Politecnico di Torino

POLITECNICO MILANO 1863

EPFL

POLITECNICO DI TORINO

MASTER OF SCIENCE
DATA SCIENCE AND ENGINEERING

TOWARDS CONTEXT BASED MONOCULAR DEPTH ESTIMATION

SUPERVISORS:

Prof. Barbara Caputo
Prof. Nicola Gatti
Prof. Sabine Süsstrunk

CANDIDATE:

Alessio Cappellato

A.Y. 2020/2021
GRADUATION SESSION OCTOBER 2021

This is only a foretaste of what is to come
and only the shadow of what is going to be
— Alan Turing

To my family

Acknowledgements

I would like to first express my sincere gratitude to Deblina Bhattacharjee, for assisting and guiding me in the work on this thesis, and Professor Sabine Süsstrunk, for welcoming me in her Images and Visual Representation Lab, allowing me to study such a stimulating research topic and providing me with all the resources I could need; I also wish to extend my appreciation to Professor Barbara Caputo and Professor Nicola Gatti, for their valuable counsel. As my supervisors, their expertise and insightful feedback have undoubtedly elevated my work and have been a source of inspiration, both academically and personally.

I would like to genuinely thank all the Professors I had the opportunity to learn from in these past five years of study across Europe, as I wouldn't have been able to achieve this goal without the knowledge and passion they passed on to me. I want to thank my fellow colleagues from all around the World with whom I shared this demanding journey studded with hard-earned accomplishments. A special mention is for the friends met during my experiences abroad, as they have been a continuous source of motivation to constantly challenge and improve myself and of light-hearted distractions when study permitted us; I will always cherish the time we spent together.

I would also like to thank my closest friends, whom I missed dearly throughout the last year, for firmly believing in me since the very beginning and for being there to celebrate our achievements together; I feel extremely grateful for their unquestioned presence in my life.

Finally, my deepest gratitude goes to my family. To my parents and grandparents, for lovingly raising me as the person I am now, for encouraging me to make my dreams come true and for backing each and every of my decisions. To my girlfriend Elena, for her unwavering support despite the prolonged physical distance, for always seeing and bringing out the best in me, and for being a warm light in my darkest moments. A last mention is for my dog Cal, for invariably welcoming me with his waggy tail after a busy day and for being a loyal companion during the long study sessions at home. Their invaluable help has been essential over the years and I cannot express enough how thankful I am for it and how fortunate I feel.

Turin, October 13, 2021 Alessio

Abstract

Monocular depth estimation is a classical computer vision task, which consists in densely predicting the spatial distance between the object depicted by each pixel and the camera with which the single RGB image is taken. This type of information is extremely useful for a variety of practical contexts, like 3D reconstruction, visual simultaneous localization and mapping (SLAM), and autonomous driving systems, because it permits reasoning about the geometrical structure of the environment and the relationship between objects in it.

Over the last years, a number of fully-convolutional encoder-decoder networks have been used to study the considered problem; their popularity is rooted in their locality and translation invariance properties, which allow a parameter-efficient modelling of highly spatially-correlated information. In this context, in the first part of this work, we improve the design of a convolutional decoder incorporating the Laplacian pyramid decomposition of the input image to guide the progressive prediction of depth residuals; this additional feature provided to the decoder retains important information on the location of object boundaries, but also uninformative noise due to intra-object variations. To overcome this issue, we propose to use contours extracted from instance segmentation masks to filter out the noise and keep only the semantically relevant Laplacian residuals. The resulting method achieves a performance improvement on most metrics and a reduction of visual artifacts.

More recently, the success of transformers and the ability of their attention mechanism to model long-range dependencies (contrarily to the limited receptive field of convolutions) have sparked many studies proving their competitiveness with convolution-based methods. In the second part of this thesis, we thus adopt a vision transformer-based paradigm both in the design of the encoder and subsequently of the decoder. In particular, we propose a multitask setting with depth estimation and semantic segmentation to conduct a thorough study on the role of attention and its impact on the cross-task interaction. We initially focus on the development of custom attention inside a columnar transformer encoder and employ double-head convolutional decoders for independent dense prediction, revealing that attention sharing is beneficial for both tasks in comparison to the individual monotask performance. Moreover, we show that the extraction of task-invariant features in a single stream further improves the results on all metrics. Finally, we adopt a pyramidal transformer encoder with shifted windows, to better leverage the power of skip connections, and extend the use of transformers to the decoding stage by proposing various monotask and multitask decoders, thereby obtaining convolution-free networks. While in the monotask setting the performance of the proposed transformer decoders is comparable with that of the convolutional ones, the improvement brought by the interaction with the segmentation task is slightly lower. Overall, we systematically outperform the state of the art and our previous results, as proved by extensive experimentation on the official NYU Depth V2 dataset, and demonstrate that transformers can achieve comparable results and surpass convolutional methods even when trained with few samples.

The PyTorch source code of all methods proposed in this thesis is available at

https://github.com/CappellatoAlessio/master-project.

Contents

List of Figures

List of Tables

Introduction

In this part, we briefly introduce the task with its peculiar characteristics and challenges, we consider several aspects that currently make it extremely relevant in numerous fields and why hardware devices previously used to obtain the same type of information are now regarded as inadequate; we then shortly mention the most important software methods adopted in the past to tackle this task, their major achievements and shortcomings; finally, we describe the relevant datasets commonly considered for this task and formally define the task itself and the metrics used to evaluate the performance of previous and our works.

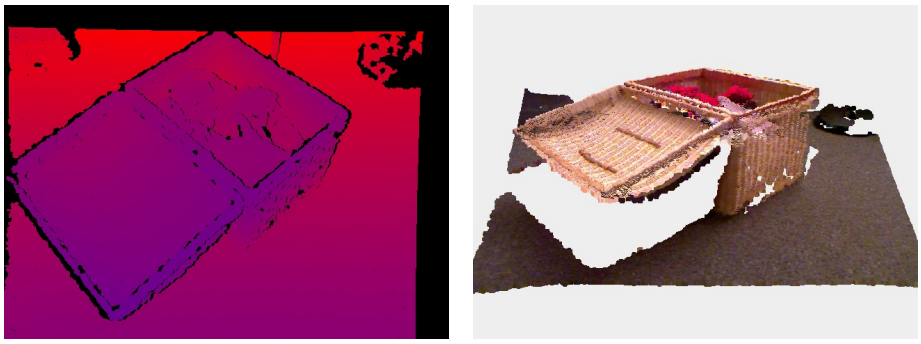
After this introduction, the work is divided in parts and then again in chapters, containing a more specific introduction, a detailed presentation of related work and of our proposed methods, quantitative and qualitative evaluations to compare with the state-of-the-art and some derived conclusions.

The task: Monocular Depth Estimation

Depth estimation is a classical task in computer vision, together with tasks like object detection, object recognition and semantic segmentation, to which it is strongly related. Together with this last one, depth estimation is a dense prediction task, meaning that the output of the model has the same resolution of the input image, or, in other words, that an output value must be predicted by the model for every single pixel of the input image, in a 1-to-1 correspondence. More specifically, depth estimation consists in the prediction of a real value (even if it can be remapped as a dense classification task) representing the spatial distance in meters between the object depicted in a given pixel of the input image and the camera.

This category of tasks is considerably more challenging than non-dense prediction tasks, because the model must be able not only to infer high-level information from the whole input image, but also to retain local information and to effectively merge these finer details with the coarser scene understanding to obtain accurate high-resolution predictions.

In particular, in this work we focus on the even more challenging monocular depth estimation sub-task, which consists in the use of a single RGB image as input (in opposition to stereo



(a) Depth map (b) Colored point cloud

Figure 1: 3D reconstruction of RGB image from depth map.

depth estimation, in which multiple viewpoints are exploited); this causes the problem to be ill-posed, because the same 2D projection can be derived from infinitely many distinct 3D scenes and because occlusions cannot be, at least partially, resolved.

Depth information (the output of depth estimation) is essential for several real-world application, in particular with respect to autonomous and robotics systems, fields that have been gaining more and more interest in the recent years, consequently increasing the importance of the considered task: depth information is indeed crucial to allow understanding and reconstruction of 3D spaces and their geometry (as shown in Figure 1), which makes it possible for agents to effectively perceive the surrounding environment and their position into it and to take informed decisions regarding their navigation in it and their interactions with it.

