simulator [51] and the dataset Matterport3D [12]. The Matterport3D dataset contains several scans of indoor environments of different buildings, and the Habitat simulator facilitates simulation and interaction in the environments. It is possible to extract RGB images, semantic segmentations and depth maps, and it is possible to interact and navigate in the 3d scans. We show an example in fig. 1.13. Since the scans are based on real images captured from the real world, and not artificially created, they can be considered photo-realistic. The use of simulated environments has several advantages over using real-world robots when running experiments, such as free access to the ground truth of various modalities, fast and parallel simulation, comparison of different methods under identical environments and setups, and it does not require physical space or real robots. Note however that good performance in a simulated environment is not a guarantee for real-world performance [40], and transferability of a policy learned in simulation to the real world is a highly non-trivial problem studied by itself [15, 62, 2], and outside the scope of the papers in this thesis. One important further detail is that when visual navigation problems are studied

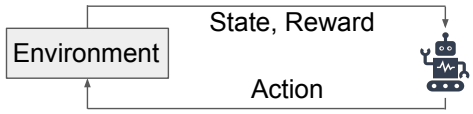


**Figure 1.14:** The basic setup in reinforcement learning. An agent should act in an environment so that it maximizes the reward it obtains. At a given time step the agent selects an action, based on the state, which contains information about the environment. Once the agent acts it obtains a reward, which is a measure of how good the action was for the specific task the agent should perform. This process is repeated for several steps until the episode ends.

in such simulated environments, then the environments are often fixed in the sense that it is not possible to e.g. move objects around or open and close doors. The 3d environments are assumed to be static and to solve the tasks it suffices to navigate and gather visual observations, and not change the state of the 3d environment. Although out of scope, there is some recent work targeting navigation problems in large-scale simulated environments where objects are not static [4, 84].

Some examples of common tasks studied in embodied learning, here narrowed down to visual navigation problems often studied in simulated environments, are listed below. See Zhu et al. [92] and Duan et al. [24] for thorough and recent surveys.

- Point goal navigation [12, 86, 14]: In point goal navigation the agent is given target coordinates relative to its current position it should navigate to. There can be obstacles in the way, so the agent should recognize those and plan to navigate around them.
- Object goal navigation [13, 11]: In object goal navigation the agent should navigate to an instance of a specific class, e.g. fridge or bed in an unknown environment. The objects are often not visible from the agent's starting position, so solving the task requires exploration of a building as well as visual recognition.
- Visual language navigation [63, 1, 83]: In visual language navigation an agent is tasked with following and executing language instructions such as "find the kitchen and walk up to the fridge" or "walk up the stairs and enter the second door to the left". This requires navigation, but also visual perception and language processing.

These problems are often successfully approached with reinforcement learning, which we will describe in more detail in the next section.

## 4.1 Reinforcement Learning

In this section we will give a brief introduction to reinforcement learning which is needed for paper 2, 3 and 4. For a more thorough text, see Sutton and Barto [78]. In fig. 1.14 we show

the basic setup for reinforcement learning (RL). An agent in this setup should *act* in a way that maximizes the reward it obtains. The learning is thus that of a policy of which action to take at each time step, so that the cumulative rewards are maximized. The agent acts in an environment, and at each time step the agent observes the state $s_t$ of the environment. The state encodes information about the current state of the environment in the form of e.g. images or feature vectors. The agent selects an action $a_t$ based on its policy $\pi(a_t|s_t)$, which maps the state to a distribution over all possible actions. After an action in the environment, the agent receives a reward $r_t$, which is a numerical value indicating how good the action was. The reward function is often handcrafted to model the specific behaviour that is wanted. The agent then selects a new action $a_{t+1}$ based on the updated state $s_{t+1}$ following the action $a_t$ in the environment, and continues to select new actions until the episode ends.

Each action will lead to a reward, but rewards should be aggregated over multiple time-steps, since the choice of $a_t$ will affect all rewards $r_{t'}$ with $t' \geq t$. If we assume that an agent has received the rewards $r_0, r_1, \ldots$ we define the discounted future reward $g_t$ at time $t$ as

$$g_t = \sum_{t'=t} \gamma^{t'-t} r_{t'} \tag{1.10}$$

where $\gamma$ is a discount factor. The return $g_t$ is a weighted sum of all future rewards starting from $t$, where rewards far in the future are discounted by $\gamma$. The value of $\gamma$ determines how far ahead an agent should plan, and an agent trained with $\gamma = 0$ will greedily select the action yielding the immediately highest reward, while an agent trained with $\gamma = 1$ will be forced to plan further ahead. A common choice is $\gamma = 0.99$.

Mathematically, the dynamics of the environment is modelled by a Markov Decision Process (MDP) [5] which consist of a state space $\mathcal{S}$, action space $\mathcal{A}$, transition probabilities $p(s'|s, a)$ from one state $s \in \mathcal{S}$ to another $s' \in \mathcal{S}$ by the action $a \in \mathcal{A}$, a corresponding reward function $R(s, a, s')$ for the transition, and a distribution $p(s_0)$ of the initial state. We further assume that we have a policy $\pi(a|s)$ which maps states to actions. A trajectory of the MDP is then generated by the following model

$$s_0 \sim p(s_0) \tag{1.11}$$

$$a_t \sim \pi(\cdot|s_t) \tag{1.12}$$

$$s_{t+1} \sim p(\cdot|s_t, a_t) \tag{1.13}$$

where we note that $s_{t+1}$ is only conditioned on $s_t$ and $a_t$, and not any $s_{t'}$ or $a_{t'}$ with $t' < t$, which is the Markov property, and similarly $a_t$ is only conditioned on $s_t$. For a given trajectory[3] $\tau = (s_0, a_0, s_1, a_1, \ldots, s_T)$ we have that

$$p(\tau) = p(s_0) \prod_{t=0}^{T-1} \pi(a_t|s_t) p(s_{t+1}|s_t, a_t) \tag{1.14}$$

[3]It is no requirement that every episode has the same length $T$, but we assume that to simplify the presentation.

It should be noted that for all but the simplest environments, it is very difficult to compute or approximate the state transitions $p(s'|s,a)$ and the corresponding reward function $R(s,a,s')$. If we do not need to, we say that we use model-free RL, in contrast to model-based RL where the MDP dynamics are estimated and then used for planning.

Two very useful functions are the Q-value $Q(s,a)$ and the value function $V(s)$ which are defined as

$$Q(s_t,a_t) = E_{s_{t+1},a_{t+1},\ldots,s_T}\left[\sum_{t'=t}^{T-1}\gamma^{t'-t}R(s_{t'},a_{t'},s_{t'+1})\right] \tag{1.15}$$

$$V(s_t) = E_{a_t,s_{t+1},a_{t+1},\ldots,s_T}\left[\sum_{t'=t}^{T-1}\gamma^{t'-t}R(s_{t'},a_{t'},s_{t'+1})\right] \tag{1.16}$$

which denotes the expected discounted future returns as given by the policy $\pi$. The Q-value is one action ahead of the value function. We note that the Q-value and the value function are related by

$$Q(s_t,a_t) = E_{s_{t+1}}\left[R(s_t,a_t,s_{t+1}) + \gamma V(s_{t+1})\right] \tag{1.17}$$

$$V(s_t) = E_{a_t}\left[Q(s_t,a_t)\right] \tag{1.18}$$

and they also fulfill the following recursive equations known as the Bellman equations

$$Q(s_t,a_t) = E_{s_{t+1},a_{t+1}}\left[R(s_t,a_t,s_{t+1}) + \gamma Q(s_{t+1},a_{t+1})\right] \tag{1.19}$$

$$V(s_t) = E_{a_t,s_{t+1}}\left[R(s_t,a_t,s_{t+1}) + \gamma V(s_{t+1})\right] \tag{1.20}$$

Another related term is the advantage, defined as $A(s,a) = Q(s,a) - V(s)$ which is interpreted as how much better it is to choose a specific action $a$ than selecting an action according to $\pi(\cdot|s)$. The Q-value, value function and advantage function all depend on the policy. Sometimes this dependency is marked explicitly by using notation such as $V^\pi(s_t)$ instead of $V(s_t)$.

Several examples of full RL setups including the environment, state space, action space and reward function are:

- Point goal navigation (see §4). The agent can navigate via the actions move forward, rotate left and rotate right, and it is also equipped with a done action to be selected when the target is reached. The state space is the RGB image, depth map and the coordinates $(r,\theta)$ of the target (sometimes referred to as GPS+compass), specified by polar coordinates relative to the agent's pose. The reward is the decrease in geodesic distance to the goal and furthermore a positive reward when the done action is selected if the agent was sufficiently close to the target.

- An agent plays an ATARI game such as pong [53]. The action space is to move the paddle up, down or keep it still. The reward can be $+1$ if the agent scores and $-1$ if the opponent scores, and all other actions get a reward of 0. The state space can e.g. be an RGB image of the screen or a vector with the positions and velocities of the paddles and the ball.
- A humanoid robot should walk forward [33]. The state space is the locations and velocities of the joints in 3d and the action space is the torque to apply per joint. The reward is proportional to the velocity forward of the robot.

## 4.2 Policy Gradient Methods

While there are many different approaches to RL [78], we will only focus on policy optimization methods. In this approach, we are optimizing only the policy $\pi_\theta(a|s)$, which we assume is parameterized by a set of parameters $\theta$. We will derive the reinforce algorithm, which is a method that approximates the gradient of the expected returns, and use that to optimize $\theta$. This approach is model-free and relies on sampling of the MDP. The only requirement is that we can compute the gradient $\nabla_\theta \pi_\theta(a|s)$ which is possible e.g. if $\pi_\theta$ is implemented as a neural network, via backpropagation.

To derive reinforce [79], we start by a simplified example utilizing a simple trick to compute gradients of the expected value of a function $g(x)$ over a parameterized distribution $f_\theta(x)$. We can see that the gradient of the expected value w.r.t. the parameters $\theta$ can be computed as

$$\begin{aligned}
\nabla_\theta E_{x\sim f_\theta(x)}\left[g(x)\right] &= \nabla_\theta \int g(x) f_\theta(x) dx && (1.21)\\
&= \int g(x) \nabla_\theta f_\theta(x) dx && (1.22)\\
&= \int g(x) \nabla_\theta \log f_\theta(x) f_\theta(x) dx && (1.23)\\
&= E_{x\sim f_\theta(x)}\left[g(x) \nabla_\theta \log f_\theta(x)\right] && (1.24)\\
&\approx \frac{1}{N} \sum_{k=1}^{N} g(x_k) \nabla_\theta \log f_\theta(x_k) && (1.25)
\end{aligned}$$

where $\{x_k\}_{k=1}^N$ are sampled from the probability distribution $f_\theta$. The latter form allows for sampling over $f_\theta(x)$ to compute the gradient of the expected value. We can thus use any gradient based method to optimize the expected values w.r.t. the parameters $\theta$. We will now derive reinforce, which extends the simple result above to a full trajectory of an MDP. The result above can be seen as the special case of an MDP with just one action, where $g(x)$ corresponds to the reward and $f_\theta(x)$ the policy.

The reinforce algorithm optimizes the sum of rewards[4] $R(\tau) = \sum_{t=0}^{T-1} r_t$ and we want to optimize the function

$$J(\theta) = E_{\tau \sim \pi_\theta}[R(\tau)] \tag{1.26}$$

where

$$\tau = (s_0, a_0, r_0, \ldots, s_T) \tag{1.27}$$

is a rollout, according to the policy $\pi_\theta$. We use the convention that $r_t = R(s_t, a_t, s_{t+1})$ is the reward after taking the action $a_t$ in the state $s_t$. To evaluate the gradient of $J(\theta)$ we will need to explicitly compute $p(\tau)$. This can be expressed using the transition probabilities of the MDP and the policy $\pi_\theta$ as

$$p(\tau) = p(s_0) \prod_{t=0}^{T-1} \pi_\theta(a_t|s_t) p(s_{t+1}|s_t, a_t) \tag{1.28}$$

The gradient of $\log p(\tau)$ will be used later and is computed as

$$\nabla_\theta \log p(\tau) = \nabla_\theta \left[ \log p(s_0) + \sum_{t=0}^{T-1} (\log \pi_\theta(a_t|s_t) + \log p(s_{t+1}|s_t, a_t)) \right] \tag{1.29}$$

$$= \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t|s_t) \tag{1.30}$$

where all the transition probabilities $p(s_{t+1}|s_t, a_t)$ vanish since they do not depend on the parameters $\theta$ of the policy. Now we are ready to compute the gradient of the objective function $J(\theta)$ as follows

$$\nabla_\theta J(\theta) = \nabla_\theta E_{\tau \sim \pi_\theta}[R(\tau)] \tag{1.31}$$

$$= \nabla_\theta \int_\tau p(\tau) R(\tau) d\tau \tag{1.32}$$

$$= \int_\tau \nabla_\theta p(\tau) R(\tau) d\tau \tag{1.33}$$

$$= \int_\tau p(\tau) R(\tau) \nabla_\theta \log p(\tau) d\tau \tag{1.34}$$

$$= \int_\tau p(\tau) R(\tau) \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t|s_t) d\tau \tag{1.35}$$

$$= E_{\tau \sim \pi_\theta} \left[ R(\tau) \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t|s_t) \right] \tag{1.36}$$

[4] To simplify the presentation we assume that $\gamma = 1$.

We have now obtained an expression similar to that of the simplified setup in (1.24), and we can thus obtain approximations of $\nabla_\theta J(\theta)$ by sampling trajectories $\tau \sim \pi_\theta$ according to the current policy $\pi_\theta$. Note that in (1.36) we multiply $\nabla_\theta \log \pi_\theta(a_t|s_t)$ with $R(\tau) = \sum_{t=0}^{T-1} r_t$, but for given $t$, the action $a_t$ only affects the rewards $r_{t'}$ with $t' \geq t$. In fact, it can be shown[5] [78] that we can discard rewards with $t' < t$ and also subtract a baseline from the rewards as follows

$$\nabla_\theta E_{\tau\sim\pi_\theta}[R(\tau)] = E_{\tau\sim\pi_\theta}\left[R(\tau)\sum_{t=0}^{T-1}\nabla_\theta \log \pi_\theta(a_t|s_t)\right] \tag{1.37}$$

$$= E_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{T-1}\left(\sum_{t'=t}^{T-1} r_{t'}\right)\nabla_\theta \log \pi_\theta(a_t|s_t)\right] \tag{1.38}$$

$$= E_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{T-1}\left(\sum_{t'=t}^{T-1} r_{t'} - b(s_t)\right)\nabla_\theta \log \pi_\theta(a_t|s_t)\right] \tag{1.39}$$

where $b$ can be any function that only depends on $s_t$. In practice, $b$ is usually an estimate of the value function $V(s_t)$ and it can greatly reduce the variance of the gradient estimation [78]. If both the policy and value function are estimated, it is typically called an actor-critic method in the RL literature where actor refers to the policy and critic to the value function.

The formulation in (1.39) allows for easy sampling. We let the agent act according to $\pi_\theta(a|s)$ and we use a sampled trajectory $\tau = (s_0, a_0, r_0, \ldots, s_T)$ to optimize the policy using the estimated gradient

$$\nabla_\theta J(\theta) \approx \frac{1}{T}\sum_{t=0}^{T-1}\left(\sum_{t'=t}^{T-1} r_{t'} - \hat{V}_\phi(s_t)\right)\nabla_\theta \log \pi_\theta(a_t|s_t) \tag{1.40}$$

where $\hat{V}_\phi$ approximates the value function $V$ and is parameterized by $\phi$. This function can in turn be optimized e.g. by a least squares objective $J_V(\phi)$ defined as

$$J_V(\phi) = \frac{1}{2T}\sum_{t=0}^{T-1}\left\|\sum_{t'=t}^{T-1} r_{t'} - \hat{V}_\phi(s_t)\right\|^2 \tag{1.41}$$

If we denote $A_t = \sum_{t'=t}^{T-1} r_{t'} - \hat{V}_\phi(s_t)$, and look at the signs in (1.40) for one index $t$ in isolation, we note that if $A_t > 0$, then the effect of a gradient ascent step will be to increase $J(\theta)$ by increasing $\pi_\theta(a_t|s_t)$. Whatever actions $a_t$ lead to positive advantages $A_t > 0$ are

[5] Sketches of the main steps of the proofs are given here. Rewrite $E_\tau[R(\tau)] = \sum_{t=0}^{T-1} E_\tau[r_t]$ and differentiate $E_\tau[r_t]$ as in the main text and use that $r_t$ only depends on the trajectory up until $t$, not $T-1$. For the baseline subtraction, note that $\pi_\theta(a_t|s_t)\nabla_\theta \log \pi_\theta(a_t|s_t) = \nabla_\theta \pi_\theta(a_t|s_t)$, so $E_{a_t}[b(s_t)\nabla_\theta \log \pi_\theta(a_t|s_t)] = \sum_{a_t} b(s_t)\nabla_\theta \pi(a_t|s_t) = b(s_t)\nabla_\theta 1 = 0$.

reinforced in the sense that $\pi_\theta(a_t|s_t)$ increases, and similarly $\pi_\theta(a_t|s_t)$ decreases for actions with negative advantages $A_t < 0$.

Algorithm 1 shows the pseudo-code for a basic version of policy gradient using the equations derived above. We assume that we update $\theta$ and $\phi$ by a single gradient ascent or descent step using learning rate $\alpha_p$ and $\alpha_v$ respectively. Since the value function $V$ depends on the policy $\pi_\theta$, the parameters $\phi$ of the estimate $\hat{V}_\phi$ need to be updated any time $\theta$ is updated. There are many variations of the basic algorithm presented here. We can use any gradient based optimization method, and e.g. ADAM [41] is also a common choice. We can also collect multiple trajectories per parameter update and average the gradients in (1.40) over multiple trajectories to reduce the variance, and for a set of collected trajectories we can perform multiple parameter updates. However, we can in general not reuse a collected trajectory after the parameters $\theta$ are updated. It is required that the trajectories are sampled from the policy $\pi_\theta$ with the current parameters $\theta$ for the gradient estimation to be correct, so after updating the policy the trajectories are in general immediately discarded and new ones are sampled. It is also possible to estimate the advantage function $A_t = \sum_{t'=t}^{T-1} r_{t'} - \hat{V}_\phi(s_t)$ differently, e.g. as in Generalized Advantage Estimation (GAE) [69], and sometimes all advantages are re-scaled prior to the gradient estimation to have mean 0 and standard deviation 1 for numerical reasons.

**Algorithm 1** Procedural code for a basic actor-critic version of policy gradient.

Initialize the parameters of the policy $\pi_\theta$ and the value function $\hat{V}_\phi$.
**for** Episode $= 1, \ldots, N_{eps}$ **do**
    Reset the environment to obtain $s_0$
    **for** $t = 0, \ldots, T-1$ **do**
        Sample an action $a_t \sim \pi_\theta(\cdot|s_t)$ and store $(s_t, a_t, r_t)$.
    **end for**
    $\theta \leftarrow \theta + \alpha_p \frac{1}{T} \sum_{t=0}^{T-1} \left( \sum_{t'=t}^{T-1} r_{t'} - \hat{V}_\phi(s_t) \right) \nabla_\theta \log \pi_\theta(a_t|s_t)$
    $\phi \leftarrow \phi - \alpha_v \nabla_\phi \left( \frac{1}{2T} \sum_{t=0}^{T-1} \left\| \sum_{t'=t}^{T-1} r_{t'} - \hat{V}_\phi(s_t) \right\|^2 \right)$
**end for**
**return** $\pi_\theta$ (optimized policy)

We will now show the equations for Proximal Policy Optimization (PPO) [70], which further alters (1.39) to at every step not change the policy too much. We assume that we have collected a trajectory $\tau = (s_0, a_0, r_0, \ldots, s_T)$ as before, but we also keep track of the old policy parameters $\theta_{\text{old}}$ used to collect $\tau$ and compare $\pi_\theta$ with $\pi_{\theta_{\text{old}}}$ to make sure it does not deviate too much. Instead of a single gradient step as in (1.40) we update the parameters with the

gradient

$$\nabla_\theta \frac{1}{T} \sum_{t=0}^{T-1} \min \left( \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)} A_t, \text{clip} \left( \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}, 1-\epsilon, 1+\epsilon \right) A_t \right) \tag{1.42}$$

where

$$\text{clip}(x, L, H) = \max(\min(x, H), L) = \begin{cases} L, & x < L \\ x, & L \leq x \leq H \\ H, & x > H \end{cases} \tag{1.43}$$

constraints $x$ to the interval $[L, H]$, $A_t$ is the advantage function, estimated by $\sum_{t'=t}^{T-1} r_{t'} - \hat{V}_\phi(s_t)$ as in actor-critic, and $\epsilon$ is a hyperparameter. While (1.40) looks quite different to (1.42), if we rewrite $\nabla_\theta \log \pi_\theta(a_t|s_t) = \nabla_\theta \pi_\theta(a_t|s_t)/\pi_\theta(a_t|s_t)$ we see the similarity. It can be shown that the difference between (1.40) and (1.42) is that in PPO the terms where $\pi_\theta(a_t|s_t)$ deviates too much from $\pi_{\theta_{\text{old}}}(a_t|s_t)$ have gradient zero. Specifically, the terms of the sum in (1.42) can be rewritten as

$$\min \left( \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)} A_t, \text{clip} \left( \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}, 1-\epsilon, 1+\epsilon \right) A_t \right) \tag{1.44}$$

$$= \begin{cases} A_t \min \left( \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}, 1+\epsilon \right), & A_t \geq 0 \\ A_t \max \left( \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}, 1-\epsilon \right), & A_t < 0 \end{cases} \tag{1.45}$$

where we see that if $A_t > 0$, the gradient is non-zero only if $\pi_\theta(a_t|s_t) < (1+\epsilon)\pi_{\theta_{\text{old}}}(a_t|s_t)$, and if $A_t < 0$ it is non-zero only if $\pi_\theta(a_t|s_t) > (1-\epsilon)\pi_{\theta_{\text{old}}}(a_t|s_t)$, which clarifies the effect of the hyperparameter $\epsilon$. Note that an action with $A_t > 0$ should be reinforced ($\pi_\theta(a_t|s_t)$ should increase) while an action with $A_t < 0$ should be suppressed ($\pi_\theta(a_t|s_t)$ should decrease).

# Bibliography


[1] P. Anderson, Q. Wu, D. Teney, J. Bruce, M. Johnson, N. Sünderhauf, I. Reid, S. Gould, and A. Van Den Hengel. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3674–3683, 2018.

[2] J. V. Baar, A. Sullivan, R. Cordorel, D. K. Jha, D. Romeres, and D. Nikovski. Sim-to-real transfer learning using robustified controllers in robotic tasks involving complex dynamics. *2019 International Conference on Robotics and Automation (ICRA)*, pages 6001–6007, 2019.

[3] V. Badrinarayanan, A. Kendall, and R. Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 39(12):2481–2495, 2017.

[4] D. Batra, A. X. Chang, S. Chernova, A. J. Davison, J. Deng, V. Koltun, S. Levine, J. Malik, I. Mordatch, R. Mottaghi, et al. Rearrangement: A challenge for embodied ai. *arXiv preprint arXiv:2011.01975*, 2020.

[5] R. Bellman. A markovian decision process. *Journal of mathematics and mechanics*, 6(5):679–684, 1957.

[6] O. Booij, B. Terwijn, Z. Zivkovic, and B. Krose. Navigation using an appearance based topological map. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 3927–3932. IEEE, 2007.

[7] G. J. Brostow, J. Shotton, J. Fauqueur, and R. Cipolla. Segmentation and recognition using structure from motion point clouds. In *European conference on computer vision*, pages 44–57. Springer, 2008.

[8] T. Brox and J. Malik. Large displacement optical flow: descriptor matching in variational motion estimation. *IEEE transactions on pattern analysis and machine intelligence*, 33(3):500–513, 2010.

[9] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Transactions on robotics*, 32(6):1309–1332, 2016.

[10] S. Caelles, K.-K. Maninis, J. Pont-Tuset, L. Leal-Taixé, D. Cremers, and L. Van Gool. One-shot video object segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 221–230, 2017.

[11] T. Campari, P. Eccher, L. Serafini, and L. Ballan. Exploiting scene-specific features for object goal navigation. In *European Conference on Computer Vision*, pages 406–421. Springer, 2020.

[12] A. Chang, A. Dai, T. Funkhouser, M. Halber, M. Niessner, M. Savva, S. Song, A. Zeng, and Y. Zhang. Matterport3d: Learning from rgb-d data in indoor environments. *International Conference on 3D Vision (3DV)*, 2017.

[13] D. S. Chaplot, D. P. Gandhi, A. Gupta, and R. R. Salakhutdinov. Object goal navigation using goal-oriented semantic exploration. *Advances in Neural Information Processing Systems*, 33, 2020.

[14] D. S. Chaplot, R. Salakhutdinov, A. Gupta, and S. Gupta. Neural topological slam for visual navigation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12875–12884, 2020.

[15] Y. Chebotar, A. Handa, V. Makoviychuk, M. Macklin, J. Issac, N. D. Ratliff, and D. Fox. Closing the sim-to-real loop: Adapting simulation randomization with real world experience. *2019 International Conference on Robotics and Automation (ICRA)*, pages 8973–8979, 2019.

[16] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):834–848, 2017.

[17] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*, 2017.

[18] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 801–818, 2018.

[19] J. Cheng, Y.-H. Tsai, S. Wang, and M.-H. Yang. Segflow: Joint learning for video object segmentation and optical flow. In *Proceedings of the IEEE international conference on computer vision*, pages 686–695, 2017.

[20] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[21] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[22] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[23] A. Dosovitskiy, P. Fischer, E. Ilg, P. Hausser, C. Hazirbas, V. Golkov, P. Van Der Smagt, D. Cremers, and T. Brox. Flownet: Learning optical flow with convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2758–2766, 2015.

[24] J. Duan, S. Yu, H. L. Tan, H. Zhu, and C. Tan. A survey of embodied ai: From simulators to research tasks. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 2022.

[25] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010.

[26] C. Farabet, C. Couprie, L. Najman, and Y. LeCun. Learning hierarchical features for scene labeling. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1915–1929, 2012.

[27] G. Floros and B. Leibe. Joint 2d-3d temporally consistent semantic segmentation of street scenes. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2823–2830. IEEE, 2012.

[28] R. Gadde, V. Jampani, and P. V. Gehler. Semantic video cnns through representation warping. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4453–4462, 2017.

[29] E. Garcia-Fidalgo and A. Ortiz. Vision-based topological mapping and localization methods: A survey. *Robotics and Autonomous Systems*, 64:1–20, 2015.

[30] G. Ghiasi and C. C. Fowlkes. Laplacian pyramid reconstruction and refinement for semantic segmentation. In *European conference on computer vision*, pages 519–534. Springer, 2016.

[31] I. Goodfellow, Y. Bengio, and A. Courville. *Deep learning*. MIT press, 2016.

[32] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[33] N. Heess, D. TB, S. Sriram, J. Lemmon, J. Merel, G. Wayne, Y. Tassa, T. Erez, Z. Wang, S. Eslami, et al. Emergence of locomotion behaviours in rich environments. *arXiv preprint arXiv:1707.02286*, 2017.

[34] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[35] P. Hu, F. Caba, O. Wang, Z. Lin, S. Sclaroff, and F. Perazzi. Temporally distributed networks for fast video semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8818–8827, 2020.

[36] P. Hu, F. Perazzi, F. C. Heilbron, O. Wang, Z. Lin, K. Saenko, and S. Sclaroff. Real-time semantic segmentation with fast attention. *IEEE Robotics and Automation Letters*, 6(1):263–270, 2020.

[37] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[38] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox. Flownet 2.0: Evolution of optical flow estimation with deep networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2462–2470, 2017.

[39] S. Jain, X. Wang, and J. E. Gonzalez. Accel: A corrective fusion network for efficient semantic segmentation on video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8866–8875, 2019.

[40] A. Kadian, J. Truong, A. Gokaslan, A. Clegg, E. Wijmans, S. Lee, M. Savva, S. Chernova, and D. Batra. Sim2real predictivity: Does evaluation in simulation predict real-world performance? *IEEE Robotics and Automation Letters*, 5:6670–6677, 2020.

[41] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[42] N. Kojima and J. Deng. To learn or not to learn: Analyzing the role of learning for navigation in virtual environments. *arXiv preprint arXiv:1907.11770*, 2019.

[43] P. Krähenbühl and V. Koltun. Efficient inference in fully connected crfs with gaussian edge potentials. *Advances in neural information processing systems*, 24:109–117, 2011.

[44] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012.

[45] A. Kundu, V. Vineet, and V. Koltun. Feature space optimization for semantic video segmentation. In *CVPR*, 2016.

[46] P. Lei and S. Todorovic. Recurrent temporal deep field for semantic video labeling. In *European Conference on Computer Vision*, pages 302–317. Springer, 2016.

[47] Y. Li, J. Shi, and D. Lin. Low-latency video semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5997–6005, 2018.

[48] G. Lin, A. Milan, C. Shen, and I. Reid. Refinenet: Multi-path refinement networks for high-resolution semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1925–1934, 2017.

[49] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[50] B. Mahasseni, S. Todorovic, and A. Fern. Budget-aware deep semantic video segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1029–1038, 2017.

