Deliverable D2.5: Learning scene representations in realistic environments

Due Date: 24/05/2022

Main Author: UNITN

Contributors: CVUT

Dissemination: Public Deliverable

This project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement No. 871245.

This project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement No. 871245.

DOCUMENT FACTSHEET

Deliverable	D2.5: Learning scene representations in realistic environments
Responsible Partner	UNITN
Work Package	WP2: Environment Mapping, Self-localisation and Simulation
Task	T2.4: Scene Representations for Localisation and Mapping
Version & Date	24/05/2022
Dissemination	Public Deliverable

CONTRIBUTORS AND HISTORY

Version	Editor	Date	Change Log
1	UNITN	20/05/2022	First Draft
2	CVUT	24/05/2022	Added CVUT contribution
3	CVUT	31/05/2022	Final version

APPROVALS

Authors/editors	UNITN, CVUT
Task Leader	CVUT
WP Leader	CVUT

This project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement No. 871245.

Contents

This project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement No. 871245.

Executive Summary

The current deliverable is part of WP2 of the H2020 SPRING project. The deliverable is a software deliverable that includes methods for performing semantic segmentation and instance segmentation at image level and in point cloud. In addition, it proposes two novel methods to efficiently perform *continual learning* on semantic segmentation and depth estimation. The deliverable reflects the progress made by the consortia in relation to T2.4. Additional tasks related to behavioral cues mapping and object affordances will be included in D2.7 at M39.

The prototype implementation of tools for the instance segmentation is built on top of the YOLACT [4] method. This 2D segmentation Convolutional Neural Network (CNN) generates prototype masks of individual objects using a projection head on the ResNet-101 backbone. The prototypes are linearly combined, leading to the real-time speed required for online data processing [1].

The following tool focuses on the same task processed on top of the point cloud from the depth estimation or MVS tools. We employed the SSTNet [44] leading the ScanNet instance segmentation benchmark[2]. It is the first end-to-end trainable CNN that constructs a segmentation tree. Tree traversal and the splitting module provide proposals for further pruned and refined objects [3].

The spatio-temporal segmentation [11] is a high-dimensional convolutional neural network for 4D spatio-temporal data. We used the published example on 3D data with a fixed time. The algorithm is based on an auto-differentiation library for sparse tensors and sparse convolution, that is, the Minkowski engine [11]. The sparse tensors outperform the classical CNN and lead to SoTA results in quality and time criteria [4].

Regarding the incremental learning model for semantic segmentation, we propose a new framework, namely uncertainty-aware contrast distillation (UCD) that has been recently published in IEEE TPAMI [70]. The model proposes a novel contrastive distillation loss that accounts for semantic association among pixels of the new and old model. The method adopts an uncertainty estimation strategy for the contrastive learning framework that leverages the joint probability of the pixel pairs belonging to the same class and weights the strength of the distillation signal accordingly. Additionally, the model has been proven to be beneficial on top of most of the traditional semantic segmentation frameworks on a number of publicly available benchmarks [5]. The work has been published in the IEEE Transaction on Pattern Analysis and Machine Intelligence in 2022 [70].

The next section includes a novel framework to estimate depth from of a monocular camera. This module was not included initially in the proposal, but it has been considered lately due to the benefit that such estimation can bring to a number of downstream tasks. The proposed models is the first involving transformers for both monocular depth estimation and surface normal prediction tasks. Transformers can successfully improve the ability of traditional convolutional neural networks to model long-range dependencies. In this work we propose a novel and effective unified attention gate structure designed to utilize and fuse multi-scale information in a parallel manner and pass information among different affinities maps in the attention gate decoders for better modeling the multi-scale affinities. The model is tested on a number of publicly available data sets and has recently been published at the International Conference on Computer Vision (ICCV) 2021 [71] [6]. The method has been recently published at the International Conference of Computer Vision 2021 (i.e. [71]).

The last section and the tool provided in D2.5 include the state-of-the-art multiview stereo method Patchmatch-Net [67]. This algorithm builds on top of the cascade pipeline composed of Patchmatch and depth estimation models. The depth is estimated on small resolution images and iteratively refined on larger and larger pictures. Such an approach allows fast depthmaps estimation without dependence on an image size, which is the limitation of the voxel-based method [7].

As per European Commission requirements, the repository will be available to the public for a duration of at least four years after the end of the SPRING project.

[1] **Instance segmentation**: https://gitlab.inria.fr/spring/wp2_mapping_localization/2d-instance-segmentation-yolact
[2] **ScanNet benchmark**: http://kaldir.vc.in.tum.de/scannet_benchmark/data_efficient/lr_semantic_instance_3d
[3] **Instance segmentation of the point cloud**: https://gitlab.inria.fr/spring/wp2_mapping_localization/3d-instance-segmentation-sstnet
[4] **Semantic segmentation for the point cloud**: https://gitlab.inria.fr/spring/wp2_mapping_localization/3d-semantic-segmentation-minkowskiengine
[5] **Incremental learning for semantic segmentation**: https://gitlab.inria.fr/spring/wp4_behavior/non-integrated-contributions/UCD.git
[6] **Depth estimation**: https://gitlab.inria.fr/spring/wp4_behavior/non-integrated-contributions/TransDepth.git
[7] **Multi-view depth estimantion**: https://gitlab.inria.fr/spring/wp2_mapping_localization/mvs-depth-estimation-patchmatchnet

This project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement No. 871245.

1 Contributions

1.1 Introduction

This deliverable **D2.5** is part of **WP2** of the H2020 SPRING project, which aims to present the result of T2.4: Scene Representations for Localization and Mapping.

In this context, we present:

- The first prototype implementation of tools for **instance segmentation** (Sec. 1.2).
- The first prototype implementation of tools for **instance segmentation of the point cloud** (Sec. 1.3).
- The first prototype implementation of tools for **semantic segmentation of the point cloud** (Sec. 1.4).
- An extended description of a novel method to perform **incremental learning** that is directly applicable to most of the SOTA methods for **semantic segmentation** (Sec. 1.5).
- An extended description of a novel method to perform **depth estimation** (Sec. 1.6)
- The first prototype implementation of tools for **Multi-View Stereo depth estimation** (Sec. 1.6.3).

1.2 Instance segmentation

Since we want our system to work in real-life scenarios, object detection should be as fast as possible. Two-stage instance segmentation approaches, e.g., Mask-RCNN [31], generates candidate region-of-interests in the first stage and classifies plus segments those regions in the second stage. Re-pooling features and further computations make them unable to run in real-time on large scale images. One-stage instance segmentation methods without re-pooling features lead to much faster processing. YOLACT [4] is a real-time instance segmentation method that split the segmentation into two parallel parts: generating prototype masks and predicting per-instance mask coefficients. Final instance masks are then produced as a linear combination of prototype masks with predicted coefficients. A running demo of YOLACT instance segmentation is available on GitLab[1].

1.3 Instance segmentation of the point cloud

One of the fundamental methods for understanding the scene is the instance segmentation of the dense point cloud in 3D space. The commonly employed scheme in SoTA methods [21, 35, 78] first learns features for individual points that are discriminative enough, and, further apply the point grouping to get individual objects. Therefore, the second step is not supervised by the main objective of instance segmentation and point-wise features may lead to fragmented segmentations if the point cloud contains irregularities. We work on recently published Semantic Superpoint Tree Network (SSTNet) [44]. It provides end-to-end solution to overcome listed challenges. It first use a backbone to learns point-wise features and assign them soft semantic scoring vectors. In parallel, SSTNet over-segment the point cloud. The points and related feature vectors from found segments are composed into superpoints by average pooling. Next, the divisive grouping in a top-down manner build the semantic superpoint tree where superpoints realize the leaves and the tree root the whole point cloud. Tree traversal and splitting module provides proposals for instance objects in a form of non-spliting decision on a tree branch. Such proposals are further refined by CliqueNet to prune some of the branch nodes and ScoreNet [35] to evaluate the generated proposals. A running demo of SSTNet is available on GitLab[2].

[1] **GitLab repository**: https://gitlab.inria.fr/spring/wp2_mapping_localization/2d-instance-segmentation-yolact

[2] **GitLab repository**: https://gitlab.inria.fr/spring/wp2_mapping_localization/3d-instance-segmentation-sstnet

Table 1.1: 3D Semantic Label Benchmark evaluation on ScanNet dataset [14] [11]

Method	Mean IOU
ScanNet [14]	30.6
SSC-UNet [28]	30.8
TangentConv [63]	43.8
SurfaceConv [50]	44.2
3DMV [15]	48.4
PointNet++SW [51]	52.3
MinkowskiNet42 (voxel size 5cm) [11]	**67.9**
SparseConvNet [28]	72.5
MinkowskiNet42 (voxel size 2cm) [11]	**73.4**

Table 1.2: Semantic segmentation results on Stanford Area 5 dataset [2] [11]

Method	Mean IOU	Mean Accuracy
PointNet [51]	41.09	48.98
SparseUNet [27]	41.72	64.62
TangentConv [63]	52.80	60.70
PointCNN [42]	57.26	63.86
SuperpointGraph [40]	58.04	66.50
MinkowskiNet20	**62.60**	**69.62**
MinkowskiNet32	**65.35**	**71.71**

1.4 Semantic Segmentation of the point cloud

This section focuses on semantic segmentation of point clouds, see Figure 1.1.

3D convolutional networks can be split in three groups. First group uses a rectangular grid and a dense representation [64, 14]. This representation, while intuitive and supported by all major public NN libraries, is however rather memory and computation demanding as most of the space in the grid is empty. To resolve this, OctNet [54] proposed to use the Octree structure to represent 3D space and convolution on it. The second group uses sparse 3D-convolutional neural net- works [60, 27]. There are two quantization methods used for high dimensions: a permutohedral lattice and a rectangular grid. SplatNet [60] uses a the former approach while SparseUNet [27] uses the latter for 3D classification and semantic segmentation. The last group consists of 3D pseudo-continuous convolutional neural networks [33, 42]. Unlike the previously mentioned methods, these define convolutions using continuous kernels in a continuous space. However, to find neighbours in a continuous space is an expensive task requiring data structures (for example KD-tree) for representation of point clouds.

Recently, there has been an increase in neural networks without 3D convolutions for 3D perception. These methods are based on either using a 2D convolutions on observable surfaces [50, 63] or using a PointNet methods [51]. PointNet based approaches use a set of input coordinates as features for a multi-layer perceptron. However, this approach processes a limited number of points and thus a sliding window for cropping out a section is required and rather size limited.

Spatio-Temporal Segmentation [11] is a high-dimensional convolutional neural network for 4D spatio-temporal data. Compared with other approaches that combine temporal data with a recurrent neural network or a shallow model (CRF), Spatio-Temporal Segmentation [11] uses a homogeneous representation and convolutions consistently throughout the networks. It is based on Minkowski Engine, an open-source auto-differentiation library for sparse tensors and sparse convolution, proposed in [11]. As can be seen in tables 1.1, 1.2, this proposed approach outperforms all previously mentioned methods. A running demo of Spatio-Temporal Segmentation is available on GitHub[3].