This type of information can also be obtained through 3D sensors, such as LiDAR (Light Detection And Ranging) and RGB-D cameras (e.g., Kinect), or even RaDAR and SoNAR. In general, these devices have several drawbacks in terms of cost, space requirements, computing time, power consumption and processing capacity, they can only provide sparse depth maps and are strongly affected by distance range (indoor vs outdoor scenes) and lighting conditions. In particular, Kinect belongs to the category of Time-of-Flight sensors, meaning that the distance is computed by measuring the travel time of a light pulse to the objects in the scene and back to the camera, hence it is suited only for short ranges and it is more accurate the closer objects are to the sensor. LiDAR is a better quality sensor employing a laser scanner which produces highly accurate measurements, but it is more costly, it consumes significant amounts of power and it is bigger in size and considerably more fragile because of its moving parts.

Traditional methods

Traditional methods employed hand-crafted features, such as perspective, lighting, occlusion, objects size and localization, haze, texture variations and gradients, and probabilistic graphical models to extract depth information from single images. Hoiem et al. [23] proposed

a method to learn and coarsely estimate the relationship between geometric structural classes (ground, sky and vertical regions), to then break down the scene into a few planar surfaces and predict depth according to their orientation with respect to the camera; Saxena et al. [46] proposed a Markov Random Field able to exploit multiscale local and global features and to predict both absolute patch-wise depth and relative depth between different patches; Delage et al. [11] proposed a dynamic Bayesian network which assumes a "floor-wall" geometry of the scene to estimate the boundary between the two and then use this perspective cue to predict depth. Other works [29, 27, 34] proposed non-parametric methods, which retrieve the top relevant image-depth pairs from a RGB-D dataset according to photometric content matching, align them to the query RGB image and use the aligned candidate depth maps to generate the prediction.

In parallel, other methods were simultaneously designed, focusing on the exploitation of geometric constraints when multiple views are available; an exhaustive study of these approaches is out of the scope of this work, but a general overview is proposed in the following for completeness.

The historically most studied setting is the one considering stereo images ('stereo depth estimation', similar to the stereopsis process in human vision), i.e., simultaneously taken images of the same scene from sufficiently different points of view: typically, the methods [48, 47, 81] proposed in this setting use color intensity gradient to identify object boundaries and estimate disparity maps by rectifying the pair of images (projecting them to a common space), matching their features and minimizing a smoothing cost function to then accordingly triangulate points mapped to matched pixel pairs in a point cloud and finally retrieve the surface structure of the scene. Stereo depth estimation is limited by the distance between the cameras, which inevitably worsens the quality of the disparity map in the regions corresponding to high distance objects, and by the need of alignment and calibration (resectioning) to estimate the parameters approximating the different cameras and the transformation between them, which are expensive techniques allowing to relate 2D points in the cameras image space with locations in the 3D scene (hence acquiring depth information) [2, 76]; furthermore, the performance regarding lowly textured regions and occluded objects is usually unsatisfactory.

'Structure from motion', another setting similar in spirit, considers a sequence of frames taken with a single camera (monocular) during a variable time span, while performing some camera motion of variable trajectory and extent: most methods [60, 57, 26] proposed in this setting extract scale- and geometry- invariant features (isolated points, line end-points or texture elements), match them to find candidate scene overlaps in input images, verify their satisfaction of geometry constraints by estimating a valid transformation between corresponding features and reconstruct the scene through triangulation of the confirmed multi-view overlaps. Some limitations of this setting are a general assumption of static and rigid scenes and of non-zero camera translation (non purely rotational motion) to induce parallax, a strong dependence on exact feature matching and the inherent scale ambiguities caused by an insufficient camera rotation angle and number of frames [56].

To address some stereo and structure from motion problems, such as occlusion and feature correspondence, the 'depth from focus' setting has been proposed: it is based on the optical relationships between objects, camera lens and image detector, which determine the distance from the camera at which objects are focused depending on focal length and lens position (while objects at different distances are more blurred the larger the difference is). This setting [53, 10] then requires to take several images (typically more than 10) of the target object with varying camera parameters and to select among them the one corresponding to the best focus of the object, by using a function measuring the image high-frequency content; once the focusing problem is solved, knowing the camera parameters that produced the solution, it is possible to compute the depth of the object through the lens equation. This setting was then generalized in 'depth from defocus' [54], in which only few images are necessary (possibly none of which in focus): for each of them, the degree of defocus of all objects in the scene is estimated either through Fourier-domain or spatial-domain based methods; once this information from all images is combined, a map of relative distances is generated and the depth map is derived by computing the focus distance (lens equation with known camera parameters). Nevertheless, this setting still suffers of several limitations, most notably it still assumes static scenes and it heavily relies on the blur level estimation techniques.

Finally, the 'shape from shading' setting is based on the optical principle of shading: the interaction between illumination, shape of the surface, reflecting properties of the surface, and image projection determines variations in the image irradiance; the problem consists then in recovering the surface orientation satisfying the information encoded in the irradiance map, either by using the gradient, which is not compatible with constraints imposed by occluding boundaries, or directly the surface-normal vectors. This extraction can be done mainly in four ways [75]: by minimizing the integral of the brightness error (with the addition of regularization terms on surface smoothness or integrability) [24], by propagating the shape from a set of surface points, by using derivatives of the intensity under local spherical assumption, or by solving the linearization of the reflectance map (which describes the radiance as a function of surface orientation, known the light source). Typically, these methods assume a Lambertian model of image formation, smoothly curved surfaces with homogeneous reflecting characteristics and independence of radiance from surface position in space, but Lambertian reflectance is not always satisfied and impossibly shaded images exist (no corresponding smooth surface given the assumptions) [25]; consequently, they do not perform well in highly textured, non-uniformly colored scenes with structures at different depths.

In general, while each setting has its specific strengths and weaknesses, an issue common to all of these approaches is a need of considerably more resources (computation time, memory, energy consumption) and data with respect to the monocular depth estimation task; moreover, the recent availability of large datasets and advancements in the field of deep learning have demonstrated that automatically extracted deep features are far superior to those employed in all previous works.

Early Deep Learning methods

The methods proposed in modern depth estimation can be mainly categorized in supervised and self-supervised. Before presenting here the most important cornerstones in supervised monocular depth estimation (the scope of this work), the self-supervised category is briefly discussed for completeness.

Self-supervised methods

Since ground-truth depth maps are not used in self-supervised training, additional information in input is needed to make up: often this includes stereo images with left and right views, monocular video sequences, visual odometry or 6D pose; nevertheless, these are still considered in the field of monocular depth estimation, because the additional data needed during training are not required anymore in test and inference.

The first case, similar to 'stereo depth estimation', was proposed by Garg et al. [16]: the CNN is similar to an autoencoder, where the encoder is trained to predict the inverse depth (disparity) map of the left view, which is then used in the warping of the right view to reconstruct the input image; the resulting photometric difference (with the addition of a smoothness term) is used as reconstruction loss. Godard et al. [19] improved the previous network by predicting both left and right disparity maps, to learn pixel-level correspondences between the two views through a loss term enforcing left-right disparity consistency; furthermore, [20] redesigned the loss function to ignore static pixels (against the moving camera assumption), reduce visual artifacts and handle occluded pixels. Aleotti et al. [1] proposed to cast the task within a Generative Adversarial Networks (GAN) for image reconstruction paradigm, in which the generator behaves as the encoder of the previous works and the discriminator has to distinguish between the resulting warp of the input left image and the right view. Tosi et al. [58] proposed a network mimicking the stereo setup even if fed with a single view (considered as left) both during training and test: in a first stage multi-scale deep features are extracted, then multi-scale disparity maps are estimated and used to warp-synthesize deep features of a virtual right view, to be able to exploit stereo matching consistency in the final refinement.

The second case, similar to 'structure from motion', was proposed by Zhou et al. [80]: two independent CNNs are used, one trained to predict the inverse depth (disparity) map at time t and the other to predict the camera pose motion between times t and $t+1$, which are then jointly used as in [16] to warp the view at time $t+1$ and reconstruct the view at time t; the two CNNs are coupled at training time by the use of a single reconstruction loss, based on view synthesis [66] (with the addition of a smoothness term). Mahjourian et al. [38] proposed to use the predictions of the two networks to enforce temporal consistency, by adding a 3D geometric loss term computed between the structured point cloud derived from the predicted depth at time t and the warping of the structured point cloud derived from the predicted depth at time $t+1$ (and, symmetrically, vice versa). Yin and Shi [74], instead, proposed to use the predictions of the two networks to estimate, first, the optical flow due to ego-motion in the static scene

geometry by assuming a rigid structure and, second, the residual optical flow due to the non-rigid motion of dynamic entities; an adaptive geometric consistency loss term between bidirectional ($t \rightarrow t+1,\ t+1 \rightarrow t$) pairs of flow estimates is added at both stages. Casser et al. [6, 7] deeply modified the network architecture to better address the difference between static background and dynamic entities: they proposed to estimate ego-motion by masking out all moving objects to only consider the background and to estimate the motion of each individual moving object separately by masking out the rest (thanks to instance segmentation masks); the warping function needed to reconstruct and compare consecutive images then becomes a sequence of warpings in which the predicted ego-motion is applied first, followed by all the predicted object motions in the corresponding segmented portions.