[51] Manolis Savva*, Abhishek Kadian*, Oleksandr Maksymets*, Y. Zhao, E. Wijmans, B. Jain, J. Straub, J. Liu, V. Koltun, J. Malik, D. Parikh, and D. Batra. Habitat: A Platform for Embodied AI Research. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.

[52] D. Mishkin, A. Dosovitskiy, and V. Koltun. Benchmarking classic and learned navigation in complex 3d environments. *arXiv preprint arXiv:1901.10915*, 2019.

[53] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[54] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos. Orb-slam: a versatile and accurate monocular slam system. *IEEE transactions on robotics*, 31(5):1147–1163, 2015.

[55] D. Nilsson and C. Sminchisescu. Semantic video segmentation by gated recurrent flow propagation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6819–6828, 2018.

[56] D. Nilsson, A. Pirinen, E. Gärtner, and C. Sminchisescu. Embodied visual active learning for semantic segmentation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 2373–2383, 2021.

[57] D. Nilsson, A. Pirinen, E. Gärtner, and C. Sminchisescu. Embodied learning for lifelong visual perception. *arXiv preprint arXiv:2112.14084*, 2021.

[58] H. Noh, S. Hong, and B. Han. Learning deconvolution network for semantic segmentation. In *Proceedings of the IEEE international conference on computer vision*, pages 1520–1528, 2015.

[59] M. Oquab, L. Bottou, I. Laptev, and J. Sivic. Is object localization for free?-weakly-supervised learning with convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 685–694, 2015.

[60] G. Papandreou, L.-C. Chen, K. P. Murphy, and A. L. Yuille. Weakly-and semi-supervised learning of a deep convolutional network for semantic image segmentation. In *Proceedings of the IEEE international conference on computer vision*, pages 1742–1750, 2015.

[61] M. Paul, M. Danelljan, L. Van Gool, and R. Timofte. Local memory attention for fast video semantic segmentation. *arXiv preprint arXiv:2101.01715*, 2021.

[62] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel. Sim-to-real transfer of robotic control with dynamics randomization. *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–8, 2018.

[63] Y. Qi, Z. Pan, S. Zhang, A. van den Hengel, and Q. Wu. Object-and-action aware model for visual language navigation. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part X 16*, pages 303–317. Springer, 2020.

[64] S. K. Ramakrishnan, D. Jayaraman, and K. Grauman. An exploration of embodied visual exploration. *International Journal of Computer Vision*, 129(5):1616–1649, 2021.

[65] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28:91–99, 2015.

[66] J. Revaud, P. Weinzaepfel, Z. Harchaoui, and C. Schmid. Epicflow: Edge-preserving interpolation of correspondences for optical flow. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1164–1172, 2015.

[67] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[68] N. Savinov, A. Dosovitskiy, and V. Koltun. Semi-parametric topological memory for navigation. In *International Conference on Learning Representations*, 2018.

[69] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015.

[70] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[71] B. Settles. Active learning literature survey. 2009.

[72] E. Shelhamer, K. Rakelly, J. Hoffman, and T. Darrell. Clockwork convnets for video semantic segmentation. In *European Conference on Computer Vision*, pages 852–868. Springer, 2016.

[73] K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. In *Advances in Neural Information Processing Systems*, pages 568–576, 2014.

[74] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015.

[75] P. Sturgess, K. Alahari, L. Ladicky, and P. H. Torr. Combining appearance and structure from motion features for road scene understanding. In *BMVC-British Machine Vision Conference*. BMVA, 2009.

[76] D. Sun, X. Yang, M.-Y. Liu, and J. Kautz. Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8934–8943, 2018.

[77] N. Sundaram, T. Brox, and K. Keutzer. Dense point trajectories by gpu-accelerated large displacement optical flow. In *ECCV*, 2010.

[78] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. 2018.

[79] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, pages 1057–1063, 2000.

[80] S. Thrun. Probabilistic robotics. *Communications of the ACM*, 45(3):52–57, 2002.

[81] S. Tripathi, S. Belongie, Y. Hwang, and T. Nguyen. Semantic video segmentation: Exploring inference efficiency. In *2015 International SoC Design Conference (ISOCC)*, pages 157–158. IEEE, 2015.

[82] Y.-H. Tsai, M.-H. Yang, and M. J. Black. Video segmentation via object flow. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3899–3908, 2016.

[83] X. Wang, Q. Huang, A. Celikyilmaz, J. Gao, D. Shen, Y.-F. Wang, W. Y. Wang, and L. Zhang. Reinforced cross-modal matching and self-supervised imitation learning for vision-language navigation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6629–6638, 2019.

[84] L. Weihs, M. Deitke, A. Kembhavi, and R. Mottaghi. Visual room rearrangement. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2021.

[85] P. Weinzaepfel, J. Revaud, Z. Harchaoui, and C. Schmid. Deepflow: Large displacement optical flow with deep matching. In *Proceedings of the IEEE international conference on computer vision*, pages 1385–1392, 2013.

[86] E. Wijmans, A. Kadian, A. Morcos, S. Lee, I. Essa, D. Parikh, M. Savva, and D. Batra. Dd-ppo: Learning near-perfect pointgoal navigators from 2.5 billion frames. In *International Conference on Learning Representations*, 2019.

[87] F. Yu and V. Koltun. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*, 2015.

[88] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.

[89] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia. Pyramid scene parsing network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2881–2890, 2017.

[90] S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, and P. H. Torr. Conditional random fields as recurrent neural networks. In *Proceedings of the IEEE international conference on computer vision*, pages 1529–1537, 2015.

[91] B. Zhou, H. Zhao, X. Puig, S. Fidler, A. Barriuso, and A. Torralba. Semantic understanding of scenes through the ade20k dataset. *International Journal of Computer Vision*, 127:302–321, 2018.

[92] F. Zhu, Y. Zhu, X. Liang, and X. Chang. Deep learning for embodied vision navigation: A survey. *arXiv preprint arXiv:2108.04097*, 2021.

# Chapter 2

# Scientific publications

# Paper I

# Semantic Video Segmentation by Gated Recurrent Flow Propagation

David Nilsson[1] and Cristian Sminchisescu[1,2]
[1]Department of Mathematics, Faculty of Engineering, Lund University
[2]Institute of Mathematics of the Romanian Academy

## Abstract

Semantic video segmentation is challenging due to the sheer amount of data that needs to be processed and labeled in order to construct accurate models. In this paper we present a deep, end-to-end trainable methodology for video segmentation that is capable of leveraging the information present in unlabeled data, besides sparsely labeled frames, in order to improve semantic estimates. Our model combines a convolutional architecture and a spatio-temporal transformer recurrent layer that is able to temporally propagate labeling information by means of optical flow, adaptively gated based on its locally estimated uncertainty. The flow, the recognition and the gated temporal propagation modules can be trained jointly, end-to-end. The temporal, gated recurrent flow propagation component of our model can be plugged into any static semantic segmentation architecture and turn it into a weakly supervised video processing one. Our experiments in the challenging CityScapes and Camvid datasets, and for multiple deep architectures, indicate that the resulting model can leverage unlabeled temporal frames, next to a labeled one, in order to improve both the video segmentation accuracy and the consistency of its temporal labeling, at no additional annotation cost and with little extra computation.

# 1 Introduction

Systems capable of computing accurate and temporally consistent semantic segmentations in video are central to scene understanding, being useful in applications in robotics, for instance grasping, or for autonomous vehicles where one naturally works with videos rather than single images, and high levels of precision are needed. Since the emergence of deep learning methods for image classification, the problem of semantic image segmentation has received increasing attention, with some of the most successful methods based on fully trainable convolutional architectures (CNN). Data for training and refining single frame, static models is now quite diverse [7, 29]. In contrast, fully trainable approaches to semantic video segmentation face the difficulty of obtaining detailed annotations for individual video frames, although datasets are emerging for the (unsupervised) video segmentation problem [11, 36, 27]. Therefore some of the existing approaches to semantic video segmentation [42, 43, 25] rely on single frame models with corresponding variables connected in time using random fields with higher-order potentials, and mostly pre-specified parameters. Fully trainable approaches to video are rare. The computational complexity of video processing further complicated matters.

One possible approach to designing semantic video segmentation models in the long run can be to only label frames, sparsely, in video, as it was done for static datasets[7, 29]. Then one should be able to leverage temporal dependencies in order to propagate and aggregate information in order to decrease uncertainty during *both* learning and inference. This would require a model that can integrate spatio-temporal dependencies across video frames.

While approaches based on CNNs appear right, they are non-trivial to adapt to video segmentation due to the amount of data that needs to be processed for dense predictions. If video processing and temporal matching were to be learned without explicit components such as optical flow warping, one possibility would be to design a model based on 3D convolutions, as used e.g. for action recognition[20, 3]. To our knowledge no such approach has been pursued for semantic video segmentation. Instead, we will take an explicit modeling approach relying on existing single-frame CNNs augmented with spatial transformer structures that implement warping along optical flow fields. These will be combined with adaptive recurrent units in order to learn to fuse the estimates from single (unlabeled) frames with the labeling information temporally propagated from nearby ones, properly gated based on their uncertainty. The proposed model is differentiable and end-to-end trainable.

# 2 Related Work

Our semantic video segmentation work relates to the different fields of semantic image segmentation, as well as, more remotely, to unsupervised video segmentation. We will here only

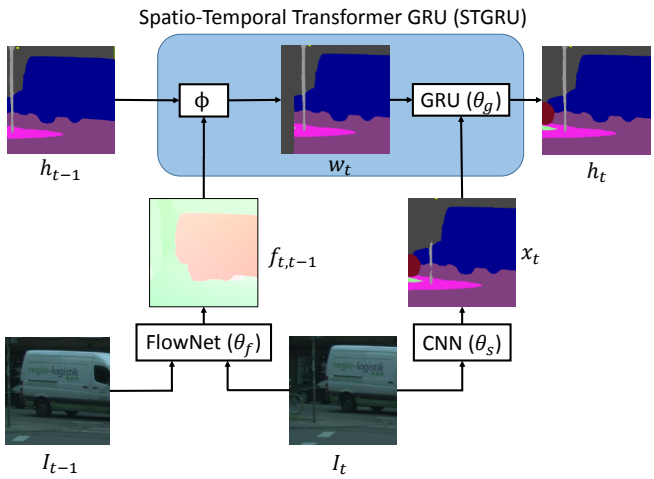


**Figure 2.1:** Overview of our Spatio-Temporal Transformer Gated Recurrent Unit (STGRU), combining a Spatial Transformer Network (§3.1) mapping $\phi$ for optical flow warping with a Gated Recurrent Unit (§3.2) to adaptively propagate and fuse semantic segmentation information over time. E.g. the pole is not accurately segmented by the static network ($x_t$), but combining $x_t$ with the segmentation at the previous timestep ($h_{t-1}$) gives a more accurate estimate $h_t$.

briefly review the vast literature with some attention towards formulations based on deep architectures which represent the foundation of our approach.

Many approaches start with the network in [24, 39] and refine it for semantic segmentation. In [15] residual connections are used making it possible to increase depth substantially. [32] obtained semantic segmentations by turning a network for classification [39] into a dense predictor by computing segmentations at different scales and combining all predictions. The network was made fully convolutional. Another successful approach is to apply a dense conditional random field (CRF) [23] as a post-processing step on top of individual pixel or frame predictions. [4] use a fully convolutional network to predict a segmentation and then apply the dense CRF in post processing. [46] realized that inference in dense CRFs can be formulated as a fixed point iteration implementable as a recurrent neural network. Another successful approach is the deep architecture of [44] where max pooling layers are replaced with dilated convolutions. The network was extended by introducing a context module where convolutions with increasingly large dilation sizes are used.

Video segmentation has received significant attention starting from early methodologies based on temporal extensions to normalized cuts [38], random field models and tracking [42, 26], motion segmentation [34] or efficient hierarchical graph-based formulations [14, 43]. More recently, proposal methods where multiple figure-ground estimates or multipart superpixel segmentations are generated at each time-step, then linked through time using optical flow

[27, 1, 35], have become popular.

The dense CRF of [23] has been used for semantic video segmentation [25, 41], most notably by [25] using pairwise potentials based on aligning the frames using optical flow. Along similar lines as our earlier version of this work[33], [10] independently present an end-to-end trainable system for semantic video segmentation that warps two-frame intermediate representations in a CNN. We differ in that we warp the segmentation outputs and we can use multiple frames forward and backward in time. In a similar fashion, [30] combines semantic segmentations by means of optical flow warping for body part segmentation in videos. In [19], video propagation is performed by filtering in a bilateral space instead of using optical flow to connect frames temporally. The temporal matching can also be performed using superpixels and optical flow, as in [16], where information in matched regions is pooled using Spatio-Temporal Data-Driven Pooling (STD2P). [21] use GANs [13] to first predict future video frames in an unsupervised manner and then use the learned features for semantic video segmentation. In [37] observe that intermediate representations in a CNN change slowly in video, and present a method to only recompute features when there is enough change, leading to significant speed-ups.

# 3 Methodology

A visual illustration of how our semantic video segmentation model aggregates information in adjacent video frames is presented in fig. 2.1. We start with a semantic segmentation at the previous time step, $h_{t-1}$ and warp it along the optical flow to align it with the segmentation at time $t$, by computing $w_t = \phi_{t-1,t}(h_{t-1})$ where $\phi$ is a mapping of labels along the optical flow. This is fed as the hidden state to a gated recurrent unit (GRU) where the other input is the estimate $x_t$ computed by a single frame CNN for semantic segmentation. The information contained in $w_t$ and $x_t$ has significant redundancy, as one expects from nearby video frames, but in regions where it is hard to find the correct segmentation, or where significant motion occurs between frames, they might contain complementary roles. The final segmentation $h_t$ combines the two segmentations $w_t$ and $x_t$ by means of learnt GRU parameters and should include segments where either of the two are very confident. Our model is end-to-end trainable and we can simultaneously refine the GRU parameters $\theta_g$, the parameters of the static semantic segmentation network $\theta_s$ and the parameters of the FlowNet $\theta_f$.

Our overall video architecture can operate over multiple timesteps both forward and backward with respect to the timestep $t$, say, where semantic estimates are obtained. The illustration of this mechanism is shown in fig. 2.2. In *training*, the model has the desirable property that it can rely only on sparsely labeled video frames, but can take advantage of the temporal coherency in the unlabeled video neighborhoods centered at the ground truth. Specifically, given an estimate of our static (per-image) semantic segmentation model at timestep $t$, as well

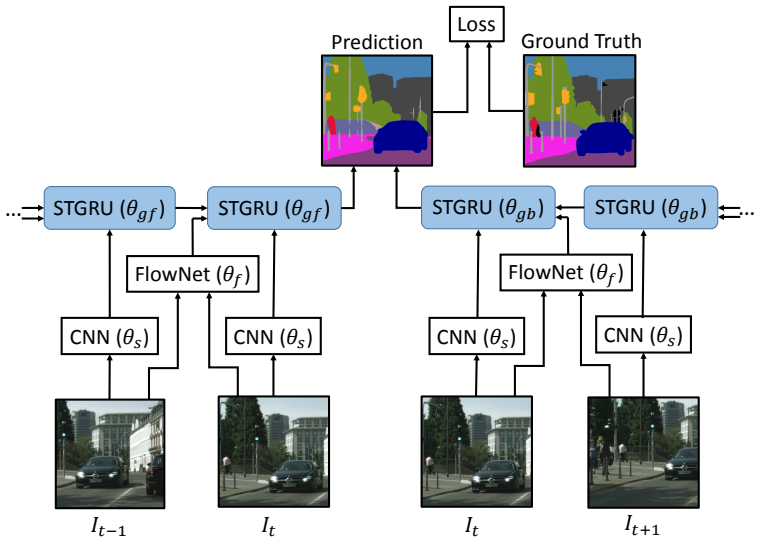


**Figure 2.2:** Illustration of our temporal architecture entitled Gated Recurrent Flow Propagation (GRFP) based on Spatio-Temporal Transformer Gated Recurrent Units (STGRU), illustrated in fig.2.1. The model can integrate both forward only and forward-backward calculations, under separate recurrent units with different parameters $\theta_{gf}$ (forward) and $\theta_{gb}$ (backward). Each of the forward and backward recurrent units have tied parameters across timesteps. The parameters of the semantic segmentation architecture ($\theta_s$) and FlowNet ($\theta_f$) are shared over time. The predictions from the forward model aggregated over frames $t-T,\ldots,t-1,t$ (in the above illustration $T=1$) and (when available and desirable) backward model aggregated over frames $t+T,\ldots t+1,t$ are fused at the central location $t$ in order to make a prediction that is compared against the ground truth available only at frame $t$ by means of a semantic segmentation loss function.

as estimates prior and posterior to it, we can warp these using the confidence gated optical flow forward and backward in time (using the Spatio-Temporal Transformer Gated Recurrent Unit, STGRU, illustrated in fig. 2.1) towards timestep $t$ where ground truth information is available, then fuse estimates in order to obtain a prediction. The resulting model is conveniently differentiable. The loss signal will then be used to backpropagate information for training both the parameters of the gated recurrent units ($\theta_{gf}$, $\theta_{gb}$), the parameters of the (per-frame) semantic segmentation network ($\theta_s$) and the parameters of the FlowNet ($\theta_f$). In *testing* the network can operate either statically, per frame, or take advantage of video frames prior and (if available) posterior to the current processing timestep.

Given these intuitions we will now describe the main components of our model: the spatio-temporal transformer warping and the gated recurrent unit, and then describe implementation and training details.

## 3.1 Spatio-Temporal Transformer Warping

We will use optical flow as input to warp the semantic segmentation estimates across successive frames. We extend the spatial transformer network [18] to operate in the spatio-temporal video domain. Elements on a two-dimensional grid $x_{ij}$ will map to $y_{ij}$ according to

$$y_{ij} = \sum_{m,n} x_{mn} k(i + f^y_{ij} - m, j + f^x_{ij} - n), \tag{2.1}$$

where $(f^x_{ij}, f^y_{ij})$ is the optical flow vector for the pixel at location $(i, j)$. We will use a bilinear interpolation kernel $k(x, y) = \max(0, 1 - |x|) \max(0, 1 - |y|)$. The mapping is differentiable and we can backpropagate gradients from $y$ to both $x$ and $f$. The sum contains only 4 non-zero terms when using the bilinear kernel, so it can be computed efficiently. The methodology has been introduced earlier by us [33] and also independently in [17, 30, 10].

## 3.2 GRUs for Semantic Video Segmentation

To connect the probability maps for semantic segmentation at different timesteps, $h_{t-1}$ and $h_t$, we will use a modified convolutional version of the Gated Recurrent Unit [5]. In particular, we will design a gating function based on the flow, so we only trust the semantic segmentation values warped from $h_{t-1}$ at locations where the flow is certain. We also use gating to predict the new segmentation probabilities, taking into account if either $h_{t-1}$ or $x_t$ have high confidence for a certain class in some region of the image.

To adapt a generic GRU for semantic video segmentation, we first change all fully connected layers to convolutions. The hidden state $h_t$ and the input variable $x_t$ are no longer vectors but tensors of size $H \times W \times C$ where $H$ is the image height, $W$ is the image width and $C$ is the number of channels, corresponding to the different semantic classes. The input $x_t$ is normalized using softmax and $x_t(i, j, c)$ models the probability that label is $c$ for pixel $(i, j)$. We let $\phi_{t-1,t}(x)$ denote the warping of a feature map $x$ from time $t - 1$ to $t$, using optical flow given as additional input, as described in section 3.1. The proposed adaptation of the GRU for semantic video segmentation is

$$w_t = \phi_{t-1,t}(h_{t-1}) \tag{2.2}$$
$$r_t = 1 - \tanh(|W_{ir} * (I_t - \phi_{t-1,t}(I_{t-1})) + b_r|) \tag{2.3}$$
$$\tilde{h}_t = W_{xh} * x_t + W_{hh} * (r_t \odot w_t) \tag{2.4}$$
$$z_t = \sigma(W_{xz} * x_t + W_{hz} * (r_t \odot w_t) + b_z) \tag{2.5}$$
$$h_t = \text{softmax}(\lambda(1 - z_t) \odot r_t \odot w_t + z_t \odot \tilde{h}_t), \tag{2.6}$$

where $W$ and $b$ denote trainable convolution weights, and biases, respectively. Instead of relying on a generic parametrization for the reset gate $r_t$, we use a confidence measure for

the flow by comparing the image $I_t$ with the warped image of $I_{t-1}$. We also discard $\tanh$ when computing $\tilde{h}_t$ and instead use softmax in order to normalize $h_t$. We multiply with a trainable parameter $\lambda$ in order to compensate for a possibly different scaling of $\tilde{h}_t$ relative to the warped $h_{t-1}$ due to the convolutions with $W_{hh}$ and $W_{xh}$. Note that $h_{t-1}$ only enters when we compute the warping $w_t$ so we only use the warped $h_{t-1}$, i.e. $w_t$.

### 3.3 Implementation

For the static (per-frame) component of our model, we rely on a deep neural network pre-trained on the CityScapes dataset and fed as input to the gated recurrent units. We conducted experiments using the Dilation architecture [44], LRR [12] and PSP [45]. The convolutions in the STGRU were all of size $7 \times 7$. We use the standard log-likelihood loss for semantic segmentation

$$L(\theta) = -\sum_{i,j} \log p(y_{ij} = c_{ij} | I, \theta) \tag{2.7}$$

where $p(y_{ij} = c_{ij} | I, \theta)$ is the softmax normalized output of the STGRU, estimating the probability of the correct class $c_{ij}$ for the pixel at $(i, j)$. The recurrent network was optimized using Adam [22] with $\beta_1 = 0.95$, $\beta_2 = 0.99$ and learning rate $2 \cdot 10^{-5}$. Due to GPU memory constraints, the per-frame semantic segmentation CNN computations had to be performed one frame at a time with only the final output saved in memory. When training the system end-to-end the intermediate activations for each frame had to be recomputed. We used standard gradient descent with momentum for the experiments where the static networks or flow networks were refined, with learning rate $2 \cdot 10^{-12}$ and momentum $0.95$. Note that the loss was not normalized, hence the small learning rate. We used FlowNet2 [17] for all experiments unless otherwise stated.

**Default setup** The GRFP model we use is, unless otherwise stated, a forward model trained using 5 frames ($T = 4$ in fig. 2.2) where the parameters of the STGRU $\theta_{gf}$ and the parameters of the static segmentation CNN $\theta_s$ are refined, while the parameters of the FlowNet $\theta_f$ are frozen.

## 4 Experiments

We perform an extensive evaluation on the challenging CityScapes and CamVid datasets, where video experiments nevertheless remain difficult to perform due to the large volume of computation. We evaluate under two different perspectives, reflecting the relevant, key aspects of our method. First we evaluate semantic video segmentation. We will compare our method with other methods for semantic segmentation and show that by using temporal

information we can improve segmentation accuracy over a network where the predictions are computed per frame and unlabeled video data is not used. In the second evaluation we use our method to compute semantic segmentations for all frames in a longer video. We will then compare its temporal consistency against the baseline method where the predictions are performed per frame. We will show quantitatively that our method gives a temporally more consistent segmentation compared to the baseline.

## 4.1 Semantic Video Segmentation

CityScapes [6] consists of sparsely annotated frames. Each labeled frame is the 20th frame in a 30 frame video snippet. There is a total of 2,975 labelled frames in the training set, 500 in the validation set and 1,525 in the test set. We use a forward model with 5 frames and apply the loss to the final frame. Notice however that due to computational considerations, while the STGRU unit parameters $\theta_{gf}$ were trained based on propagating information from 5 frames, the unary network parameters $\theta_s$ were refined based on back-propagating gradient from the 3 STGRU units closest to the loss. The images had size $512 \times 512$ in training, whereas in testing their size was increased to the full resolution $1024 \times 2048$ as more memory was available compared to the training setup.

We used Dilation10 [44], LRR [12] or PSP [45] as backend to our model. We obtain improved performance by using the proposed video methodology compared to the static per-frame baseline for all deep architectures used for static processing. We show the results on the validation set in table 2.1. In this experiment, we only refined the parameters of the GRU and not the parameters of the PSP network.

In table 2.2 we show semantic segmentation results of our model on the CityScapes test set, along with the performance of a number of state of the art static semantic segmentation models.

We used our GRFP methodology trained using Dilation10, LRR-4x and PSP as baseline models and in all cases we show improved labelling accuracy. Notice that our methodology can be used with any semantic segmentation method that processes each frame independently. Since we showed improvements using all baselines, we can predict that other single-frame methods can benefit from our proposed video methodology as well.

In table 2.3 we show the mean IoU over classes versus the number of frames used for inference for our model based on Dilation10. One can see that under the current representation, *in inference*, not much gain is achieved by the forward model beyond propagating information from 4 frames. The results are presented in more detail in table 2.4 where we show the estimates produced by the pre-trained Dilation10 network and the per-frame Dilation network with parameters refined by our model, GRFP(1), as well as the results of our GRFP

**Table 2.1:** Average class (cls) and category (cat) IoU on the CityScapes validation set for various single frame baselines we tried our model GRFP on. By using our video methodology we can see labelling improvements for all baselines we tried, showing that our method is applicable to many different single frame semantic segmentation CNNs. With SSc and MSc we mean single scale and multi scale testing, see [45] for details.

| Method | IoU cls | IoU cat |
|---|---|---|
| GRFP(PSP-MSc, FlowNet2) | 81.3 | 90.7 |
| PSP-MSc [45] | 80.9 | 90.5 |
| GRFP(PSP-SSc, FlowNet2) | 80.2 | 90.2 |
| PSP-SSc [45] | 79.7 | 89.9 |
| GRFP(LRR-4x, FlowNet2) | 73.6 | 88.3 |
| LRR-4x [12] | 72.5 | 87.8 |
| GRFP(Dilation10, FlowNet2) | 69.5 | 86.4 |
| Dilation10 [44] | 68.7 | 86.3 |

**Table 2.2:** Average class (cls) and category (cat) IoU on the CityScapes test set. We use Dilation10, LRR-4x and PSP as baselines, and we are able to improve the average class IoU with 1.0, 1.1 and 0.4 percentage points, respectively. Notice that our GRFP methodology proposed for video is applicable to most of the other semantic segmentation methods that predict each frame independently – they can all benefit from potential performance improvements at no additional labeling cost.

| Method | IoU cls | IoU cat |
|---|---|---|
| GRFP(PSP-Msc, FlowNet2) | 80.6 | 90.8 |
| NetWarp [10] | 80.5 | 91.0 |
| PSP-Msc [45] | 80.2 | 90.6 |
| PEARL [21] | 75.4 | 89.2 |
| GRFP(LRR-4x, FlowNet2) | 72.9 | 88.6 |
| LRR-4x [12] | 71.8 | 88.4 |
| Adelaide_context [28] | 71.6 | 87.3 |
| DeepLabv2-CRF [4] | 70.4 | 86.4 |
| GRFP(Dilation10, FlowNet2) | 68.1 | 86.6 |
| Dilation10 [44] | 67.1 | 86.5 |
| DPN [31] | 66.8 | 86.0 |
| FCN 8s [32] | 65.3 | 85.7 |

model operating over 5 frames GRFP(5). Notice that while the average of our GRFP(1) is almost identical to the one of the pre-trained Dilation10, the individual class accuracies are different. It is apparent that most of our gains come from contributions due to temporal propagation and consistency reasoning in our STGRU models.

Figure 2.4 shows several illustrative situations where our proposed GRFP methodology outperforms the single frame Dilation10 baseline. In particular, our method is capable to more accurately segment the car, the right wall, and the left pole. In all cases it is apparent that inference for the current frame becomes easier when information is integrated over a longer temporal window, as in GRFP. in fig. 2.3 we show several illustrative examples of the flow gat-

**Table 2.3:** Average class (cls) and category (cat) IoU on the validation set of CityScapes when using a different number of frames for inference. The model was trained using 5 frames. Notice that using more than one frame improves performance which for this dataset, however, saturates beyond 4 frames.

| Frames | IoU cls | IoU cat |
|---|---|---|
| 1 | 68.8 | 86.3 |
| 2 | 69.2 | 86.4 |
| 3 | 69.4 | 86.4 |
| 4 | 69.5 | 86.4 |
| 5 | 69.5 | 86.4 |

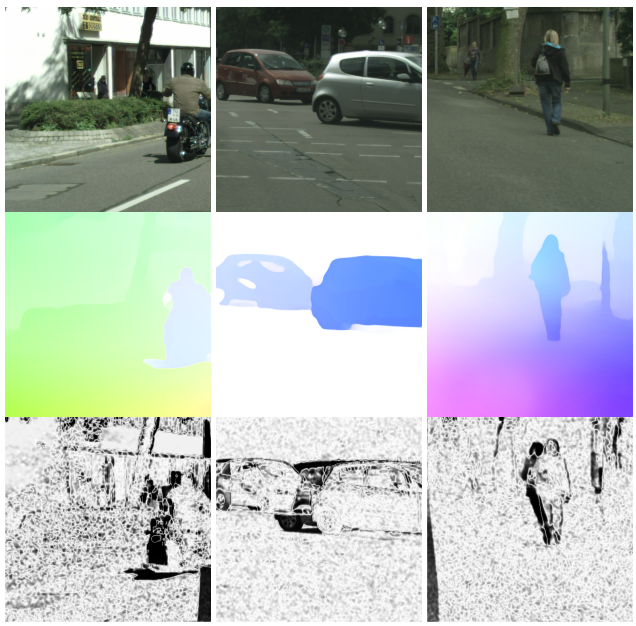

**Figure 2.3:** Illustration of the flow gating as estimated by our Spatio-Temporal Transformer Gated Recurrent Unit. We show three examples each containing a video frame, the optical flow and its confidence as estimated by our STGRU model. White regions indicate a confident flow estimate ($r_t = 1$) whereas black regions are uncertain ($r_t = 0$). Occluded regions are black, as expected.

ing learned by our STGRU units. Notice that the areas our model learns to discard ($r_t = 0$) correspond to occluded regions.