1.5 Incremental Semantic Segmentation

A fundamental and challenging problem in deep learning is catastrophic forgetting, *i.e.* the tendency of neural networks to fail to preserve the knowledge acquired from old tasks when learning new tasks. The problem is even more critical when dealing with sensitive data, that cannot be collected or preserved on storage for further model trainings. This problem has been widely investigated in the research community and several Incremental Learning (IL) approaches have been proposed in the past years. While earlier works in computer vision have mostly focused on

[3] **GitLab repository**: https://gitlab.inria.fr/spring/wp2_mapping_localization/3d-semantic-segmentation-minkowskiengine

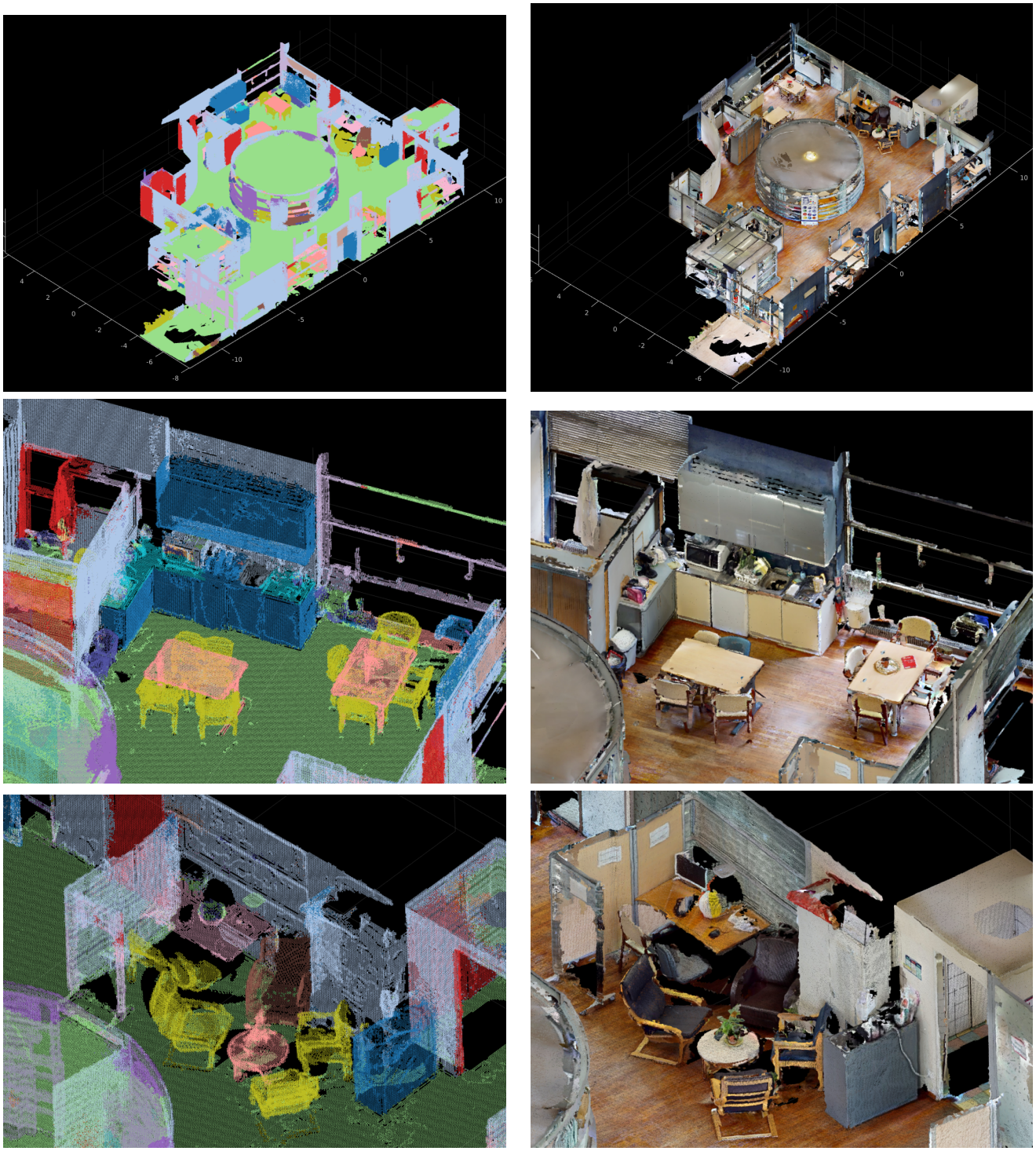


Figure 1.1: Semantic segmentation results using [11] on Broca Living Lab dataset.

image classification and object detection, more recently some IL approaches for semantic segmentation have been introduced. These previous works showed that, despite its simplicity, knowledge distillation can be effectively employed to alleviate catastrophic forgetting. In the following approach, we follow this research direction and, inspired by recent literature on contrastive learning, we propose a novel distillation framework, Uncertainty-aware Contrastive Distillation (UCD). In a nutshell, UCD is operated by introducing a novel distillation loss that takes into account all the images in a mini-batch, enforcing similarity between features associated to all the pixels from the same classes, and pulling apart those corresponding to pixels from different classes. In order to mitigate catastrophic forgetting, we contrast features of the new model with features extracted by a frozen model learned at the previous incremental step. The code of this approach is publicly available on GitLab [4]. The work has been published in the IEEE Transaction

[4] **GitLab repository**: https://gitlab.inria.fr/spring/wp4_behavior/non-integrated-contributions/UCD.git

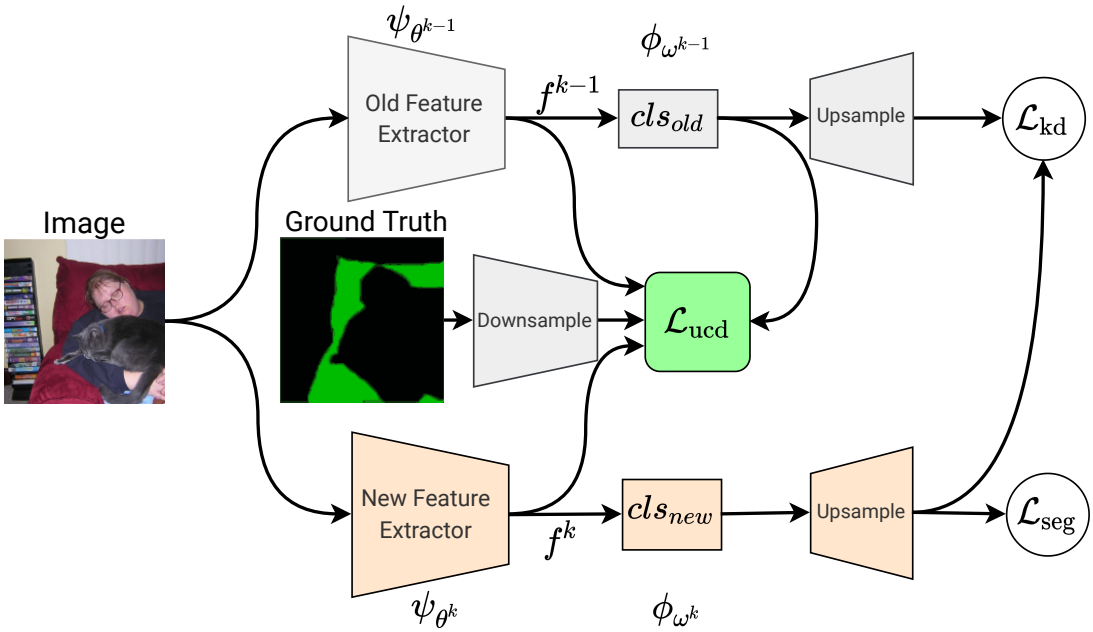


Figure 1.2: Overview of the proposed architecture at the incremental step k. The grey blocks denote the network trained on old tasks, while the light blue one indicates the network trained on old task. The proposed network is made of a feature extractor and a classifier and it is trained on three losses. Our uncertainty-aware contrastive distillation framework corresponds to the green box. Downsample and Upsample denote the resolution decrease from $[H, W]$ to $[\frac{H}{16}, \frac{W}{16}]$ and the resolution increase $[\frac{H}{16}, \frac{W}{16}]$ to $[H, W]$, respectively.

on Pattern Analysis and Machine Intelligence in 2022 [70].

1.5.1 Proposed Approach

In this section, we first provide a formalization of the problem of IL for semantic segmentation. Then, we describe the proposed Uncertainty-aware Contrastive Distillation (UCD).

Problem Definition and Notation

Let $\mathcal{I}$ denote the input image space, $I \in \mathcal{I}$ an image and $\mathcal{Y}$ the label space. In semantic segmentation, given I we want to assign a label in $\mathcal{Y}$, representing its semantic category, to each pixel of I. Pixels that are not assigned to other categories are assigned to the background class B. The problem can be addressed within a supervised framework defining a training set $\mathcal{T} = \{I_n, M_n\}_{n=1}^{N}$ of pairs of images I_n and associated segmentation maps $M_n \in \mathcal{Y}^{H \times W}$, where H and W denote the image height and width respectively.

In the general IL setting, training is realized over multiple learning steps, and this is also the case in IL for semantic segmentation. In the following k will denote the index of the incremental learning step. Each step is associated to a different training set $\mathcal{T}^k = \{I_n^k, M_n^k\}_{n=1}^{N^k}$. At each learning step k we are interested in learning a model Φ^k able to segment the classes of the label space of the k-th step, $\mathcal{Y}^k$. Importantly, the training set with N^k image-segmentation pairs is available only during the training of the corresponding incremental step. Moreover, Φ^k should be trained to mitigate catastrophic forgetting, that is to be able to also recognise the semantic classes of the previous learning steps $\cup_{k'=1}^{k-1}\mathcal{Y}^{k'}$ without having access to $\cup_{k'=1}^{k-1}(\mathcal{I}^{k'}, \mathcal{Y}^{k'})$. Following [5], we explore two settings for the splitting the datasets into different tasks: (i) the overlapped setting, which only ensures label separation across tasks $\mathcal{Y}^k \cap \mathcal{Y}^{k'} = \emptyset, \forall k' \in \{1, ..., k-1\}$; and (ii) the disjoint setting, where in addition to the label separation the same training image must not belong to more than one task, formally: $\forall I_n^k \in \mathcal{T}^k, I_n^k \notin \cup_{k'=1}^{k-1}\mathcal{T}^{k'}$.

In this work we assume that the model Φ^k is implemented by the composition of a generic backbone ψ_{θ^k} with parameters θ^k and a classifier ϕ_{ω^k} with parameters ω^k. The features extracted with the backbone are denoted by $f \in \mathbb{R}^{\frac{H}{16} \times \frac{W}{16} \times D}$, *i.e.* $f^k = \psi_{\theta^k}(I)$, meaning that f^k are extracted with the backbone at the k-th step. The output segmentation map can be computed from our pixel-wise predictor $Y_p^k = \{\arg\max(\text{upsample}(\phi_{\omega^k}(f_p^k)[p, c]))\}_{p \in I}$, where $\phi_{\omega^k}(f_p^k)[p, c]$ is the probability for class c in pixel p (which stands for a tuple (H, W) where H and W are the height and width of an image).