In general, self-supervised methods perform worse than supervised ones, because they suffer more heavily from lack of generalization (they cannot rectify their own bias) and from scale inconsistency and ambiguity, since they can only exploit geometric constraints as a proxy supervisory signal instead of actual ground-truth maps of the target depth information to be predicted.

Supervised methods

The use of deep CNNs for supervised learning in the setting of monocular depth estimation was introduced by Eigen et al. [14]: the proposed architecture leverages multi-scale information and consists in a first stream making a coarse prediction based on a global view of the input image and a second stream extracting features on a local scale; the output of the former stream is injected in the latter and further processed to locally refine the coarse prediction with finer details. They treated the task as a dense regression problem and introduced a new task-specific term in the loss function to be minimized during training (together with the element-wise mean squared error): a scale-invariant error was proposed to focus on the measurement of depth relations within the scene instead of on the global scale (which is the major ambiguity inherent to the considered setting and was found to be responsible of a large fraction of the computed error). Eigen and Fergus [13] improved the previous work by proposing a general multi-purpose model, able to be applied with minor modifications to three different dense prediction tasks: depth estimation, surface normal estimation and semantic segmentation. The proposed architecture is deeper in terms of number of convolutional layers, leverages an additional scale of higher resolution, and does not predict an unrefined global-scale depth map: the first stream extracts coarse features based on a global view of the input image, the second stream refines them with finer features extracted with a narrower field of view and produces a mid-resolution prediction, the third stream locally refines the intermediate prediction with higher-resolution details and produces a final output of higher resolution than [14]. Furthermore, regarding the depth estimation task, they added a first-order matching loss term comparing depth gradients to enforce local structural similarity between prediction and ground truth.

Liu et al. [33] proposed a deep convolutional neural field (DCNF) model to exploit the feature extraction capabilities of CNNs and the task formulation as a continuous conditional random field (CRF) learning problem. The input image is over-segmented into superpixels and the image patches centred around each superpixel centroid are considered; to compute the unary potentials, a shared CNN predicts separately for every input patch the depth value of the corresponding superpixel (assumed homogeneous); to compute the pairwise potentials, a shared fully-connected layer predicts separately for every pair of neighbouring superpixels their similarity; unary and pairwise potentials are then fed into the continuous CRF to obtain the final prediction. Alternatively, they also proposed to replace the architecture of the unary part with a CNN fed a single time with the whole image, followed by a superpixel pooling layer to obtain up-sampled superpixel-level features and by fully-connected layers to compute the corresponding unary potentials.

Laina et al. [30] was the first work to propose a fully-convolutional encoder-decoder structure, based on ResNet [21] with the final fully-connected layer substituted with a sequence of four novel up-sampling blocks: the concept of projection residual connection was extended to up-convolutions by adding a 3×3 convolution after the up-convolution and a projection connection between lower-resolution input (up-sampled with its own up-convolution) and higher-resolution output. They also proposed the use of the reverse Huber as loss function, balancing between the $\mathfrak{L}_1$ norm for errors in the neighbourhood of 0 and the $\mathfrak{L}_2$ norm for larger errors, to address the heavy-tailed distribution of depth values that can be observed in the considered datasets.

Xu et al. [69] proposed a fully-convolutional encoder-decoder architecture predicting intermediate depth maps of increasing resolution at each of the four decoding layers; to integrate the complementary multi-scale information and produce a final prediction map, these side outputs are then fused through continuous CRFs, in two possible ways: a unified multi-scale CRF that simultaneously combines all predictions and enforces smoothness constraints between neighboring pixels and different scales, or a cascade of multiple scale-specific CRFs that gradually refine the lower-resolution observations of the previous models with the corresponding higher-resolution side output. They also described how to implement the two proposed CRFs-based models as sequential deep networks, by introducing a stack of novel elementary blocks to perform mean-field updates. Xu et al. [70] improved the previous work by moving the continuous CRF from the side to the bottleneck between the fully-convolutional encoder and decoder, to directly operate on the extracted features instead of on the prediction maps: a unified multi-scale CRF models the relationship between corresponding feature maps at the last scale and at each of the previous intermediate scales through an attention mechanism controlling the inter-scale information flow and enforcing pixel-level structural constraints; the jointly estimated multi-scale features and attention maps are then used to refine the deepest features before passing them to the decoder for the final prediction.

Cao et al. [5] proposed to remap the dense task from a continuous regression problem to a classification problem with depth ranges uniformly discretized in the log space, in order

to naturally obtain a confidence in the form of probability distribution in addition to the class prediction. The proposed architecture is a fully-convolutional encoder-decoder network producing softmaxed dense score maps, which are then refined in post-processing by a fully-connected CRF (the pairwise potential of all pairs of pixels is considered) able to improve estimates with low confidence. The loss function used during training was a modified version of cross entropy loss: an information gain matrix is introduced to tune the contribution of each pixel according to its predicted probability distribution and absolute distance with respect to ground truth (while in typical classification tasks there is no similarity relationship between separate classes).

Fu et al. [15] proposed to recast the task as an ordinal regression problem with depth ranges spacing-increasingly discretized, to allow relatively larger errors in the prediction of larger depths and focus on a more precise prediction of smaller depths. The proposed architecture presents a fully-convolutional encoder as feature extractor, with some modifications in the last blocks: the max-pooling down-sampling operators were removed and the convolutional layers were replaced by dilated convolutions to enlarge their field-of-view without reducing the feature spatial resolution. The proposed decoder consists of three parallel branches: an atrous spatial pyramid pooling (ASPP) module with three different dilation rates to capture multi-scale information without the need of skip connections, a 1×1 convolution to extract cross-feature interactions, and a full-image encoder to summarize the feature maps with a global view through average-pooling and a small fully-connected layer; the outputs of the branches are then concatenated (the global feature vector is copied to restore its spatial resolution) and processed to obtain the multi-channel dense ordinal predictions. The ordinal loss function used explicitly models the discrete labels as a well-ordered set such that the model learns to predict for every depth range if the pixel depth value is larger than the left extremum of that range and the last depth range to have a positive estimate to this question is considered to be the predicted depth range (since the left extremum of the immediately larger depth range, which is the right extremum of the considered depth range, is estimated to be larger than the pixel depth value); this allows to tune the penalty based on the distance from the true label.

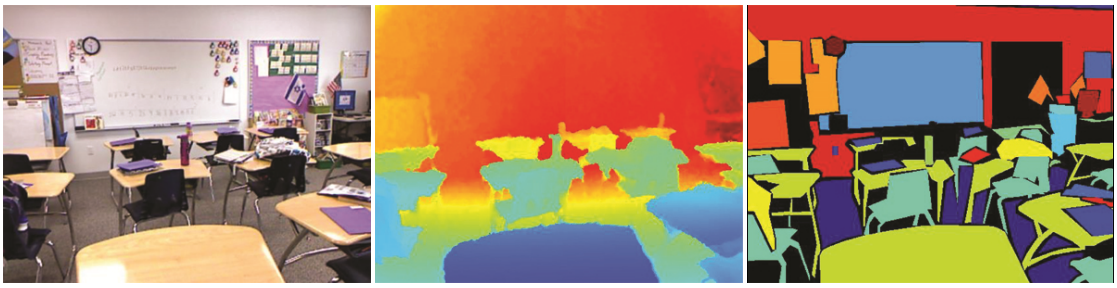
Evaluation

Datasets

In the following sections, the datasets most commonly considered for the task are presented; a summary regarding dataset size, image resolution, distance range and additional annotation can be found in Table 1.

Table 1: Datasets for Monocular Depth Estimation.

Dataset	Training frames	Resolution	Range (m)	Annotations
NYU Depth V2 [50]	795	480 × 640	10	Depth+Segmentation
KITTI [59]	23488	352 × 1216	50-80	Depth



(a) RGB image (b) Depth map (c) Segmentation mask

Figure 2: NYU Depth V2 [50] official dataset.