**Combining forward and backward models.** In table 2.5 we show the accuracy on the CityScapes validation set for various settings where we used the forward and backward models and averaged the predictions using Dilation10. This joint model was described in fig. 2.2. The best results were obtained when a forward-backward model was trained by averaging the predictions using 5 frames going forward (using $I_{t-4}, I_{t-3}, \ldots, I_t$) with the predictions using 5 frames going backward (using $I_{t+4}, I_{t+3}, \ldots, I_t$). Better results were obtained when the forward and backward models were trained jointly, and not independently.

**Table 2.4:** Average class IoUs on the CityScapes validation set for the Dilation10 baseline, the GRFP model using 5 frames, GRFP(5), and the refined Dilation10 net that the GRFP learns, which is equivalent to GRFP(1).

| Class | Dilation10 | GRFP(5) | GRFP(1) |
|---|---|---|---|
| Road | 97.2 | 97.3 | 97.1 |
| Sidewalk | 79.5 | 80.1 | 79.2 |
| Building | 90.4 | 90.5 | 90.4 |
| Wall | 44.9 | 50.6 | 46.8 |
| Fence | 52.4 | 53.3 | 53.0 |
| Pole | 55.1 | 55.3 | 55.2 |
| Traffic light | 56.7 | 57.5 | 56.7 |
| Traffic sign | 69.0 | 68.7 | 68.9 |
| Vegetation | 91.0 | 91.1 | 91.0 |
| Terrain | 58.7 | 59.6 | 58.7 |
| Sky | 92.6 | 92.7 | 92.5 |
| Person | 75.7 | 76.2 | 75.7 |
| Rider | 50.0 | 50.3 | 50.1 |
| Car | 92.2 | 92.4 | 92.2 |
| Truck | 56.2 | 57.4 | 55.8 |
| Bus | 72.6 | 73.9 | 72.8 |
| Train | 53.2 | 53.4 | 54.5 |
| Motorcycle | 46.2 | 48.8 | 46.3 |
| Bicycle | 70.1 | 71.0 | 70.4 |
| Average | 68.7 | 69.5 | 68.8 |

**Table 2.5:** Average class IoU on the CityScapes validation set for various forward-backward models and for models refining FlowNet2 and training end-to-end. See Fig. 2.2 for how the parameters are defined. The first three models are described in more detail in table 2.4. **(a)** A forward-backward model evaluated with $T = 2$ using the same parameters for the backward STGRU as the best forward model GRFP(5) both in the forward direction and backward direction. We used $\theta_s$ and $\theta_{gf}$ as for GRFP(5) and set $\theta_{gb} = \theta_{gf}$. **(b)** as (a) but with $T = 4$. **(c)** We used $\theta_{gf}$ and $\theta_s$ from GRFP(5) but refined $\theta_{gb}$ independently. We used $T = 4$. **(d)** We refined all parameters $\theta_{gf}$, $\theta_{gb}$ and $\theta_s$ jointly and used $T = 4$. **(e)** We used the setting in GRFP(5) and trained the Dilation10 network, the flow network and the recurrent network jointly. It was evaluated in forward mode with $T = 4$.

| Method | Dilation10 | GRFP(5) | GRFP(1) | fwbw(a) | fwbw(b) | fwbw(c) | fwbw(d) | FlowNet2(e) |
|---|---|---|---|---|---|---|---|---|
| IoU | 68.7 | 69.5 | 68.8 | 69.5 | 69.6 | 69.6 | 69.8 | 69.5 |

**Joint training including optical flow.** To make our model entirely end-to-end trainable we also refine its optical flow component. In training, we jointly estimate all the component STGRU, Dillation10 and Flownet2 parameters. The model produced competitive results, see (e) in table 2.5, although the performance was not improved compared to models where we only refined the parameters of the static network and those of the STGRU. We note that the error signal passed to the FlowNet comes from a loss based on semantic segmentation. This is a very weak form of supervision for refining optical flow.

**Table 2.6:** Average IoU on the test set of CamVid for different video segmentation methods all based on the per-frame Dilation8 CNN. Note that out method GRFP obtains a higher score than the static Dilation8 model it is based on.

| Method | IoU |
|---|---|
| Dilation8 [44] | 65.3 |
| FSO [25] | 66.1 |
| GRFP(Dilation8, FlowNet2) | 66.1 |
| VPN [19] | 66.7 |
| NetWarp [10] | 67.1 |

**Table 2.7:** Temporal consistency (%) for demo videos Stuttgart_00, Stuttgart_01 and Stuttgart_02 in the CityScapes dataset. Notice that our GRFP semantic video segmentation method achieves a more consistent solution than single frame baselines.

| Method | Vido | Vid1 | Vid2 | Avg |
|---|---|---|---|---|
| GRFP(PSP-SSc, FlowNet2) | 88.15 | 90.99 | 85.64 | 88.26 |
| PSP-Ssc | 85.82 | 88.93 | 82.92 | 85.89 |
| GRFP(LRR-4x, FlowNet2) | 84.78 | 88.73 | 81.72 | 85.08 |
| LRR-4x | 80.74 | 86.22 | 77.88 | 81.61 |
| FSO [25] | 91.31 | 93.32 | 89.01 | 91.21 |
| GRFP(Dilation10, FlowNet2) | 84.29 | 88.87 | 81.96 | 85.04 |
| Dilation10 | 79.18 | 86.13 | 76.77 | 80.69 |

**CamVid** To show that our method is not limited to CityScapes we also provide additional experiments on the CamVid dataset [2]. This dataset consists of 4 videos that are annotated at 1 Hz. We follow the setup in [44, 25] where all images are downsampled to $480 \times 640$, and we use the same split with 367 training images, 100 validation images and 233 test images. We use our GRFP methodology with Dilation8 [44] as static network and FlowNet2 [17] as flow network. The results are shown in table 2.6. We can see that the segmentation accuracy is improved by using additional video frames as input, and our labeling results are on par with the state-of-the-art[25]. Qualitative examples are shown in fig. 2.5.

**Temporal Consistency** We evaluate the temporal consistency of our semantic video segmentation method by computing trajectories in video using [40] and calculating for how many of the trajectories the labelling is the same in all frames, following the evaluation methodology in [25]. We use the demo videos provided in the CityScapes dataset, that are 600, 1100 and 1200 frames long, respectively. Due to computational considerations, we only used the middle $512 \times 512$ crop of the larger CityScapes images. The results are given in table 2.7 where improvements are achieved for all videos and for all methods compared to per-frame baselines. This can also be seen qualitatively in the videos provided in the supplementary

**Table 2.8:** Timing of the different components of out GRFP methodology for a Titan X GPU. We show improved segmentation accuracy and temporal consistency by incurring an additional runtime of 75 ms per frame if we use FlowNet1 or 335 ms per frame if we use FlowNet2.

| | Dilation10 | LRR-4x |
|---|---|---|
| Segmentation module | 350 ms | 200 ms |
| FlowNet2/FlowNet1 | 300/40 ms | 300/40 ms |
| STGRU | 35 ms | 35 ms |

**Table 2.9:** Assessing the robustness of our methodology w.r.t. optical flow quality. We report the average class IoU on the validation set of CityScapes. The optical flows computed using FlowNet1[9] or the method of Farnebäck[8] have lower accuracy than FlowNet2, yet our GRFP methodology can still improve the labelling accuracy over per-frame processing baselines.

| Method | IoU cls |
|---|---|
| GRFP(LRR-4x, FlowNet2) | 73.6 |
| GRFP(LRR-4x, FlowNet1) | 73.4 |
| GRFP(LRR-4x, Farnebäck) | 73.0 |
| LRR-4x | 72.5 |

material[1]. There is significantly less flickering and noise when using the proposed GRFP semantic video segmentation methodology compared to models that rely on single-frame estimates. Note that while the temporal consistency is lower for our method compared to FSO, the run-time is significantly faster. Our method takes about 0.7 s per frame while FSO takes more than 10s per frame.

**Timing.** We report timings for the different components of our method when using a Titan X GPU. We report results for both FlowNet1 [9] and FlowNet2 [17]. Table 2.8 show timings per frame for the three main components of our framework: the static prediction, the flow, and the STGRU computations. We report the time to process (testing, not training) one frame in a video with resolution $512 \times 512$. With the proposed methodology we achieve both improved temporal consistency and labelling accuracy at an additional runtime cost of 335 ms per frame with FlowNet2 and 75 ms with FlowNet1.

**Effect of Low Optical Flow Quality.** To assess the robustness of our methodology to inaccurate optical flow modules we perform experiments where Flownet1[9] or the optical flow method of Farnebäck [8] were used at test time instead of (the most competitive) FlowNet2 for a GRFP model trained with LRR-4x and FlowNet2. The average end-point-error (EPE) on KITTI 2012 is 25.3 pixels for Farnebäck, 9.1 pixels for FlowNet1, whereas the best FlowNet2 model has an EPE of 1.8 pixels. Based on results in table 2.9 we conclude that our method can compensate and still improve over per-frame processing baselines, even when the optical flow has significantly lower quality then FlowNet2.

[1]Available at `https://openaccess.thecvf.com/content_cvpr_2018/Supplemental/2131-supp.zip`

# 5 Conclusions

We have presented a deep, end-to-end trainable methodology for semantic video segmentation – including the capability to jointly refine the static recognition, optical flow and temporal propagation modules –, that is capable of taking advantage of the information present in unlabeled frames in order to improve estimates. Our model combines a convolutional architecture and a spatio-temporal transformer recurrent layer that learns to temporally propagate semantic segmentation information by means of optical flow, adaptively gated based on its locally estimated uncertainty. Our experiments on the challenging CityScapes and CamVid datasets, and for different deep semantic components, indicate that our resulting model can successfully propagate information from labeled video frames towards nearby unlabeled ones, in order to improve both the accuracy of the semantic video segmentation and the consistency of its temporal labeling, at no additional annotation cost, and with little supplementary computation.

**Acknowledgments:** This work was supported by the European Research Council Consolidator grant SEED, CNCS-UEFISCDI PN-III-P4-ID-PCE-2016-0535, the EU Horizon 2020 Grant DE-ENIGMA, and SSF.

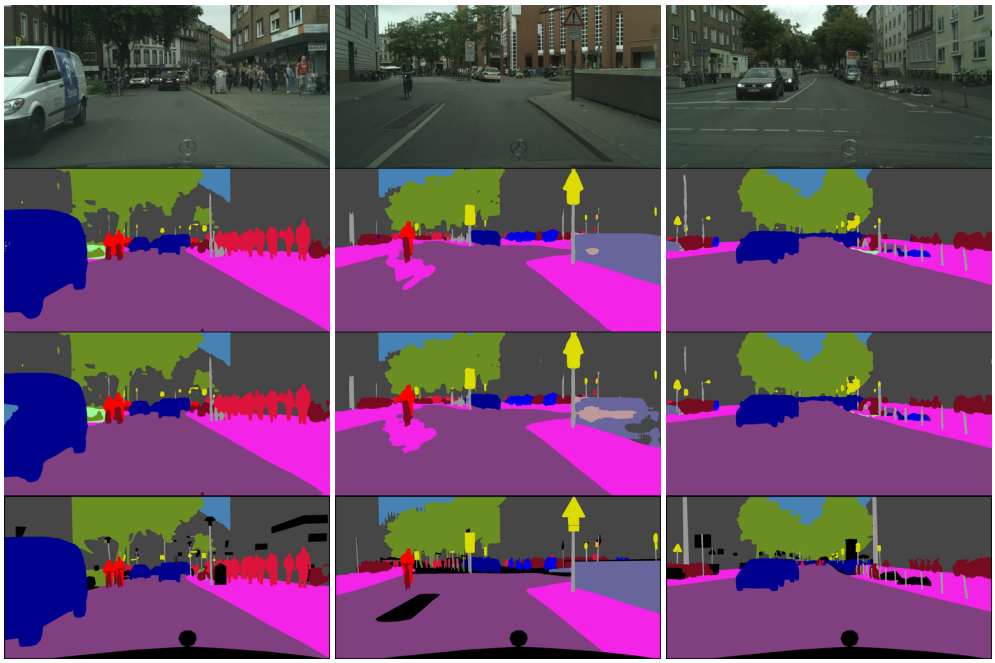

**Figure 2.4:** From top to bottom: the image, video segmentation by GRFP, static segmentation by Dilation10, and the ground truth. Notice the more accurate semantic segmentation of the car in the left example, the right wall in the middle example, and the left pole in the right example. For the first two examples the static method fails where the some object has a uniform surface over some spatial extent. The pole in the right image may be hard to estimate based on the current frame alone, but the inference problem becomes easier if earlier frames are considered, a property our GRFP model has.

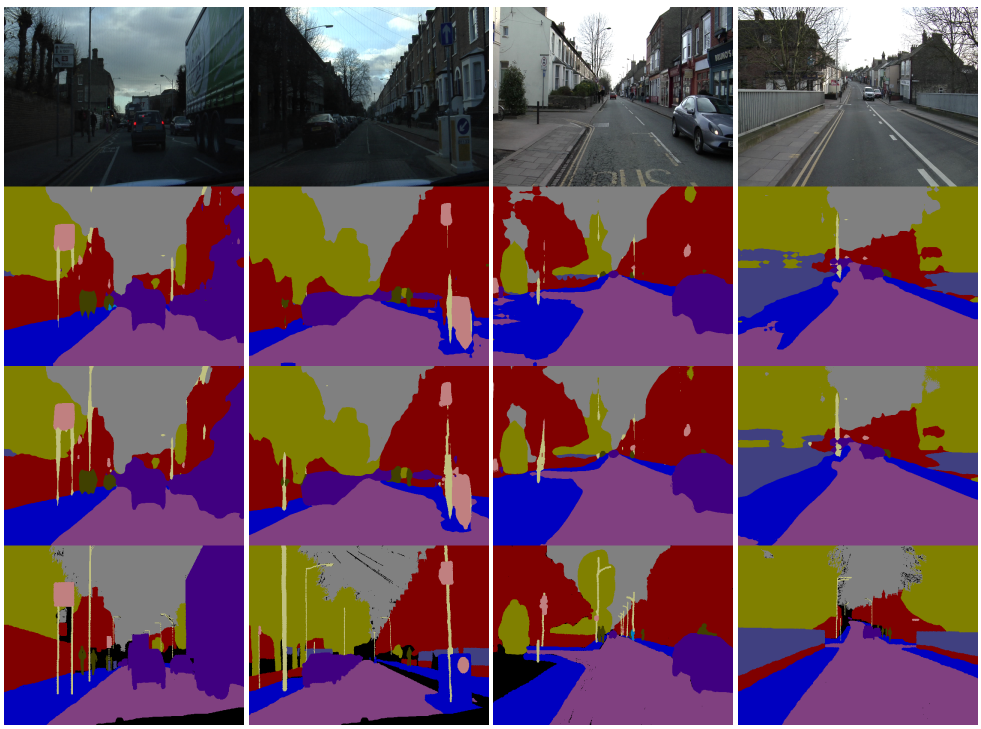

**Figure 2.5:** Qualitative examples from the CamVid test set. From top to bottom: the image, static segmentation by Dilation8, video segmentation by GRFP, and the ground truth. In the two examples to the left, notice that poles are better segmented by our video method. In the two right images, the sidewalks are better segmented using video.

# Bibliography


[1] D. Banica, A. Agape, A. Ion, and C. Sminchisescu. Video object segmentation by salient segment chain composition. In *ICCV, IPGM Workshop*, 2013.

[2] G. J. Brostow, J. Fauqueur, and R. Cipolla. Semantic object classes in video: A high-definition ground truth database. *Pattern Recognition Letters*, 30(2):88–97, 2009.

[3] J. Carreira and A. Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *CVPR*, 2017.

[4] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Semantic image segmentation with deep convolutional nets and fully connected crfs. In *ICLR*, 2015.

[5] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[6] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. The cityscapes dataset for semantic urban scene understanding. In *CVPR*, 2016.

[7] M. Everingham, S. A. Eslami, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The pascal visual object classes challenge: A retrospective. *IJCV*, 2015.

[8] G. Farnebäck. Two-frame motion estimation based on polynomial expansion. *Image analysis*, pages 363–370, 2003.

[9] P. Fischer, A. Dosovitskiy, E. Ilg, P. Häusser, C. Hazırbaş, V. Golkov, P. van der Smagt, D. Cremers, and T. Brox. Flownet: Learning optical flow with convolutional networks. In *ICCV*, 2015.

[10] R. Gadde, V. Jampani, and P. V. Gehler. Semantic video cnns through representation warping. In *ICCV*, 2017.

[11] F. Galasso, N. Nagaraja, T. Cardenas, T. Brox, and B.Schiele. A unified video segmentation benchmark: Annotation, metrics and analysis. In *ICCV*, 2013.

[12] G. Ghiasi and C. C. Fowlkes. Laplacian pyramid reconstruction and refinement for semantic segmentation. In *ECCV*, 2016.

[13] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *NIPS*, 2014.

[14] M. Grundmann, V. Kwatra, M. Han, and I. Essa. Efficient hierarchical graph-based video segmentation. In *CVPR*, 2010.

[15] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[16] Y. He, W.-C. Chiu, M. Keuper, and M. Fritz. Std2p: Rgbd semantic segmentation using spatio-temporal data-driven pooling. *CVPR*, 2017.

[17] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox. Flownet 2.0: Evolution of optical flow estimation with deep networks. In *CVPR*, 2017.

[18] M. Jaderberg, K. Simonyan, A. Zisserman, et al. Spatial transformer networks. In *NIPS*, 2015.

[19] V. Jampani, R. Gadde, and P. V. Gehler. Video propagation networks. *CVPR*, 2017.

[20] S. Ji, W. Xu, M. Yang, and K. Yu. 3d convolutional neural networks for human action recognition. In *PAMI*, 2013.

[21] X. Jin, X. Li, H. Xiao, X. Shen, Z. Lin, J. Yang, Y. Chen, J. Dong, L. Liu, Z. Jie, et al. Video scene parsing with predictive feature learning. *ICCV*, 2017.

[22] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[23] P. Krähenbühl and V. Koltun. Efficient inference in fully connected crfs with gaussian edge potentials. In *NIPS*, 2011.

[24] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.

[25] A. Kundu, V. Vineet, and V. Koltun. Feature space optimization for semantic video segmentation. In *CVPR*, 2016.

[26] J. Lezama, K. Alahari, J. Sivic, and I. Laptev. Track to the future: Spatio-temporal video segmentation with long-range motion cues. In *CVPR*, 2011.

[27] F. Li, T. Kim, A. Humayun, D. Tsai, and J. M. Rehg. Video segmentation by tracking many figure-ground segments. In *CVPR*, 2013.

[28] G. Lin, C. Shen, I. Reid, et al. Efficient piecewise training of deep structured models for semantic segmentation. In *CVPR*, 2016.

[29] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *ECCV*, 2014.

[30] S. Liu, C. Wang, R. Qian, H. Yu, and R. Bao. Surveillance video parsing with single frame supervision. In *CVPR*, 2017.

[31] Z. Liu, X. Li, P. Luo, C.-C. Loy, and X. Tang. Semantic image segmentation via deep parsing network. In *ICCV*, 2015.

[32] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015.

[33] D. Nilsson and C. Sminchisescu. Semantic video segmentation by gated recurrent flow propagation. *arXiv preprint arXiv:1612.08871*, 2016.

[34] P. Ochs, J. Malik, and T. Brox. Segmentation of moving objects by long term video analysis. *PAMI*, 2014. doi: 10.1109/TPAMI.2013.242.

[35] A. Papazoglou and V. Ferrari. Fast object segmentation in unconstrained video. In *ICCV*, 2013. ISBN 978-1-4799-2840-8. doi: 10.1109/ICCV.2013.223.

[36] F. Perazzi, J. Pont-Tuset, B. McWilliams, L. Van Gool, M. Gross, and A. Sorkine-Hornung. A benchmark dataset and evaluation methodology for video object segmentation. In *CVPR*, 2016.

[37] E. Shelhamer, K. Rakelly, J. Hoffman, and T. Darrell. Clockwork convnets for video semantic segmentation. In *Computer Vision–ECCV 2016 Workshops*, pages 852–868. Springer, 2016.

[38] J. Shi and J. Malik. Normalized cuts and image segmentation. *PAMI*, 2000. ISSN 0162-8828. doi: 10.1109/34.868688.

[39] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.

[40] N. Sundaram, T. Brox, and K. Keutzer. Dense point trajectories by gpu-accelerated large displacement optical flow. In *ECCV*, 2010.

[41] S. Tripathi, S. Belongie, Y. Hwang, and T. Nguyen. Semantic video segmentation: Exploring inference efficiency. In *SoC Design Conference (ISOCC), 2015 International*, pages 157–158. IEEE, 2015.

[42] D. Tsai, M. Flagg, A. Nakazawa, and J. M. Rehg. Motion coherent tracking using multi-label MRF optimization. *IJCV*, 2012. doi: 10.1007/s11263-011-0512-5.

[43] C. Xu, C. Xiong, and J. J. Corso. Streaming hierarchical video segmentation. In *ECCV*, 2012.

[44] F. Yu and V. Koltun. Multi-scale context aggregation by dilated convolutions. In *ICLR*, 2016.

[45] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia. Pyramid scene parsing network. In *CVPR*, 2017.

[46] S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, and P. H. Torr. Conditional random fields as recurrent neural networks. In *ICCV*, 2015.

# Paper II

# Embodied Visual Active Learning for Semantic Segmentation

David Nilsson[1,2], Aleksis Pirinen[1], Erik Gärtner[1,2], Cristian Sminchisescu[1,2]
[1]Department of Mathematics, Faculty of Engineering, Lund University
[2]Google Research

## Abstract

We study the task of *embodied visual active learning*, where an agent is set to explore a 3d environment with the goal to acquire visual scene understanding by actively selecting views for which to request annotation. While accurate on some benchmarks, today's deep visual recognition pipelines tend to not generalize well in certain real-world scenarios, or for unusual viewpoints. Robotic perception, in turn, requires the capability to refine the recognition capabilities for the conditions where the mobile system operates, including cluttered indoor environments or poor illumination. This motivates the proposed task, where an agent is placed in a novel environment with the objective of improving its visual recognition capability. To study embodied visual active learning, we develop a battery of agents – both learnt and pre-specified – and with different levels of knowledge of the environment. The agents are equipped with a semantic segmentation network and seek to acquire informative views, move and explore in order to propagate annotations in the neighbourhood of those views, then refine the underlying segmentation network by online retraining. The trainable method uses deep reinforcement learning with a reward function that balances two competing objectives: *task performance, represented as visual recognition accuracy*, which requires exploring the environment, and the necessary *amount of annotated data* requested during active exploration. We extensively evaluate the proposed models using the photorealistic Matterport3D simulator and show that a fully learnt method outperforms comparable pre-specified counterparts, even when requesting fewer annotations.

# 1 Introduction

Imagine a household robot in a home it has never been before and equipped with a visual sensing module to perceive its environment and localize objects. If the robot fails to recognize some objects, or to adapt to changes in the environment, over time, it may not be able to properly perform its tasks. Much of the recent success of visual perception has been achieved by deep CNNs, e.g. in image classification [23, 41, 15], semantic segmentation [25, 6] and object detection [35, 34]. Such systems may however be challenged by unusual viewpoints or domains, as noted e.g. by Ammirato et al. [2] and Yang et al. [53]. Moreover, a mobile household robot should ideally operate with lightweight, re-trainable and task-specific perception models, rather than large and comprehensive ones, which could be demanding computationally and not tailored to the needs of a specific house.

In practice, even in closed but large environments, developing robust scene understanding by exhaustive approaches may be difficult, as looking everywhere requires an excessive amount of annotation labor. All views are however not equally informative, as a view containing many diverse objects is likely more useful than one covering a single semantic class, e.g. a wall. This suggests that in learning visual perception one does not have to label exhaustively. As new, potentially difficult arrangements appear in an evolving environment, it would be useful to identify those automatically, based on the task and demand, rather than programmatically, by periodically re-training a complete model. Moreover, the agent could make the most out of its embodiment by propagating a given ground truth annotation using motion – as measured by the perceived optical flow – in that neighborhood. The agent can then self-train, online, for increased performance. The key questions are how should one explore the environment, how to select the most informative views to annotate, and how to make the most out of them. We analyze these questions in an *embodied visual active learning* framework, illustrated in fig. 3.1.

To ground the embodied visual active learning task, in this work we measure visual perception ability as semantic segmentation accuracy. The agent is equipped with a semantic segmentation system and must move around and request annotations in order to refine it. After exploring the scene the agent should be able to accurately segment all views in the explored area. This requires an exploration policy covering different objects from diverse viewpoints and selecting sufficiently many annotations to train the perception model. The agent can also propagate annotations to different nearby viewpoints using optical flow and then self-train. We develop a battery of methods, ranging from pre-specified ones to a fully trainable deep reinforcement learning-based agent, which we evaluate extensively in the photorealistic Matterport3D environment [4].

In summary, our main contributions are:

- We study the task of *embodied visual active learning*, where an agent should explore

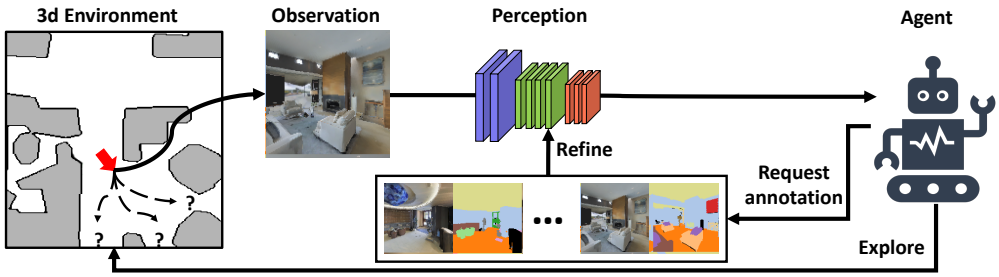


**Figure 3.1: Embodied visual active learning**. An agent in a 3d environment must explore and occasionally request annotation in order to efficiently refine its visual perception. The navigation component makes this task significantly more complex than traditional active learning, where the data pool over which the agent queries annotations, either in the form of image collections or pre-recorded video streams, is static and given.

a 3d environment to acquire visual scene understanding by actively selecting views for which to request annotation. The agent then propagates information by moving in the neighborhood of those views and self-trains;

- In our setup, visual learning and exploration can inform and guide one another since the recognition system is selectively and gradually refined during exploration, instead of being trained at the end of a trajectory on a full set of densely annotated views;
- We develop a variety of methods, both learnt and pre-specified, to tackle our task in the context of semantic segmentation;
- We perform extensive evaluation in a photorealistic 3d environment and show that a fully learnt method outperforms comparable pre-specified ones.

## 2 Related Work

The embodied visual active learning setup leverages several computer vision and machine learning concepts, such as embodied navigation, active learning and active vision. There is substantial recent literature on embodied agents navigating in real or simulated 3d environments, especially given the recent emergence of large-scale simulators [37, 22, 49, 9, 36].

We here briefly review variants of embodied learning. In Embodied Question Answering [8, 47, 54], an agent is given a question, e.g. "What color is the car?". The agent must typically explore the environment quite extensively in order to be able to answer. Zhu et al. [57], Mousavian et al. [28] task the agent with reaching a target view using as few steps as possible. The agent receives the current view and the target as inputs in each step. In point-goal navigation [27, 38, 37, 14] the agent is given coordinates of a target to reach using visual information and ego-motion. In visual exploration [33, 10, 7, 32, 55, 5] the task is to explore

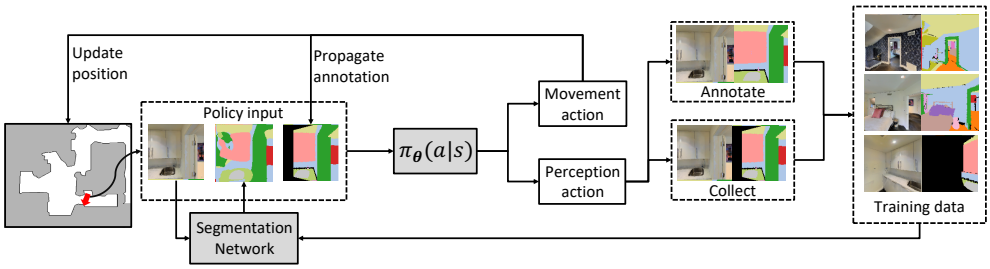


**Figure 3.2: Embodied visual active learning for semantic segmentation**. A first-person agent is placed in a room and a deep network predicts the semantic segmentation of the agent's view. Based on the view and its segmentation, the agent can either select a *movement action* to change position and viewpoint, or select a *perception action* (`Annotate` or `Collect`). `Annotate` adds the current view and its ground truth segmentation to the pool of training data for the segmentation network, while `Collect` is a cheaper version (no additional supervision required) where the current view and the last annotated view – propagated to the agent's current position using optical flow – is added to the training set. The propagated annotation is also a policy input for the learnt agent in §3.3. After a perception action, the segmentation network is refined on the current training set. The embodied visual active learning process is considered successful if, after selecting a limited number of `Annotate` actions or an exploration budget is exhausted, the segmentation network can accurately segment any other view in the environment where the agent operates. Note that the map (left) is *not* provided as input to the learnt agent in §3.3.

an unknown environment as quickly as possible, by covering the whole scene area. In Ammirato et al. [2], Yang et al. [53] an agent is tasked to navigate an environment to increase the accuracy of a pre-trained recognition model, e.g. by moving around occluded objects. This is in contrast to our work where the goal is to collect views for *training* a perception model. Whereas in Ammirato et al. [2], Yang et al. [53] the agent is spawned close to the target object, we cannot make such assumptions, as our task is not only to accurately recognize a single object or view, but to do so for *all* views in the potentially large area explored by the agent.