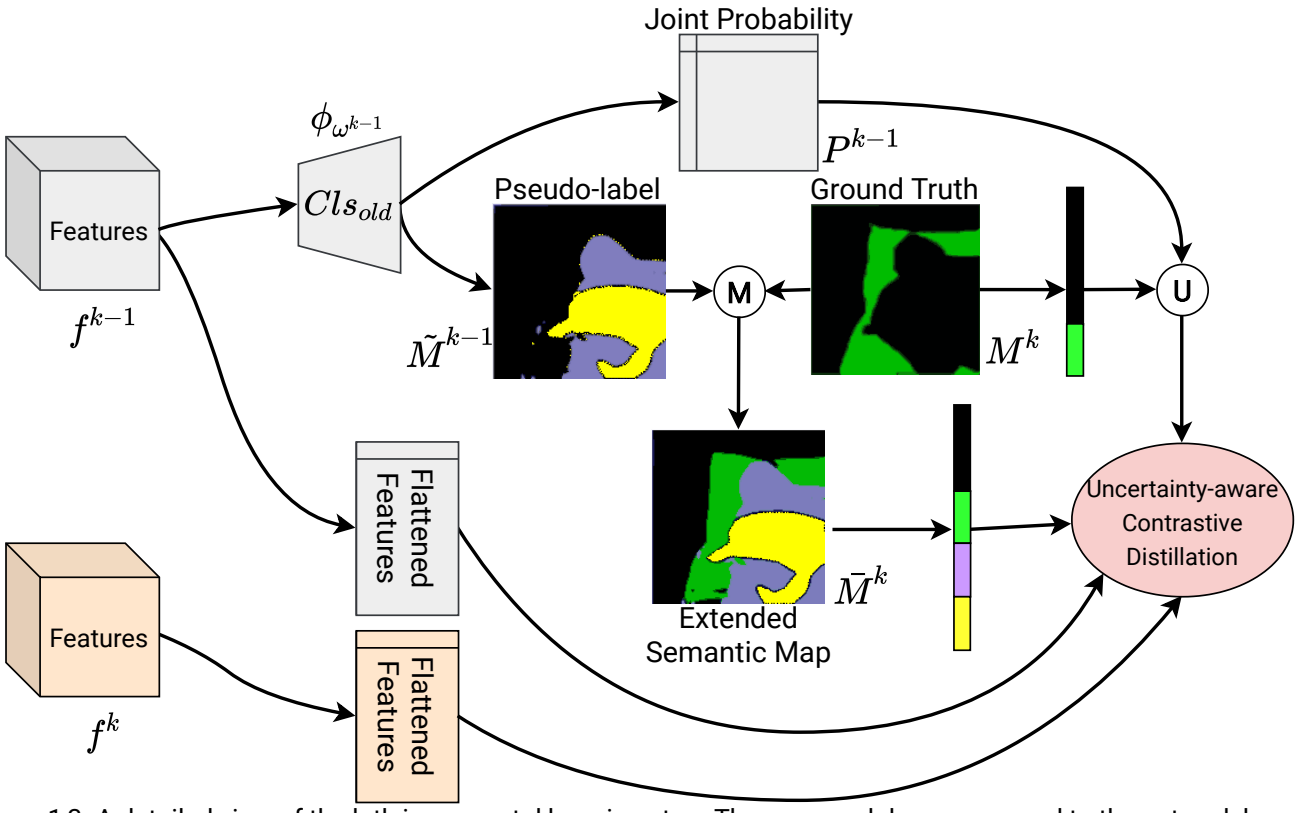


Figure 1.3: A detailed view of the k-th incremental learning step. The grey modules correspond to the network learned at the previous incremental learning step, which is frozen at step k while the blue modules correspond to the network trained at step k. Ⓤ is an uncertainty estimation operation. Ⓜ is a merger operation.

Overall Idea

Existing semantic segmentation frameworks are currently based on the use of the cross-entropy loss to align the probability distributions predicted by the new and old models in a pixel-wise fashion. However, this is suboptimal for two reasons: (i) the cross-entropy loss compares pixel-wise predictions independently and ignores relationships between pixels; (ii) due to the use of the softmax, the cross-entropy loss only depends on the relationships among logits and cannot directly supervise the learned feature representations.

A supervised contrastive framework [37] seems like a natural solution to address these problems. In fact, as recently shown in [68], by employing a contrastive loss, it is possible to explicitly model semantic relationships between pixels and learn an embedding space by pulling representations of pixel samples of the same class close and pushing those associated to different classes apart.

We propose to leverage from this idea and introduce a distillation loss that does not operate in a pixel-wise fashion but rather considers inter-pixel dependencies. In particular, we consider both relationships among representations of the new model, and between representations of the new and old models. We show that this enables better knowledge transfer and mitigates catastrophic forgetting.

The overall pipeline used in the k-th incremental learning step is illustrated in Fig. 1.2. The two streams represent the backbone for the current and previous learning steps respectively. Notably, at learning step k, the backbone associated with the old tasks $\psi_{\theta^{k-1}}$ and the classifier $\phi_{\omega^{k-1}}$ are frozen and used only during training. When a new batch of data sampled from the current task k is received, the images are forwarded into both networks, producing two sets of features and probability distributions. The old classifier outputs probabilities for old classes only, while the new classifier also predicts the classes of the current task. In order to train the new classifier we need to provide supervision on all classes. Naturally, before introducing the contrastive distillation loss we explain how to extend the semantic map using the pseudo-labels for the old classes.

Extended Semantic Map

In this model, we also make use of the pseudo-labels generated by the old model to counteract catastrophic forgetting. The pseudo-labels are obtained as follows:

$$\tilde{M}^{k-1}_{n,p} = \arg\max(\phi_{\omega^{k-1}}(f^{k-1}_{n,p})). \tag{1.1}$$

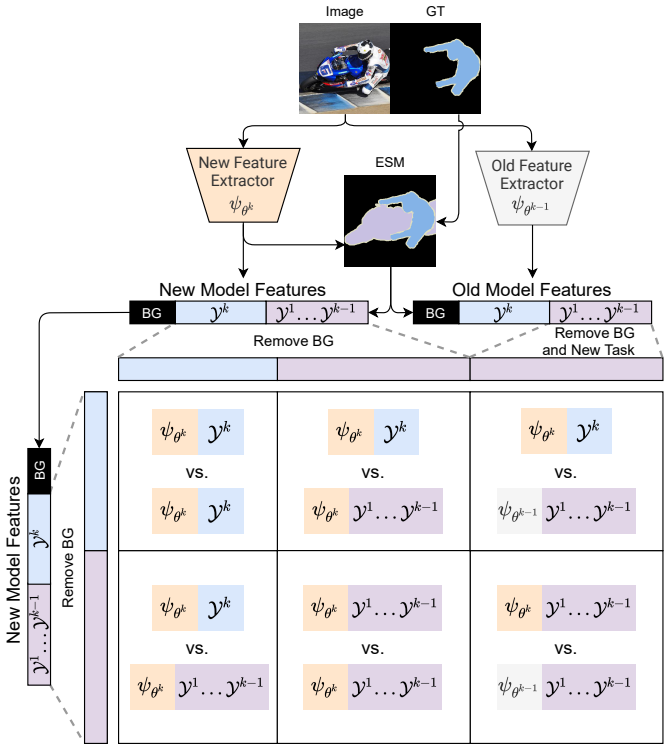


Figure 1.4: Visualization of the pairwise similarity matrix employed for efficient computation of the contrastive distillation loss. On the left side of the matrix we have the anchor features that are being compared with the contrast features on top. Blue and purple features and elements of the matrix belong to the new and old tasks respectively. Grey and orange colors represent the old and new feature extractors.

For each pixel in the image, $\tilde{M}^{k-1}$ contains the index of the most likely class according to the old model. Note that, since the old model is frozen during task k it is not able to predict the classes in $\mathcal{Y}^k$, in fact it will probably confuse them with the background. However, since we have access to the ground truth for the current task, we can use it to correct the pseudo-label. In practice, we superimpose the ground truth (excluding the background) on top of the pseudo-labels, generating $\bar{M}^k$, the Extended Semantic Map (ESM). This merger operation, denoted as Ⓜ, can be summarised by the following equation:

$$\bar{M}^k_{n,p} = \begin{cases} M^k_{n,p} & \text{if } M^k_{n,p} \neq 0, \\ \tilde{M}^{k-1}_{n,p} & \text{otherwise.} \end{cases} \tag{1.2}$$

With this simple mechanism, we are able to associate an ESM to each sample in the batch, and virtually to the whole dataset, namely: $\bar{\mathcal{T}}^k = \{I^k_n, \bar{M}^k_n\}_{n=1}^{N^k}$. Note that, since the pseudo-labels are estimated by a neural network, they might be noisy. The uncertainty associated with the ESM will be handled further in the deliverable, but first we introduce the contrastive distillation loss.

Contrastive Distillation Loss

With this extended semantic map, we can now construct a contrastive feature distillation loss in the following way. For a given training image I^k_n we can extract features with the previous and current backbones, and denote them by f^k_n and f^{k-1}_n respectively. We must now interpret these two tensors as a list of $\frac{H}{16} \times \frac{W}{16}$ features of dimension C. At this point, we need to mask out the pixels that do not contain interesting information. For f^k_n we ignore the background pixels (*i.e.* the pixels that do not belong to any of the new and old classes), since they can contain multiple objects, making their representations very heterogeneous. We do this at batch-level by collecting the following set of indices:

$$\mathcal{R}^k = \left\{(n,p) \mid \bar{M}^k_{n,p} \neq 0,\, p \in I^k_n, n \in \mathcal{B}\right\}, \tag{1.3}$$

where p represents the coordinates of a pixel in an image and $\mathcal{B}$ is the current mini-batch. Similarly, for the features f^{k-1}_n generated by the old model we need to apply an analogous masking mechanism. However, since the old model does not contain any knowledge about the new task, we need to mask out the pixels that belong to the current task, in addition to the background. As before, we gather the following indices:

$$\mathcal{S}^k = \left\{(n,p) \mid \bar{M}^k_{n,p} \notin \mathcal{Y}^k \cup \{0\}\,,\, p \in I^k_n, n \in \mathcal{B}\right\}. \tag{1.4}$$

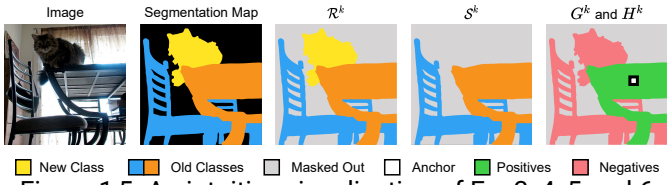


Figure 1.5: An intuitive visualization of Eq. 3, 4, 5 and 6

Apart from the reasons listed above, filtering the indices of the representations of the old and new model into $\mathcal{R}^k$ and $\mathcal{S}^k$ is also useful to reduce the computational footprint of our contrastive distillation loss. In order to mitigate catastrophic forgetting, we want to contrast the new and old representations. Nonetheless, it is also critical that new features remain consistent with themselves, especially for pixels of new classes. More precisely, we examine all possible new-new pairs of features, i.e. both features extracted from ψ_{θ^k}, and all possible old-new pairs of features, i.e. one feature extracted from ψ_{θ^k} and one from $\psi_{\theta^{k-1}}$ (subject to their membership to $\mathcal{R}^k$ and $\mathcal{S}^k$). To make them easier to grasp, we show an intuitive visualization of $\mathcal{R}^k$ and $\mathcal{S}^k$ in Fig. 1.5. Moreover, for each anchor feature we select, we need to construct a set of positives and a set of negatives in order to perform the contrast. Given the index a of an anchor, we select as positives all the pixels that belong to the same class:

$$\begin{aligned} G^k_{\mathcal{R}}(a) &= \left\{ f^k_r \mid \bar{M}^k_a = \bar{M}^k_r, r \in \mathcal{R}^k \setminus \{a\} \right\}, \\ G^k_{\mathcal{S}}(a) &= \left\{ f^{k-1}_r \mid \bar{M}^k_a = \bar{M}^k_r, r \in \mathcal{S}^k \right\}, \\ G^k(a) &= G^k_{\mathcal{R}}(a) \cup G^k_{\mathcal{S}}(a), \end{aligned} \tag{1.5}$$

where $G^k_{\mathcal{R}}(a)$ and $G^k_{\mathcal{S}}(a)$ are the positives mined from the new and old features respectively, and $G^k(a)$ is the full set of positives for f_a. Similarly for the negatives:

$$\begin{aligned} H^k_{\mathcal{R}}(a) &= \left\{ f^k_r \mid \bar{M}^k_a \neq \bar{M}^k_r, r \in \mathcal{R}^k \right\}, \\ H^k_{\mathcal{S}}(a) &= \left\{ f^{k-1}_r \mid \bar{M}^k_a \neq \bar{M}^k_r, r \in \mathcal{S}^k \right\}, \\ H^k(a) &= H^k_{\mathcal{R}}(a) \cup H^k_{\mathcal{S}}(a) \end{aligned} \tag{1.6}$$

The positives $G^k(a)$ and negatives $H^k(a)$ are shown in Fig. 1.5 to highlight the relationship among anchor, positives and negatives. Note that in this second case we are selecting all the pixels that do not share the same class with a (negatives). We can now write the contrastive distillation loss as follows:

$$\mathcal{L}_{\text{cd}} = \frac{1}{|\mathcal{R}^k|} \sum_{a \in \mathcal{R}^k} \frac{-1}{|G^k(a)|} \sum_{f^+ \in G^k(a)} \mathcal{L}_{\text{lc}} \left(f^k_a, f^+, a \right), \tag{1.7}$$

where $\mathcal{L}_{\text{lc}}$ is the log-contrast loss involving an anchor feature f^k_a generated by the new model, a positive feature f_g and the set of negatives for the anchor, namely:

$$\mathcal{L}_{\text{lc}} = \log \frac{\exp\left(\delta(f^k_a, f^+)/\tau\right)}{\sum_{f^- \in H^k(a)} \exp\left(\delta(f^k_a, f^-)/\tau\right)}, \tag{1.8}$$

where δ denotes the cosine similarity $\delta\left(u, v\right) = \frac{u}{||u||} \cdot \frac{v}{||v||}$ and τ is the temperature of the contrast. In practice to compute the contrastive distillation loss efficiently using modern deep learning libraries we compute a matrix multiplication over the whole batch to obtain the pairwise similarity matrix (see Fig. 1.4). Each element in the matrix contains the cosine similarity between two pixels. Then we sum over the rows using masks for positives and negatives to obtain the loss value.