NYU Depth V2

The NYU Depth V2 dataset by Silberman et al. [50] (2012) is the most commonly used dataset for indoor monocular depth estimation, with a distance range of 10 meters. It is the extension of the NYU Depth dataset [49], which only contained ~100k frames, from 64 different indoor scenes and 7 scene types. This new version, instead, contains more than 400k frames, from 464 different indoor scenes, 3 cities and 26 scene types; the dataset consists of video sequences of RGB and depth pairs recorded by a Kinect camera, with a resolution of 480 × 640 pixels and a variable sampling rate between 20 – 30 FPS. Officially, 249 scenes are reserved for training (for a total amount of ~240k frames) and 215 scenes are reserved for test.

A subset of 1449 pairs constitutes the official dataset, with 795 training images from the 249 training scenes and 654 test images from the 215 test scenes; these images were hand selected

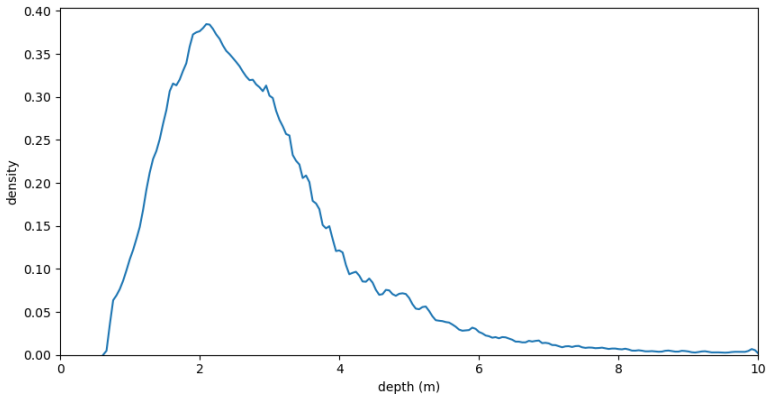


Figure 3: NYU Depth V2 [50] depth distribution.

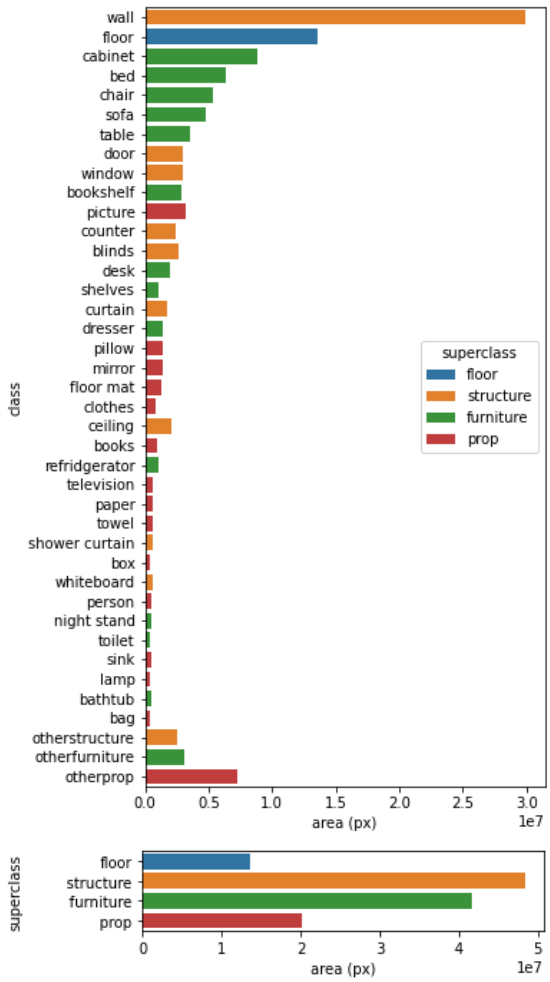


Figure 4: NYU Depth V2 [50] class distribution.

to maximise diverse content and minimize similarities between frames. These pairs have been synchronized in time and aligned in space (the raw depth maps have been projected onto the RGB space); furthermore the sparse depth maps have been in-painted with the colorization by Levin et al. [32]. The depth distribution is shown in Figure 3.

The official dataset additionally provides dense labeling for semantic segmentation: 35064 distinct objects are labelled in 894 classes (plus the 'void' 0-class, representing unlabeled pixels) and multiple instances of the same object class in an image are distinguished with a counter for instance segmentation. An example triplet of RGB image, depth map and segmentation mask is depicted in Figure 2. Furthermore, each class is mapped to one of the 4 superclasses: 'floor', 'structure' (non-floor parts of the room), 'furniture' (large objects) and 'prop' (small objects). Commonly, the 894 classes are reduced to 40, with the top-37 classes in area (number of pixels), including the 'floor' class, and the remaining aggregated according to the corresponding superclass ('otherstructure', 'otherfurniture' and 'otherprop'). The class and superclass distributions are shown in Figure 4, from which the heavy unbalance of the dataset is clearly noticeable.

KITTI

The KITTI Raw dataset by Geiger et al. [17, 18] (2013) is the most commonly used dataset for outdoor monocular depth estimation, with a distance range of 120 meters. It contains ~94k frames, from 'city', 'residential', 'road', 'campus' and 'person' scenes; the dataset consists of video sequences of stereo RGB and grayscale images and depth point clouds recorded by a Velodyne LiDAR, with a resolution of 352 × 1216 pixels, 100k−120k 3D irregularly spaced points per frame and a sampling rate of 10 FPS. These images and point cloud pairs have been synchronized in time and rectified in space; furthermore, 3D bounding box tracklet labels for

(a) RGB image (b) Depth map

Figure 5: KITTI [59] dataset.

over 200k objects and 8 classes, optical flow maps and odometry trajectories are provided.

The KITTI Depth dataset by Uhrig et al. [59] (2017) introduced the depth maps derived from projected LiDAR point clouds and aligned with the corresponding frames of the KITTI Raw dataset. An example pair of RGB image and depth map is depicted in Figure 5.

The vast majority of works uses the so called Eigen split [14], which selected 56 scenes from the 'city', 'residential' and 'road' categories and split them in 28 for training and 28 for test. Each training scene contains on average ~800 frames, but both (left and right) images are kept and treated as unassociated, while stationary frames are filtered out to avoid identical images; the training set contains then 23488 pairs, while only 697 are selected for the test set. The maximum depth is usually set to either 50 or 80 meters.

Metrics

In this section, we formally define the task and the metrics commonly used to evaluate models performance.

Since distance is a continuous measure, the vast majority of works consider the Monocular Depth Estimation task as a pixel-level continuous regression problem and even the works remapping the task to its discretized classification or ordinal regression versions try to eventually derive a continuous value from the discrete predictions. Moreover, as discussed above, self-supervised frameworks present several drawbacks which, together with their lower performance, make them less competitive with respect to the supervised ones, category in which this work falls.

Let $\mathscr{I} \subset \mathbb{R}^{3\times H\times W}$ be the space of RGB images and $\mathscr{D} \subset \mathbb{R}^{H\times W}$ be the space of corresponding depth maps, where H, W are respectively the height and width of the images, and assume the availability of a pairs training dataset $\mathscr{S} = \{(I_i, D_i)\}_{i=1}^{N}, I_i \in \mathscr{I}\ and\ D_i \in \mathscr{D}, \forall i \in [1, N]$, then the task is to learn a non-linear mapping $\Phi : \mathscr{I} \xrightarrow{\mathscr{S}} \mathscr{D}$.

Consider now a pairs test dataset $\mathscr{T} = \{(I_j, D_j)\}_{j=1}^{M}, I_j \in \mathscr{I}\ and\ D_j \in \mathscr{D}, \forall j \in [1, M]$ and let $D_j^* = \Phi(I_j) \in \mathscr{D}, \forall j \in [1, M]$ be the model prediction, T be the set of pixels and $d_i^{(*)} \in \mathbb{R}$ be the depth value of pixel $i \in T$ of $D^{(*)}$, then the most commonly used quantitative evaluation metrics, proposed by Eigen et al. [14], are Root Mean Square Error (RMSE, Equation 1), its

logarithmic version (RMSElog, Equation 2), thresholded accuracy (δ_t, Equation 3), Squared Relative difference (SqRel, Equation 4) and Absolute Relative difference (AbsRel, Equation 5), to which are sometimes added Absolute Difference (AbsDiff, Equation 6) and its logarithmic version (log10, Equation 7). The smaller the error, the better (Equations 1, 2, 4, 5, 6, 7); the higher the accuracy, the better (Equation 3).

$$\sqrt{\frac{1}{|T|}\sum_{i\in T}\left\|d_i - d_i^*\right\|^2} \tag{1}$$

$$\sqrt{\frac{1}{|T|}\sum_{i\in T}\left\|\log\left(d_i\right) - \log\left(d_i^*\right)\right\|^2} \tag{2}$$

$$\frac{1}{|T|}\left|\left\{\max\left(\frac{d_i}{d_i^*}, \frac{d_i^*}{d_i}\right) = \delta_i < 1.25^t\right\}_{i\in T}\right|,\ t \in \{1,2,3\} \tag{3}$$

$$\frac{1}{|T|}\sum_{i\in T}\frac{\left\|d_i - d_i^*\right\|^2}{d_i} \tag{4}$$

$$\frac{1}{|T|}\sum_{i\in T}\frac{\left|d_i - d_i^*\right|}{d_i} \tag{5}$$

$$\frac{1}{|T|}\sum_{i\in T}\left|d_i - d_i^*\right| \tag{6}$$

$$\frac{1}{|T|}\sum_{i\in T}\left|\log_{10}\left(d_i\right) - \log_{10}\left(d_i^*\right)\right| \tag{7}$$

In the following we also define the problem formulation and performance metrics for the semantic segmentation supervised task, because some of the relevant work considered is done on this related task and some of our proposed methods adopt it as a secondary task.