There are relations to curiosity-driven learning [30, 52], in that we also seek an agent which visits novel views (states). In Pathak et al. [30], exploration is aided by giving rewards based on the prediction error of a self-supervised inverse-dynamics model. This is a task-independent exploration strategy useful to search 2d or 3d environments during training. In our setup, exploration is task-specific in that it is aimed specifically at refining a visual recognition system in a novel environment. Moreover, we use semi-dense rewards for both visual learning and for exploration. Hence we are not operating using sparse rewards where curiosity approaches often outperform other methods.

Our work is also related to Song et al. [42], Pot et al. [31], Zhong et al. [56], Wang et al. [46]. Differently from us, Song et al. [42] uses hand-crafted annotation and exploration strategies, aiming to label all voxels in a 3d reconstruction by selecting a subset of frames covering all voxels. This is a form of exhaustive annotation and a visual perception system is not trained. Hence the system can only analyze objects in annotated voxels. In our setup the agent is instead tasked with both exploration and the selection of views to annotate, and we learn a perception module aiming to generalize to unseen views. In contrast to us, Pot et al. [31], Zhong et al. [56], Wang et al. [46] do not consider an agent choosing where to move in the

environment, nor which parts to label. Instead, they use all views seen when following a pre-specified path for training a visual recognition system. Pot et al. [31] use an object detector obtained by self-supervised learning and clustering. Zhong et al. [56], Wang et al. [46] use constraints from SLAM to improve a given segmentation model. This approach could in principle complement our label propagation, and is orthogonal to our main proposals.

Next-best-view (NBV) prediction [18, 50, 20, 17, 43, 13] is superficially similar to our task. In Jayaraman and Grauman [18] an agent is trained to reveal parts of a panorama and a model is built to complete all views of the panorama. Our setup allows free movement in an environment, hence it features a navigation component which makes our task more comprehensive. While NBV typically integrates information from all predicted views, our task requires the adaptive selection of only a subset of the views encountered during the agent's navigation trajectory.

Active learning [40, 11, 26, 48, 29, 12] can be seen as the static version of our setup, as it considers approaches for learning what parts of a larger *pre-existing* and *static* training set should be fed into the training procedure, and in what order. We instead consider the active learning problem in an embodied setup, where an agent can move and actively select views for which to request annotation. Embodiment makes it possible to use motion to propagate annotations, hence effectively generate new ones at no additional annotation cost. In essence, our work lays groundwork towards marrying the active vision and the active learning paradigms.

# 3 Embodied Visual Active Learning

Embodied visual active learning is an interplay between a first-person agent, a 3d environment and a trainable perception module. See fig. 3.1 for a high-level abstraction and fig. 3.2 for details of the particular task considered in this paper. The perception module processes images (views) observed by the agent in the environment. The agent can request annotations for views in order to refine the perception module. It should ideally request very few annotations as these are costly. The agent can also generate more annotations for free by neighborhood exploration using label propagation, such that when trained on that data the perception module becomes increasingly more accurate in the explored environment. To assess how successful an agent is on the task, we test how accurate the perception module is on multiple random viewpoints selected uniformly in the area explored by the agent.

**Task overview.** The agent begins each episode randomly positioned and rotated in a 3d environment, with a randomly initialized semantic segmentation network. The ground truth segmentation mask for the first view is given for the initial training of the segmentation network. The agent can choose *movement actions* (`MoveForward`, `MoveLeft`, `MoveRight`, `RotateLeft`, `RotateRight` with 25 cm movements and 15 degree rotations), or *perception*

*actions* (`Annotate`, `Collect`). If the agent moves or rotates, the ground truth mask is propagated using optical flow. At any time, the agent may choose to insert the propagated annotation into its training set with the `Collect` action, or to ask for a new ground truth mask with the `Annotate` action. After an `Annotate` action the propagated annotation mask is re-initialized to the ground truth annotation. After each perception action, the segmentation network $\mathcal{S}$ is refined on the training set, which is expanded with the new data point.

The agent's performance is evaluated at the end of the episode. The goal is to maximize the mIoU and mean accuracy of the segmentation network on the views in the area explored by the agent. Specifically, a set of *reference views* are randomly sampled within a disc of radius $r$ centered at the starting location, and the segmentation network is evaluated on these. Hence to perform well the agent is required to explore its surroundings, and it should refine its perception module in regions of high uncertainty.

## 3.1 Methods for the Proposed Task

We develop several methods to evaluate and study the embodied visual active learning task. All methods except the RL-agent issue the `Collect` action when 30% of the propagated labels are unknown and `Annotate` when 85% are unknown. The intuition is that the prespecified methods should request annotation when most pixels are unlabeled. The specific hyperparameters of all models were set based on a validation set.

**Random.** Uniformly selects random movement actions. This baseline is thus a lower bound in terms of embodied exploration for this task.

**Rotate.** Continually rotates left. This method is useful in comparing with trainable agents that move and explore, i.e. to monitor what improvements can be expected from embodiment.

**Bounce.** Explores by walking straight forward until it hits a wall, then samples a new random direction and moves forward until it collides with a new wall, and so on. This agent quickly explores the environment.

**Frontier exploration.** This method builds a map, online, by using using depth and motion from the simulator [51]. All pixels with depth within a 4m threshold are back-projected in 3d and then classified as either obstacles or navigable, based on height relative to the ground plane. This agent is confined to move within the reference view radius $r$, which is a choice to its advantage[2] as annotated views will more likely be similar to reference views that reside within that same radius.

**Space filler.** Follows a shortest space filling curve within the reference view radius $r$, and

[2] This ensures it is evaluated under ideal conditions in contrast to the RL-agent in §3.3.

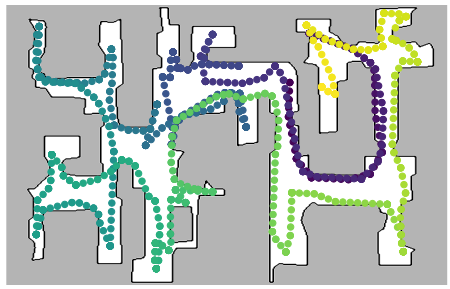

**Figure 3.3:** An example of a space filling curve in a Matterport3D floor plan. Methods based on the space filler assume complete spatial knowledge of the environment.

as $r$ increases the entire environment is explored. This baseline makes strong and somewhat less general (or depending on the application, altogether unrealistic) assumptions in order to create a path: knowing the floor plan in advance, as well as which locations are reachable from the start. It also only moves within the reference view radius, and knows the shortest geodesic paths to take on the curve. Hence, this method can be considered an upper bound for other methods. The space filling curve is computed by placing a grid of nodes onto the floor plan (1m resolution, using a sampling and reachability heuristic), and then finding the shortest path around it with an approximate traveling salesman solver. Fig. 3.3 shows a space filling curve in a Matterport3D floor plan.

**RL-agent.** This fully trainable method we develop jointly learns exploration and perception actions in a reinforcement learning framework. See the full description in §3.3.

## 3.2 Semantic Segmentation Network

Each method uses the same FCN-inspired deep network [25] for semantic segmentation. The network consists of 3 blocks of convolutional layers, each containing 3 convolutional layers with kernels of size $3 \times 3$. The first convolutional layer in each block uses a stride of 2, which halves the resolution. For each block the number of channels doubles, using 64, 128 and 256 channels respectively. Multiple predictions are made using the final convolutional layers of each block. The multi-scale predictions are resized to the original image resolution using bilinear interpolation and are finally summed up, resulting in the final segmentation estimate. Note that we have deliberately chosen to make the network small so that it can be efficiently refined on new data.

At the beginning of each episode, the parameters are initialized randomly, and we train the network on the very first view, for which we always supply the ground truth segmentation. Each time `Annotate` or `Collect` is selected, we refine the network. Mini-batches of size 8, which always include the latest added labeled image, are used in training. We use random cropping and scaling for data augmentation. The network is refined either until it has

trained for 1000 iterations or until the accuracy of a mini-batch exceeds $95\%$. We use a standard cross-entropy loss averaged over all pixels. The segmentation network is trained using stochastic gradient descent with learning rate $0.01$, weight decay $10^{-5}$ and momentum $0.9$. To propagate semantic labels, we compute optical flow between consecutive viewpoints using PWC-Net [44]. The optical flow is computed bidirectionally and only pixels where the difference between the forward and backward displacements is less than 2 pixels are propagated [45]. We found that labels were reliably tracked over several frames when using 2 pixels as a threshold.

## 3.3 Reinforcement Learning Agent

To present the reinforcement-learning agent for our task, we begin with an explanation of the state-action representation and policy network, followed by the reward structure and finally policy training.

**Actions, states and policy.** The agent is represented as a deep stochastic policy $\pi_\theta(a_t|s_t)$ that samples an action $a_t$ in state $s_t$ at time $t$. The actions are `MoveForward`, `MoveLeft`, `MoveRight`, `RotateLeft`, `RotateRight`, `Annotate` and `Collect`. The full state is $s_t = \{\boldsymbol{I}_t, \boldsymbol{S}_t, \boldsymbol{P}_t, \boldsymbol{F}_t\}$ where $\boldsymbol{I}_t \in \mathbb{R}^{127\times127\times3}$ is the image, $\boldsymbol{S}_t = \mathcal{S}_t(\boldsymbol{I}_t) \in \mathbb{R}^{127\times127\times3}$ is the semantic segmentation mask predicted by the deep network $\mathcal{S}_t$ (this network is refined over an episode; $t$ indexes the network parameters at time $t$), $\boldsymbol{P}_t \in \mathbb{R}^{127\times127\times3}$ is the propagated annotation, and $\boldsymbol{F}_t \in \mathbb{R}^{7\times7\times2048}$ is a deep representation of $\boldsymbol{I}_t$ (a ResNet-50 backbone feature map).

The policy consists of a *base processor*, a *recurrent module* and a *policy head*. The base processor consists of two learnable components: $\phi_{img}$ and $\phi_{res}$. The 4-layer convolutional network $\phi_{img}$ takes as input the depth-wise concatenated triplet $\{\boldsymbol{I}_t, \boldsymbol{S}_t, \boldsymbol{P}_t\}$, producing $\phi_{img}(\boldsymbol{I}_t, \boldsymbol{S}_t, \boldsymbol{P}_t) \in \mathbb{R}^{512}$. Similarly, the 2-layer convolutional network $\phi_{res}$ yields an embedding $\phi_{res}(\boldsymbol{F}_t) \in \mathbb{R}^{512}$ of the ResNet features $\boldsymbol{F}_t$. An LSTM [16] with 256 cells constitutes the recurrent module, which takes as input $\phi_{img}(\boldsymbol{I}_t, \boldsymbol{S}_t, \boldsymbol{P}_t)$ and $\phi_{res}(\boldsymbol{F}_t)$. The input has length 1024. The hidden LSTM state is fed to the policy head, consisting of a fully-connected layer followed by a 7-way softmax which produces action probabilities.

**Rewards.** In training, the main reward is related to the mIoU improvement of the final segmentation network $\mathcal{S}_T$ over the initial $\mathcal{S}_0$ on a reference set $\mathcal{R}$. The set $\mathcal{R}$ is constructed at the beginning of each episode by randomly selecting views within a geodesic distance $r$ from the agent's starting location, and contains views with corresponding ground truth semantic segmentation masks. At the end of an episode of length $T$, the underlying perception module is evaluated on $\mathcal{R}$. Specifically, after an episode (with $T$ steps), the agent receives as final reward:

$$R_T = \text{mIoU}(\mathcal{S}_T, \mathcal{R}) - \text{mIoU}(\mathcal{S}_0, \mathcal{R}) \tag{3.1}$$

To obtain a denser signal, tightly coupled with the final objective, we also give a reward proportional to the improvement of $\mathcal{S}$ on the reference set $\mathcal{R}$ after each `Annotate` (*ann*) and `Collect` (*col*) action:

$$R_t^{ann} = \text{mIoU}(\mathcal{S}_t, \mathcal{R}) - \text{mIoU}(\mathcal{S}_{t-1}, \mathcal{R}) - \epsilon^{ann} \tag{3.2}$$

$$R_t^{col} = \text{mIoU}(\mathcal{S}_t, \mathcal{R}) - \text{mIoU}(\mathcal{S}_{t-1}, \mathcal{R}) \tag{3.3}$$

To ensure the agent does not request costly annotations too frequently, each `Annotate` action is penalized with a negative reward $-\epsilon^{ann}$ (we set $\epsilon^{ann} = 0.01$), as seen in (3.2). Such a penalty is *not* given for the free `Collect` action. Moreover, the dataset we use has 40 different semantic classes, but some are very rare and apply only to small objects, and some might not even be present in certain houses. We address this imbalance by computing the mIoU using only the 10 largest classes, ranked by the number of pixels in the set of reference views for the current episode.

While the rewards (3.1) - (3.3) should implicitly encourage the agent to explore the environment in order to request annotations for distinct, informative views, we empirically found useful to include an additional explicit exploration reward. Denote the positions the agent has visited up to time $t-1$ in its current episode by $\{\boldsymbol{x}_i\}_{i=1}^{t-1} = \{(x_i, y_i)\}_{i=1}^{t-1}$, and let $\boldsymbol{x}_t = (x_t, y_t)$ denote its current position. We define the exploration (*exp*) reward based on a kernel density estimate of the agent's visited locations:

$$R_t^{exp} = a - bp_t(\boldsymbol{x}_t) \coloneqq a - \frac{b}{t-1} \sum_{i=1}^{t-1} k(\boldsymbol{x}_t, \boldsymbol{x}_i) \tag{3.4}$$

where $a$ and $b$ are hyperparameters (both set to 0.003). Here $p_t(\boldsymbol{x}_t)$ is a Gaussian kernel estimate of the density with bandwidth 0.3m. It is large for previously visited positions and small for unvisited positions, thereby encouraging the agent's expansion towards new places in the environment. The exploration reward is only given for movement actions. Note that the pose $\boldsymbol{x}_i$ is only used to compute the reward $R_t^{exp}$ and is not available to the policy via the state space.

**Policy training.** The policy network is trained using PPO [39] based on the RLlib reinforcement learning package [24], as well as OpenAI Gym [3]. For optimization we use Adam [21] with batch size 512, learning rate $10^{-4}$ and discount rate 0.99. During training, each episode consists of 256 actions. The agent is trained for 4k episodes, which totals 1024k steps.

Our system is implemented in TensorFlow [1], and it takes about 3 days to train an agent using 4 Nvidia Titan X GPUs. An episode of length 256 took on average about 3 minutes using a single GPU, and during training we used 4 workers with one GPU each, collecting rollouts independently. The runtime per episode varies depending on how frequently the agent decides to annotate, as training the segmentation network is the bottleneck and accounts for

approximately 90% of the run-time. We used optical flow from the simulator to speed up policy training. For evaluation, the RL-agent and all other methods use PWC-Net to compute optical flow. The ResNet-50 feature extractor is pre-trained on ImageNet [19] with weights frozen during policy training.

# 4 Experiments

In this section we provide empirical evaluations of various methods. The primary metrics are mIoU and segmentation accuracy but we emphasize that we test the exploration and annotation selection capability of *policies* – the mIoU and accuracy measure how well agents explore in order to refine their perception. Differently from accuracy, the mIoU does not become overly high by simply segmenting large background regions (such as walls), hence it is more representative of overall semantic segmentation quality.

**Experimental setup.** We evaluate the methods on the Matterport3D dataset [4] using the embodied agent framework Habitat [37]. This setup allows the agent to freely explore photorealistic 3d models of large houses, that have ground truth annotations for 40 diverse semantic classes. Hence it is a suitable environment for evaluation. To assess the generalization capability of the RL-agent we train and test it in *different* houses. We use the same 61, 11 and 18 houses for training, validation and testing as Chang et al. [4]. The RL-agent and all pre-specified methods except the space filler are *comparable* in terms of assumptions, cf. §3.1. The space filler assumes full spatial knowledge of the environment (ground truth map) and hence has inherent advantages over the other methods.

During RL-agent training we randomly sample starting positions and rotations from the training houses at the start of each episode. An episode ends after 256 actions. Hyperparameters of the learnt and pre-specified agents are tuned on the validation set. For validation and testing we use 3 and 4 starting positions per scene, respectively, so each agent is tested for a total of 33 episodes in validation and 72 episodes in testing. The reported metrics are the mean over all these runs. All methods are evaluated on the same starting positions in the same houses. The reference views used to evaluate the semantic segmentation performance are obtained by sampling 32 random views within a 5 m geodesic distance of the agent's starting position at the beginning of each episode. In training the reference views are sampled randomly. During validation and testing, for fairness, we sample the same views for a given starting position when we test different agents. Note that there is no overlap between reference views during policy training and testing, since training, validation and testing houses are non-overlapping.

Recall that the RL-agent's policy parameters are denoted by $\boldsymbol{\theta}$. Let $\boldsymbol{\theta}_{seg}$ denote the parameters of the underlying semantic segmentation network, in order to clarify when we reset,

freeze and refine $\boldsymbol{\theta}$ and $\boldsymbol{\theta}_{seg}$, respectively. For RL-training, we refine $\boldsymbol{\theta}$ during policy estimation in the training houses. When we evaluate the policy on the validation or test houses we freeze $\boldsymbol{\theta}$ and only use the policy for inference. The parameters of the segmentation network $\boldsymbol{\theta}_{seg}$ are always reset at the beginning of an episode, regardless of which house we deploy the agent in, and regardless of whether the policy network is training or not. During an episode, we refine $\boldsymbol{\theta}_{seg}$ exactly when the agent selects the `Annotate` or `Collect` actions (this applies also to all the other methods described in §3.1). Thus annotated views in an episode are used to refine $\boldsymbol{\theta}_{seg}$ in that episode only, and are not used in any other episodes.

## 4.1 Main Results

We measure the performances of the agents in two settings: (a) with unlimited annotations but limited total actions (max 256, as during RL-training), or (b) for a limited annotation budget (max 100) but unlimited total actions. All methods were tuned on the validation set in a setup similar to (a) with 256 step episodes. Note however that the number of annotations can differ for different methods in a 256 step episode. The setup (b) is used to assess how the different methods compare for a fix number of annotations.

**Fixed episode length.** Table 3.1 and fig. 3.4 show results on the test scenes for episodes of length 256. The RL-agent outperforms the comparable pre-specified methods in mIoU and accuracy, although frontier exploration – which uses perfect pose and depth information, and is idealized to always move within the reference view radius – yields similar accuracy after about 170 steps. The RL-agent uses much fewer annotations than other methods, hence those annotated views are more informative. The space filler, which assumes perfect knowledge of the map, outperforms the RL-agent but uses significantly more annotations. Note that the *Rotate* baseline saturates, supporting the intuition that an agent has to move around in order to increase performance in complex environments.

**Fixed annotation budget.** In table 3.2 and fig. 3.5 we show test results when the annotation budget is limited to 100 images per episode. As expected, the space filler yields the best results, although the RL-agent gets comparable performance when using up to 15 annotations. The RL-agent outperforms comparable pre-specified methods in mIoU and accuracy. Frontier exploration obtains similar accuracy. We also see that the episodes of the RL-agent are longer.

**Qualitative examples.** Fig. 3.6 shows examples of views that the RL-agent choose to annotate. The agent explores large parts of the space and the annotated views are diverse, both in their spatial locations and in the types of semantic classes they contain. Fig. 3.7 shows how the segmentation network's performance on two reference views improves during an episode. The two views are initially poorly segmented, but as the agent explores and acquires annotations for novel views, the accuracy on the reference views increases.

**Table 3.1:** Comparison of different agents for a fixed episode length of 256 actions on the Matterport3D test scenes. The RL-agent gets higher mIoU using far fewer annotations than comparable pre-specified methods, implying that the RL-agent's policy selects more informative views to annotate.

| Method | mIoU | Acc | # Ann | # Coll |
|---|---|---|---|---|
| Space filler | 0.439 | 0.769 | 24.7 | 23.9 |
| RL-agent | 0.394 | 0.727 | 16.7 | 15.2 |
| Frontier exploration | 0.385 | 0.735 | 24.2 | 21.6 |
| Bounce | 0.357 | 0.708 | 29.6 | 26.0 |
| Rotate | 0.295 | 0.661 | 34.3 | 32.7 |
| Random | 0.204 | 0.566 | 29.1 | 19.5 |

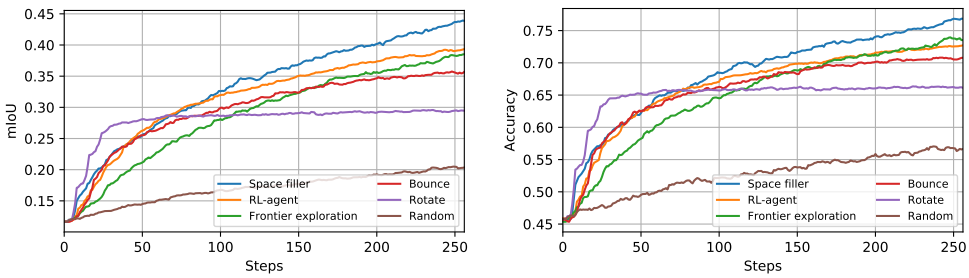


**Figure 3.4:** Mean segmentation accuracy and mIoU versus number of actions (steps), evaluated on the Matterport3D test scenes. The RL-agent was trained on 256-step episodes. This agent fairly quickly outperforms all other comparable pre-specified agents. *Rotate* is strong initially since it quickly gathers many annotations in a 360 degree arc, but is eventually outperformed by most other methods that move around in the houses. Frontier exploration yields similar accuracy as the RL-agent after about 170 steps, but uses significantly more annotations (cf. table 3.1) and assumes perfect pose and depth information. The space filler, which assumes full knowledge of the environment, yields the best results after about 100 steps.

## 4.2 Ablation Studies of the RL-agent

Ablation results of the RL-agent on the validation set are in table 3.3. We compare to the following versions: i) Policy without visual features $\phi_{img}$; ii) Policy without ResNet features $\phi_{res}$; iii) No additional exploration reward (3.4), i.e. $R_t^{exp} = 0$; iv) No `Collect` action and $\boldsymbol{P}_t$ is not an input to $\phi_{img}$; and v) Only exploration trained, using the heuristic strategy for annotations. We trained the ablated models for 4,000 episodes as for the full model.

Both the validation accuracy and mIoU are higher for the full RL-model compared to all ablated variants, justifying design choices. The model not relying on propagating annotations and using the `Collect` action performs somewhat worse than the full model despite a comparable amount of annotations. The learnt annotation strategy yields higher mIoU and accuracy compared to the heuristic one, at comparable number of annotations. The exploration reward is important in encouraging the agent to navigate to unvisited positions – without it performance is worse, despite a comparable number of annotations. The agent trained without the exploration reward uses an excessive number of `Collect` actions, so this agent often stands still instead of moving. Finally, omitting either visual or ResNet features

**Table 3.2:** Comparison of different agents for a fixed budget of 100 annotations on Matterport3D test scenes. The RL-agent gets a higher mIoU than comparable pre-specified agents, despite not being trained in this setting.

| Method | mIoU | Acc | # Steps | # Coll |
|---|---|---|---|---|
| Space filler | 0.600 | 0.863 | 1048 | 91 |
| RL-agent | 0.507 | 0.796 | 1541 | 94 |
| Frontier exploration | 0.485 | 0.796 | 998 | 84 |
| Bounce | 0.464 | 0.776 | 861 | 87 |
| Rotate | 0.303 | 0.668 | 752 | 96 |
| Random | 0.242 | 0.595 | 910 | 64 |

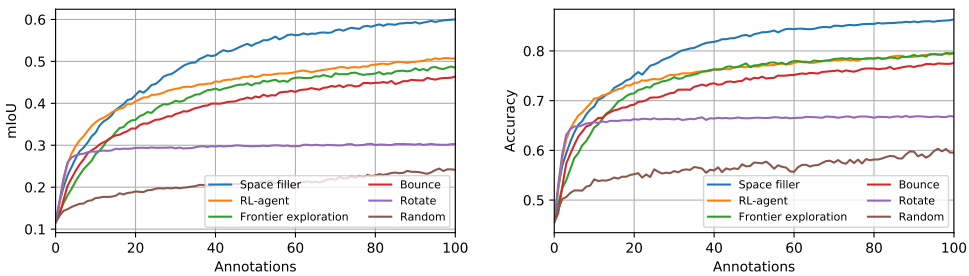


**Figure 3.5:** Mean segmentation accuracy and mIoU for a varying number of requested annotations evaluated on the Matterport3D test scenes. The RL-agent outperforms all comparable pre-specified methods (although frontier exploration matches it in accuracy after about 40 annotations), indicating that it has learnt an exploration policy which generalizes to novel scenes. The space filler, as expected, outperforms the RL-agent, except for less than 15 annotations. Thus the RL-agent is best before and around its training regime, where on average annotates 16.7 times per episode, cf. table 3.1.

from the policy significantly harms accuracy for the resulting recognition system.

## 4.3 Analysis of Annotation Strategies

In this section we examine how different annotation strategies affect the task performance on the validation set for the space filler and bounce methods. Specifically, the annotation strategies are:

- **Threshold perception.** This is the variant evaluated in §4.1, i.e. it issues the `Collect` action when 30% of the propagated labels are unknown and `Annotate` when 85% are unknown.
- **Learnt perception.** We train a simplified RL-agent where the movement actions are restricted to follow the exploration trajectory of the baseline method (space filler and bounce, respectively). This model has 3 actions: move along the baseline exploration path, `Annotate` and `Collect`. All other training settings are identical to the full RL-agent.

**Table 3.3:** Ablation study of different RL-based model variants for 256-step episodes on the validation set. The full RL-agent outperforms all ablated models at a comparable or lower number of requested annotations.

| Variant | mIoU | Acc | # Ann | # Coll |
|---|---|---|---|---|
| Full model | 0.427 | 0.732 | 16.4 | 16.4 |
| No collect nor $\boldsymbol{P}_t$ | 0.415 | 0.727 | 17.9 | 0.0 |
| Only exploration | 0.411 | 0.727 | 16.1 | 14.4 |
| $R_t^{exp} = 0$ | 0.401 | 0.719 | 17.7 | 47.4 |
| No $\phi_{img}$ | 0.378 | 0.696 | 14.3 | 3.8 |
| No ResNet | 0.375 | 0.705 | 23.3 | 0.3 |

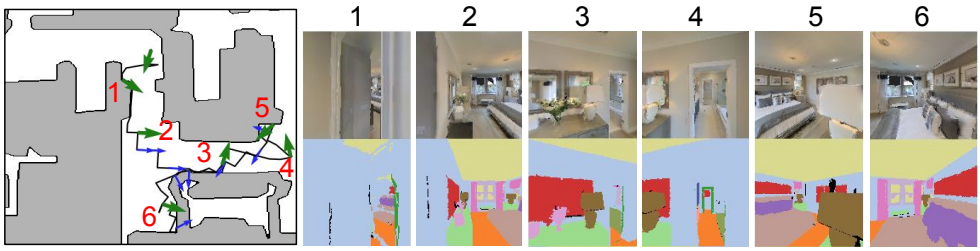


**Figure 3.6:** The first six requested annotations by the RL-agent in a room from the test set. Left: Map showing the agent's trajectory and the six first requested annotations (green arrows). The initially given annotation is not indicated with a number. Blue arrows indicate `Collect` actions. Right: For each annotation (numbered 1 - 6) the figures show the image seen by the agent and the ground truth received when the agent requested annotations. As can be seen, the agent quickly explores the room and requests annotations containing diverse semantic classes.

- **Random perception.** In each step, this variant follows the baseline exploration trajectory with 80% probability, while annotating views and collecting propagated labels with 10% probability each.

As can be seen in table 3.4, the best results for the space filler are obtained by using the threshold strategy, which also annotates slightly less frequently than other variants. Using learnt perception actions yields better results compared to random perception actions, and takes slightly fewer annotations per episode. Similar results carry over to the bounce method in table 3.5, i.e. the best results are again obtained by the threshold variant. The model with a learnt annotation strategy fails to converge to anything better than heuristic perception strategies. In fact, it converges to selecting `Collect` almost 40% of the time, which indicates a lack of movement for this variant.