Overall, $\mathcal{L}_{\text{cd}}$ pushes the features of all same-class pixels together, while pulling the pixels with different class apart. Very importantly, the proposed loss considers all features within the same batch as potential pairs, thus adding many more positive pairs, and regularising *de facto* the learning. We recall that the feature extractor from the previous learning step is frozen. Therefore, features f^{k-1} are also frozen and used to attract the new features f^k towards a representation that encodes the classes of the previous learning step, hence the name *contrastive distillation loss*.

Uncertainty-aware Contrastive Distillation

As discussed in the previous section, the contrastive distillation loss requires to have pseudo-labels of the semantic classes of the previous incremental learning steps. In order to achieve this, we exploit the semantic classifier learned

at the previous step, that outputs a per-class probability tensor $P^{k-1} \in [0,1]^{\frac{H}{16} \times \frac{W}{16} \times T^{k-1}}$, $\sum_{l=1}^{T^{k-1}} P^{k-1}_{p,l} = 1, \forall pcx$ (recall that T^{k-1} is the total number of classes until step $k-1$). This probability map is extended with the ground-truth at learning step k in the following way:

$$\bar{P}^k_p = \begin{cases} [P^{k-1}_p, 0_{L^k}] & \text{if } M^k_p = 0, \\ \mathbb{I}_{M^k_p} & \text{otherwise,} \end{cases} \tag{1.9}$$

where, 0_d is the zero vector of dimension d and $\mathbb{I}_c$ is a one-hot vector at coordinate c. In other words, the new probability tensor copies from the previous one when the ground-truth points to the background, and is a one-hot vector with the ground-truth class otherwise.

Therefore, the extended probability tensor is $\bar{P}^k \in [0,1]^{H \times W \times T^k}$. By arg-maxing this tensor over the classes, we obtain the pseudo-labels in the extended semantic map $\bar{M}^K_n$. The direct application of these pseudo-labels in the contrastive distillation loss discussed in Eq. (1.7). Our intuition is that the arg-max operation takes a hard-decision that does not account for the uncertainty. We have therefore developed a principled strategy to account for the uncertainty in the classification of each pixel using the previous backbone.

The extended probability map can used be also to produce an estimate of the uncertainty of the pseudo-label. Indeed, one can compute the probability of two pixels a (anchor) and g (positive) belonging to the same class at incremental step k using:

$$\sigma^k_{a,g} = \bar{P}^k_a \cdot \bar{P}^k_g. \tag{1.10}$$

This formulation naturally takes into account two important aspects that are particularly relevant for contrastive learning: (i) the *confidence* of the network in predicting a certain class; (ii) the *alignment* of the two probability distributions. if the network is not confident (high entropy) or the alignment is poor then $\sigma^k_{a,g}$ is low, reducing the strength of the attraction. This mechanism intrinsically accounts for the uncertainty, reducing the noise and therefore leading to better performance.

Finally, we can use this probability to modify $\mathcal{L}_{\text{cd}}$ in (1.7) into $\mathcal{L}_{\text{ucd}}$ – the *uncertainty-aware contrastive distillation* – defined as:

$$\mathcal{L}_{\text{ucd}} = \frac{1}{|\mathcal{R}^k|} \sum_{a \in \mathcal{R}^k} \frac{-1}{|G(a)|} \sum_{f^+_g \in G(a)} \sigma^k_{a,g} \mathcal{L}_{\text{lc}}, \tag{1.11}$$

where g is the pixel index associated to f^+_g.

We also summary all step of in Fig 1.3 and explain more detail about contrastive distillation with uncertainty (red block in Fig 1.3).

UCD as a generic framework

The proposed uncertainty-aware contrastive distillation framework can be integrated seamlessly on top of virtually any other baseline. In the following we describe how to integrate UCD with two state of the art methods.

MiB-UCD. Recently, Cermelli *et al.* [5] proposed Modeling the Background for Incremental Learning (MiB), a distillation-based approach to learn the predictor ϕ_{θ^k} at step k given a training set $\mathcal{T}^k$ by distilling knowledge using the predictions of model $\phi_{\theta^{k-1}}$ on the previous step. In particular, they propose to minimize the following loss function:

$$\mathcal{L} = \frac{1}{|\mathcal{T}^k|} \sum_{(I^k_n, M^k_n) \in \mathcal{T}^k} \left(\mathcal{L}_{\text{seg}}(I^k_n, M^k_n) + \lambda \mathcal{L}_{\text{kd}}(I^k_n) \right) \tag{1.12}$$

where $\lambda > 0$ is a hyperparameter balancing the importance of two loss terms $\mathcal{L}_{\text{seg}}(x,y)$ and $\mathcal{L}_{KD}(x)$. These are defined in order to specifically take into account the role of the background class B, which contains different semantic categories over subsequent time steps. In particular, the loss $\mathcal{L}_{\text{seg}}$ is a revisited cross-entropy loss defined as follows:

$$\mathcal{L}_{\text{seg}}(I^k_n, M^k_{n,p}) = -\frac{1}{N} \sum_{p \in I} \log \tilde{\phi}_{\omega^k}(f^k_n)[p, M^k_{n,p}], \tag{1.13}$$

where $f^k_n = \psi_{\theta^k}(I^k_n)$ and:

$$\tilde{\phi}_{\omega^k}(f^k_n)[p,c] = \begin{cases} \phi_{\omega^k}(f^k_n)[p,c] & \text{if } c \neq 0, \\ \sum_{q \in \mathcal{Y}^{t-1}} \phi_{\omega^k}(f^k_n)[p,q] & \text{if } c = 0. \end{cases} \tag{1.14}$$

This loss is meant to compensate the fact that the training set $\mathcal{T}^k$ might include also pixels associated to the categories previously observed in the old task.

Furthermore, a distillation loss [34] is introduced in order to encourage ϕ_{ω^k} to produce probabilities close to the ones produced by $\phi_{\omega^{k-1}}$. The distillation loss writes:

$$\mathcal{L}_{\text{kd}}(I_n^k) = -\frac{1}{|I|}\sum_{i\in I}\sum_{c\in\mathcal{Y}^{k-1}} \phi_{\omega^{k-1}}(f_n^{k-1})[i,c]\log\hat{\phi}_{\omega^k}(f_n^k)[i,c]\,, \tag{1.15}$$

where:

$$\hat{\phi}_{\omega^k}(f_n^k)[p,c] = \begin{cases}\phi_{\omega^k}(x)[p,c] & \text{if } c \neq \texttt{B}\\ \sum_{q\in\mathcal{Y}^k}\phi_{\omega^k}(f_n^k)[p,q] & \text{if } c = \texttt{B}\,.\end{cases} \tag{1.16}$$

This loss is intended to account for the fact that $\phi_{\theta^{k-1}}$ might predict as background pixels of classes that we are currently trying to learn. This aspect is crucial to perform a correct distillation of the old model into the new one.

Nonetheless, distilling the output probabilities might not be sufficient in order to defy catastrophic forgetting. For this reason, MiB seems to be a natural candidate for enhancement using UCD. The overall loss MiB-UCD is:

$$\mathcal{L} = \mathcal{L}_{\text{seg}} + \lambda_{\text{kd}}\mathcal{L}_{\text{kd}} + \lambda_{\text{ucd}}\mathcal{L}_{\text{ucd}}, \tag{1.17}$$

PLOP-UCD. A different approach for incremental learning in semantic segmentation was explored in [18]. While MiB [5] performs distillation on output probabilities, Pseudo-label and LOcal POD (PLOP) [18] proposes to use intermediate representations to transfer richer information. This is achieved using a technique called Pooled Output Distillation (POD) [19] that was already shown to work well in classification settings. In addition, in order to preserve details that would be neglected by the pooling, PLOP uses a multi-scale distillation scheme. In practice, for every layer l of the network, pooled features (Local POD embeddings) are collected at several scales and concatenated to obtain a feature vector $\mathcal{F}_l$. This process is carried out on both old and current models, and then the concatenated features are compared using an $_2$ loss:

$$\mathcal{L}_{\text{pod}} = \frac{1}{L}\sum_{l=1}^{L}\left\|\mathcal{F}_l^k - \mathcal{F}_l^{k-1}\right\|^2, \tag{1.18}$$

where L is the total number of layers of the network.

Besides Local POD, PLOP also introduces a simpler way of solving background shift: a pseudo-labeling strategy for background pixels. Predictions of the old model for background pixels are used as clues regarding their real class, if they belong to any of the old classes. An argmax operation is applied on every pixel, subject to a class-specific confidence threshold. In other words, in the case of non-background pixels they copy the ground truth label; otherwise, if the old model is confident enough, the most likely class predicted by the old model is selected. The pseudo-label S is then used in the classification loss as follows:

$$\mathcal{L}_{\text{pseudo}} = -\frac{\nu}{|I|}\sum_{p\in I}\log\phi_{\omega^k}(f_n^k)[p,S_{n,p}]\,, \tag{1.19}$$

where ν is the ratio of accepted old classes pixels over the total number of such pixels.

Combining these two ideas, PLOP was shown to be able to outperform MiB in the overlapped setting, while its superiority on the harder disjoint setting is still unclear. Nonetheless, we believe PLOP can take advantage of our contrastive distillation loss. In fact, although it already performs distilation of intermediate features, PLOP lacks two critical aspects: (i) it does not take into account the relationships between pixel representations for different images (ii) it does not use the pseudo-label for feature distillation. Also, the way PLOP estimates the confidence of the pseudo-label is very primitive and cursed with class-specific hyper-parameter tuning. On the contrary, our UCD loss naturally accounts for the uncertainty in a more elegant way. The overall loss of PLOP-UCD is the following:

$$\mathcal{L} = \mathcal{L}_{\text{pseudo}} + \lambda_{\text{pod}}\mathcal{L}_{\text{pod}} + \lambda_{\text{ucd}}\mathcal{L}_{\text{ucd}}, \tag{1.20}$$

1.5.2 Experiments

In this section we give a snapshot of the results of UCD on PASCAL-VOC 20212 dataset against state of the art approaches. Our experimental results are then shown and illustrated. Additional experiments are available in the original paper [70].

Table 1.3: Mean IoU on the Pascal-VOC 2012 dataset for different incremental class learning scenarios. $*$ means results come from re-implementation. † means a updated version. ♯ means the conditional GAN model pretrained on ImageNet [16] is used. UDC means uncertainty-aware contrastive distillation. Best among table in **bold**, best among part in underlined.