Let $\mathscr{C} \subset \mathbb{N}^{H\times W}$ be the space of semantic masks corresponding to $\mathscr{I}$, where each pixel belongs to one and only one class among $K \in \mathbb{N}$, and assume the availability of a pairs training dataset $\mathscr{S} = \{(I_i, C_i)\}_{i=1}^N, I_i \in \mathscr{I}\ and\ C_i \in \mathscr{C}, \forall i \in [1,N]$, then the task is to learn a non-linear mapping $\Psi : \mathscr{I} \xrightarrow{\mathscr{S}} \mathscr{C}$.

Consider now a pairs test dataset $\mathscr{T} = \left\{\left(I_j, C_j\right)\right\}_{j=1}^M, I_j \in \mathscr{I}\ and\ C_j \in \mathscr{C}, \forall j \in [1,M]$ and let $C_j^* = \Psi\left(I_j\right) \in \mathscr{C}, \forall j \in [1,M]$ be the model prediction, T be the set of pixels and $c_i^{(*)} \in \{0,\ldots,K-1\}$ be the class of pixel $i \in T$ of $C^{(*)}$, then the most commonly used quantitative evaluation metrics are mean Intersection over Union (mIoU, Equation 8), mean Accuracy (mAcc, Equation 9) and all Accuracy (aAcc, Equation 10); the higher they are, the better.

$$\frac{1}{K}\sum_{k=0}^{K-1}\frac{\left|\left\{c_i = k \wedge c_i^* = k\right\}_{i\in T}\right|}{\left|\left\{c_i = k \vee c_i^* = k\right\}_{i\in T}\right|} \tag{8}$$

$$\frac{1}{K}\sum_{k=0}^{K-1}\frac{\left|\left\{c_i = k \wedge c_i^* = k\right\}_{i\in T}\right|}{\left|\{c_i = k\}_{i\in T}\right|} \tag{9}$$

$$\frac{\left|\left\{c_i = c_i^*\right\}_{i\in T}\right|}{|T|} \tag{10}$$

More precisely, models predict a confidence distribution over the semantic classes $C^{**} \in [0,1]^{K\times H\times W}$, where $c_{i,k}^{**} \in [0,1]\,,\ \forall i \in T, k \in \{0,\ldots,K-1\}\ \textit{and}\ \sum_{k=0}^{K-1} c_{i,k}^{**} = 1, \forall i \in T$; C^* is derived from C^{**} by simply applying $c_i^* = \arg\max_{k\in\{0,\ldots,K-1\}} c_{i,k}^{**}$.

Convolutional Neural Networks Part I

1 Contour-filtered Laplacian pyramid

In this chapter we will see how we improved a convolutional decoder based on the Laplacian pyramid decomposition of the input image, by utilizing object contours obtained from instance segmentation maps. The Laplacian residual at each level of the pyramid is the result of a different band-pass filter on the RGB image and retains local spatial information that is inevitably lost in the encoding process due to the multiple downsamplings occurring in the typical convolutional backbone through striding or pooling. In general, this type of information can be significantly relevant to guide the prediction of depth residuals, because it allows to better model boundaries and edges, where changes in depth values are sharper and difficult to correctly estimate when simply upsampling the deep features produced by the encoder; however, it also contains high amounts of noise due not only to textured regions but also to imperceptible variations of color, brightness and gamma between neighbouring pixels, which can mislead the decoding and produce visual artifacts. To solve this issue, we propose to filter out this noise inside object boundaries and consider only the residual information located in correspondence of instance contours at each scale; this allows the decoder to focus on the resolution of depth ambiguities between the two sides of object boundaries, progressively adding finer details and smaller instances to the coarser scene representation modeled in the upper levels. Experiments on the official NYU Depth V2 dataset demonstrate that our method effectively reduces the undesired artifacts caused by the Laplacian pyramid, improves performance on most metrics with respect to the state-of-the-art literature, in particular with a reduction of AbsRel by 4.15%, and achieves comparable results on the remaining ones.

1.1 Introduction and background

As already presented in the Introduction, most modern methods proposed for supervised learning of the monocular depth estimation task rely on convolutional neural networks to extract hierarchical multi-scale deep features from the single RGB image in input, which are then used in a subsequent step to produce the corresponding (almost) full-resolution dense depth prediction map. Since the first work [14] introduced this setting, there has been a noticeable trend towards a widespread adoption of fully-convolutional architectures with an encoder-decoder structure, in which the encoding backbone is typically borrowed from state-of-the-art works in the object detection and recognition fields; moreover, to speed up convergence and improve sample efficiency, the backbone is usually initialized with the weights learnt during training on the original task (learning is transferred between related tasks in the form of pre-trained weights), while the downstream modules are randomly initialized.

1.1.1 Convolutional layers

Before reviewing the most recent and relevant works in this field, on which we build, it is necessary to describe the elementary block common to all of them: the convolutional layer. A convolutional layer is used to perform a series of 2D convolutions on the input tensor, using several different kernels whose parameters are learnable: consider a tensor of shape (C_{in}, H_{in}, W_{in}), where H_{in} is the height, W_{in} is the width and C_{in} is the number of channels (for example, in RGB images $C_{in} = 3$), and a block of $C_{out} \times C_{in}$ convolutional kernels of size (k_H, k_W) applied with stride (s_H, s_W) and padding (p_H, p_W), then the convolutional layer's parameters are weights of shape $(C_{out}, C_{in}, k_H, k_W)$ and biases of shape (C_{out}); the output of this layer will be of shape $(C_{out}, H_{out}, W_{out})$, $H_{out} = \left\lfloor \frac{H_{in}+2\times p_H - k_H}{s_H} + 1 \right\rfloor$ *and* $W_{out} = \left\lfloor \frac{W_{in}+2\times p_W - k_W}{s_W} + 1 \right\rfloor$, where each element $c \in C_{out}, h \in H_{out}, w \in W_{out}$ is the result of the total sum over the element-wise product between the c-th filter and a slice of the padded input tensor of shape (C_{in}, k_H, k_W) centered in the appropriate projection of (h, w) on the input tensor, plus the c-th bias. Each filter can be seen as a collection of C_{in} kernels, which are applied separately to the respective input channel and produce one version of channel each; these versions are then aggregated by summation (plus the bias) so that each filter outputs a single channel.