In table 3.3 we saw that a learnt exploration method with a heuristic annotation strategy yields worse results than a fully learnt model. Conversely, the results from table 3.4 and table 3.5 show that a heuristic exploration method using a learnt annotation strategy yields worse results than an entirely heuristic model. Together these results indicate that it is necessary to learn how to annotate and explore jointly to provide the best results, given comparable environment knowledge.

**Table 3.4:** Results for different model variants of the space filler method. We report the mean on the validation scenes. The threshold perception strategy – which is the one used in the main evaluations in §4.1 – yields the best results.

| Variant | mIoU | Acc | # Ann | # Coll |
|---|---|---|---|---|
| Threshold perception | 0.472 | 0.770 | 20.8 | 19.9 |
| Learnt perception | 0.454 | 0.755 | 22.8 | 37.4 |
| Random perception | 0.446 | 0.747 | 24.2 | 24.4 |

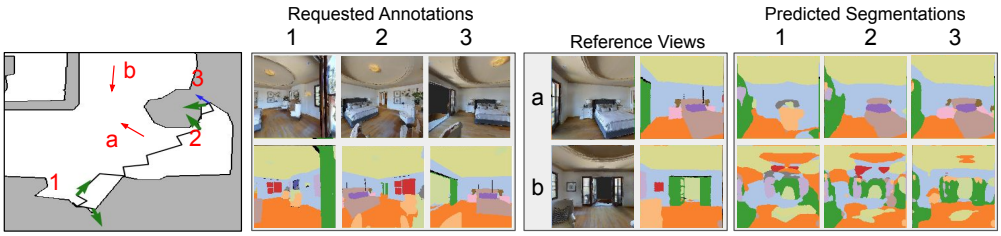


**Figure 3.7:** Example of the RL-agent's viewpoint selection and how its perception improves over time. We show results of two reference views after the first three annotations of the RL-agent. Left: Agent's movement path is drawn in black on the map. The annotations (green arrows) are numbered 1 - 3, and the associated views are shown immediately right of the map (the initially given annotation is not shown). Red arrows labeled *a* - *b* indicate the reference views. Right: Reference views and ground truth masks, followed by predicted segmentation after one, two and three annotations. Notice clear segmentation improvements as the agent requests more annotations. Specifically, note how reference view *a* improves drastically with annotation *2* as the bed is visible in that view, and with annotation *3* where the drawer is seen. Also note how segmentation improves for reference view *b* after the door is seen in annotation *3*.

## 4.4 Pre-training the Segmentation Network

Recall that our semantic segmentation network is randomly initialized at the beginning of each episode. In this section we evaluate the effect of instead pre-training the segmentation network[3] on the 61 training houses using about 20,000 random views. In table 3.6 we compare using this pre-trained segmentation network as initialization for the RL-agent with the case of random initialization. We also show results when not further fine-tuning the pre-trained segmentation network, i.e. when not performing any embodied visual active learning.

The weak result obtained when not fine-tuning (first row) indicates significant appearance differences between the houses. This is further suggested by the fact that the RL-agent gets a surprisingly modest boost from pre-training the segmentation network (third row vs second row). Note the different number of annotated views used here – the agent without pre-training uses only 16.4 views on average, while the other uses about $20,000 + 14.4$ annotated views, if we count all the images used for pre-training. Due to relatively marginal gains for a large number of annotated images, we decided to evaluate all agents without pre-training the segmentation network.

[3]In this pre-training experiment, we use the same architecture and hyperparameters for the segmentation network as when it is trained and deployed in the embodied visual active learning task.

**Table 3.5:** Results for different model variants of the bounce method. We report the mean on the validation scenes. The threshold perception strategy – which is the one used in the main evaluations in §4.1 – yields the best results, but also uses the largest amount of annotations on average.

| Variant | mIoU | Acc | # Ann | # Coll |
|---|---|---|---|---|
| Threshold perception | 0.388 | 0.706 | 27.4 | 24.5 |
| Learnt perception | 0.375 | 0.699 | 14.6 | 98.8 |
| Random perception | 0.379 | 0.698 | 25.9 | 24.6 |

**Table 3.6:** Results for different training regimes for the semantic segmentation network. A pre-trained segmentation network generalizes poorly to unseen environments (first row), and there is relatively little gain for the RL-agent by having a pre-trained segmentation network (third row). Note that pre-training uses over 1000x more annotations compared to performing embodied active visual learning from scratch.

| Variant | mIoU | Acc | # Ann | # Coll |
|---|---|---|---|---|
| Pre-train, no RL | 0.208 | 0.549 | 20k | 0.0 |
| No pre-train, RL | 0.427 | 0.732 | 16.4 | 16.4 |
| Pre-train, RL | 0.461 | 0.780 | 20k + 14.4 | 13.3 |

# 5 Conclusions

In this paper we have explored the *embodied visual active learning* task for semantic segmentation and developed a diverse set of methods, both pre-designed and learning-based, in order to address it. The agents can explore a 3d environment and improve the accuracy of their semantic segmentation networks by requesting annotations for informative viewpoints, propagating annotations via optical flow at no additional cost by moving in the neighborhood of those views, and self-training. We have introduced multiple baselines as well as a more sophisticated fully learnt model, each exposing different assumptions and knowledge of the environment. Through extensive experiments in the photorealistic Matterport3D environment we have thoroughly investigated the various methods and shown that the fully learning-based method outperforms comparable non-learnt approaches, both in terms of accuracy and mIoU, while relying on fewer annotations.

**Acknowledgments:** This work was supported in part by the European Research Council Consolidator grant SEED, CNCS-UEFISCDI PN-III-P4-ID-PCE-2016-0535 and PCCF-2016-0180, the EU Horizon 2020 Grant DE-ENIGMA, Swedish Foundation for Strategic Research (SSF) Smart Systems Program, as well as the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation.

# Bibliography


[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016.

[2] P. Ammirato, P. Poirson, E. Park, J. Košecká, and A. C. Berg. A dataset for developing and benchmarking active vision. In *ICRA*, pages 1378–1385. IEEE, 2017.

[3] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

[4] A. Chang, A. Dai, T. Funkhouser, M. Halber, M. Niessner, M. Savva, S. Song, A. Zeng, and Y. Zhang. Matterport3d: Learning from rgb-d data in indoor environments. *International Conference on 3D Vision*, 2017.

[5] D. S. Chaplot, D. Gandhi, S. Gupta, A. Gupta, and R. Salakhutdinov. Learning to explore using active neural slam. In *ICLR*, 2020.

[6] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *TPAMI*, 40(4):834–848, 2017.

[7] T. Chen, S. Gupta, and A. Gupta. Learning exploration policies for navigation. In *ICLR*, 2019.

[8] A. Das, S. Datta, G. Gkioxari, S. Lee, D. Parikh, and D. Batra. Embodied question answering. In *CVPR*, volume 5, page 6, 2018.

[9] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun. CARLA: An open urban driving simulator. In *CoRL*, pages 1–16, 2017.

[10] K. Fang, A. Toshev, L. Fei-Fei, and S. Savarese. Scene memory transformer for embodied agents in long-horizon tasks. In *CVPR*, June 2019.

[11] M. Fang, Y. Li, and T. Cohn. Learning how to active learn: A deep reinforcement learning approach. 2017.

[12] D. Feng, X. Wei, L. Rosenbaum, A. Maki, and K. Dietmayer. Deep active learning for efficient training of a lidar 3d object detector. In *2019 IEEE Intelligent Vehicles Symposium (IV)*, pages 667–674, 2019.

[13] E. Gärtner, A. Pirinen, and C. Sminchisescu. Deep reinforcement learning for active human pose estimation. In *AAAI*, pages 10835–10844, 2020.

[14] S. Gupta, J. Davidson, S. Levine, R. Sukthankar, and J. Malik. Cognitive mapping and planning for visual navigation. In *CVPR*, pages 2616–2625, 2017.

[15] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016.

[16] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[17] D. Jayaraman and K. Grauman. Look-ahead before you leap: end-to-end active recognition by forecasting the effect of motion. In *ECCV*, pages 489–505. Springer, 2016.

[18] D. Jayaraman and K. Grauman. Learning to look around: Intelligently exploring unseen environments for unknown tasks. In *CVPR*, 2018.

[19] Jia Deng, Wei Dong, R. Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *CVPR*, 2009. ISBN 978-1-4244-3992-8. doi: 10.1109/CVPRW.2009.5206848.

[20] E. Johns, S. Leutenegger, and A. J. Davison. Pairwise decomposition of image sequences for active multi-view recognition. In *CVPR*, pages 3813–3822, 2016.

[21] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[22] E. Kolve, R. Mottaghi, W. Han, E. VanderBilt, L. Weihs, A. Herrasti, D. Gordon, Y. Zhu, A. Gupta, and A. Farhadi. AI2-THOR: An Interactive 3D Environment for Visual AI. *arXiv preprint arXiv:1712.05474*, 2017.

[23] A. Krizhevsky, I. Sutskever, and G. Hinton. Imagenet classification with deep convolutional neural networks. In *NeurIPS*, 2012.

[24] E. Liang, R. Liaw, R. Nishihara, P. Moritz, R. Fox, K. Goldberg, J. Gonzalez, M. Jordan, and I. Stoica. Rllib: Abstractions for distributed reinforcement learning. In *International Conference on Machine Learning*, pages 3053–3062, 2018.

[25] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, pages 3431–3440, 2015.

[26] E. Lughofer. Single-pass active learning with conflict and ignorance. *Evolving Systems*, 3(4): 251–271, 2012.

[27] D. Mishkin, A. Dosovitskiy, and V. Koltun. Benchmarking classic and learned navigation in complex 3d environments. *arXiv preprint arXiv:1901.10915*, 2019.

[28] A. Mousavian, A. Toshev, M. Fišer, J. Košecká, A. Wahid, and J. Davidson. Visual representations for semantic target driven navigation. In *ICRA*, pages 8846–8852. IEEE, 2019.

[29] A. Pardo, M. Xu, A. Thabet, P. Arbelaez, and B. Ghanem. Baod: Budget-aware object detection. *arXiv preprint arXiv:1904.05443*, 2019.

[30] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell. Curiosity-driven exploration by self-supervised prediction. In *ICML*, volume 2017, 2017.

[31] E. Pot, A. Toshev, and J. Kosecka. Self-supervisory signals for object discovery and detection. *arXiv preprint arXiv:1806.03370*, 2018.

[32] W. Qi, R. T. Mullapudi, S. Gupta, and D. Ramanan. Learning to move with affordance maps. In *ICLR*, 2020.

[33] S. K. Ramakrishnan, D. Jayaraman, and K. Grauman. An exploration of embodied visual exploration. *arXiv preprint arXiv:2001.02192*, 2020.

[34] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *CVPR*, pages 779–788, 2016.

[35] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *NeurIPS*, pages 91–99, 2015.

[36] M. Savva, A. X. Chang, A. Dosovitskiy, T. Funkhouser, and V. Koltun. MINOS: Multimodal indoor simulator for navigation in complex environments. *arXiv:1712.03931*, 2017.

[37] M. Savva, A. Kadian, O. Maksymets, Y. Zhao, E. Wijmans, B. Jain, J. Straub, J. Liu, V. Koltun, J. Malik, et al. Habitat: A platform for embodied ai research. In *ICCV*, pages 9339–9347, 2019.

[38] A. Sax, B. Emi, A. R. Zamir, L. J. Guibas, S. Savarese, and J. Malik. Mid-level visual representations improve generalization and sample efficiency for learning visuomotor policies. In *CoRL*, 2019.

[39] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[40] B. Settles. Active learning literature survey. Technical report, University of Wisconsin-Madison Department of Computer Sciences, 2009.

[41] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[42] S. Song, L. Zhang, and J. Xiao. Robot in a room: Toward perfect object recognition in closed environments. *CoRR, abs/1507.02703*, 2015.

[43] S. Song, A. Zeng, A. X. Chang, M. Savva, S. Savarese, and T. Funkhouser. Im2pano3d: Extrapolating 360 structure and semantics beyond the field of view. In *CVPR*, pages 3847–3856, 2018.

[44] D. Sun, X. Yang, M.-Y. Liu, and J. Kautz. Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume. In *CVPR*, pages 8934–8943, 2018.

[45] N. Sundaram, T. Brox, and K. Keutzer. Dense point trajectories by gpu-accelerated large displacement optical flow. In *ECCV*, pages 438–451. Springer, 2010.

[46] K. Wang, Y. Lin, L. Wang, L. Han, M. Hua, X. Wang, S. Lian, and B. Huang. A unified framework for mutual improvement of slam and semantic segmentation. In *ICRA*, pages 5224–5230. IEEE, 2019.

[47] E. Wijmans, S. Datta, O. Maksymets, A. Das, G. Gkioxari, S. Lee, I. Essa, D. Parikh, and D. Batra. Embodied Question Answering in Photorealistic Environments with Point Cloud Perception. In *CVPR*, 2019.

[48] M. Woodward and C. Finn. Active one-shot learning. *arXiv preprint arXiv:1702.06559*, 2017.

[49] F. Xia, A. R. Zamir, Z.-Y. He, A. Sax, J. Malik, and S. Savarese. Gibson env: real-world perception for embodied agents. In *CVPR*. IEEE, 2018.

[50] B. Xiong and K. Grauman. Snap angle prediction for 360 panoramas. In *ECCV*, pages 3–18, 2018.

[51] B. Yamauchi. A frontier-based approach for autonomous exploration. In *CIRA*, pages 146–151. IEEE, 1997.

[52] H.-K. Yang, P.-H. Chiang, K.-W. Ho, M.-F. Hong, and C.-Y. Lee. Never forget: Balancing exploration and exploitation via learning optical flow. *arXiv preprint arXiv:1901.08486*, 2019.

[53] J. Yang, Z. Ren, M. Xu, X. Chen, D. J. Crandall, D. Parikh, and D. Batra. Embodied amodal recognition: Learning to move to perceive objects. In *ICCV*, pages 2040–2050, 2019.

[54] L. Yu, X. Chen, G. Gkioxari, M. Bansal, T. L. Berg, and D. Batra. Multi-target embodied question answering. In *CVPR*, 2019.

[55] L. Zheng, C. Zhu, J. Zhang, H. Zhao, H. Huang, M. Niessner, and K. Xu. Active scene understanding via online semantic reconstruction. In *Computer Graphics Forum*, volume 38, pages 103–114. Wiley Online Library, 2019.

[56] F. Zhong, S. Wang, Z. Zhang, and Y. Wang. Detect-slam: Making object detection and slam mutually beneficial. In *WACV*, pages 1001–1010. IEEE, 2018.

[57] Y. Zhu, R. Mottaghi, E. Kolve, J. J. Lim, A. Gupta, L. Fei-Fei, and A. Farhadi. Target-driven visual navigation in indoor scenes using deep reinforcement learning. In *ICRA*, pages 3357–3364. IEEE, 2017.

# A Introduction

In this supplementary material we provide additional insights into our proposed models for the embodied active visual learning task. Details of the semantic segmentation network are given in §B, and of the RL-agent's policy network in §C. In §D we provide an algorithmic description of how the RL-agent operates within our task. In §E we compare with additional variants / hyperparameter configurations of the bounce method and conclude that the variants used in the main paper provide the best results in terms of mIoU.

# B Semantic Segmentation Network

Fig. 3.8 contains a schematic overview of the semantic segmentation network. We deliberately made it small so that it would be very quick to refine it with new data. It consists of 3 blocks of convolutional layers, each containing 3 convolutional layers with kernels of size $3 \times 3$. The first convolutional layer in each block uses a stride of 2, which halves the resolution. For each block the number of channels doubles. Multiple predictions are made using the final convolutional layers of each block. The multi-scale predictions are resized to the original image resolution using bilinear interpolation and are finally summed up, resulting in the final segmentation. The network resembles FCN [25] by predicting the semantic segmentation at multiple scales. In training we use a standard cross-entropy loss averaged over all pixels. The segmentation network is trained using stochastic gradient descent with learning rate $0.01$, weight decay $10^{-5}$ and momentum $0.9$.

# C Policy Network of the RL-Agent

See Fig. 3.9 for an overview of the policy network. The policy consists of two branches, where the first processes the image and segmentation inputs, and the second processes the extracted deep features.

# D Pseudo Code

The full procedure of our RL-model for embodied visual active learning for semantic segmentation is described in Algorithm 1. It describes among other things how the states and segmentation network are updated during an episode.

**Table 3.7:** Results of different model variants of the bounce method. We report the mean on the validation scenes. Threshold perception is used in the main paper.

| Variant | mIoU | Acc | # Ann | # Coll |
|---|---|---|---|---|
| Threshold perception | 0.388 | 0.706 | 27.4 | 24.5 |
| Learned perception | 0.375 | 0.699 | 14.6 | 98.8 |
| Random perception | 0.379 | 0.698 | 25.9 | 24.6 |
| Version 1 | 0.353 | 0.677 | 11.4 | 0.0 |
| Version 2 | 0.372 | 0.695 | 9.8 | 9.6 |
| Version 3 | 0.381 | 0.701 | 20.0 | 10.4 |

# E Variants of Bounce

We tried multiple perception strategies of the bounce baseline on the validation set and present in Table 3.7 the results of 3 different versions (in addition to those perception strategies described in §4.3):

- **Version 1.** Recall that the bounce method samples a random rotation after bouncing in a wall, and then begins moving in that direction. This version annotates prior to walking in a new direction (it issues no `Collect` actions).
- **Version 2.** Issues `Annotate` after rotating towards a new direction, and issues `Collect` four steps (1 m) after that.
- **Version 3.** Issues `Annotate` after rotating towards a new direction, and issues `Annotate` and `Collect` with 10% probability each when walking forward.

We see that the third version – with more frequent annotations and collects compared to versions 1 and 2 – obtains the best performance in terms of accuracy and mIoU on the validation set for 256-step episodes. However, it does not outperform the threshold strategy.

**Algorithm 1** Procedural code for the RL-agent in the embodied visual active learning for semantic segmentation task.

Initialize parameters of the segmentation network $\mathcal{S}$
Initialize location $(x_1, y_1)$ and rotation $\phi_1$ randomly and let $\boldsymbol{x}_1 = (x_1, y_1, \phi_1)$
Extract image $\boldsymbol{I}_1$ and receive associated annotation mask $\boldsymbol{A}_1$ at $\boldsymbol{x}_1$; initialize training set $\mathcal{D} = \{(\boldsymbol{I}_1, \boldsymbol{A}_1)\}$
Perform initial training of $\mathcal{S}$ on $\mathcal{D}$
Initialize propagated annotation $\boldsymbol{P}_1 = \boldsymbol{A}_1$
Compute segmentation $\boldsymbol{S}_1 = \mathcal{S}(\boldsymbol{I}_1)$ and deep features $\boldsymbol{F}_1$
Initialize agent state $s_1 = (\boldsymbol{I}_1, \boldsymbol{S}_1, \boldsymbol{P}_1, \boldsymbol{F}_1)$
**for** $t = 1, \ldots, T$ **do**
    Sample action $a_t \sim \pi_\theta(\cdot|s_t)$
    **if** $a_t \in \{$`MoveForward`, `MoveLeft`, `MoveRight`, `RotateLeft`, `RotateRight`$\}$ **then**
        Set $\boldsymbol{x}_{t+1}$ according to movement
        Propagate annotation $\boldsymbol{P}_{t+1} = \text{flow}(\boldsymbol{P}_t)$
    **else**
        Set $\boldsymbol{x}_{t+1} = \boldsymbol{x}_t$
        Set $\boldsymbol{P}_{t+1} = \boldsymbol{P}_t$
    **end if**
    Obtain view $\boldsymbol{I}_{t+1}$ associated to $\boldsymbol{x}_{t+1}$
    **if** $a_t =$ `Annotate` **then**
        Obtain annotation mask $\boldsymbol{A}_{t+1}$ at $\boldsymbol{x}_{t+1}$
        Update training set $\mathcal{D} = \mathcal{D} \cup \{(\boldsymbol{I}_{t+1}, \boldsymbol{A}_{t+1})\}$
        Refine $\mathcal{S}$ on $\mathcal{D}$
        Reset propagated annotation $\boldsymbol{P}_{t+1} = \boldsymbol{A}_{t+1}$
    **else if** $a_t =$ `Collect` **then**
        Update training set $\mathcal{D} = \mathcal{D} \cup \{(\boldsymbol{I}_{t+1}, \boldsymbol{P}_{t+1})\}$
        Refine $\mathcal{S}$ on $\mathcal{D}$
    **end if**
    Compute segmentation $\boldsymbol{S}_{t+1} = \mathcal{S}(\boldsymbol{I}_{t+1})$ and deep features $\boldsymbol{F}_{t+1}$
    Update agent state $s_{t+1} = (\boldsymbol{I}_{t+1}, \boldsymbol{S}_{t+1}, \boldsymbol{P}_{t+1}, \boldsymbol{F}_{t+1})$
**end for**
**return** $\mathcal{S}_\star$ (*trained segmentation network)*

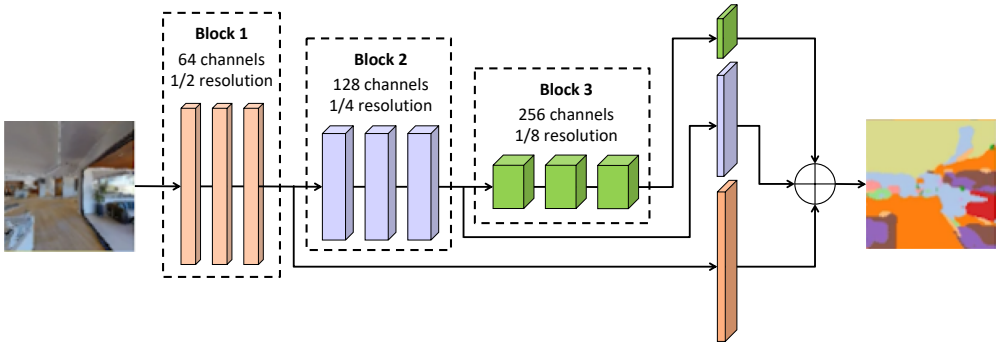


**Figure 3.8:** Architecture of the deep network we use for semantic segmentation. The input image is processed sequentially through 3 blocks, each containing 3 convolutional layers. The first convolutional layer in each block uses a stride of 2, which halves the resolution for each block, and at the same time we double the number of channels. The semantic segmentation is predicted for multiple resolutions and are summed together to predict the semantic segmentation.

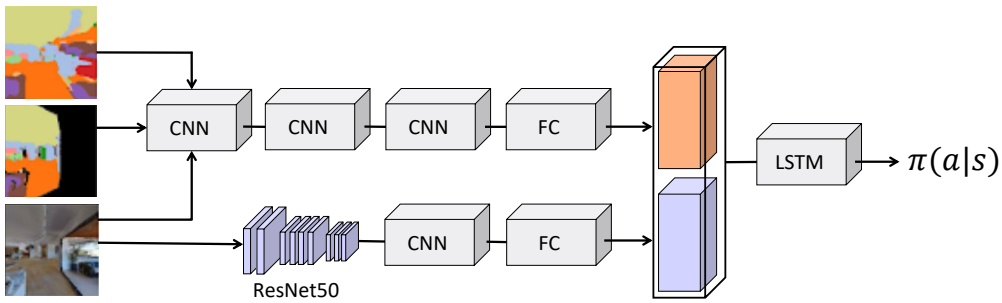


**Figure 3.9:** The policy network architecture for the RL-agent. The network has three inputs: the current RGB image $\boldsymbol{I} \in \mathbb{R}^{127\times127\times3}$ (bottom), the current segmentation prediction $\boldsymbol{S} \in \mathbb{R}^{127\times127\times3}$ (top), and the current optical flow propagated ground truth segmentation $\boldsymbol{P} \in \mathbb{R}^{127\times127\times3}$ (middle). All three inputs are stacked depthwise and then processed by three convolutional layers and a fully connected layer (this processing subnetwork is called $\phi_{img}$ in §3.3 of the main paper). The first layer has 32 filters, kernel size $8 \times 8$, and stride 4. The second layer has 64 filters, kernel size $4 \times 4$, and stride 2. The third layer has 64 filters, kernel size $3 \times 3$, and stride 1. Finally, the fully connected layer has 512 outputs. In addition, the RGB image is passed through an image feature extractor (ResNet-50), called $\phi_{res}$ with output $\boldsymbol{F}_t$ in the main paper. The deep features $\boldsymbol{F}_t$ are subsequently passed through a convolutional layer with 128 filters, kernel size $2 \times 2$ and stride 2. Finally, these features are processed by a fully connected layer with 512 outputs. These two input branches are then concatenated and fed to an LSTM with 256 cells. The hidden state of the LSTM is finally passed to a softmax layer to produce the action distribution.

# Paper III

# Embodied Learning for Lifelong Visual Perception

David Nilsson[1], Aleksis Pirinen[2], Erik Gärtner[1], Cristian Sminchisescu[1,3]
[1]Department of Mathematics, Faculty of Engineering, Lund University
[2]RISE Research Institutes of Sweden
[3]Google Research

## Abstract

We study lifelong visual perception in an embodied setup, where we develop new models and compare various agents that navigate in buildings and occasionally request annotations which, in turn, are used to refine their visual perception capabilities. The purpose of the agents is to recognize objects and other semantic classes in the whole building at the end of a process that combines exploration and active visual learning. As we study this task in a lifelong learning context, the agents should use knowledge gained in earlier visited environments in order to guide their exploration and active learning strategy in successively visited buildings. We use the semantic segmentation performance as a proxy for general visual perception and study this novel task for several exploration and annotation methods, ranging from frontier exploration baselines which use heuristic active learning, to a fully learnable approach. For the latter, we introduce a deep reinforcement learning (RL) based agent which jointly learns both navigation and active learning. A point goal navigation formulation, coupled with a global planner which supplies goals, is integrated into the RL model in order to provide further incentives for systematic exploration of novel scenes. By performing extensive experiments on the Matterport3D dataset, we show how the proposed agents can utilize knowledge from previously explored scenes when exploring new ones, e.g. through less granular exploration and less frequent requests for annotations. The results also suggest that a learning-based agent is able to use its prior visual knowledge more effectively than heuristic alternatives.

# 1 Introduction

Over the last decade we have witnessed rapid progress in deep learning-based visual perception [22, 39, 14, 32, 24]. Impressive as these developments have been on their own, many contemporary state-of-the-art perception models may nevertheless deteriorate in performance (or altogether fail) if attached e.g. to embodied systems which operate in unusual or entirely new circumstances [1, 29, 10]. Even when a given perception system does not break down entirely, it may still produce poorer predictions compared to the training and evaluation settings for which it was originally developed. One possible direction to limit model overfitting – and which is often pursued – is to work with ever larger datasets, but such a procedure is typically associated with a high cost due to the need to gather and annotate massive amounts of data. Moreover, the expressive capacity of even the deepest perception model is inherently limited, so it may be better to focus the model training on data which is most relevant for the conditions where the model is to be used.

The above motivates the need for principled and adaptive methods that can effectively refine visual perception systems in case forms of supervision become available (e.g. from a human-in-the-loop) in the present use-cases of those systems. In this work we develop and study methods for lifelong perception refinement in an embodied context, where a visual perception model trained in previously explored environments is to be refined such that it becomes as accurate as possible in a new context. A naive approach for this would be one where an embodied agent looks around and queries for annotations for all viewpoints observed in the new environment it explores. This would however require an impractically large amount of annotation and is thus clearly not feasible in practice. Moreover, this type of approach is particularly wasteful since we assume that the agent has already obtained a coarse perception model a priori. Depending on the agent's prior experience, the different viewpoints it observes in its new environment may be of highly varying importance. For example, viewpoints which contain large portions of the same semantic classes and/or appearances as the agent has seen in the past may not have to be annotated, and thus the agent may focus on those novel parts of the scene which it is most uncertain about.

In this paper we thoroughly study this proposed *embodied lifelong learning* task. We present, evaluate and compare agents of various degrees of sophistication – providing both heuristic alternatives and a fully learnable one – and with varying amounts of prior visual experience. An overview of the task is shown in Fig. 4.1. We let visual perception be measured as semantic segmentation accuracy, given that semantic segmentation is a core perception task that requires fine (pixel-wise) predictions of each view, but note that our framework is applicable in the context of any perception task. To this end, we equip an agent with a semantic segmentation network and the agent is then tasked to move around and query for annotations in order to refine it. After exploring a novel building the agent should be able to accurately segment views in the whole building using a limited number of annotations. Note that which

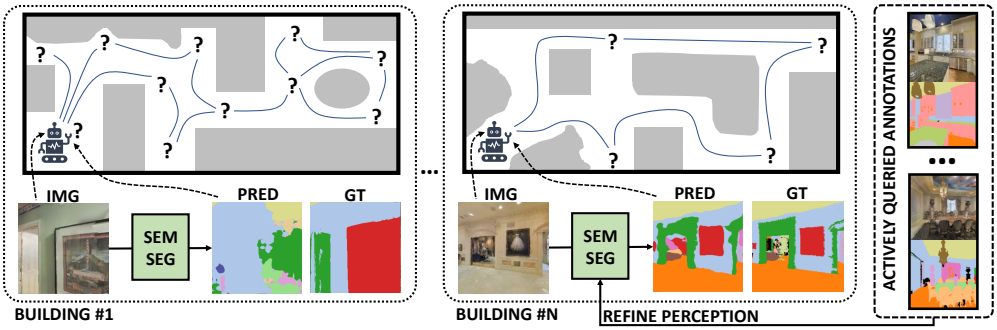


**Figure 4.1:** Overview of the embodied lifelong learning setup. A first-person agent selectively determines viewpoints for which to query annotations as it actively explores a new building. Its visual perception system – here a semantic segmentation network – is continually refined as the agent queries for more data. As the agent operates in a lifelong framework, its exploration and active learning may differ in later buildings compared to earlier ones. For example, the number of annotations (indicated in the respective maps with question marks), as well as the exploration trajectories, may vary. In this example, in the first building (left), the perception of the agent's current view is poor, and the agent decides to query for the ground truth to refine its perception before continuing to explore. Comparing with a later building (right), the agent's corresponding viewpoint is much more accurate – potentially because it already learnt to recognize e.g. the 'painting' category (red in the ground truth masks) in the first building – and in this case the agent decides to ignore querying for that annotation.

and how many viewpoints should be annotated may be affected by the agent's prior visual perception performance obtained from earlier explored buildings. To empirically investigate this new task, we compare and evaluate all the proposed methods in the Matterport3D simulator [4] which provides photorealistic renderings of a large set of indoor scenes.