Method	19-1						15-5						15-1					
	Disjoint			Overlapped			Disjoint			Overlapped			Disjoint			Overlapped		
	1-19	20	all	1-19	20	all	1-15	16-20	all	1-15	16-20	all	1-15	16-20	all	1-15	16-20	all
FT	5.8	12.3	6.2	6.8	12.9	7.1	1.1	33.6	9.2	2.1	33.1	9.8	0.2	1.8	0.6	0.2	1.8	0.6
PI[77]	5.4	14.1	5.9	7.5	14.0	7.8	1.3	34.1	9.5	1.6	33.3	9.5	0.0	1.8	0.4	0.0	1.8	0.4
EWC[38]	23.2	16.0	22.9	26.9	14.0	26.3	26.7	37.7	29.4	24.3	35.5	27.1	0.3	4.3	1.3	0.3	4.3	1.3
RW[6]	19.4	15.7	19.2	23.3	14.2	22.9	17.9	36.9	22.7	16.6	34.9	21.2	0.2	5.4	1.5	0.0	5.2	1.3
LwF[43]	53.0	9.1	50.8	51.2	8.5	49.1	58.4	37.4	53.1	58.9	36.6	53.3	0.8	3.6	1.5	1.0	3.9	1.8
LwF-MC[53]	63.0	13.2	60.5	64.4	13.3	61.9	67.2	41.2	60.7	58.1	35.0	52.3	4.5	7.0	5.2	6.4	8.4	6.9
ILT[46]	69.1	16.4	66.4	67.1	12.3	64.4	63.2	39.5	57.3	66.3	40.6	59.9	3.7	5.7	4.2	4.9	7.8	5.7
ILT†[48]	69.4	16.5	66.7	67.4	12.4	64.7	63.3	39.6	57.4	66.4	40.8	60.0	4.4	6.4	4.9	5.5	8.0	6.1
SDR[47]	69.9	37.3	68.4	69.1	32.6	67.4	73.5	47.3	67.2	75.4	52.6	69.9	59.2	12.9	48.1	44.7	21.8	39.2
RECALL♯[45]	65.2	**50.1**	65.8	67.9	**53.5**	68.4	66.3	**49.8**	63.5	66.6	50.9	64.0	**66.0**	**44.9**	**62.1**	65.7	**47.8**	**62.7**
UCD	73.4	33.7	71.5	71.4	47.3	70.0	71.9	49.5	66.2	77.5	**53.1**	71.3	53.1	13.0	42.9	49.0	19.5	41.9
MiB[5]	69.6	25.6	67.4	70.2	22.1	67.8	71.8	43.3	64.7	75.5	49.4	69.0	46.2	12.9	37.9	35.1	13.5	29.7
MiB+SDR[47]	70.8	31.4	68.9	71.3	23.4	69.0	**74.6**	44.1	**67.3**	76.3	50.2	70.1	59.4	14.3	48.7	47.3	14.7	39.5
MiB+UCD	74.3	28.4	72.0	73.7	34.0	71.7	73.0	**46.2**	66.3	**78.5**	50.7	**71.5**	53.3	14.4	43.5	51.9	13.1	42.2
PLOP* [18]	75.1	38.2	73.2	75.0	39.1	73.2	66.5	39.6	59.8	74.7	49.8	68.5	49.0	13.8	40.2	65.2	22.4	54.5
PLOP+ UCD	**75.7**	31.8	**73.5**	**75.9**	39.5	**74.0**	67.0	39.3	60.1	75.0	51.8	69.2	50.8	13.3	41.4	**66.3**	21.6	55.1
Joint	77.4	78.0	77.4	77.4	78.0	77.4	79.1	72.6	77.4	79.1	72.6	77.4	79.1	72.6	77.4	79.1	72.6	77.4

PASCAL-VOC 2012

The Pascal VOC2012 dataset [22] contains 11,530 training/validation images of a variable size and with semantic labels correspondint to 20 classes plus background. Following previous works [46, 57, 5, 62], we propose two different experimental settings: *disjoint* and *overlapped*. In the first setting [46], each learning step contains a unique set of images, whose pixels belong to classes seen either in the current or in the previous learning steps. In other words, each image is seen in only one learning step. On the contrary, in the *overlapped* setting [57], a few classes appear on both tasks. Additionally, results are proposed in multiple experiments, using different class-incremental procedures: the first adds only one class during the new task (19-1), the second adds 5 classes at once (15-5) and the third adds 5 classes one at the time averaging the final results (15-1). Results are reported in terms of mIoU.

Network Details

The proposed method is implemented in PyTorch. The experiments are conducted on two Nvidia RTX 6000 GPUs during training while one for testing. The feature extractor used in this work includes a backbone network and a decoder. In detail, the ResNet-101 [32] is chosen as backbone while features are decoded using the Deeplab-v3 architecture [8]. The weights are initialized on ImageNet [55] and trained using momentum and weight decay proposed in [8]. The initial learning rate is set to 10^{-2} for the first learning step, then it is diminished for the following steps to 10^{-3}. We train the model with a batch-size of 24 for 30 epochs for Pascal-VOC 2012. The input image is cropped to meet the input dimension of 512×512 and transformed using the augmentation protocol proposed in [8]. The temperature value of the contrastive loss has been set to 0.07, while the weighting parameters λ_{ucd}, $\lambda_{pod}$and λ_{kd} are set to 0.01, 0.01 and 10 respectively.

Experimental Results

Results are presented in this section in terms overall accuracy against the current state of the art (i.e. Table 1.3) and with qualitative examples in Figure 1.6.

Comparison with state of the art methods

Comparison on VOC 2012. Table 1.3 shows the performance of UCD with respect to competing method for class-incremental learning including PI [77], EWC [38], RW [6], LwF [43], LwF-MC [53] and works specially designed for image segmentation such as ILT [46], MiB [5], PLOP [18] and SDR [47]. As mentioned earlier, fine tuning and joint training are also reported as a lower and upper bound reference.

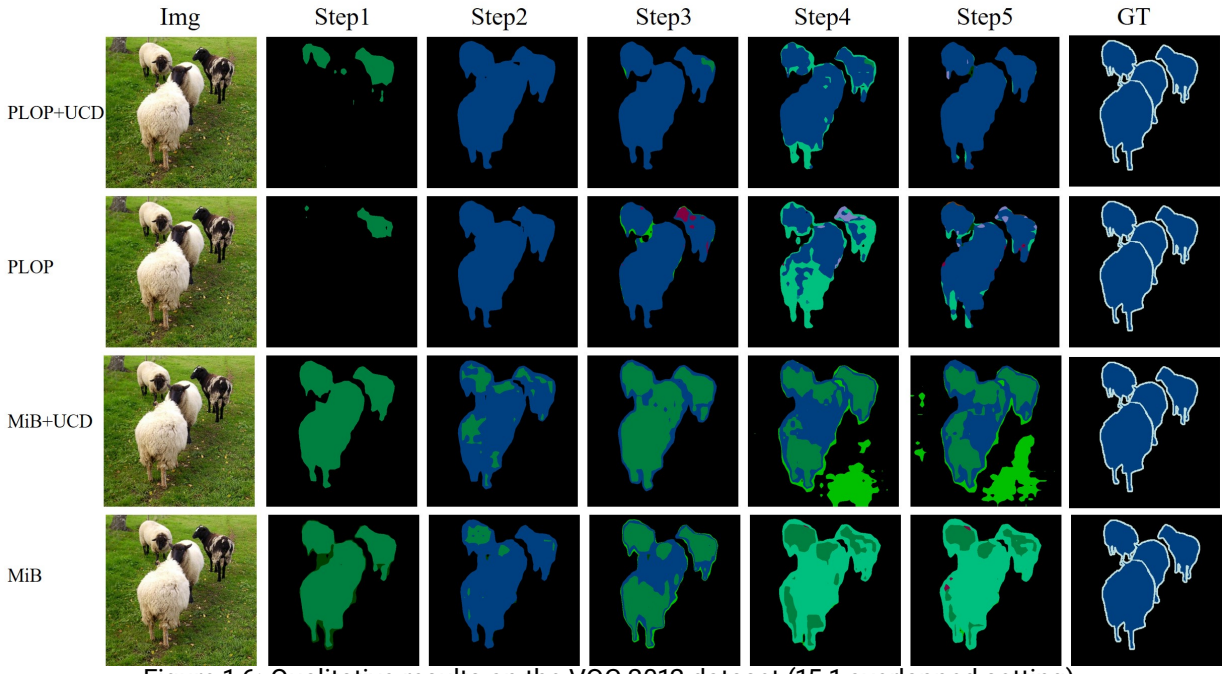


Figure 1.6: Qualitative results on the VOC 2012 dataset (15-1 overlapped setting)

Specifically, PLOP has been modified in order to not consider the background class in the accuracy computation for sake of fairness in the results comparison with the competing works. The training model proposed in this work (i.e. UCD) can be applied on top of other methods therefore Table 1.3 shows results of the best performing incremental learning algorithms for semantic segmentation evaluated using the UCD procedure. Results show that UCD generally improves the performances of other standalone models (i.e. MiB and PLOP), in both *overlapped* and *disjoint* settings. However, UCD has a weaker performance than MiB+SDR [47],mainly in disjoint setup of VOC 2012. The first reason is that SDR uses attraction and repulsion losses with prototypes (calculated with a running average), which is fundamentally different from the contrastive loss. Averaging features into prototypes changes balancing between the classes in the loss. Probably, this makes results of old classes perform better because old classes already account for the majority of the total classes. Another reason is that in some cases it is difficult for the network to generate good extended semantic maps in the disjoint setting. This probably affects UCD more than SDR, because SDR counteracts this noise by using running averages and has additional regularization mechanisms. Another can notice that neither of the two methods mentioned above is outperforming the other in all settings, while adding UCD is generally beneficial. Qualitative results are shown in Figure 1.6 for 15-1 scenario [5]. Note that classes such as human, bike, and cow are included in the old task but are still well represented in the resulting segmentation map when UCD is involved. However, the improvements brought by UCD is larger for MiB than PLOP, even in thecases where MiB was better than PLOP. One reason is, in most of the settings, PLOP already achieves very good results, outperforming MiB by a large margins. This makes it hard for UCD to produce an extra improvement over PLOP. In addition, it seems PLOP is not well suited for the disjoint setting and therefore makes UCD less effective as well. Finally, since PLOP already has a feature distillation mechanism, it might be that the two feature distillation losses take care of similar aspects of knowledge transfer.

1.6 Depth Estimation

The capability of estimating depth facilitates a number of downstream tasks to perform better. The absence of a dedicated and forward-looking depth sensor on the robot suggested us to propose a model that is able to infer a depth map from a single monocular image. The goal of this module is therefore to estimate the pixel-wise depth map starting from a simple RGB image. Due to their remarkable inference capability for semantic segmentation tasks we decided to adopt a Transformer architecture [66] as a backbone for our model. Initially designed for natural language processing tasks, Transformers have emerged as alternative architectures with innate global self-attention mechanisms to capture long-range dependencies. For the depth estimation module, we propose TransDepth, an architecture that benefits from both convolutional neural networks and transformers. To avoid the network losing its ability to capture

[5] More results are available on the original paper [70]

local-level details due to the adoption of transformers, we propose a novel decoder that employs attention mechanisms based on gates. Notably, this work has been published as the first paper that applies transformers to pixel-wise prediction problems involving continuous labels (*i.e.*, monocular depth prediction and surface normal estimation). The code of this approach is publicly available on GitLab [6]. The method has been recently published at the International Conference of Computer Vision 2021 (i.e. [71]).