Convolutional layers are particularly relevant in the field of computer vision because of the way in which they explicitly leverage the peculiar characteristics of spatial information in 2D images, while immensely reducing the parameter space with respect to the previously used fully-connected layers, hence allowing the design of much deeper networks. In particular, fully-connected layers generate each output element by computing the weighted sum of all elements in the input tensor, which means that the number of learnable parameters scales linearly with the input and output resolution and can easily become intractable for even relatively low-resolution images and shallow networks. On the other hand, convolutional layers can be seen as a memory-efficient regularized version of this linear transformation:

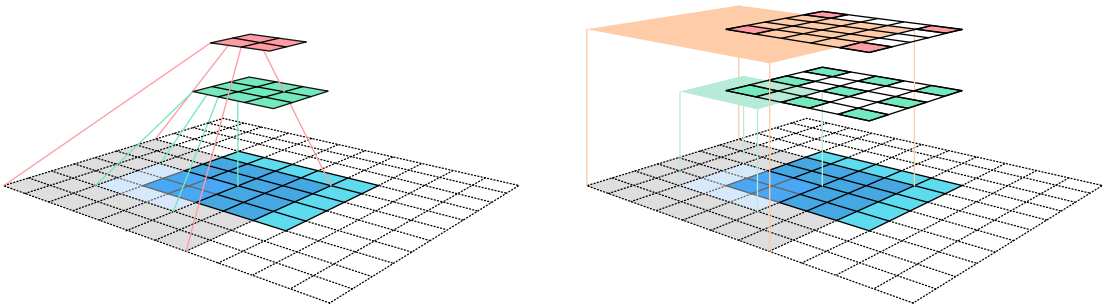


Figure 1.1: Alternative visualizations of the virtual receptive field of stacked convolutional layers, with $k_H = k_W = 3,\ s_H = s_W = 2,\ p_H = p_W = 1$.

each output element is computed as a weighted sum of only the input elements inside the kernel's receptive field (its center and the neighbouring elements); furthermore, the kernel weights and bias used in the sum are shared across all spatial locations (differently to 'locally-connected layers'), which means that this operation is translation invariant and the number of learnable parameters is constant (equal to the kernel size). This regularization relies on the assumption that useful kernels can be learnt to extract features that are general enough to be potentially present in any part of the input tensor from a consistently organized set of local inputs (the spatial relationship between any pair of neighbouring elements is always the same); the assumption is inspired by the connectivity pattern between cortical neurons in the animal visual cortex.

An important concept is the one regarding the receptive field of convolutional layers. When considering a convolutional layer in isolation, the receptive field of its kernels is defined by the kernel size (k_H, k_W): if a kernel is applied on a raw RGB image, it will not be able to extract high-level information, but will be constrained to recognize, for example, edges, gradients and some basic pattern; in fact, traditional methods presented in the Introduction commonly used fixed hand-crafted convolutional kernels to extract this type of low-level visual cues. If, instead, we consider a stack of convolutional layers, each layer applies its kernels on the output of the previous layer: this means that the receptive field with respect to the previous layer is still defined by the kernel size (k_H, k_W), but the virtual receptive field with respect to the raw RGB image in input is much larger and is defined by the kernel size and stride of all the preceding layers. The application of an example stack of two identical convolutional layers ($k_H = k_W = 3,\ s_H = s_W = 2,\ p_H = p_W = 1$) on a $H_{in} = W_{in} = 5$ input is visualized in Figure 1.1 in two alternative but equivalent ways. This property allows stacked layers to recognize higher-level information the deeper they are in the network, by exploiting the spatial relationship embedded in the lower-level information extracted by the upstream layers.

An advancement of convolutional layers based on their receptive field properties is the use of dilated kernels: a (d_H, d_W)-dilated convolutional layer with kernel size (k_H, k_W) can be seen as having a virtual kernel of size $(d_H \times (k_H - 1) + 1, d_W \times (k_W - 1) + 1)$, where only $k_H \times k_W$ elements are non-zero and learnable, leaving $d_H - 1$ zero elements on the vertical

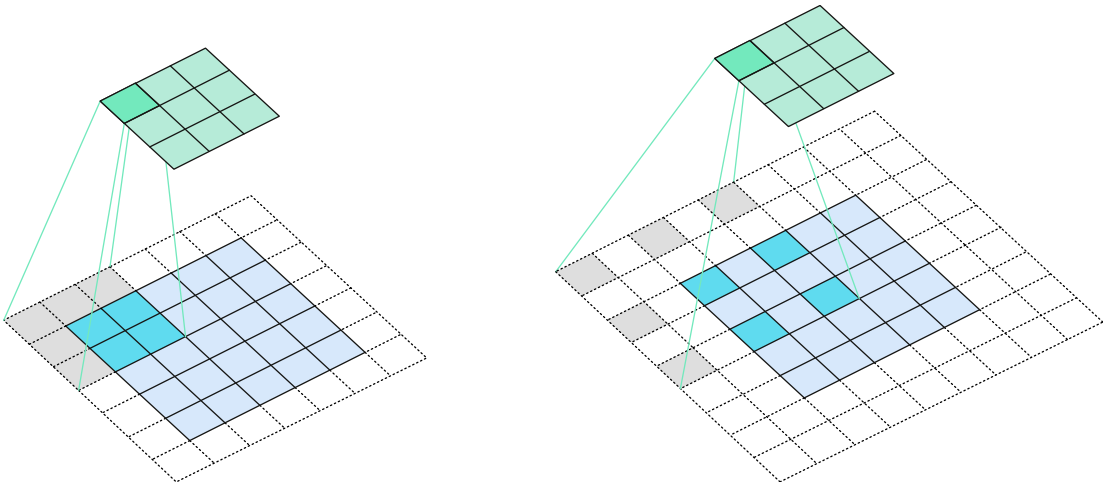
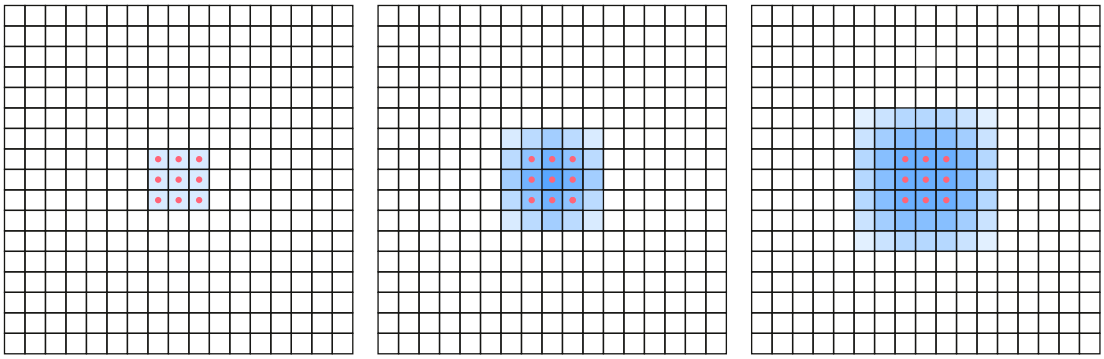
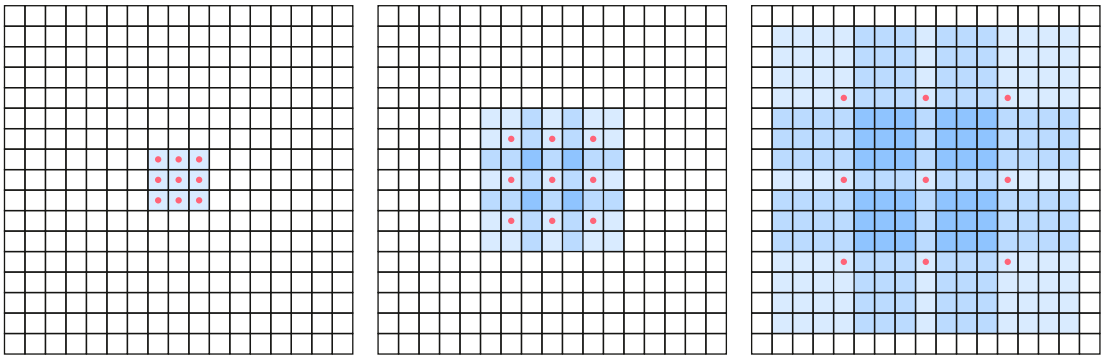


Figure 1.2: Comparison of convolutional kernels with dilation rates of $d_H = d_W = 1$ (left) and $d_H = d_W = 2$ (right).