In summary, our main contributions are:

- We introduce and study the embodied active learning task in a lifelong learning setup where experience and annotations in one scene can guide and inform the exploration and active learning in successive ones.
- We introduce several heuristic as well as a fully learnable RL formulation for the task, and study how they are respectively affected by the level of prior visual knowledge.
- We provide extensive experimental evaluation of our proposed methods and show that a fully learnable RL agent can use prior visual perception to guide its exploration and active learning more effectively compared to heuristic alternatives, both in terms of exploration granularity and frequency of annotations.

## 2 Related Work

Our work is broadly related to embodied learning, active learning, as well as lifelong learning, and we here mention some related work for each subproblem.

**Embodied learning.** With the emergence of large scale simulators [33, 45] of indoor scans [4] there has been much recent work on applying RL to visual navigation problems. In point goal navigation [44] an agent is given a navigation target in polar coordinates $(r, \theta)$ and the task is to reach the goal, which often requires navigating around obstacles. There is a large body of work for this problem, and in general approaches based on RL [44, 34] perform well. A related problem is visual exploration where an agent is given egocentric observations and should navigate to explore as much of a scene as possible [5, 31]. Another setup related to our paper is studied by Ammirato et al. [1], Yang et al. [47] and concerns how to locally explore to get a better view of a partially occluded object.

How to explore a scene to obtain relevant views for learning visual perception is studied by Chaplot et al. [6]. In their setup the refinement of the visual model is not interactive, but rather a post-processing step – the visual perception is updated after exploring several scenes, while we refine the visual perception online whenever the agent queries for annotations. Furthermore, in Chaplot et al. [6] all views of the explored trajectories are labelled, while we also task the agent to select only a small subset of frames to label. Different but related problem setups include learning along fixed navigation trajectories by self-supervision and querying for labels of a subset of the data [41, 30], as well as formulations which consider joint improvement of SLAM and visual perception [48, 43]. Another work related to ours is Nilsson et al. [27], to which we however differ in several key respects. In our formulation the exploration is global, which is enforced by integrating a global planner with our proposed agents (thus we do not restrict the problem to recognizing only a local neighborhood of where an agent is spawned, as Nilsson et al. [27]). Furthermore, we study the task in a lifelong setup instead of considering independent scenes, and a key contribution of our work is showing how the behavior of agents can improve as they explore multiple scenes consecutively instead of independently and in separation.

**Active learning.** In active learning [37] the problem is to select which data among a large pool of unlabelled data to label to maximally improve a model. Common approaches include RL [11, 21], coverage of a feature space [36], uncertainty estimation [12, 20], or generative modelling [40, 9]. Our paper differs from the standard active learning formulation in that we do not assume access to a pool of unlabelled data. Instead we consider embodied agents that need to explore an environment to gather the data considered for annotations.

We next briefly review methods for active learning (AL) for semantic segmentation. Siddiqui et al. [38] propose an AL strategy that exploits viewpoint consistency in multi-view datasets. Their method queries labels for those views and superpixels which are not consistently segmented across views from different directions. Golestaneh and Kitani [13] show that self-consistency can greatly improve the performance of a model. They suggest an AL framework where small image patches that need to be labeled are iteratively extracted by selecting image patches which have high uncertainty (high entropy) under equivariant transformations. The approach by Kasarla et al. [17] uses entropy- and region-based AL, with annotation at the su-

perpixel level in images, along with the use of fully connected CRFs for label propagation. Mackowiak et al. [25] study region-based AL by estimating labelling cost and uncertainty of unlabelled regions, and Casanova et al. [2] present a deep reinforcement learning-based approach, where an agent learns a policy to select a subset of image patches to label.

**Lifelong learning.** In typical lifelong learning [28, 42], or continual learning, the task is to sequentially train a model for several different tasks in a specific order, while not forgetting the previously learned tasks [19, 18]. Approaches to continual learning for semantic segmentation include storing prototype features of old classes [26], pooling schemes to preserve spatial relations [8] and strategies to cope with the ambiguity and semantic shift of the background class which in different learning stages can overlap with foreground objects at other stages [3].

# 3 Embodied Lifelong Learning

We here describe the proposed embodied lifelong learning task in more detail – see Fig. 4.1 for an overview. An agent equipped with a visual perception model (here a semantic segmentation network; see §3.2) is spawned in a random location in a building, and its task is to actively move around and explore the environment, while restrictively querying annotations for viewpoints in order to refine its visual perception. To this end, the agent is equipped with four actions `MoveForward`, `RotateLeft`, `RotateRight` and `Annotate`. In §3.1 we describe how we approach global navigation to ensure systematic exploration of the scenes; we use the same core navigation module within all methods that we evaluate (see §4). These agents range from heuristic to a fully learnable one. The latter approachs is based on deep RL and is explained in §3.3.

## 3.1 Global Navigation

For an embodied agent to learn a visual perception model that performs well in a given environment, the agent should systematically explore the scene so that it gets an opportunity to observe a diverse set of viewpoints containing various objects. To this end we explicitly map the scene as the agent navigates and provide goal locations for the agent to navigate to. We thus decouple the global navigation from the local exploration and visual learning. A high level overview is given in Fig. 4.2.

The pose and depth at every time step is sent to the mapper which uses the new information to update its map. Using the map, a navigation target is selected to expand the boundary of the map. We use depth and pose to compute a map of the environment. Specifically, given a point $\boldsymbol{x}$ in the image plane with depth $d$ we can compute the 3d point $\boldsymbol{X}$ using the camera

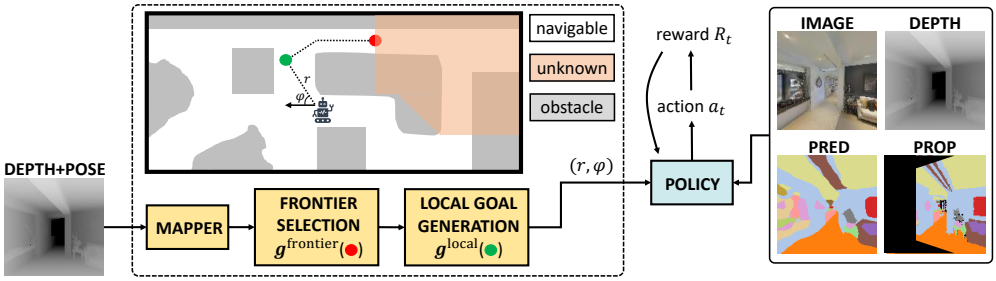


**Figure 4.2:** To ensure systematic exploration of a scene we combine a mapper with a local point goal navigation policy. The mapper constructs a 2d top down map of the environment using the agent's pose and depth, marking each point either as navigable, obstacle or unknown. Each agent (be it heuristic or learnable) uses a hierarchical approach to navigation, where a frontier point $\boldsymbol{g}^{\text{frontier}}$ is first selected; then the agent obtains a local goal $\boldsymbol{g}^{\text{local}}$ on the shortest path to $\boldsymbol{g}^{\text{frontier}}$. For the learnable RL-agent (see §3.3), following prior work on point goal navigation, we give the target location $\boldsymbol{g}^{\text{local}}$ as policy input via polar coordinates $(r, \varphi)$, where $r$ and $\varphi$ denote distance and rotation, respectively, to $\boldsymbol{g}^{\text{local}}$ relative to the agent's pose. After each movement the agent is rewarded in proportion to the decrease in geodesic distance to $\boldsymbol{g}^{\text{local}}$, as well as by a constant reward once it reaches $\boldsymbol{g}^{\text{local}}$, see (4.2). At this point the mapper provides the agent with a new local navigation target. Note that we also provide rewards related to visual perception refinement, see (4.1).

pose. Once we have the 3d points corresponding to the pixels, we threshold based on the height relative to the ground plane to classify as being navigable or an obstacle. For each point on the 2d grid map we keep a counter of the classifications and use majority voting to determine if it is navigable or an obstacle.

We use classical frontier exploration [46] for the global policy. A frontier point is defined as a navigable point on the map that has a neighbouring point for which it is unknown whether it is navigable or an obstacle. The closest frontier point is always selected, and once reached a new frontier point is selected until no more frontier points remain. To ease the navigation problem, as frontier points might be far away from the current location, we use a local planner as well. Using the map, we compute the shortest path to the selected frontier point $\boldsymbol{g}^{\text{frontier}}$ using Dijkstra's algorithm. To select a local navigation target $\boldsymbol{g}^{\text{local}}$, we split the shortest path based on curvature, whenever the direction changes, and use this sequence as navigational targets. This was found on the validation set to be more robust than using a fixed distance between local targets.

## 3.2 Semantic Segmentation

For semantic segmentation we use ResNet-50 [14] pre-trained on ImageNet [16]. We make the network fully convolutional [24] and change the output resolution from $1/32$ to $1/8$ of the input image resolution by removing strides and adding dilations in the ResNet blocks originally having a lower resolution than $1/8$ in the same way as e.g. Chen et al. [7]. We replace the final classification layer with a new one with the correct number of classes, i.e. 40 in Matterport3D, followed by upsampling by bilinear interpolation to the original input

image size.

At the beginning of each episode we either keep the semantic segmentation parameters from the previous episode if we evaluate in the lifelong setup or we initialize all parameters from ImageNet pre-training except for the last classification layer which is initialized randomly. During episodes, we refine the segmentation network whenever the agent selects the `Annotate` action. The image and ground truth that the agent queried for are added to the training set of segmentation network. As in Nilsson et al. [27] we refine the segmentation network either until the accuracy of a minibach is $95\%$ or for at most 1000 iterations. The minibatches always include the last image the agent queried for annotation, and we use random left-right flipping and scaling for data augmentation. We train the semantic segmentation network with minibatches of size 4, and we use SGD with learning rate $3 \cdot 10^{-3}$, weight decay $10^{-4}$ and momentum $0.9$.

### 3.3 Reinforcement Learning for Embodied Lifelong Learning

In this section we describe our proposed reinforcement learning (RL) agent for the embodied lifelong learning task. The global navigation procedure in §3.1 is set as a prior for the RL-agent's exploration. We next describe in detail the policy of the RL-agent and then explain the reward functions used during training.

**Policy overview.** To integrate local navigation with RL we follow prior work on point goal navigation [33]. To this end, we specify the current local navigation target $\boldsymbol{g}_t^{\text{local}}$ by $(r_t, \varphi_t)$ which is the position in polar coordinates of $\boldsymbol{g}_t^{\text{local}}$ relative to the agent's pose. We add $(r_t, \varphi_t)$ to the state space and use it as input to the deep stochastic policy $\pi_\theta(a_t|s_t)$, where we recall that the actions are `MoveForward`, `RotateLeft`, `RotateRight` and `Annotate`. The state space $s_t$ further contains images and segmentations. Specifically, we let $s_t = \{\boldsymbol{I}_t, \boldsymbol{S}_t, \boldsymbol{P}_t, \boldsymbol{D}_t, r_t, \varphi_t\}$ where $\boldsymbol{I}_t \in \mathbb{R}^{127\times127\times3}$ is the agent's current view, $\boldsymbol{S}_t = \mathcal{S}_t(\boldsymbol{I}_t) \in \mathbb{R}^{127\times127\times3}$ is the semantic segmentation of $\boldsymbol{I}_t$ computed using the current version $\mathcal{S}_t$ of the segmentation network, $\boldsymbol{P}_t \in \mathbb{R}^{127\times127\times3}$ is the last annotation propagated to the agent's current pose, and $\boldsymbol{D}_t \in \mathbb{R}^{127\times127}$ is the depth. Note that we use depth as input to the policy since it is common practice in point goal navigation and often significantly improves the performance [33, 44]. The propagated mask $\boldsymbol{P}_t$ is the latest annotation, but propagated to the agent's current view using optical flow. We compute optical flow between consecutive viewpoints using depth and pose. We thus make the same assumptions on available input modalities as the mapper in §3.1. The policy consists of two branches processing the visual information and navigation targets separately, and these are followed by a recurrent network whose hidden state is used to compute the policy. We give further details in the appendix.

**Rewards.** We here describe the reward function. It consists of two components – a *perception reward* to encourage sparse annotations of informative viewpoints which improve the

overall perception, and an *exploration reward* to encourage the agent to explore the whole building.

Perception reward: The reward $R_t^{\text{seg}}$ for semantic segmentation is only non-zero when the agent requests annotation ($a_t = \text{Annotate}$), and we define it in that case as

$$R_t^{\text{seg}} = \text{mIoU}(\mathcal{S}_{t+1}, \mathcal{R}) - \text{mIoU}(\mathcal{S}_t, \mathcal{R}) - \epsilon^{\text{ann}} + \lambda^{\text{acc}} \mathbf{1}(\text{acc}(\mathcal{S}_t, \boldsymbol{I}_t) < \tau^{\text{acc}}) \quad (4.1)$$

where $\mathcal{S}_{t+1}$ and $\mathcal{S}_t$ are respectively the segmentation networks (i.e. parameter configurations) at time $t+1$ and $t$ (i.e. after and before refinement), $\mathcal{R}$ is the set of reference views in the building, $\epsilon^{\text{ann}} = 0.01$ is a penalty (hyperparameter) paid for each `Annotate` action, $\lambda^{\text{acc}} = 0.01$ is a hyperparameter, $\mathbf{1}(\cdot)$ evaluates to one if the condition is true and zero otherwise, and $\tau^{\text{acc}} = 0.7$ is a threshold hyperparameter.

Overall, the perception reward (4.1) is similar to that proposed in Nilsson et al. [27], but we found it useful for the lifelong setup to include an additional term rewarding asking for annotations when the accuracy $\text{acc}(\mathcal{S}_t, \boldsymbol{I}_t)$ of the segmentation network $\mathcal{S}_t$ (prior to refinement) of agent's current view $\boldsymbol{I}_t$ was below a threshold. The reference views $\mathcal{R}$ are 32 randomly sampled views from the entire floor of the scene the agent explores. We compute the mIoU by only selecting the 10 most common classes in the reference views, to exclude very small objects and unusual classes, since the label distribution is very unbalanced in the Matterport3D scenes.

Exploration reward: This reward encourages the agent to move closer to its current local goal $\boldsymbol{g}_t^{\text{local}}$ and is given by

$$R_t^{\text{exp}} = d(\boldsymbol{x}_t, \boldsymbol{g}_t^{\text{local}}) - d(\boldsymbol{x}_{t+1}, \boldsymbol{g}_t^{\text{local}}) + \lambda_g \mathbf{1}(d(\boldsymbol{x}_{t+1}, \boldsymbol{g}_t^{\text{local}}) < \epsilon) \quad (4.2)$$

where $\boldsymbol{x}_t$ is the agent's position at step $t$ and $d(\boldsymbol{x}, \boldsymbol{y})$ is the geodesic distance between the points $\boldsymbol{x}$ and $\boldsymbol{y}$ on the map. We use $\lambda_g = 1$ and $\epsilon = 1$. Note that in contrast to point goal navigation we do not end an episode when a goal is reached, and instead a new goal is given, as described in §3.1.

Full reward: The full reward is given by a convex combination of (4.1) and (4.2), i.e. $R_t = \lambda R_t^{\text{exp}} + (1-\lambda) R_t^{\text{seg}}$ with $\lambda = 0.01$. This reward thus balances between visual learning (i.e. improving the agent's perception) and environment exploration.

**Policy training.** We train the policy using PPO [35]. To train the policy in the lifelong setup, we reset the segmentation network every 10th episode, and run 4 episodes of length 500 per scene. We pre-train the policy for $10^6$ steps for standard point goal navigation, which does not require refinement of the segmentation model, and we then train the policy in the full segmentation environment for an additional $10^6$ steps which takes about 4 days using 2 Nvidia Titan X GPUs. See the appendix for more details and ablations.

# 4 Experiments

In this section we empirically evaluate various methods for our proposed task. Our main focus is to investigate differences in agent behavior (in terms of exploration and annotations strategies) between the episodic and the lifelong setups, as well as compare how the proposed RL-agent performs compared to several baseline methods for the task. We also provide additional experiments including ablation studies of the RL-agent in the appendix.

**Experimental setup.** All methods are evaluated on the Matterport3D dataset [4] and we use the simulator framework Habitat [33] in which an agent can navigate in the photorealistic scenes. We use the official split [4] which contains 61 training scenes, 11 validation scenes and 18 test scenes.

When we evaluate on the test set of Matterport3D, we run one episode per scene, for a total of 18 episodes. In a given scene, all methods we evaluate start at the same position. We end the episodes either when the agent has explored all of the scene or after 2000 steps. In each scene we train and evaluate the semantic segmentation network using train and test sets in that scene. The training data is obtained by the queries of the agent we evaluate and the test set is obtained by randomly sampling 32 views uniformly in the whole scene (we always use the same random views for a given scene when we evaluate different methods). Hence we obtain one mIoU score per scene, and we only use the 10 most common labels per scene when computing the mIoU due to very unbalanced data. When we report mIoU we always refer to the average over all 18 test scenes. Similarly, when we report metrics as a function of the exploration, we refer to the area marked as navigable by the mapper (cf. §3.1) normalized by the total area (separately within each scene) and then average metrics over all scenes.

**Episodic and lifelong setups.** We differentiate between episodes where the segmentation network is reset at the start of episodes or not by the terms *episodic* and *lifelong*. In the episodic case we reset the segmentation network at the beginning of the episode. Thus no prior segmentation network learning or annotations are kept and the agents begin each episode tabula rasa. In the lifelong setup we keep the parameters of the segmentation network from the previous episode (e.g. at the beginning of the 10th episode the perception system has been trained in nine different previous buildings). Thus the agent will typically be able to recognize some objects and/or semantic classes already at the beginning of the episode, and we expect the agent's exploration to be less granular and that it annotates less frequently since similar scenes or objects might have been seen before. Note that when evaluating the RL-agent (cf. §3.3) in the respective settings, the exact same policy is used to be able to assess the agent's adaptability with respect to its current perception. We use the same ordering of the 18 test scenes for all methods we evaluate, and found that the ordering had a negligible impact (see appendix).

**Table 4.1:** Comparison between episodic and lifelong setups for the accuracy oracle, RL-agent and baselines (uniform, random). We see that the average area explored since the last annotation ($\Delta A/$ annot, $\mathrm{m}^2$) when requesting a new annotation is higher on average in the lifelong compared to the episodic setup for both the accuracy oracle and the RL-agent, indicating adaptive behavior depending on the current segmentation accuracy. This is not the case for the two simpler heuristics. The average navigable area added to the mapper per step ($\Delta A/$ step, $\mathrm{m}^2$) is identical for episodic and lifelong for the accuracy oracle and the two baselines, as is expected since they explore identically, but for the RL-agent we see that in the lifelong setup it on average explores faster. Finally, we see that the improvements of the average mIoU after annotations 1 - 50 and 51 - 100 in the lifelong setup compared to the episodic are larger for the accuracy oracle and RL-agent than for the baselines, and especially for the early part of episodes (annotations 1 - 50). This indicates that both the RL-agent and the accuracy oracle adapt their annotation strategies when evaluated in the lifelong setup.

| Model | Setup | $\Delta A$ / annot | $\Delta A$ / step | mIoU (1-50) | mIoU (51-100) |
|---|---|---|---|---|---|
| Accuracy oracle | Episodic | 1.162 | 0.056 | 0.306 | 0.429 |
| | Lifelong | 1.256 (+0.094) | 0.056 | 0.340 (+0.034) | 0.448 (+0.019) |
| RL-agent | Episodic | 0.876 | 0.057 | 0.286 | 0.397 |
| | Lifelong | 1.070 (+0.194) | 0.059 | 0.324 (+0.038) | 0.407 (+0.010) |
| Uniform | Episodic | 1.086 | 0.057 | 0.297 | 0.402 |
| | Lifelong | 1.086 ($\pm$0) | 0.057 | 0.311 (+0.014) | 0.409 (+0.007) |
| Random | Episodic | 1.114 | 0.057 | 0.296 | 0.395 |
| | Lifelong | 1.063 (-0.051) | 0.057 | 0.303 (+0.007) | 0.393 (-0.002) |

**Evaluated methods.** We evaluate the RL-agent (cf. §3.3) and several baselines that make use of the global exploration methodology described in §3.1. Since we use a global mapper and planner, we let the baseline agents consist of variants which follow shortest paths to the generated navigation goals. We evaluate such agents with different annotation strategies. The baselines are:

- **Accuracy oracle.** Annotates if the semantic segmentation accuracy of its current view is below a threshold (set to 0.7 based on a validation set). Since this baseline requires access to ground truth segmentations we call it an oracle, and it should be considered an upper bound.
- **Uniform annotations.** Annotates with a fixed frequency, which we set to every 20th step to match the average frequency of the accuracy oracle (making them comparable). This is used to show how much can be gained by choice of annotations (as the accuracy oracle) compared to heuristics.
- **Random annotations.** Requests annotations randomly with probability 5% and thus it moves according to the global mapper with 95% probability. This baseline also annotates with the same average frequency as the accuracy oracle.

We further compare to a pre-trained segmentation model using 10,000 random views from the training scenes of Matterport3D. We use 50 of the 61 training scenes for training data and 11 for validation and use the same ResNet model (cf. §3.2) as for all other evaluations.

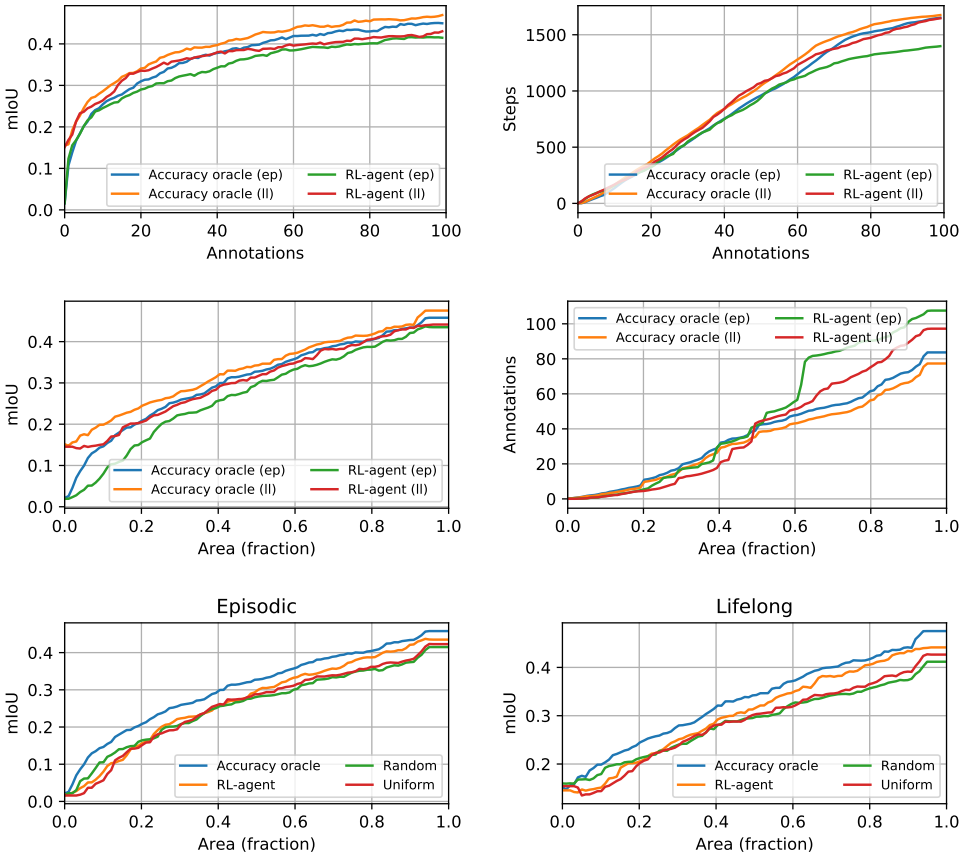


**Figure 4.3:** Comparison between episodic (ep) and lifelong (ll) setups for the RL-agent and the accuracy oracle, and comparison of the RL-agent to the accuracy oracle and heuristic baselines for both the episodic and lifelong setup. In the four top plots we see that for a given number of annotations or area explored, both methods get a higher mIoU in the lifelong setup compared to the episodic. We can also see adaptive behavior in the annotation frequencies. For a given number of annotations, both methods take more steps on average in the lifelong setup than the episodic, indicating less frequent annotations when evaluated in the lifelong setting. We can see the same behavior if we look at the number of annotations for a given explored area – both methods annotate less frequently in the lifelong setup than in the episodic one. In the two bottom plots we compare all baselines. As expected, the accuracy oracle obtains the highest mIoU. The RL-agent obtains the second highest mIoU at the end of exploration. Moreover, we see a larger improvement of the RL-agent compared to uniform and random in the lifelong setup than in the episodic one.

## 4.1 Comparing Lifelong and Episodic Setups

In this section we provide comparisons between the episodic and lifelong setups for the RL-agent and the accuracy oracle, see Table 4.1 and Fig. 4.3. We see that the mIoU is higher at the beginning of episodes, either for the first annotations or for limited observed area, for lifelong compared to episodic. This is expected since in the lifelong setup the semantic segmentation network has already been trained in previously explored scenes, and thus there

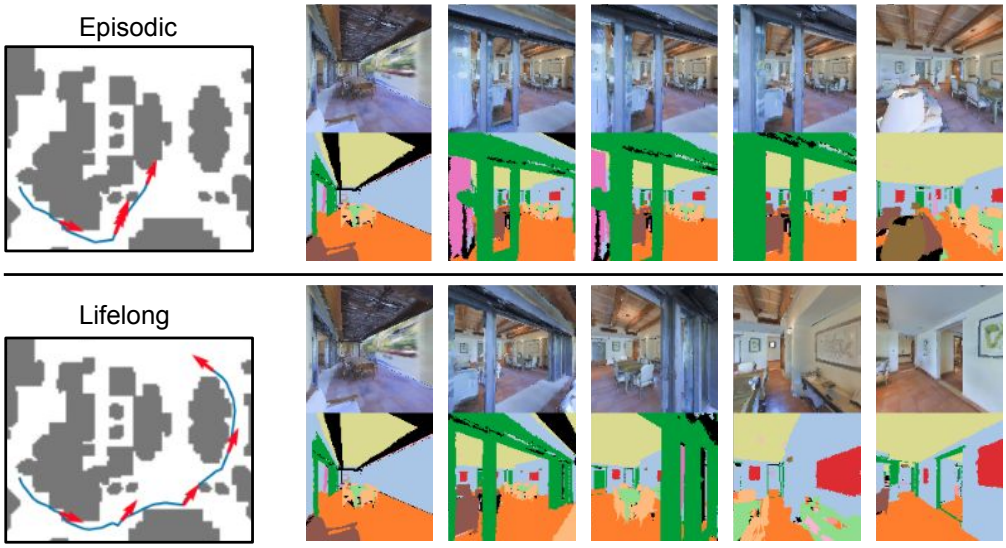


**Figure 4.4:** The first five requested annotations when evaluating the RL-agent in the episodic versus lifelong setup. The agent requests annotations at a sparser rate when evaluated lifelong in this scene. For more qualitative examples, please see the appendix.

is some transfer to the current one. Similarly, for both methods we observe differences in annotation frequency between the two settings, e.g. the average area explored per requested annotation is higher in the lifelong setup for both of these methods. We see that while both methods annotate more sparsely in the lifelong setup compared to the episodic, the mIoU is still higher for a comparable number of annotations or level of exploration.

In Fig. 4.4 we show example trajectories of the RL-agent on a scene, and we compare the requested annotations in the episodic and lifelong setups. We see that the RL-agent annotates more sparsely in the lifelong setup than in the episodic one, and thus explores more of the scene per annotation. This can also be seen more generally in Table 4.1 – the average explored area per annotation is higher in the lifelong setup than the episodic for the RL-agent. Thus the agent is able to adapt its behavior based on its current visual perception performance.

## 4.2 Comparisons with the Heuristic Baselines

In Fig. 4.3 and Table 4.1 we provide comparisons of the RL-agent to the baselines which explore with frontier exploration and request annotations using different heuristic strategies. By comparing the accuracy oracle, uniform and random, we see that for a given area explored, the accuracy oracle gets a higher mIoU. As seen in Table 4.1, all methods annotate at a similar frequency with respect to the area, but only for the accuracy oracle do we observe a difference between the area explored per annotation between episodic and lifelong. We also see that the

gap in mIoU between the episodic and lifelong setups is greater for the accuracy oracle than uniform and random. These results suggest that uniform and random are unable to adapt with respect to the performance of their perception models.