1.6.1 The Proposed TransDepth

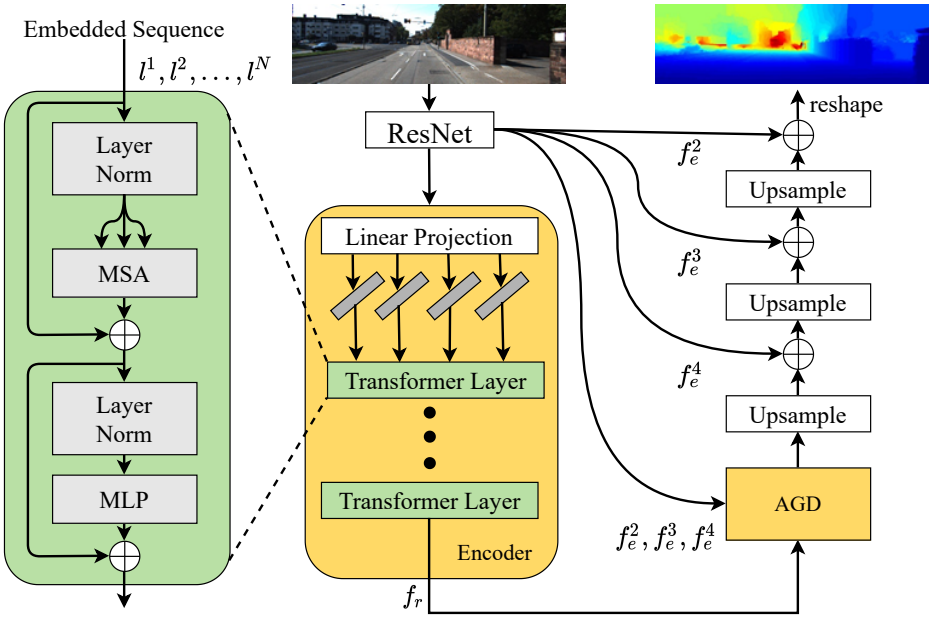


Figure 1.7: The overview of the proposed TransDepth. The symbols © and $\oplus$ denote concatenation and addition operations, respectively. AG is short for attention gate.

Our work aims to solve limited receptive fields by adding Transformer layers and enhancing the learned representation by an attention gate decoder.

Transformer for Depth Prediction

An overview of the network is depicted in Figure 1.7. Unlike the previous works [79, 7, 17] reshaping the image $I \in \mathbb{R}^{H \times W \times 3}$ into a sequence of flattened 2D patches $I_p \in \mathbb{R}^{N \times (p^2 \cdot 3)}$, we propose a hybrid model. As shown in Figure 1.7, the input sequence comes from a ResNet backbone [32]. Then the patch embedding is applied to patches extracted from the final feature output of a CNN. This patch embedding's kernel size should be $p \times p$, which means that the input sequence is obtained by simply flattening the spatial dimensions of the feature map and projecting to the Transformer dimension. In this case, we also remove position embedding because the original physical meaning is missing while mapping the vectorized patches I_p into a latent embedding space l_p using a linear projection. The input of the first Transformer layer is calculated as follow:

$$z_0 = [l^1 E; l^2 E; \cdots; l^N E], \tag{1.21}$$

where z_0 is mapped into a latent N-dimensional embedding space using a trainable linear projection layer and E is the patch embedding projection. There are L Transformer layers which consist of multi-headed self-attention (MSA) and multi-layer perceptron (MLP) blocks. At each layer ℓ, the input of the self-attention block is a triplet of Q (query), K (key), and V (value), similar with [66], computed from $z_{\ell-1} \in \mathbb{R}^{L \times C}$ as:

$$Q = z_{\ell-1} \times W_Q, K = z_{\ell-1} \times W_K, V = z_{\ell-1} \times W_V, \tag{1.22}$$

where $W_Q, W_K, W_V \in \mathbb{R}^{C \times d}$ are the learnable parameters of weight matrices and d is the dimension of Q, K, V. The self-attention is calculated as:

$$\mathrm{AH} = \mathrm{softmax}(\frac{Q \times K^T}{\sqrt{d}}) \cdot V, \tag{1.23}$$

[6] **GitLab repository**: https://gitlab.inria.fr/spring/wp4_behavior/non-integrated-contributions/TransDepth.git

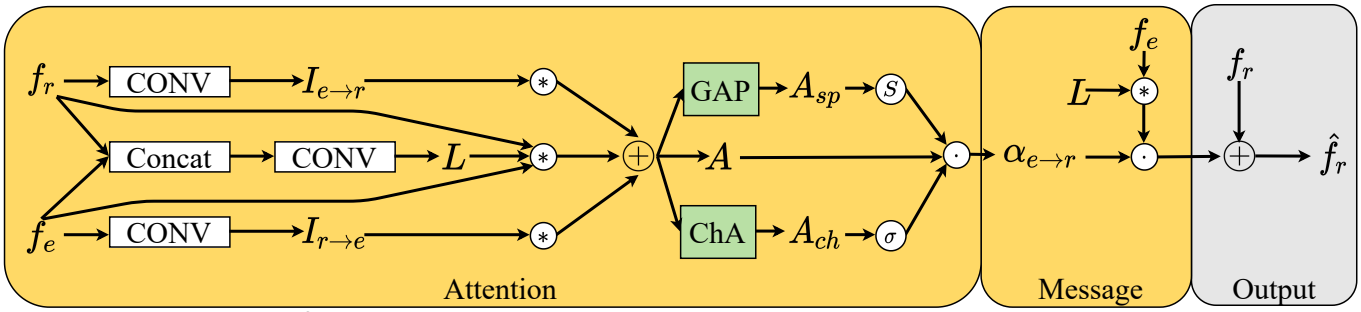


Figure 1.8: The overview of the proposed attention gate module. The symbols $\odot$, $\oplus$, σ, $*$, and s denote element-wise multiplication, element-wise addition, sigmoid, convolution, and softmax operation, respectively.

where AH is short for attention head and d is the dimension of self-attention block. MSA means the attention head will be calculated m times by independent weight matrices. The final $\mathrm{MSA}(z_{\ell-1})$ is defined as:

$$\mathrm{MSA}(z_{\ell-1}) = z_{\ell-1} + \mathrm{concat}(\mathrm{AH}_1; \mathrm{AH}_2; \cdots; \mathrm{AH}_m) \times W_o, \tag{1.24}$$

where $W_o \in \mathbb{R}^{md\times C}$. The output of MSA is then transformed by a MLP block with residual skip as the layer output as:

$$z_\ell = \mathrm{MLP}(\mathrm{LN}(z'_\ell)) + z'_\ell, \tag{1.25}$$

where $\mathrm{LN}(\cdot)$ means the layer normalization operator and $z'_\ell = \mathrm{MSA}(z_{\ell-1})$. The structure of a Transformer layer is illustrated in the left part of Figure 1.7. After the Transformer layer, the output will be recovered to the original feature shape.

Attention Gate Decoder

Given an input image $\mathbf{I}$ and a generic front-end CNN model, we consider a set of S multi-scale feature maps $\mathbf{F}=\{f^i\}_{i=1}^N$. Being a generic framework, these feature maps can be the output of S intermediate CNN layers or of another representation, thus s is a *virtual* scale. Opposite to previous works adopting simple concatenation or weighted averaging schemes [79], we propose to combine the multi-scale feature maps by learning a set of latent kernels ($I_{r\to e}$, $I_{e\to r}$, L) with a novel structure Attention-Gated module sketched in Figure 1.8. We choose f^N as a receive feature only, f_r, while $\{f^i\}_{i=1}^{N-1}$ are chosen as emitting features, f_e, in all tasks. The influence of the fusion of different scales is explained in the ablation part.

In detail, the whole attention gate can be divided into two parts, i.e., attention and message. We propose to bring together recent advances in pixel-wise prediction by formulating a novel attention gate mechanism for the attention part. Inspired by [24], where two spatial- and channel-wise predictions are computed, we opt to infer different spatial and channel attention variables. Our attention tensor can be defined by:

$$\begin{aligned} A^i_{sp} &= \frac{1}{C}\sum_{c=1}^{C}(\omega_{sp} * A^i)[c,h,w], \\ A^i_{ch} &= \frac{1}{HW}\sum_{h,w=1}^{H,W}(\omega_{sp} * A^i)[c,h,w], \\ \alpha^i_{e\to r} &= \mathrm{softmax}(A^i_{sp}) \cdot \sigma(A^i_{ch}) \cdot A^i, \end{aligned} \tag{1.26}$$

where i means f^i is chosen as an emitting feature. Different from [24], we adapt a local conditional kernel before generating attention. The kernels $I_{r\to e}$, $I_{e\to r}$, and L are predicted from the input features using a linear transformation as follows:

$$\begin{aligned} \mathbf{L}^{i,j} &= \mathbf{W}_L{}^{i,j}\mathrm{concat}(f^i_e, f^j_r) + \mathbf{b}_L{}^{i,j}, \\ \mathbf{I}^{i,j}_{r\to e} &= \mathbf{W}_{I^{i,j}_{r\to e}} f^i_e + \mathbf{b}_{I^{i,j}_{r\to e}}, \\ \mathbf{I}^{i,j}_{e\to r} &= \mathbf{W}_{I^{i,j}_{e\to r}} f^j_r + \mathbf{b}_{I^{i,j}_{e\to r}}. \end{aligned} \tag{1.27}$$

Then, the integrated attention is defined as follow:

$$A^i = \mathbf{I}^i_{e\to r} * f_r + \mathbf{I}^i_{r\to e} * f^i_e + f_r * L * f^i_e. \tag{1.28}$$

Compared with the attention part, the message is easy to be calculated by $L^i * f_r$. Finally, the output of our attention gate decoder is:

$$\hat{f}^i_e = \mathrm{concat}(L^1 * f^1_e \cdot \alpha^1_{e\to r} + f_r, ..., L^{N-1} * f^{N-1}_e \cdot \alpha^{N-1}_{e\to r} + f_r). \tag{1.29}$$

Table 1.4: Depth Estimation: KITTI dataset. K: KITTI. CS: CityScapes [13]. CS→K: CS pre-training. D: Depth supervision. M, Se, V, S: Monocular, segmentation, video, stereo. Sup: supervise.

Method	Sup	Data	Error (lower is better)				Accuracy (higher is better)		
			abs rel	sq rel	rms	log rms	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
CC [52]	M+Se	K	0.140	1.070	5.326	0.217	0.826	0.941	0.975
Bian [3]	M+V	K+CS	0.137	1.089	5.439	0.217	0.830	0.942	0.975
DeFeat [59]	M	K	0.126	0.925	5.035	0.200	0.862	0.954	0.980
S^3Net [9]	M+Se	K	0.124	0.826	4.981	0.200	0.846	0.955	0.982
Monodepth2 [25]	M	K	0.115	0.903	4.863	0.193	0.877	0.959	0.981
pRGBD [65]	M	K	0.113	0.793	4.655	0.188	0.874	0.960	0.983
Johnston [36]	M	K	0.106	0.861	4.699	0.185	0.889	0.962	0.982
SGDepth [39]	M+Se	K+CS	0.107	0.768	4.468	0.180	0.891	0.963	0.982
Shu [58]	M	K	0.104	0.729	4.481	0.179	0.893	0.965	0.984
DORN [23]	D	K	0.072	0.307	2.727	0.120	0.932	0.984	0.994
Yin [75]	M	K	0.072	-	3.258	0.117	0.938	0.990	0.998
PackNet [30]	V	K+CS	0.071	0.359	3.153	0.109	0.944	0.990	0.997
FAL-Net [26]	S	K+CS	0.068	0.276	2.906	0.106	0.944	0.991	0.998
PGA-Net [69]	D	K	0.063	0.267	**2.634**	0.101	0.952	0.992	0.998
BTS [41]	M	K	**0.061**	0.261	2.834	0.099	0.954	0.992	0.998
Baseline	M	K	0.106	0.753	3.981	0.104	0.888	0.967	0.986
Ours w/ AGD	M	K	0.065	0.261	2.766	0.101	0.953	0.993	0.998
Ours w/ ViT	M	K	0.064	0.258	2.761	0.099	0.955	0.993	0.999
Ours w/ AGD+ViT (Full)	M	K	0.064	**0.252**	2.755	**0.098**	**0.956**	**0.994**	**0.999**

Once the hidden variables are updated, we use them to address several different discrete prediction tasks, including monocular depth estimation and surface normal estimation. Following previous works, the network optimization loss for depth prediction, updated from [20], is:

$$\mathcal{L}_{depth} = \alpha \sqrt{\frac{1}{T}\sum_i g_i^2 - \frac{\lambda}{T^2}(\sum_i g_i)^2}, \tag{1.30}$$

where $g_i = \log \hat{d}_i - \log d_i$ with the ground truth depth d_i and the predicted depth $\hat{d}_i$. We set λ and α to 0.85 and 10, same with [41]. The angular loss is chosen as the surface normal loss.