(a) Fixed dilation rate $d_H = d_W = 1$



(b) Exponential dilation rates $d_H = d_W \in \{1, 2, 4\}$

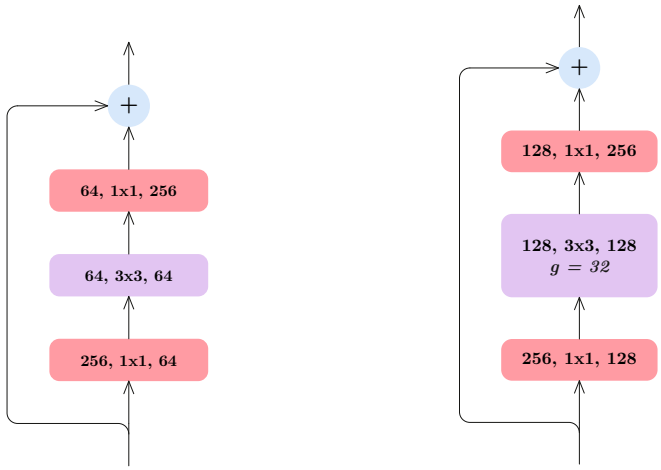
Figure 1.3: Comparison of the virtual receptive field of stacked convolutional layers with different dilation rates ($k_H = k_W = 3,\ s_H = s_W = 1$).

dimension and $d_W - 1$ zero elements on the horizontal dimension between each of them; in practice, the convolution operator is modified, rather than the kernel, to produce the output with $H_{out} = \left\lfloor \frac{H_{in}+2\times p_H - d_H \times (k_H - 1) - 1}{s_H} + 1 \right\rfloor$ *and* $W_{out} = \left\lfloor \frac{W_{in}+2\times p_W - d_W \times (k_W - 1) - 1}{s_W} + 1 \right\rfloor$. The example depicted in Figure 1.2 shows the conceptual difference between kernels of dense and dilated convolutions. While this variation seems to not follow the locality assumption, because immediately neighbouring input elements are not considered in favour of longer-range dependencies, when applied downstream with respect to other convolutional layers it has the effect of enlarging its virtual receptive field, by making it more sparsely populated (reducing the overlaps), possibly without ignoring any element of the upstream layers' output and of the raw RGB image in input (if the stack is correctly designed); the described behaviour is shown in Figure 1.3 and compared with the behaviour of a stack of dense convolutions.

The last variation of convolutional layers regards the connections between input and output channels: it is possible to avoid the interaction between subsets of channels by separating them into g groups. The filters are divided in groups of C_{out}/g so that each filter belongs to one and only one group and is a collection of C_{in}/g kernels; these kernels are then applied only on the slice of the padded input tensor of shape $\left(C_{in}/g, k_H, k_W\right)$ containing the input channels that belong to the respective group.

1.1.2 Residual Networks

The most commonly used encoding backbones are residual networks, the first version of which was proposed by He et al. [21] to address the difficulty in training deeper neural networks due to the degradation (of training accuracy) and vanishing gradient problems. In theory, inserting additional layers to a successfully trained architecture can only increase its performance (at least at training time), since in the worst case the added layers can learn the identity function; in practice, this behaviour did not emerge as convolutional layers seem to be unable to converge in reasonable time to an approximation of this state. They then proposed to explicitly redesign the layers to learn a residual mapping $\mathscr{F}(x) = \mathscr{H}(x) - x$ with respect to the layer input x rather than an unreferenced mapping $\mathscr{H}(x)$, because in the extreme case discussed above it is easier to optimize $\mathscr{F}(x) = 0$ than $\mathscr{H}(x) = x$; while the identity function optimality assumption might be too strong in real cases, the underlying mapping to be fit by the layer is intuitively closer to the identity than to the zero mapping. This novel mathematical formulation of the layers was implemented by using 'shortcut connections' skipping one or multiple layers and element-wise additions between their output and the input; when the dimensions of output and input do not match, a projection is applied to the shortcut connection to learn appropriate matching skip weights. The proposed network follows the VGG-nets [51] philosophy: residual blocks are sequentially divided in four stages, inside which spatial resolution and number of channels are constant and between which the spatial resolution is halved by convolutional layers with stride of 2 (hence without any intermediate pooling layer) and the number of channels is doubled.



(a) ResNet (b) ResNeXt

Figure 1.4: Comparison of residual blocks.

The second version was proposed by Xie et al. [68], inspired by the 'split-transform-merge' strategy of Inception models [55]: in an Inception block, the input is split in multiple lower-dimensional embeddings, each of which is transformed by distinct filters on separate parallel branches, whose intermediate results are then merged via concatenation; this architecture constrains the parameter space to a strict subspace of an equivalent single layer transforming a higher-dimensional embedding. However, the networks belonging to this family have been carefully designed for a specific task on a specific dataset, with a high number of different hyper-parameters for each transformation in each individual block. They then proposed to integrate the 'split-transform-merge' strategy in the previously designed residual blocks, while still following the design rules set in [51, 21]. This 'Network-in-Neuron' resulted in the use of a new 'cardinality' dimension defining the size of the set of multi-branch same-topology transformations inside a block, as opposed to the dimensions of width (number of channels) and depth (number of blocks); they set the template transformation to be the bottleneck from [21] and implemented the resulting residual blocks using grouped convolutions with cardinality g as described in subsection 1.1.1. The two versions of residual blocks are compared in Figure 1.4. They also showed that increasing cardinality yields better results than comparably increasing width or depth.

1.2 Related work

In this section, the most relevant works addressing the considered task with the use of convolutional neural networks are reviewed in depth, by emphasising the previous difficulties they intended to tackle, their proposed solutions and technical implementation, and their resulting strengths and weaknesses. A summary of the discussed and other related methods can be found in Table 1.1 for NYU Depth V2 [50] and Table 1.2 for KITTI [59] with 0-80m cap,

Table 1.1: Quantitative evaluations of related works on NYU Depth V2 [50].
The top part contains some early methods briefly presented in the Introduction; the bottom part contains the related work detailed in this chapter. For each metric, the best result is highlighted in bold and the second-best is underlined; the data reported are provided by the respective papers.

Method	**Frames**	**AbsRel** ↓	**log10** ↓	δ_1 ↑	δ_2 ↑	δ_3 ↑	**RMSE** ↓	**RMSElog** ↓
Eigen et al. [14]	120k	0.215	-	0.611	0.887	0.971	0.907	0.285
Liu et al. [33]	795	0.213	0.087	0.650	0.906	0.976	0.759	-
Laina et al. [30]	12k	0.127	0.055	0.811	0.953	0.988	0.573	0.195
Fu et al. [15]	120k	0.115	0.051	0.828	0.965	0.992	0.509	-
Chen et al. [9]	12k	0.138	-	0.826	0.964	0.990	0.496	0.174
Wang et al. [63]	50k	0.115	0.049	0.871	0.975	0.993	0.519	-
Kim et al. [28]	24k	0.111	0.047	0.878	0.981	0.995	0.388	-
Lee et al. [31]	24k	0.110	0.047	0.885	0.978	0.994	0.392	-
Ye et al. [72]	795	-	0.063	0.784	0.948	0.986	0.474	**0.081**
Chen et al. [8]	50k	0.111	0.048	0.878	0.977	0.994	0.514	-
Liu et al. [35]	50k	0.113	0.048	0.878	0.978	0.995	0.504	-
Yin et al. [73]	29k	0.108	0.048	0.875	0.976	0.994	0.416	-
Bhat et al. [3]	50k	**0.103**	**0.044**	**0.903**	**0.984**	**0.997**	**0.364**	-
Song et al. [52]	36k	0.110	0.047	0.885	0.979	0.995	0.393	-

with a comparison of their performance and of the size of their training set. For each metric, the best result is highlighted in bold and the second-best is underlined; the data reported are provided by the respective papers.

1.2.1 Chen et al. [9]

The work by Chen et al. [9] is based on [15]: they considered the task as an ordinal regression problem with depth ranges spacing-increasingly discretized and replaced the convolutional layers in the last stages of the encoding backbone (ResNet) with dilated convolutional layers to extend their receptive field while maintaining the spatial resolution, hence avoiding the over-downsampling typical of classification networks. Differently from [15], they argued that an ASPP module is not an adequate decoder to capture multi-scale features, because the discretized and predefined dilation rates are not able to extract continuous context information and are consequently prone to produce grid artifacts; instead, they proposed a context aggregation module: it consists in two parallel branches, one performing global average pooling to obtain the image-level context and the other performing self-attention to obtain pixel-level context in the form of long-range similarities, whose output features are concatenated (the global feature vector is copied to restore its spatial resolution) and used to produce the final estimates. The relevance of the proposed decoder is supported by studies which showed that depth maps can be decomposed into piece-wise smooth segments bounded by object contours, hence the ability to leverage the intra- and inter- object long-range context is fundamental to accurately estimate depth. Furthermore, they proposed a solution to the stepped artifacts and discretization error issues caused by a hard-threshold-based generation

Table 1.2: Quantitative evaluations of related works on KITTI [59] with 0 - 80m cap. The top part contains some early methods briefly presented in the Introduction; the bottom part contains the related work detailed in this chapter. For each metric, the best result is highlighted in bold and the second-best is underlined; the data reported are provided by the respective papers.