If we compare the RL-agent to uniform and random, we see that the mIoU is similar in the beginning of episodes, but at the end of exploration the RL-agent gets a higher mIoU. We also see that there is a larger difference in the lifelong setup. The increase in mIoU in the lifelong setup compared to the episodic one is greater for the RL-agent than the heuristic baselines. This indicates that the RL-agent is able to adapt its active learning strategy based on its current perception performance.

### 4.3 Pre-training the Segmentation Model

If we pre-train the segmentation model on the training scenes of Matterport3D we obtain an mIoU of 0.330 when we evaluate on the test scenes. We note in Fig. 4.3 that this performance is comparable to about 20 annotations in a given scene for the accuracy oracle or RL-agent. The conclusion is that a few ($\sim$ 20) scene-specific annotations are as useful as significantly more (10,000) annotations from different buildings. Although surprising, we attribute this to significant appearance differences of the scenes in Matterport3D. This further shows that our proposed task is of importance since a few carefully selected scene-specific annotations are more useful than pre-training with significantly more annotations.

## 5 Conclusions

We have introduced and studied the novel *embodied lifelong learning* task in the context of semantic segmentation, where agents are tasked to explore large buildings and request limited annotations to refine their visual perception. The refinement occurs both within the currently explored building, and improves over the lifetimes of the agents as they tune their perception in an increasing number of buildings. We further introduced a reinforcement learning methodology to jointly learn exploration and visual learning. Our experiments on Matterport3D, covering both the learned method as well as several heuristic ones, show that a trainable method using RL annotates less frequently in the lifelong setup while also achieving more accurate visual perception than when trained from scratch. Overall, different from the compared heuristic alternatives, the learning-based agent exhibits behavior which adapts to the current performance of its visual perception model.

**Acknowledgments:** This work was supported in part by the European Research Council Consolidator grant SEED, CNCS-UEFISCDI PN-III-P4-ID-PCE-2016-0535 and PCCF-2016-0180, the EU Horizon 2020 Grant DE-ENIGMA, Swedish Foundation for Strategic Research (SSF) Smart Systems

Program, as well as the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation.

# Bibliography


[1] P. Ammirato, P. Poirson, E. Park, J. Košecká, and A. C. Berg. A dataset for developing and benchmarking active vision. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pages 1378–1385. IEEE, 2017.

[2] A. Casanova, P. O. Pinheiro, N. Rostamzadeh, and C. J. Pal. Reinforced active learning for image segmentation. *arXiv preprint arXiv:2002.06583*, 2020.

[3] F. Cermelli, M. Mancini, S. R. Bulo, E. Ricci, and B. Caputo. Modeling the background for incremental learning in semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9233–9242, 2020.

[4] A. Chang, A. Dai, T. Funkhouser, M. Halber, M. Niessner, M. Savva, S. Song, A. Zeng, and Y. Zhang. Matterport3d: Learning from rgb-d data in indoor environments. *International Conference on 3D Vision (3DV)*, 2017.

[5] D. S. Chaplot, D. Gandhi, S. Gupta, A. Gupta, and R. Salakhutdinov. Learning to explore using active neural slam. In *International Conference on Learning Representations*, 2019.

[6] D. S. Chaplot, H. Jiang, S. Gupta, and A. Gupta. Semantic curiosity for active visual learning. In *European Conference on Computer Vision*, pages 309–326. Springer, 2020.

[7] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):834–848, 2017.

[8] A. Douillard, Y. Chen, A. Dapogny, and M. Cord. Plop: Learning without forgetting for continual semantic segmentation. *arXiv preprint arXiv:2011.11390*, 2020.

[9] M. Ducoffe and F. Precioso. Adversarial active learning for deep networks: a margin based approach. *arXiv preprint arXiv:1802.09841*, 2018.

[10] L. Engstrom, B. Tran, D. Tsipras, L. Schmidt, and A. Madry. Exploring the landscape of spatial robustness. In *International Conference on Machine Learning*, pages 1802–1811. PMLR, 2019.

[11] M. Fang, Y. Li, and T. Cohn. Learning how to active learn: A deep reinforcement learning approach. *arXiv preprint arXiv:1708.02383*, 2017.

[12] Y. Gal, R. Islam, and Z. Ghahramani. Deep bayesian active learning with image data. In *International Conference on Machine Learning*, pages 1183–1192. PMLR, 2017.

[13] S. A. Golestaneh and K. M. Kitani. Importance of self-consistency in active learning for semantic segmentation. *arXiv preprint arXiv:2008.01860*, 2020.

[14] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[15] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[16] Jia Deng, Wei Dong, R. Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009. ISBN 978-1-4244-3992-8. doi: 10.1109/CVPRW.2009.5206848.

[17] T. Kasarla, G. Nagendar, G. M. Hegde, V. Balasubramanian, and C. Jawahar. Region-based active learning for efficient labeling in semantic segmentation. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1109–1117. IEEE, 2019.

[18] R. Kemker, M. McClure, A. Abitino, T. Hayes, and C. Kanan. Measuring catastrophic forgetting in neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[19] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

[20] A. Kirsch, J. van Amersfoort, and Y. Gal. Batchbald: Efficient and diverse batch acquisition for deep bayesian active learning. In *NeurIPS*, 2019.

[21] K. Konyushkova, R. Sznitman, and P. Fua. Discovering general-purpose active learning strategies. *arXiv preprint arXiv:1810.04114*, 2018.

[22] A. Krizhevsky, I. Sutskever, and G. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, 2012.

[23] E. Liang, R. Liaw, R. Nishihara, P. Moritz, R. Fox, K. Goldberg, J. Gonzalez, M. Jordan, and I. Stoica. Rllib: Abstractions for distributed reinforcement learning. In *International Conference on Machine Learning*, pages 3053–3062. PMLR, 2018.

[24] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[25] R. Mackowiak, P. Lenz, O. Ghori, F. Diego, O. Lange, and C. Rother. Cereals-cost-effective region-based active learning for semantic segmentation. *arXiv preprint arXiv:1810.09726*, 2018.

[26] U. Michieli and P. Zanuttigh. Continual semantic segmentation via repulsion-attraction of sparse and disentangled latent representations. *arXiv preprint arXiv:2103.06342*, 2021.

[27] D. Nilsson, A. Pirinen, E. Gärtner, and C. Sminchisescu. Embodied visual active learning for semantic segmentation. *AAAI*, 2021.

[28] G. I. Parisi, R. Kemker, J. L. Part, C. Kanan, and S. Wermter. Continual lifelong learning with neural networks: A review. *Neural Networks*, 113:54–71, 2019.

[29] K. Pei, Y. Cao, J. Yang, and S. Jana. Towards practical verification of machine learning: The case of computer vision systems. *arXiv preprint arXiv:1712.01785*, 2017.

[30] E. Pot, A. Toshev, and J. Kosecka. Self-supervisory signals for object discovery and detection. *arXiv preprint arXiv:1806.03370*, 2018.

[31] S. K. Ramakrishnan, D. Jayaraman, and K. Grauman. An exploration of embodied visual exploration. *International Journal of Computer Vision*, pages 1–34, 2021.

[32] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[33] M. Savva, A. Kadian, O. Maksymets, Y. Zhao, E. Wijmans, B. Jain, J. Straub, J. Liu, V. Koltun, J. Malik, et al. Habitat: A platform for embodied ai research. *arXiv preprint arXiv:1904.01201*, 2019.

[34] A. Sax, J. O. Zhang, B. Emi, A. Zamir, S. Savarese, L. Guibas, and J. Malik. Learning to navigate using mid-level visual priors. In *Conference on Robot Learning*, pages 791–812. PMLR, 2020.

[35] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[36] O. Sener and S. Savarese. Active learning for convolutional neural networks: A core-set approach. In *International Conference on Learning Representations*, 2018.

[37] B. Settles. Active learning literature survey. Technical report, University of Wisconsin-Madison Department of Computer Sciences, 2009.

[38] Y. Siddiqui, J. Valentin, and M. Nießner. Viewal: Active learning with viewpoint entropy for semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9433–9443, 2020.

[39] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[40] S. Sinha, S. Ebrahimi, and T. Darrell. Variational adversarial active learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5972–5981, 2019.

[41] S. Song, L. Zhang, and J. Xiao. Robot in a room: Toward perfect object recognition in closed environments. *CoRR, abs/1507.02703*, 2015.

[42] S. Thrun and T. M. Mitchell. Lifelong robot learning. *Robotics and autonomous systems*, 15(1-2): 25–46, 1995.

[43] K. Wang, Y. Lin, L. Wang, L. Han, M. Hua, X. Wang, S. Lian, and B. Huang. A unified framework for mutual improvement of slam and semantic segmentation. In *ICRA*, pages 5224–5230. IEEE, 2019.

[44] E. Wijmans, A. Kadian, A. Morcos, S. Lee, I. Essa, D. Parikh, M. Savva, and D. Batra. Dd-ppo: Learning near-perfect pointgoal navigators from 2.5 billion frames. *arXiv preprint arXiv:1911.00357*, 2019.

[45] F. Xia, A. R. Zamir, Z. He, A. Sax, J. Malik, and S. Savarese. Gibson env: Real-world perception for embodied agents. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9068–9079, 2018.

[46] B. Yamauchi. Frontier-based exploration using multiple robots. In *Proceedings of the second international conference on Autonomous agents*, pages 47–53, 1998.

[47] J. Yang, Z. Ren, M. Xu, X. Chen, D. J. Crandall, D. Parikh, and D. Batra. Embodied amodal recognition: Learning to move to perceive objects. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2040–2050, 2019.

[48] F. Zhong, S. Wang, Z. Zhang, and Y. Wang. Detect-slam: Making object detection and slam mutually beneficial. In *WACV*, pages 1001–1010. IEEE, 2018.

# Appendix

## A Overview

In this appendix material we provide additional insights into our proposed embodied lifelong learning task as well as the RL-based approach for modeling the task. Qualitative examples are shown in §B, studies of the scene ordering are shown in §C, ablations of the RL-based agent are performed in §D, further details of the RL training are given in §E and details of the hyperparameter tuning of the accuracy oracle in §F.

## B Qualitative Examples

We show predicted segmentations before and after refinements in Fig. 4.5. As expected, since in the episodic case the classification layer is initialized randomly, the segmentation is inaccurate at the start of episodes, but it is improved after the first annotation. We emphasize that we refine the segmentation model with *one* annotated image at a time, so in the final rows, the segmentation model is trained with just 4 annotated images. The lifelong agent has prior knowledge, and we can see that some classes e.g. wall, ceiling and floor are fairly accurate in the beginning of episodes, while several smaller objects are not correctly segmented at the beginning but are refined later on, as the segmentation model is retrained with scene-specific annotations that the agent requested.

In Fig. 4.6 we show a qualitative example of how the accuracy oracle and RL-agent annotates in the episodic and lifelong setups. We can see that both methods annotates less frequently in the lifelong setup than the episodic in this scene.

## C Impact of Scene Orderings

When we evaluate the agents in the lifelong setup we use a fixed ordering of the 18 test scenes which was selected without any special considerations. We use same ordering for all evaluations of different methods in the main paper. To see how robust the performance is if we change the ordering, we try multiple random orderings. In Table 4.2 we show results for the accuracy oracle, for episodic and lifelong as in the main paper and additionally for lifelong with 3 different random orderings of the 18 scenes. We see that the metrics are very close to the results for lifelong as reported in the main paper, and conclude that the specific ordering has a negligible effect on performance.

**Table 4.2:** We evaluate the accuracy oracle baseline on three different random scene orderings of the 18 test scenes. The first two rows are identical to that of the results table in the main paper. We see that the metrics are very close in all evaluations in the lifelong setup meaning that the specific ordering used in the main paper has little influence on the results.

| Model | Setup | $\Delta A$ / annot | $\Delta A$ / step | mIoU (1-50) | mIoU (51-100) |
|---|---|---|---|---|---|
| Accuracy oracle | Episodic | 1.162 | 0.056 | 0.306 | 0.429 |
| | Lifelong | 1.256 | 0.056 | 0.340 | 0.448 |
| | Lifelong, ordering 1 | 1.264 | 0.057 | 0.340 | 0.451 |
| | Lifelong, ordering 2 | 1.275 | 0.057 | 0.339 | 0.444 |
| | Lifelong, ordering 3 | 1.258 | 0.057 | 0.338 | 0.442 |

# D Ablation Studies for the RL-Agent

Various ablations of the RL-agent[4] proposed in the main paper are given in Table 4.3. We compare to the following versions:

- **Episodic training.** When training the full RL-agent we reset the segmentation network every 10th episode, while for this version we reset it every episode. The agent thus trains in the same way as it is evaluated in the episodic setup.
- **No accuracy reward.** We discard the term giving a reward if the accuracy of the agent's current view is below a threshold. Hence we let $\lambda^{\text{acc}} = 0$ in the segmentation reward $R_t^{\text{seg}}$.
- **No navigation pre-training.** We train the policy directly in the full segmentation environment without the navigation pre-training (whereas the full RL-agent was pre-trained for $10^6$ steps for point goal navigation).
- **No global exploration.** We use a spatial coverage reward similar to that in Nilsson et al. [27] instead of the global exploration with point goal navigation to local navigation targets.

For consistency with the test results in the main paper we report the average mIoU after 1-50 and 51-100 annotations. However, mIoU(51-100) is slightly inconclusive since the agents sometimes stop before 100 annotations and the annotation frequencies differ significantly, and much more than the methods we compare on the test set in the main paper. Due to this we also report the average number of annotations per episode (rightmost column in Table 4.3). Since more frequent annotations lead to a larger coverage of the objects in the scene we can expect a higher mIoU if the frequency is significantly higher, but this comes at a cost of requiring more annotations.

[4] We refer to the RL-agent in the main paper as the *full* RL-agent (or sometimes simply the RL-agent).

**Table 4.3:** Ablation study of the RL-agent proposed in the main paper. These models are evaluated on the 11 validation scenes of Matterport3D. We also include the accuracy oracle for reference. The evaluated metrics are the same as in the main paper for consistency, and we have added a final column (right) which shows the average number of annotations per episode. Note that the full RL-agent (2nd row) queries for significantly fewer annotations per episode in the lifelong compared to the episodic setup, which we can also see for episodic training (3rd row) but not the other ablated versions. Moreover, despite fewer annotations (11.7% less in lifelong than episodic), the *increase* in mIoU is simultaneously by far the highest for the RL-agent, when looking at lifelong vs episodic.

| Model | Setup | $\Delta A$ / annot | $\Delta A$ / step | mIoU (1-50) | mIoU (51-100) | Annots |
|---|---|---|---|---|---|---|
| Accuracy oracle | Episodic | 1.452 | 0.077 | 0.368 | 0.471 | 65.8 |
| | Lifelong | 1.530 (+0.078) | 0.077 | 0.395 (+0.027) | 0.490 (+0.019) | 62.4 (-3.4) |
| RL-agent | Episodic | 0.998 | 0.066 | 0.301 | 0.395 | 95.9 |
| | Lifelong | 1.076 (+0.078) | 0.065 | 0.362 (+0.061) | 0.446 (+0.051) | 84.7 (-11.2) |
| Episodic training | Episodic | 0.765 | 0.092 | 0.312 | 0.399 | 131.6 |
| | Lifelong | 0.856 (+0.091) | 0.071 | 0.336 (+0.024) | 0.424 (+0.025) | 106.2 (-25.4) |
| No acc reward | Episodic | 0.529 | 0.058 | 0.308 | 0.417 | 143.2 |
| | Lifelong | 0.551 (+0.022) | 0.071 | 0.335 (+0.027) | 0.455 (+0.038) | 147.0 (+3.8) |
| No nav pre-training | Episodic | 0.558 | 0.059 | 0.309 | 0.424 | 137.3 |
| | Lifelong | 0.557 (-0.001) | 0.068 | 0.322 (+0.013) | 0.407 (-0.017) | 134.9 (-2.4) |
| No global exploration | Episodic | 0.924 | 0.033 | 0.305 | 0.354 | 55.8 |
| | Lifelong | 0.665 (-0.259) | 0.033 | 0.342 (+0.037) | 0.395 (+0.041) | 79.8 (+24.0) |

We can see that the full RL-agent gets a significantly larger difference in mIoU between the episodic and lifelong setup than the ablated versions. The mIoU is higher in the lifelong setup despite using fewer average number of annotations per episode. Compared to episodic training (resetting the segmentation network every episode) we see that the full RL-agent gets a significantly higher mIoU in the lifelong setup. If we do not use an explicit accuracy reward we see that the annotation frequency is the same in the episodic and lifelong setups, and the difference in mIoU is smaller. If we do not pre-train for navigation the mIoU does not significantly improve in the lifelong setup compared to the episodic. Finally we see that if we use a spatial coverage reward instead of the global navigation with local navigation targets, the exploration is significantly worse, since the average area explored per step ($\Delta A$ / step) is about half of the other methods.

# E Further Details of the RL-Agent

For the policy training we use a simplified network for semantic segmentation to significantly speed up training. With this modification training is about 5 times faster than using Resnet-50. We use a nine layer network with three blocks with three convolutional layers each, followed by upsampling and classification. The blocks have 64, 128 and 256 channels respectively, and the first convolutional layer in each block uses stride 2, so the resolutions are $1/2$, $1/4$ and $1/8$ per block. During training we use image size $127 \times 127$ but for all evaluations we use Resnet-50 with image size $256 \times 256$.

The policy consists of two branches processing the visual information and navigation targets separately, and these are followed by a recurrent network whose hidden state is used to com-

pute the policy. The first branch is a four-layer CNN $\phi_{\text{img}}(\boldsymbol{I}_t, \boldsymbol{S}_t, \boldsymbol{P}_t, \boldsymbol{D}_t) \in \mathbb{R}^{512}$ which processes the image, segmentations and depth. The second branch is $\phi_{\text{nav}}(r_t, \varphi_t) \in \mathbb{R}^{128}$ which processes the navigation targets. It consists of two fully connected layers of size 64 and 128. We concatenate $\phi_{\text{img}}$ and $\phi_{\text{nav}}$ to a vector of length 640 and use that as an input to an LSTM [15] with 256 hidden states. Finally we compute the policy $\pi(a_t|s_t)$ from the hidden state using a fully-connected layer of size 4 followed by softmax to get a probability distribution over the actions.

The policy network contains $\phi_{\text{img}}$ which processes the image, segmentations and depth. This network consists of 3 convolutional layers, with 32, 64 and 64 channels respectively, filter sizes 8, 4 and 3, and strides 4, 2 and 1. This is followed by global average pooling and a fully-connected layer with 512 units.

We train the policy using PPO [35] and use the implementation in RLlib [23]. We use learning rate $3 \cdot 10^{-4}$. To train the policy in the lifelong setup, we reset the segmentation network every 10th episode, and run 4 episodes of length 500 per scene. We pre-train the policy for $10^6$ steps for standard point goal navigation, where we replace the predicted segmentation $\boldsymbol{S}_t$ and the propagated segmentation $\boldsymbol{P}_t$ with the ground truth segmentation in the state space and removed the `Annotate` action. We then train the policy in the full segmentation environment for an additional $10^6$ steps. The pre-training is significantly faster since the segmentation network is not used and not refined. Training $10^6$ steps in the full environment takes about 4 days using 2 Nvidia Titan X GPUs, and the pre-training takes 10 hours on a single GPU.

# F Accuracy Oracle Thresholding

We empirically found the 0.7 threshold to give the largest mIoU. We show in Table 4.3 that the accuracy oracle with the 0.7 threshold obtains mIoU(1-50)=0.395, mIoU(51-100)=0.490 using on average 62.4 annotations on the validation scenes. If we use 0.6 as threshold we get mIoU(1-50)=0.357, mIoU(51-100)=0.426 using on average 47.7 annotations and for 0.8 as threshold we get mIoU(1-50)=0.367, mIoU(51-100)=0.454 using on average 92.0 annotations. With frequent annotations, once an agent reaches 50 annotations it will only have explored a small area, while with infrequent annotations it will annotate sparsely and miss large parts of the scene.

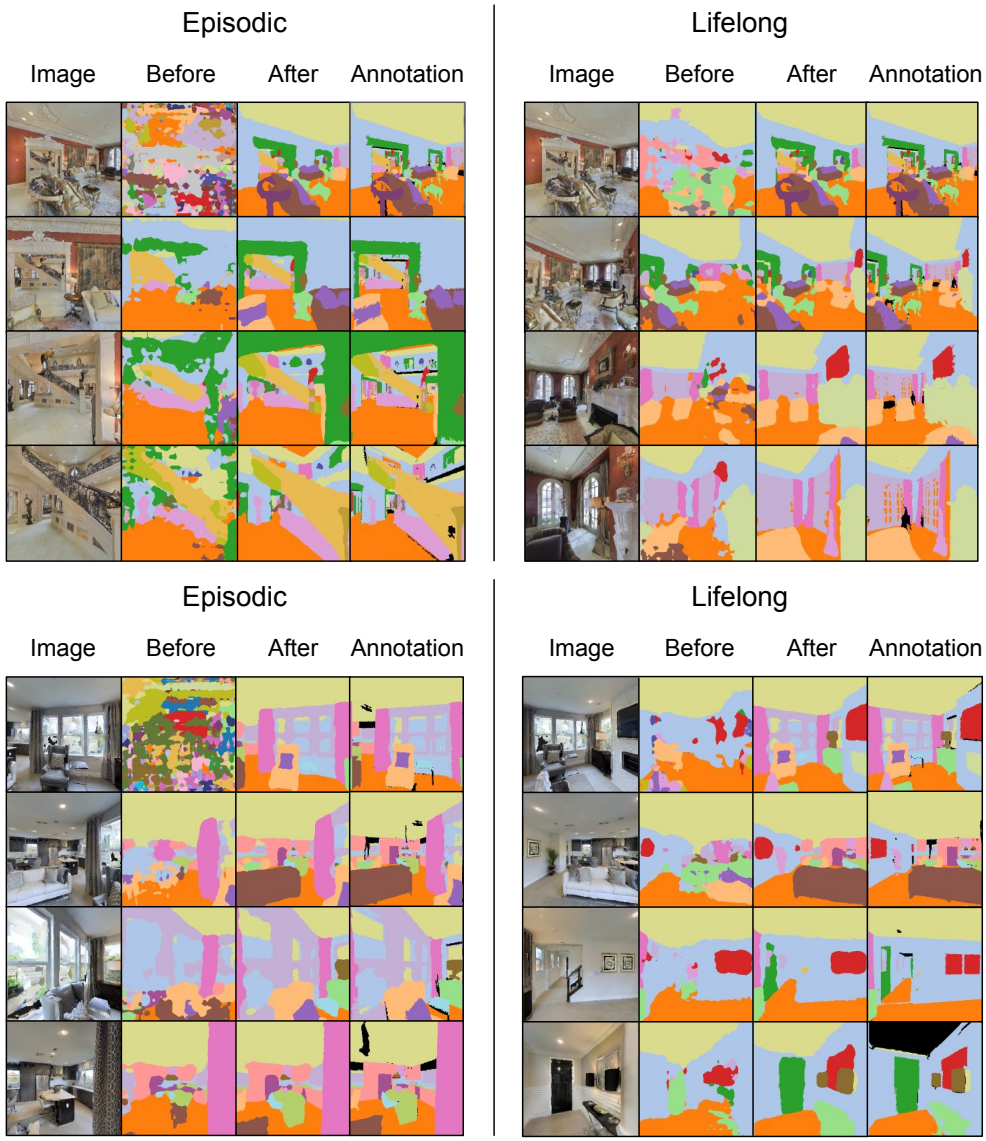


**Figure 4.5:** The first four queried annotations, as well as the predicted segmentations before and after refinement, when evaluating the RL-agent in the episodic and lifelong setups in two different buildings from the test set. We can see in the lifelong setup there is some prior knowledge, however it is mostly contained to wall, ceiling and floor. We also note that many of the objects are seen from unusual viewpoints, e.g. the couch in the first images in the first scene. Note e.g. the painting on the left in the second image for the second scene for the lifelong agent. It is correctly recognized despite not being seen in the first annotation.

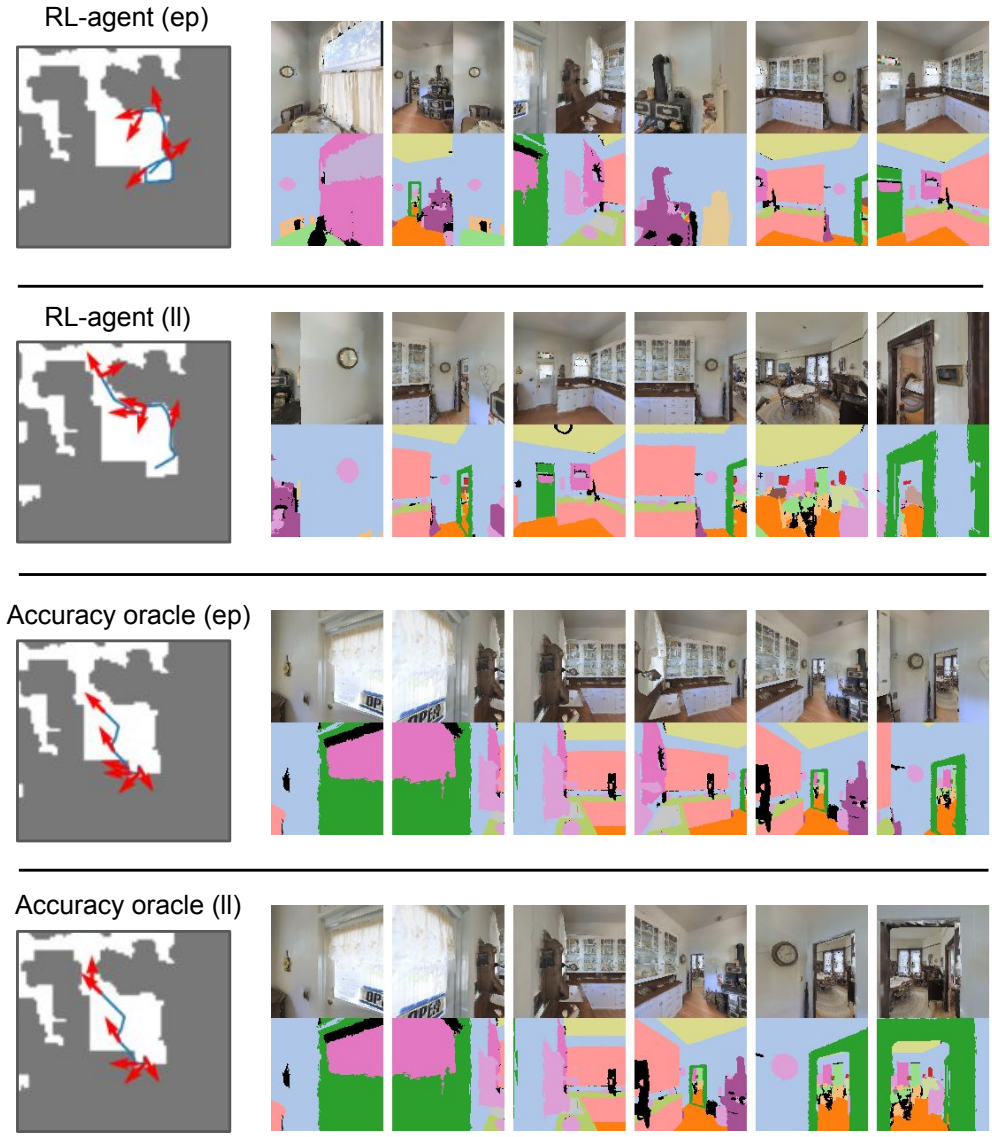


**Figure 4.6:** The first six queried annotations when evaluating the accuracy oracle and the RL-agent in the episodic (ep) and lifelong (ll) setups in a building from the test set. We can see that both agents request annotations at a sparser rate when evaluated lifelong in this scene.

# Paper IV

# Online Learning of Semantic Segmentation via Embodied Region-Based Active Learning

David Nilsson[1], Erik Gärtner[1], Aleksis Pirinen[2], Cristian Sminchisescu[1,3]
[1]Department of Mathematics, Faculty of Engineering, Lund University
[2]RISE Research Institutes of Sweden
[3]Google Research

## Abstract

We study the embodied visual active learning problem for semantic segmentation with fine-grained and hence cost-efficient annotations. In the embodied visual active learning problem, an agent is tasked to navigate in a scene and occasionally, for carefully selected views, request annotations in order to improve its visual perception. Earlier works have studied this problem using active learning for entire images, but this can be cost-inefficient if the perception is already accurate for most parts of an image queried for annotation. In this paper we instead consider region-based active learning, where an agent only queries for the ground truth of small regions in images. We propose a reinforcement learning (RL) based approach to this problem, and develop a policy architecture which controls both the navigation and the active learning. Specifically, for the active learning component the agent must predict *where* in each image in addition to *when* along the navigation trajectory to query for segmentation labels. We experimentally validate our proposed method on the Matterport3D dataset and show that our RL-agent outperforms pre-specified baselines. We also compare to earlier work [23] which assumes full image annotations, and show that our proposed region-based approach obtains a similar segmentation performance despite only using 26% of the annotations on average.