1.6.2 Results on Monocular Depth Estimation

We compare the proposed method with the leading monocular depth estimation models, i.e., [52, 3, 59, 9, 25, 65, 36, 39, 58, 30, 23, 75, 26, 41]. Comparison results on the KITTI dataset are shown in Table 1.4. Our method performs favorably versus all previous fully- and self-supervised methods, achieving the best results on the majority of the metrics. Our approach employs the supervised setting using single monocular images in the training and testing phase. Compared with recent SOTA, i.e., FAL-Net, BTS, and PGA-Net, our method is better by a large margin. Meanwhile, unlike FAL-Net using stereo split, two-step training, and post-processing, our method is end-to-end without extra post-processing. The more important thing is that "Ours w/ ViT" has outperformed the SOTA. It can support our standpoint that adding a linear Transformer makes networks improve their ability to capture long-range dependencies. In other words, our network becomes more straightforward but more potent by adapting the linear Transformer.

1.6.3 Multi-View Stereo depth estimation

The robot pose is estimated either in global map (by InLoc [61] and HLOC [56]) or tracked locally in real-time (by ORB-SLAM [49]). This information leads to globally aligned cameras that can be employed in Multi-View Stereo (MVS) depth estimation method. The latest MVS algorithms are either the voxel-based or the depth map based methods. The voxel-based methods divide the space into the regular grid, employ plane-sweep stereo [12] to estimate the cost volumes from multiple views, regularize them using convolutional neural networks (CNN) and regress the depth. The baseline voxel-based method is MVSNet [73]. In recent time, many extensions of MVSNet, i.e., [10, 72, 76, 74] were published. The voxel-based methods are limited to small image resolution because of the hardware requirements to keep the cost volumes in GPU memory. Therefore, we work on top of the SoTA depth map based method Patchmatchnet [67]. PatchmatchNet is a cascade formulation of Patchmatch module enhanced by adaptive propagation based on deep features. Features are extracted by Feature Pyramid Network (FPN) at multiple image resolutions.

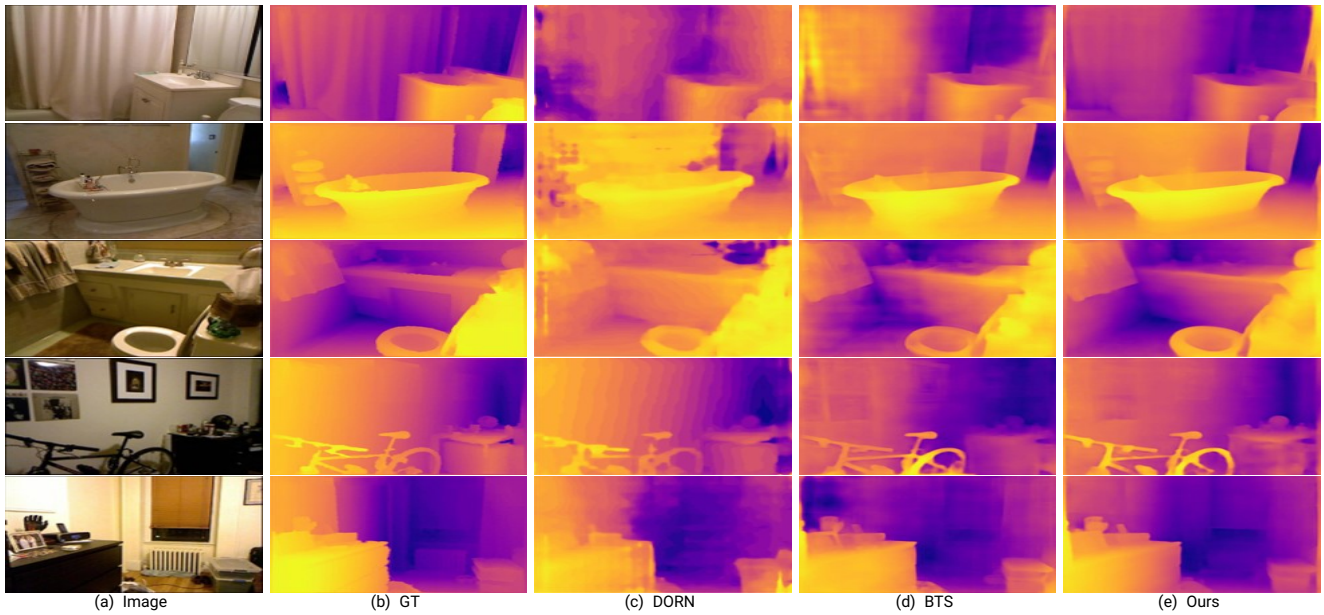
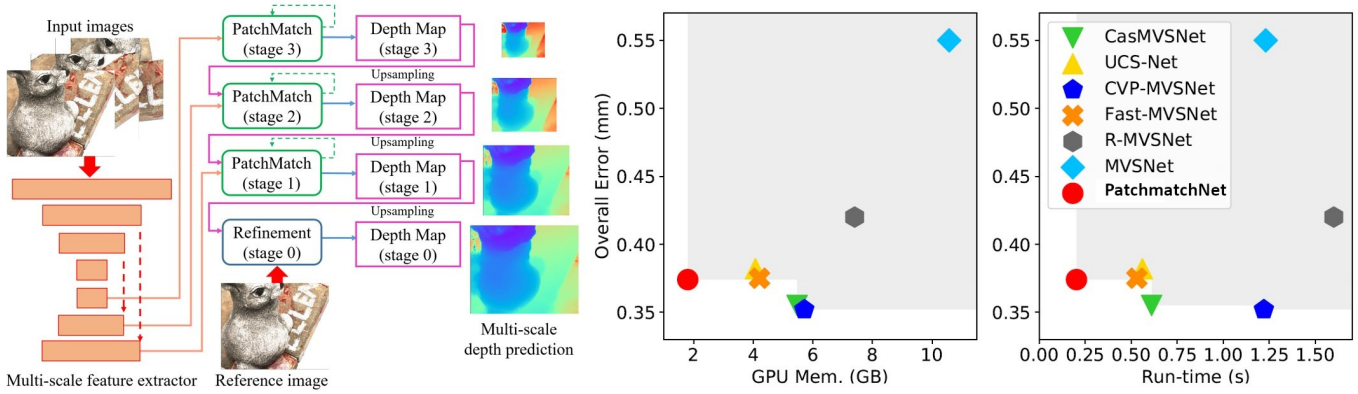


Figure 1.9: Qualitative examples on the NYU depth dataset.



(a) Structure of PatchmatchNet.

(b) Comparison with state-of-the-art learning-based multi-view stereo methods on DTU. Relationship between error, GPU memory and run-time with image size 1152 x 864.

Figure 1.10: a) The multi-scale feature extractor share the features with PatchMatch blocks and Depth Map estimator in a coarse-to-fine manner. The straightforward structure of neural network leads to small memory requirements and fast run-time as shown at b). Source [67].

Patchmatch module generate random hypotheses of the matches at first. Further, the approach iterates between propagation of the hypotheses to the neighbouring patches, and computation of the matching cost for all the hypothesis and selection of best one. Such approach do not divide the space into the regular grid to keep low memory requirements, feature high processing speed, and stay independent of the disparity range in the images. The running demo is publicly available on GitLab [7].

[7] **GitLab repository**: https://gitlab.inria.fr/spring/wp2_mapping_localization/mvs-depth-estimation-patchmatchnet

This project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement No. 871245.

Table 1.5: MVS depth estimation [67]: DTU dataset [1].

Method	Acc.(mm)	Comp.(mm)	Overall(mm)
CasMVSNet [29]	0.325	0.385	0.355
UCS-Net [10]	0.338	0.349	**0.344**
CVP-MVSNet [72]	**0.296**	0.406	0.351
Fast-MVSNet [76]	0.336	0.403	0.370
R-MVSNet [74]	0.383	0.452	0.417
MVSNet [73]	0.396	0.527	0.462
PatchmatchNet [67]	0.427	**0.277**	0.352

This project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement No. 871245.

2 Conclusions

In this deliverable, we presented models, for the instance and semantic segmentation, incremental learning for semantic segmentation, and depth estimation. We tested the models mainly on a publicly available dataset that represents a realistic environment. In the next few months, we will focus on testing such models in a relevant environment, including the results in deliverable D2.7.

This project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement No. 871245.

Bibliography

[1] Henrik Aanæs, Rasmus Ramsbøl Jensen, George Vogiatzis, Engin Tola, and Anders Bjorholm Dahl. Large-scale data for multiple-view stereopsis. *International Journal of Computer Vision*, 120(2):153–168, 2016. 20

[2] Iro Armeni, Ozan Sener, Amir R. Zamir, Helen Jiang, Ioannis Brilakis, Martin Fischer, and Silvio Savarese. 3d semantic parsing of large-scale indoor spaces. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016. 6

[3] Jiawang Bian, Zhichao Li, Naiyan Wang, Huangying Zhan, Chunhua Shen, Ming-Ming Cheng, and Ian Reid. Unsupervised scale-consistent depth and ego-motion learning from monocular video. In *NeurIPS*, 2019. 18

[4] Daniel Bolya, Chong Zhou, Fanyi Xiao, and Yong Jae Lee. Yolact++: Better real-time instance segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 2020. 4, 5

[5] Fabio Cermelli, Massimiliano Mancini, Samuel Rota Bulo, Elisa Ricci, and Barbara Caputo. Modeling the background for incremental learning in semantic segmentation. In *CVPR*, 2020. 8, 12, 13, 14

[6] Arslan Chaudhry, Puneet K Dokania, Thalaiyasingam Ajanthan, and Philip HS Torr. Riemannian walk for incremental learning: Understanding forgetting and intransigence. In *ECCV*, 2018. 14

[7] Jieneng Chen, Yongyi Lu, Qihang Yu, Xiangde Luo, Ehsan Adeli, Yan Wang, Le Lu, Alan L Yuille, and Yuyin Zhou. Transunet: Transformers make strong encoders for medical image segmentation. *arXiv*, 2021. 16

[8] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv*, 2017. 14

[9] Bin Cheng, Inderjot Singh Saggu, Raunak Shah, Gaurav Bansal, and Dinesh Bharadia. s^3 net: Semantic-aware self-supervised depth estimation with monocular videos and synthetic data. In *ECCV*, 2020. 18

[10] Shuo Cheng, Zexiang Xu, Shilin Zhu, Zhuwen Li, Li Erran Li, Ravi Ramamoorthi, and Hao Su. Deep stereo using adaptive thin volume representation with uncertainty awareness. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2524–2534, 2020. 18, 20

[11] Christopher Choy, JunYoung Gwak, and Silvio Savarese. 4d spatio-temporal convnets: Minkowski convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3075–3084, 2019. 4, 6, 7