# 1 Introduction

In many application of computer vision, convolutional neural networks (CNNs) achieve excellent performance, but they can fail for unusual viewpoints, or on domains they are not trained on [1, 34]. In robot vision, for example, objects are often observed from viewpoints which differ from those in carefully curated datasets. This motivates a principled methodology for adapting a visual perception system to the specific domain a robot is deployed in.

In this paper we study embodied visual active learning, where an agent should develop visual perception via embodied exploration of a scene. This is achieved by having the agent explore and observe different viewpoints in the scene, where it for certain views requests annotations to refine its visual perception. Earlier works [22, 23, 6] have studied the embodied visual active learning problem, but they only consider active learning where the full image is annotated whereas we study the problem with more fine-grained annotations for only parts of each image. An overview of our setup is shown in Fig. 5.1. Our region-based approach is useful in case the segmentation fails only at small parts of an image, so that only those regions should be annotated rather than whole images, where multiple parts might already be accurate and annotating those parts might not improve the perception.

We compare multiple methods for this problem on the photorealistic Matterport3D dataset [5], and we propose to use an RL based approach to learn a policy for the problem. As in prior work, an agent is tasked to both navigate and select views for active learning, but different to earlier, we additionally require the agent to select *where* in an image to query for annotations. The region prediction is also modelled via RL, where the action space is augmented to both contain actions for movements and when to annotate, but different to before also predictions for where in the image to annotate.

In summary, our contributions are:

- We study the embodied visual active learning problem with more granular annotations than earlier works. We use region-based active learning where just parts of images are annotated whereas prior work for this problem use full image annotations.
- We provide a methodology based on RL to handle the fine-grained annotations, and experimentally show that it performs better than multiple baselines.
- We compare multiple methods on the Matterport3D dataset, including our RL approach as well as agents that navigate via frontier exploration and use heuristic active learning. Specifically, we show that with our region-based annotations we can reach the same visual recognition performance as in prior work which used full image annotations, but with only a quarter of the annotation cost.

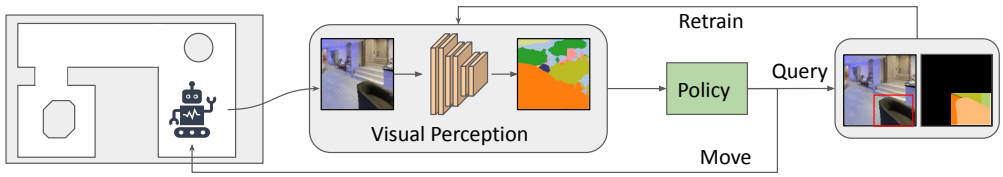


**Figure 5.1:** An overview of our setup. An agent is placed in an environment, with the goal of acquiring visual perception in the form of accurate semantic segmentation. At each time, the agent can either select to move in the environment or to query for the correct labelling of a part of its current view. If it queries for annotation, the agent must select a specific region in the image to annotate. This is then used to refine the semantic segmentation network. The goal of an agent is to explore a whole scene while getting as accurate perception as possible with a limited number of annotations.

## 2 Related Work

We here present related work. We focus on the two areas most relevant for our work which is embodied learning with an emphasis toward methods that use embodiment to improve visual perception, and then we present related work on active learning for semantic segmentation.

**Embodied learning.** We will here review work using embodied learning, especially navigation, to improve a visual perception system. Active learning in an embodied context has been studied before [22, 23], but unlike our work, they considered only the labelling of full images. A similar setup is studied by Chaplot et al. [6] where an agent find navigation trajectories giving useful views to refine segmentations. They use a curiosity reward based on 3d consistency, encouraging the agent to observe objects from multiple views. In contrast to our paper all views along the trajectories are labelled, and there is no online re-training during the navigation. Chaplot et al. [7] considers self-supervised instance segmentation improvement via exploration to gather pseudo-labels via 3d consistency, where the exploration is trained via a reward function encouraging observing highly confident predictions. Using pseudo-labels obtained via 3d was also explored in Fang et al. [9] where the labels are gathered via embodied exploration. Kotar and Mottaghi [18] improve a pre-trained object detector at test time via embodiment without requiring explicit supervision. There is also work on how to move to improve the detection score or reducing the uncertainty of a visual model, e.g. moving to get an non-occluded view either via embodiment [37, 1] or by predicting the next best view [24, 27, 15], however in these approaches the viewpoints change while the visual perception in general is not refined with additional data. There is also work on improving visual perception using methods for self-supervised domain adaption, and using embodiment to gather informative views for such systems to work well [40]. Related is also work on mutual improvement of segmentation and SLAM [33, 39], but these works consider pre-collected trajectories and not how to explore.

**Active learning for semantic segmentation.** In active learning [29, 25], the task is to among a large pool of unlabelled data select the ones to labels and use as training data for a learning system. Successful approach for general computer vision tasks are via core-sets [28] which selects diverse examples measured by distance between deep features, VAAL [32] which learns a VAE [17] to discriminate between labelled and unlabelled data, uncertainty estimation of CNNs [10], learning the acquisition function [12] or by estimating the task loss for unlabelled data [38].

We now present prior work on active learning for semantic segmentation. Unlike our online setup, most of the work presented here is pool based active learning, which means that all unlabelled data is available at all time, and none of the presented methods feature embodiment like in our setup. The methods that are presented differ in how fine-grained the annotations are, e.g. images, regions, superpixels or even single pixels, and the criteria used to select what to annotate. Casanova et al. [4] present an RL approach to region-based active learning for semantic segmentation, by learning the acquisition function via Q-learning. CEREALS [20] is proposed for region-based active learning by estimating the labelling cost of regions combined with uncertainty estimations based on the CNN outputs. Another line of work [35, 8] design acquisition functions by predicting which regions in the image that are difficult to segment. Golestaneh and Kitani [11] rank unlabelled imaged by seeing how robust they are to equivariant transformations such as left-right flipping, and select regions to annotate that are not robust to such changes. It is also possible use multi-view datasets and consider robustness of the predictions to viewpoint changes for the active learning [31]. Belharbi et al. [2] propose to in addition to the oracle labelling also use pseudo-labels by weakly supervised learning. There are also works considering very fine-grained annotations such as annotating single pixels at a time [30] or using superpixels [3, 16].

# 3 Methodology

In this section we present our methodology, of which we show an overview in Fig. 5.2. We start by describing our setup, specifically how the regions are selected and which actions an agent can take. We then describe our approach to the problem which uses RL. Finally we describe our semantic segmentation network and its training details.

## 3.1 Setup

As in earlier works [22, 23], an agent is tasked with exploring and occasionally requesting annotations for informative views in the exploration trajectory. However, these works consider the annotation to be the whole image. In this paper we further task an agent with selecting parts of images to annotate instead of the whole image. For simplicity we partition the image

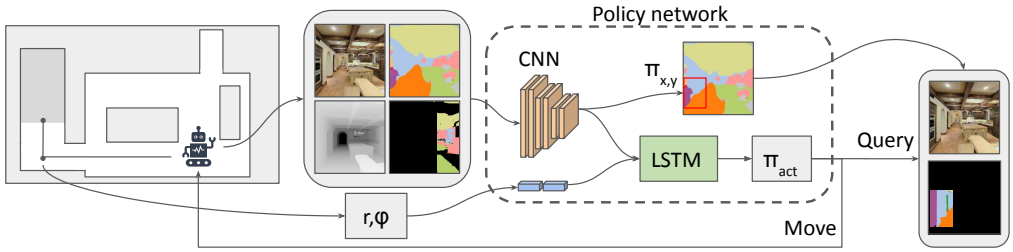


**Figure 5.2:** Overview of our policy architecture. The first part of the policy, $\pi_{act}$ is to select either a movement action (move forward, rotate left or rotate right) or to request an annotation. If the agent selects an annotation, then the region selected by $\pi_{x,y}$ is annotated and used to refine the segmentation network. The inputs to the CNN encoder are the image $I_t$, the depth $D_t$, the predicted segmentation $S_t$ and the propagated annotations $P_t$. A navigation target is given on the shortest path to the current frontier point, which is used to ensure systematic and exhaustive exploration. The navigation targets $(r, \varphi)$ are given in polar coordinates.

into $N_r \times N_r$ equally sized patches. We use $N_r = 4$ in all experiments unless otherwise stated.

We assume that the action space is a vector $a = (a_{act}, a_x, a_y)$ where $a_{act}$ is the agent's action in the environment and $(a_x, a_y)$ is the region selection, since an agent predicts both movements and positions of the bounding box. Specifically, we assume that the policy $\pi(a|s)$ is factorized as

$$\pi(a|s) = \pi(a_{act}, a_x, a_y|s) = \pi_{act}(a_{act}|s)\pi_x(a_x|s)\pi_y(a_y|s) \tag{5.1}$$

where $a_{act} \in \{\text{MoveForward, RotateLeft, RotateRight, Annotate}\}$ is the action of the agent and $a_x, a_y \in 1, \ldots, N_r$[5] is the bounding box selection.

## 3.2 Reinforcement Learning

In this section we describe our approach to the problem using RL. Specifically we describe the state space, policy network, reward function and training details.

The state space $s_t = \{I_t, S_t, P_t, D_t, r_t, \varphi_t\}$ consists of the images $\{I_t, S_t, P_t, D_t\}$ of size $127 \times 127$ where $I_t$ is the image, $S_t$ the predicted segmentation, $P_t$ the annotations propagated to the current view via optical flow, and $D_t$ which is the depth. We also provide navigational targets $\{r_t, \varphi_t\}$ in polar coordinates relative the agent's pose as given by frontier exploration [36] as in earlier work [23] to ensure global and systematic exploration of the scene. A frontier point is selected, and a local navigation target is selected along the shortest path to the frontier point.

[5] We also tried to parameterize the bounding box selection in the action space via the joint distribution as $\pi_{x,y}(a_x, a_y|s)$ for the RL training but found that it performed slightly worse.

Our model is shown in Fig. 5.2. We predict $\pi(a_x, a_y|s)$ by applying a convolution with one output channel following the last layer of the encoder of the policy network and then resizing to $N_r \times N_r$. We go to the marginalized distributions using $\pi_x(a_x|s) = \sum_{a_y} \pi(a_x, a_y|s)$ and $\pi_y(a_y|s) = \sum_{a_x} \pi(a_x, a_y|s)$. The LSTM [14] has 256 units. The CNN consists of 3 layers with filter sizes 32, 64 and 64, kernel sizes 8, 4 and 3 and strides 4, 2 and 1 respectively, and the last convolutional layer is followed by global average pooling and a dense layer with 512 outputs. The navigation targets $(r, \varphi)$ are processed prior to the LSTM by 2 dense layers with 64 and 128 outputs.

The reward function is similar to earlier work [23], but adapted for the region-based setup. It is given by by combining exploration (exp) rewards with segmentation (seg) rewards as $R_t = \lambda R_t^{\text{exp}} + (1 - \lambda) R_t^{\text{seg}}$ with $\lambda = 0.001$. The segmentation reward is only non-zero when the agent selects to annotate ($a_{act}$ =Annotate) and is in that case given by

$$R_t^{\text{seg}} = \text{mIoU}(\mathcal{S}_{t+1}, \mathcal{R}) - \text{mIoU}(\mathcal{S}_t, \mathcal{R}) - \epsilon^{\text{ann}} + \lambda^{\text{acc}} \mathbf{1}(\text{acc}(\mathcal{S}_t, I_t(a_x, a_y))) < \tau^{\text{acc}}) \tag{5.2}$$

where $\mathcal{S}_t$ is the segmentation network at step $t$, and $\mathcal{R}$ is a set of 32 randomly sampled reference views for the current episode, $I_t(a_x, a_y)$ is the region selected to be annotated and $\text{acc}(\mathcal{S}_t, I_t(a_x, a_y))$ is the accuracy of the selected region prior to the online refinement of the semantic segmentation network. We used $\epsilon^{\text{ann}} = 0.001$, $\lambda^{\text{acc}} = 0.003$ and $\tau^{\text{acc}} = 0.5$. The exploration reward is given by

$$R_t^{\text{exp}} = d(x_t, g_t^{\text{local}}) - d(x_{t+1}, g_t^{\text{local}}) + \lambda_g \mathbf{1}(d(x_{t+1}, g_t^{\text{local}}) < \epsilon) \tag{5.3}$$

where $d(\cdot, \cdot)$ is the geodesic distance, $x_t$ is the position of the agent and $g_t^{\text{local}}$ is the local navigation target on the shortest path to the currently selected frontier point. We use $\lambda_g = 1$ and $\epsilon = 1$.

The policy is trained with PPO [26] using RLlib [19] with learning rate $3 \cdot 10^{-4}$ for $10^6$ environment steps. We train the system using 2 GPUs, and it takes about 4 days to train a model. We use a simplified semantic segmentation network from [22] during policy training to speed it up.

### 3.3 Semantic Segmentation

For the semantic segmentation, we largely follow the setup in [23], namely we use a ResNet-50 [13] pretrained on ImageNet. At the start of each episode we initialize the last classification layer randomly. We use a standard cross-entropy loss, and mini-batches if size 4, which always include the latest annotated region. The cross-entropy loss is only computed for pixels in the regions that are annotated, but we always provide the full images as inputs to the segmentation network since the annotated regions themselves not always provide adequate context

for accurate predictions, see the images and regions in Fig. 5.5. Every time an agent annotates the segmentation network is refined online until the accuracy of a mini-batch exceeds 95% or for at most 1000 iterations.

# 4 Experiments

In this section we present experimental results. We start by the describing the experimental setup, followed by the test results. We then show ablation studies of the RL-agent and finally experiments where the region sizes vary.

## 4.1 Experimental setup

All methods are evaluated on Matterport3D [5] using the Habitat simulator [21], and largely follow the experimental setup in [23], which is described again in this section. We use the official split of Matterport3D with 61 training scenes, 11 validation scenes and 18 test scenes. When we evaluate on the validation set or test set, we always start at the same position for all methods considered, and the reference views used to assess the segmentation accuracy are identical for a given scene. We evaluate one episode per scene and end an episode either when the agent has taken 2000 actions or when the agent has finished exploring, which we for practical purposes define as when 95% of the navigable area has been seen.

Our main metric is the mIoU on a set of reference view, sampled uniformly in each scene. When comparing different agents in a given scene, the segmentation performance is always compared on identical reference view for fairness. Due to a very unbalanced distribution of labels in the scenes, we only look at the mIoU of the 10 most common classes among the reference views of the current scene. We specifically look at the mIoU as a function of the number of annotations and as a function of explored area. We always average the mIoUs over all scenes when we report, and when we report mIoU as a function of area, we normalize the area per scene to a value between 0 and 1. When reporting mIoU(i-j) we look at the average mIoU for annotations i to j, averaged over all scenes. Note that mIoU(i-j) is proportional to the area under the curve between annotations $i$ and $j$ when plotting mIoU versus the number of annotations.

We compare our RL method to several baselines listed below. The baselines differs in their annotation policies while the navigation policy follows shortest paths via frontier exploration.

- Region Oracle. The agent select the least accurate region to annotate whenever the segmentation accuracy on the region is less than a threshold, which was selected to be

**Table 5.1:** Results on the test set of Matterport3D. The region oracle consistently obtains the highest mIoU, but since it uses the ground truth for the region selection it should be considered an upper bound which we compare other methods to. The RL-agent outperforms the baseline selecting random regions to annotate. Compared to using full images for supervision, we see that all region-based methods obtain a much higher mIoU for a given number of annotations. Note that for [23], each annotation is for the full image which corresponds to 16 regions, so 100 region-based annotations are $100/16$ full image annotations.

| Method | mIoU(1-100) | mIoU(101-200) | Final mIoU | Average annotations |
|---|---|---|---|---|
| Region Oracle | 0.219 | 0.324 | 0.443 | 456 |
| RL - Region-Based | 0.210 | 0.304 | 0.422 | 453 |
| Random Regions | 0.201 | 0.293 | 0.391 | 452 |
| RL - Full images [23] | 0.141 | 0.230 | 0.435 | 1737 |

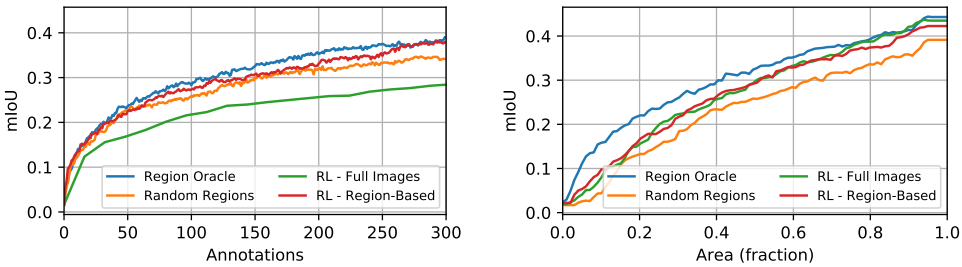


**Figure 5.3:** Results on the test scenes. We can see that the region oracle is the most efficient, but we emphasize that since it uses the ground truth for the region selection, we can not directly compare to it; rather, we can see how close other methods are to it. The RL-agent is better than random regions both when considering the mIoU as a function of the number of annotations and area. Finally we see that compared to full image supervision [23], the region-based RL-agent reaches comparable performance as a function of the explored area, but for a given number of annotations, the region-based methods are all significantly better than the method annotating full images.

50% based on experiments on the validation set. This baseline requires constant access to ground truth data and should be considered an upper bound.

- Random Regions. This agent annotates with a fix probability, and selects a random region uniformly in the image to annotate. The annotation probability was set to 25% to match the frequency of the region oracle.

We also compare our method to the earlier work in [23]. In contrast to us, they use full image supervision. We always take that into account when reporting annotations so that 1 full image annotation corresponds to 16 region-based annotations since we used $N_r = 4$.

## 4.2 Main Results

In Fig. 5.3 and Table 5.1 we show the results on the test set. We can see that the region oracle obtains the highest mIoU, both as a function of the number of annotations and as a function of the area, and that the RL-agent using region-based annotations outperforms the baseline

**Table 5.2:** Ablations of the RL-agent. We show results on the validation split of Matterport3D. As can be seen, the model with the region prediction branch consistently obtains a higher mIoU than the model without.

| Method | mIoU(1-100) | mIoU(101-200) | Final mIoU | Average annotations |
|---|---|---|---|---|
| Full model | 0.235 | 0.339 | 0.406 | 412 |
| No region branch | 0.217 | 0.319 | 0.399 | 410 |

selecting random regions to annotate. We note that the difference between the region oracle and the other methods is especially large in the beginning of episodes where little area has been explored, and that the difference is not as large if we consider the mIoU per annotation.

We compare our proposed region-based active learning methodology to earlier work [23] using full image supervision. We make the assumption that annotation effort is proportional to image area which is reasonable for cluttered scenes with multiple objects. We note that it is significantly more labelling efficient to use region-based supervision than full image supervision. For a given level of annotations, the mIoU is significantly higher for the region-based agent. We also note that the final mIoU is similar for the RL-agents, but that the region-based agent uses only about $26\%$ of the annotations compared to the agent using full image supervision.

We show qualitative examples in Fig. 5.5, with five consecutive regions the RL-agent selected to be annotated from two different scenes. Each annotated region should ideally contribute towards general visual perception of the agent in the scene, and we can see in the refined segmentations that the selected region is refined, as expected, but often also parts outside of the selected region.

## 4.3 Ablation Studies of the RL-agent

To verify the effectiveness of our proposed region-based RL-agent we compare the following models.

- Full model. The RL-agent as described in §3.2.
- No region prediction branch. We append branches for $\pi_x$ and $\pi_y$ directly following the LSTM (cf. Fig. 5.2) instead of creating a separate branch for the bounding box prediction. Note that this is a naive alteration of the policy architecture in [23].

We can see in Table 5.2 that the full model outperforms the ablated version on the validation set of Matterport3D.

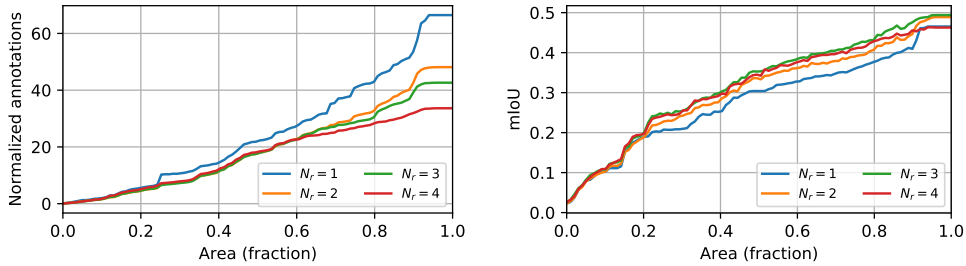


**Figure 5.4:** We vary the number of regions $N_r$ (an image has $N_r \times N_r$ regions that can be queried for annotations) for the region oracle with threshold $0.7$, on the validation set. The exploration trajectories are identical for fair comparison and only the annotations differ. We report normalized annotations which means annotated area in terms of full images, so 1 normalized annotation is e.g. 16 annotations with $N_r = 4$ or 1 annotation with $N_r = 1$. We see that for a given area, it is in general possible to obtain a higher mIoU using fewer normalized annotations as $N_r$ increases.

## 4.4 Different Region Sizes

To investigate the effect of the number of regions $N_r \times N_r$ we split our image into, we here provide experiments for the region oracle baseline with $N_r \in \{1, 2, 3, 4\}$ in Fig. 5.4. We see that for a given explored area, as we increase $N_r$, the mIoU is in general higher despite on average asking for less annotations. This shows that the main assumption behind our work is valid, namely that region-based annotations are more efficient than annotating whole images.

# 5 Conclusions

We have presented a setup and methodology for the embodied visual active learning problem where the annotations are based on image regions rather than full images. We have shown that region-based annotations are more efficient in terms of gained mIoU per annotation area compared to using full images. We presented a methodology based on RL to handle both the navigation and selection of which regions to annotate. Our proposed RL-agent outperforms baselines, and is much more efficient compared to prior work using full image supervision.

**Figure 5.5:** Examples of the selected regions of the RL-agent. We show five consecutive annotations from two different scenes. We show the image, the ground truth segmentation, and the predicted segmentations before and after the online retraining. We can see that for most of the annotations, there is something incorrect in the regions which is corrected after the refinement. We can also see that although only the selected regions are annotated, sometimes also incorrect segmentations outside of the region are corrected after the refinement, e.g. the ceiling in the top image to the left.

# Bibliography


[1] P. Ammirato, P. Poirson, E. Park, J. Košecká, and A. C. Berg. A dataset for developing and benchmarking active vision. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1378–1385. IEEE, 2017.

[2] S. Belharbi, I. Ben Ayed, L. McCaffrey, and E. Granger. Deep active learning for joint classification & segmentation with weak annotator. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 3338–3347, 2021.

[3] L. Cai, X. Xu, J. H. Liew, and C. S. Foo. Revisiting superpixels for active learning in semantic segmentation with realistic annotation costs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10988–10997, 2021.

[4] A. Casanova, P. O. Pinheiro, N. Rostamzadeh, and C. J. Pal. Reinforced active learning for image segmentation. In *International Conference on Learning Representations*, 2019.

[5] A. Chang, A. Dai, T. Funkhouser, M. Halber, M. Niessner, M. Savva, S. Song, A. Zeng, and Y. Zhang. Matterport3d: Learning from rgb-d data in indoor environments. *International Conference on 3D Vision (3DV)*, 2017.

[6] D. S. Chaplot, H. Jiang, S. Gupta, and A. Gupta. Semantic curiosity for active visual learning. In *European Conference on Computer Vision*, pages 309–326. Springer, 2020.

[7] D. S. Chaplot, M. Dalal, S. Gupta, J. Malik, and R. R. Salakhutdinov. Seal: Self-supervised embodied active learning using exploration and 3d consistency. *Advances in Neural Information Processing Systems*, 34, 2021.

[8] P. Colling, L. Roese-Koerner, H. Gottschalk, and M. Rottmann. Metabox+: A new region based active learning method for semantic segmentation using priority maps. In *ICPRAM*, 2021.

[9] Z. Fang, A. Jain, G. Sarch, A. W. Harley, and K. Fragkiadaki. Move to see better: Self-improving embodied object detection. *arXiv preprint arXiv:2012.00057*, 2020.

[10] Y. Gal, R. Islam, and Z. Ghahramani. Deep bayesian active learning with image data. In *International Conference on Machine Learning*, pages 1183–1192. PMLR, 2017.

[11] S. A. Golestaneh and K. M. Kitani. Importance of self-consistency in active learning for semantic segmentation. *arXiv preprint arXiv:2008.01860*, 2020.

[12] M. Haußmann, F. Hamprecht, and M. Kandemir. Deep active learning with adaptive acquisition. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pages 2470–2476, 2019.

[13] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[14] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[15] D. Jayaraman and K. Grauman. Learning to look around: Intelligently exploring unseen environments for unknown tasks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1238–1247, 2018.

[16] T. Kasarla, G. Nagendar, G. M. Hegde, V. Balasubramanian, and C. Jawahar. Region-based active learning for efficient labeling in semantic segmentation. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1109–1117. IEEE, 2019.

[17] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[18] K. Kotar and R. Mottaghi. Interactron: Embodied adaptive object detection. *arXiv preprint arXiv:2202.00660*, 2022.

[19] E. Liang, R. Liaw, R. Nishihara, P. Moritz, R. Fox, K. Goldberg, J. Gonzalez, M. Jordan, and I. Stoica. Rllib: Abstractions for distributed reinforcement learning. In *International Conference on Machine Learning*, pages 3053–3062. PMLR, 2018.

[20] R. Mackowiak, P. Lenz, O. Ghori, F. Diego, O. Lange, and C. Rother. Cereals-cost-effective region-based active learning for semantic segmentation. *arXiv preprint arXiv:1810.09726*, 2018.

[21] Manolis Savva*, Abhishek Kadian*, Oleksandr Maksymets*, Y. Zhao, E. Wijmans, B. Jain, J. Straub, J. Liu, V. Koltun, J. Malik, D. Parikh, and D. Batra. Habitat: A Platform for Embodied AI Research. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.

[22] D. Nilsson, A. Pirinen, E. Gärtner, and C. Sminchisescu. Embodied visual active learning for semantic segmentation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 2373–2383, 2021.

[23] D. Nilsson, A. Pirinen, E. Gärtner, and C. Sminchisescu. Embodied learning for lifelong visual perception. *arXiv preprint arXiv:2112.14084*, 2021.

[24] S. K. Ramakrishnan, D. Jayaraman, and K. Grauman. Emergence of exploratory look-around behaviors through active observation completion. *Science Robotics*, 2019.

[25] P. Ren, Y. Xiao, X. Chang, P.-Y. Huang, Z. Li, B. B. Gupta, X. Chen, and X. Wang. A survey of deep active learning. *ACM Computing Surveys (CSUR)*, 54(9):1–40, 2021.

[26] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[27] S. Seifi and T. Tuytelaars. Where to look next: Unsupervised active visual exploration on 360 deg input. *arXiv preprint arXiv:1909.10304*, 2019.

[28] O. Sener and S. Savarese. Active learning for convolutional neural networks: A core-set approach. *arXiv preprint arXiv:1708.00489*, 2017.

[29] B. Settles. Active learning literature survey. 2009.

[30] G. Shin, W. Xie, and S. Albanie. All you need are a few pixels: semantic segmentation with pixelpick. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1687–1697, 2021.

[31] Y. Siddiqui, J. Valentin, and M. Nießner. Viewal: Active learning with viewpoint entropy for semantic segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9433–9443, 2020.

[32] S. Sinha, S. Ebrahimi, and T. Darrell. Variational adversarial active learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5972–5981, 2019.

[33] K. Wang, Y. Lin, L. Wang, L. Han, M. Hua, X. Wang, S. Lian, and B. Huang. A unified framework for mutual improvement of slam and semantic segmentation. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 5224–5230. IEEE, 2019.

[34] G. Wilson and D. J. Cook. A survey of unsupervised deep domain adaptation. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 11(5):1–46, 2020.

[35] S. Xie, Z. Feng, Y. Chen, S. Sun, C. Ma, and M. Song. Deal: Difficulty-aware active learning for semantic segmentation. In *Proceedings of the Asian Conference on Computer Vision*, 2020.

[36] B. Yamauchi. A frontier-based approach for autonomous exploration. In *Proceedings 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation CIRA'97.'Towards New Computational Principles for Robotics and Automation'*, pages 146–151. IEEE, 1997.

[37] J. Yang, Z. Ren, M. Xu, X. Chen, D. J. Crandall, D. Parikh, and D. Batra. Embodied amodal recognition: Learning to move to perceive objects. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2040–2050, 2019.

[38] D. Yoo and I. S. Kweon. Learning loss for active learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 93–102, 2019.

[39] F. Zhong, S. Wang, Z. Zhang, and Y. Wang. Detect-slam: Making object detection and slam mutually beneficial. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1001–1010. IEEE, 2018.

[40] R. Zurbrügg, H. Blum, C. Cadena, R. Siegwart, and L. Schmid. Embodied active domain adaptation for semantic segmentation via informative path planning. *arXiv preprint arXiv:2203.00549*, 2022.