[12] Robert T Collins. A space-sweep approach to true multi-image matching. In *Proceedings CVPR IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 358–363. IEEE, 1996. 18

[13] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *CVPR*, 2016. 18

[14] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *CVPR*, 2017. 6

[15] Angela Dai and Matthias Nießner. 3dmv: Joint 3d-multi-view prediction for 3d semantic scene segmentation. *CoRR*, abs/1803.10409, 2018. 6

[16] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009. 14

[17] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021. 16

[18] Arthur Douillard, Yifu Chen, Arnaud Dapogny, and Matthieu Cord. Plop: Learning without forgetting for continual semantic segmentation. *CVPR*, 2021. 13, 14

[19] Arthur Douillard, Matthieu Cord, Charles Ollion, Thomas Robert, and Eduardo Valle. Podnet: Pooled outputs distillation for small-tasks incremental learning: Supplementary material. *ECCV*, 2020. 13

[20] David Eigen, Christian Puhrsch, and Rob Fergus. Depth map prediction from a single image using a multi-scale deep network. In *NeurIPS*, 2014. 18

[21] Francis Engelmann, Martin Bokeloh, Alireza Fathi, Bastian Leibe, and Matthias Nießner. 3d-mpa: Multi-proposal aggregation for 3d semantic instance segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9031–9040, 2020. 5

[22] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *IJCV*, 2010. 14

[23] Huan Fu, Mingming Gong, Chaohui Wang, Kayhan Batmanghelich, and Dacheng Tao. Deep ordinal regression network for monocular depth estimation. In *CVPR*, 2018. 18

[24] Jun Fu, Jing Liu, Haijie Tian, Yong Li, Yongjun Bao, Zhiwei Fang, and Hanqing Lu. Dual attention network for scene segmentation. In *CVPR*, 2019. 17

[25] Clément Godard, Oisin Mac Aodha, Michael Firman, and Gabriel J Brostow. Digging into self-supervised monocular depth estimation. In *ICCV*, 2019. 18

[26] Juan Luis Gonzalez and Munchurl Kim. Forget about the lidar: Self-supervised depth estimators with med probability volumes. In *NeurIPS*, 2020. 18

[27] Ben Graham. Sparse 3d convolutional neural networks. *CoRR*, abs/1505.02890, 2015. 6

[28] Benjamin Graham, Martin Engelcke, and Laurens van der Maaten. 3d semantic segmentation with submanifold sparse convolutional networks. *CoRR*, abs/1711.10275, 2017. 6

[29] Xiaodong Gu, Zhiwen Fan, Siyu Zhu, Zuozhuo Dai, Feitong Tan, and Ping Tan. Cascade cost volume for high-resolution multi-view stereo and stereo matching. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2495–2504, 2020. 20

[30] Vitor Guizilini, Rares Ambrus, Sudeep Pillai, Allan Raventos, and Adrien Gaidon. 3d packing for self-supervised monocular depth estimation. In *CVPR*, 2020. 18

[31] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017. 5

[32] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 14, 16

[33] Pedro Hermosilla, Tobias Ritschel, Pere-Pau Vázquez, Àlvar Vinacua, and Timo Ropinski. Monte carlo convolution for learning on non-uniformly sampled point clouds. *CoRR*, abs/1806.01759, 2018. 6

[34] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv*, 2015. 13

[35] Li Jiang, Hengshuang Zhao, Shaoshuai Shi, Shu Liu, Chi-Wing Fu, and Jiaya Jia. Pointgroup: Dual-set point grouping for 3d instance segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4867–4876, 2020. 5

[36] Adrian Johnston and Gustavo Carneiro. Self-supervised monocular trained depth estimation using self-attention and discrete disparity volume. In *CVPR*, 2020. 18

[37] Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. Supervised contrastive learning. *NeurIPS*, 2020. 9

[38] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 2017. 14

[39] Marvin Klingner, Jan-Aike Termöhlen, Jonas Mikolajczyk, and Tim Fingscheidt. Self-supervised monocular depth estimation: Solving the dynamic object problem by semantic guidance. In *ECCV*, 2020. 18

[40] Loïc Landrieu and Martin Simonovsky. Large-scale point cloud semantic segmentation with superpoint graphs. *CoRR*, abs/1711.09869, 2017. 6

[41] Jin Han Lee, Myung-Kyu Han, Dong Wook Ko, and Il Hong Suh. From big to small: Multi-scale local planar guidance for monocular depth estimation. *arXiv*, 2019. 18

[42] Yangyan Li, Rui Bu, Mingchao Sun, and Baoquan Chen. Pointcnn. *CoRR*, abs/1801.07791, 2018. 6

[43] Zhizhong Li and Derek Hoiem. Learning without forgetting. *TPAMI*, 2017. 14

[44] Zhihao Liang, Zhihao Li, Songcen Xu, Mingkui Tan, and Kui Jia. Instance segmentation in 3d scenes using semantic superpoint tree networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2783–2792, 2021. 4, 5

[45] Andrea Maracani, Umberto Michieli, Marco Toldo, and Pietro Zanuttigh. Recall: Replay-based continual learning in semantic segmentation. *ICCV*, 2021. 14

[46] Umberto Michieli and Pietro Zanuttigh. Incremental learning techniques for semantic segmentation. In *ICCV Workshops*, 2019. 14

[47] Umberto Michieli and Pietro Zanuttigh. Continual semantic segmentation via repulsion-attraction of sparse and disentangled latent representations. *CVPR*, 2021. 14, 15

[48] Umberto Michieli and Pietro Zanuttigh. Knowledge distillation for incremental learning in semantic segmentation. *CVIU*, 2021. 14

[49] Raul Mur-Artal, Jose Maria Martinez Montiel, and Juan D Tardos. Orb-slam: a versatile and accurate monocular slam system. *IEEE transactions on robotics*, 31(5):1147–1163, 2015. 18

[50] Hao Pan, Shilin Liu, Yang Liu, and Xin Tong. Convolutional neural networks on 3d surfaces using parallel frames. *CoRR*, abs/1808.04952, 2018. 6

[51] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *CoRR*, abs/1706.02413, 2017. 6

[52] Anurag Ranjan, Varun Jampani, Lukas Balles, Kihwan Kim, Deqing Sun, Jonas Wulff, and Michael J Black. Competitive collaboration: Joint unsupervised learning of depth, camera motion, optical flow and motion segmentation. In *CVPR*, 2019. 18

[53] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *CVPR*, 2017. 14

[54] Gernot Riegler, Ali Osman Ulusoy, and Andreas Geiger. Octnet: Learning deep 3d representations at high resolutions. *CoRR*, abs/1611.05009, 2016. 6

[55] Samuel Rota Bulò, Lorenzo Porzi, and Peter Kontschieder. In-place activated batchnorm for memory-optimized training of dnns. In *CVPR*, 2018. 14

[56] Paul-Edouard Sarlin, Cesar Cadena, Roland Siegwart, and Marcin Dymczyk. From coarse to fine: Robust hierarchical localization at large scale. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12716–12725, 2019. 18

[57] Konstantin Shmelkov, Cordelia Schmid, and Karteek Alahari. Incremental learning of object detectors without catastrophic forgetting. In *ICCV*, 2017. 14

[58] Chang Shu, Kun Yu, Zhixiang Duan, and Kuiyuan Yang. Feature-metric loss for self-supervised learning of depth and egomotion. In *ECCV*, 2020. 18

[59] Jaime Spencer, Richard Bowden, and Simon Hadfield. Defeat-net: General monocular depth via simultaneous unsupervised representation learning. In *CVPR*, 2020. 18

[60] Hang Su, Varun Jampani, Deqing Sun, Subhransu Maji, Evangelos Kalogerakis, Ming-Hsuan Yang, and Jan Kautz. Splatnet: Sparse lattice networks for point cloud processing. *CoRR*, abs/1802.08275, 2018. 6

[61] Hajime Taira, Masatoshi Okutomi, Torsten Sattler, Mircea Cimpoi, Marc Pollefeys, Josef Sivic, Tomas Pajdla, and Akihiko Torii. Inloc: Indoor visual localization with dense matching and view synthesis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7199–7209, 2018. 18

[62] Onur Tasar, Yuliya Tarabalka, and Pierre Alliez. Incremental learning for semantic segmentation of large-scale remote sensing data. *J-STARS*, 2019. 14

[63] Maxim Tatarchenko, Jaesik Park, Vladlen Koltun, and Qian-Yi Zhou. Tangent convolutions for dense prediction in 3d. *CoRR*, abs/1807.02443, 2018. 6

[64] Lyne P. Tchapmi, Christopher B. Choy, Iro Armeni, JunYoung Gwak, and Silvio Savarese. Segcloud: Semantic segmentation of 3d point clouds. *CoRR*, abs/1710.07563, 2017. 6

[65] Lokender Tiwari, Pan Ji, Quoc-Huy Tran, Bingbing Zhuang, Saket Anand, and Manmohan Chandraker. Pseudo rgb-d for self-improving monocular slam and depth prediction. In *ECCV*, 2020. 18

[66] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017. 15, 16

[67] Fangjinhua Wang, Silvano Galliani, Christoph Vogel, Pablo Speciale, and Marc Pollefeys. Patchmatchnet: Learned multi-view patchmatch stereo. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14194–14203, 2021. 4, 18, 19, 20

[68] Wenguan Wang, Tianfei Zhou, Fisher Yu, Jifeng Dai, Ender Konukoglu, and Luc Van Gool. Exploring cross-image pixel contrast for semantic segmentation. *arXiv*, 2021. 9

[69] Dan Xu, Xavier Alameda-Pineda, Wanli Ouyang, Elisa Ricci, Xiaogang Wang, and Nicu Sebe. Probabilistic graph attention network with conditional kernels for pixel-wise prediction. *TPAMI*, 2020. 18

[70] Guanglei Yang, Enrico Fini, Dan Xu, Paolo Rota, Mingli Ding, Moin Nabi, Xavier Alameda-Pineda, and Elisa Ricci. Uncertainty-aware contrastive distillation for incremental semantic segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022. 4, 8, 13, 15

[71] Guanglei Yang, Hao Tang, Zhun Zhong, Mingli Ding, Ling Shao, Nicu Sebe, and Elisa Ricci. Transformer-based source-free domain adaptation. *arXiv*, 2021. 4, 16

[72] Jiayu Yang, Wei Mao, Jose M Alvarez, and Miaomiao Liu. Cost volume pyramid based depth inference for multi-view stereo. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4877–4886, 2020. 18, 20

[73] Yao Yao, Zixin Luo, Shiwei Li, Tian Fang, and Long Quan. Mvsnet: Depth inference for unstructured multi-view stereo. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 767–783, 2018. 18, 20

[74] Yao Yao, Zixin Luo, Shiwei Li, Tianwei Shen, Tian Fang, and Long Quan. Recurrent mvsnet for high-resolution multi-view stereo depth inference. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5525–5534, 2019. 18, 20

[75] Wei Yin, Yifan Liu, Chunhua Shen, and Youliang Yan. Enforcing geometric constraints of virtual normal for depth prediction. In *ICCV*, 2019. 18

[76] Zehao Yu and Shenghua Gao. Fast-mvsnet: Sparse-to-dense multi-view stereo with learned propagation and gauss-newton refinement. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1949–1958, 2020. 18, 20

[77] Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. *PLMR*, 2017. 14

[78] Biao Zhang and Peter Wonka. Point cloud instance segmentation using probabilistic embeddings. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8883–8892, 2021. 5

[79] Sixiao Zheng, Jiachen Lu, Hengshuang Zhao, Xiatian Zhu, Zekun Luo, Yabiao Wang, Yanwei Fu, Jianfeng Feng, Tao Xiang, Philip HS Torr, et al. Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers. In *CVPR*, 2021. 16, 17