Method	**Frames**	**AbsRel** ↓	**SqRel** ↓	δ_1 ↑	δ_2 ↑	δ_3 ↑	**RMSE** ↓	**RMSElog** ↓
Eigen et al. [14]	20k	0.190	1.515	0.692	0.899	0.967	7.156	0.270
Liu et al. [33]	700	0.217	-	0.656	0.881	0.958	7.046	-
Fu et al. [15]	23k	0.072	0.307	0.932	0.984	0.994	2.727	0.120
Chen et al. [9]	22k	0.083	0.437	0.919	0.982	0.995	3.599	0.127
Wang et al. [63]	40k	0.096	0.655	0.893	0.963	0.983	4.327	0.171
Kim et al. [28]	23k	0.060	0.231	0.958	0.993	**0.999**	2.650	0.094
Lee et al. [31]	23k	<u>0.059</u>	0.245	0.956	0.993	<u>0.998</u>	2.756	0.096
Ye et al. [72]	23k	0.112	-	0.842	0.947	0.973	4.978	0.210
Liu et al. [35]	22k	0.071	0.306	0.933	0.983	0.995	2.848	0.121
Yin et al. [73]	23k	0.072	-	0.938	0.990	<u>0.998</u>	3.258	0.117
Bhat et al. [3]	26k	**0.058**	**0.190**	**0.964**	**0.995**	**0.999**	**2.360**	**0.088**
Song et al. [52]	23k	<u>0.059</u>	<u>0.212</u>	<u>0.962</u>	<u>0.994</u>	**0.999**	<u>2.446</u>	<u>0.091</u>

of the final prediction, by fully exploiting the confidence of the estimated depth distribution in the computation of continuous depth values; to improve the consistency between the produced attention map and the ground truth depth distribution, a KL divergence term between the two is also added to the loss function.

1.2.2 Wang et al. [63]

The approach adopted by Wang et al. [63] to address the problems of loss of local details in the encoder, discontinuous kernel of dilated convolutions and their inability to capture relationships between faraway pixels is fundamentally different. They proposed to keep the residual-based encoding backbone unmodified and to obtain the high-resolution spatial information through skip connections, concatenating shallower generalized appearance features to the deep specialized semantic features at the output of the decoder. They also proposed a bottleneck channel-spatial attention module consisting in two parallel branches, one performing channel-wise self-attention to adaptively module the contribution of each channel and one performing spatial self-attention, whose output features are element-wise summed and passed to a dense decoding module with cascaded dilated convolutions (as opposed to the parallel dilated convolutions in ASPP modules), which gradually recovers the feature resolution through dense upsampling while considering multi-scale information with a wider and wider receptive field. Moreover, they introduced a distance-aware regression loss to improve prediction of far objects, together with a gradient term to better model edges and a term comparing the surface normals to enhance planar structure understanding.

1.2.3 Kim et al. [28]

Inspired by studies which showed that CNNs can learn to leverage monocular visual cues and semantic features to infer the depth map of a scene, Kim et al. [28] focused on the extraction of multi-scale contextual information from every stage of the encoding. They proposed the use of attentive skip connections, in which the intermediate encoded features go through three parallel branches to highlight the significant regions and control the flow of information: one performing spatial attention through two cascaded ASPP modules, one performing channel-wise attention through global average pooling and a shallow multi-layer perceptron, and one acting as a residual connection; the three outputs are element-wise summed and delivered to the decoding stage of the corresponding resolution. The proposed decoder is a mirrored version of the residual-based encoder, in which convolutional layers are replaced by transposed convolutional (or 'deconvolutional') layers to gradually increase spatial resolution instead of reducing it. Furthermore, they proposed a bottleneck global context module to understand the image-level context from the deepest features, through a branch performing channel-wise average pooling and a branch performing spatial max pooling followed by a shallow multi-layer perceptron; the two output features are element-wise summed and used as the input of the decoder.

1.2.4 Lee et al. [31]

Similarly to [15], Lee et al. [31] replaced the convolutional layers in the last stages of the residual-based encoding backbone with dilated convolutions and used an ASPP module with five different dilation rates as a dense contextual feature extractor at the bottleneck. Differently, they added downstream a multi-scale decoder with skip connections coming from the respective stage of the encoder and proposed to insert at every decoding stage a local planar guidance layer: it learns suitable features to estimate 4-dimensional local plane coefficients (separately, the 3D unit normal vector and the distance between the plane and origin) and uses them under the local planar assumption to compute the ray-plane intersection and consequently upsample the intermediate depth estimate. The output of each local planar guidance layer has then full resolution and is densely passed both to the following decoding stage (after the appropriate downsampling) and through a shortcut to the final prediction layer, where they are combined to reach a coarse-to-fine consensus.

1.2.5 Ye et al. [72]

To address with a different approach the issue of loss of spatial information in the encoding as the cause of unsatisfactory prediction of depth details, Ye et al. [72] proposed a dual stream network with a contextual stream extracting high-level lower-resolution features and a spatial stream preserving the higher-resolution information. The contextual stream consists in an encoder-decoder structure, with the convolutional layers in the last stages of the residual-based backbone replaced with dilated convolutional layers; the decoder contains a dual

attention module: one branch performs channel-wise self-attention and the other branch performs nonlocal spatial self-attention, in which the global pairwise similarity is replaced by a nonlocal affinity computed between each pixel and its neighbours in a window around it, to reduce the computational complexity of the operation; the two outputs are summed and passed to an ASPP module to capture multi-scale information and infer a low-resolution depth map. In the spatial stream, the image edge content is isolated by an high-pass filter and shortly processed to focus on low-level features. The heterogeneous features, both in terms of semantic content and resolution, generated by the two streams are integrated in a refinement module that hierarchically upsamples the low-resolution intermediate depth estimate of the contextual stream through stacked residual transposed convolutional blocks and then fuses them together by concatenating them and employing a global-pooling-based channel attention as feature selection and combination. Furthermore, they proposed to use a deep-supervision loss function, by applying it to both the low- and high- resolution depth maps produced at different stages by the network.

1.2.6 Chen et al. [8]

Chen et al. [8] pointed out that previous methods could not accurately and simultaneously predict depth for large-scale geometrical regions (e.g., walls, floor, ceiling) and local highly-detailed regions with edges and small objects; they argued that while multi-scale features were commonly extracted and gradually fused together, the underlying scene structure was not explicitly taken in consideration. They proposed an encoder-decoder architecture with a residual-based backbone and skip connections towards the adaptive dense feature fusion module at the bottleneck: for each scale, all encoded features are passed to a scale-specific multi-scale feature fusion module that resizes them to the resolution of its scale and refines them with residual blocks to adaptively select the most relevant features for the considered scale from every scale; this is crucial, because features at different scales not only differ in resolution, but most importantly in the semantic information they represent, naturally forming a hierarchy of constraints between pixel depths. The output of this module is a fused feature pyramid in which the upper levels contain the global layout and the lower levels contain local details, which are used in the residual pyramid decoder to progressively predict higher- and higher- resolution depth maps in a coarse-to-fine way, by refining at every stage the previous lower-resolution prediction with the addition of structure details. They also adopted the use of deep-supervision on the predicted depth maps at all scales.

1.2.7 Liu et al. [35]

Liu et al. [35] followed [8] in addressing the challenge of precisely predicting depth of objects widely varying in size in complex scenes. They proposed an encoder-decoder architecture with a residual-based backbone and a decoder of attention-based feature distillation blocks to fuse the features coming from the previous decoding stage with the intermediate features at the corresponding resolution coming from the skip connections and enhance the

feature discriminative modulation during upsampling. The proposed residual decoding block consists in a multi-step distillation and progressive refinement module to enrich the feature representation through residual asymmetric convolutions robust to rotational distortions, followed by a joint attention aggregation module with a branch performing channel attention through global average pooling and a shallow multi-layer perceptron and a branch performing spatial contrast aware attention. Furthermore, they introduced a method to enhance the structural effectiveness of the combination of loss terms commonly used during training (pixel-wise, gradient, normals and SSIM): the estimated depth map is iteratively decomposed into sub-band images by a discrete wavelet transform at different frequencies and each loss term is applied on a predefined subset of them to better model horizontal edges, vertical edges, and corners and reduce the loss of these high-frequency details.

1.2.8 Yin et al. [73]

Yin et al. [73] focused on the development of a novel loss term, with the goal of demonstrating the importance of high-order geometric constraints in the 3D space. While several previous works used some geometric constrained loss, almost all of them were locally extracted from a small neighbourhood and did not leverage the global structure of the scene; since the supervision ground truth depth maps of the most widespread datasets are obtained through the 3D sensors presented in the Introduction, the depth values of neighbouring pixels can fluctuate significantly, introducing considerable noise in pixel-wise and locally computed metrics. To address this issue they introduced the concept of 'virtual norm', a stable 3D geometric constraint modelling long-range structural relationships between pixels: given the depth map predicted by a generic network, the corresponding point cloud is generated as an intermediate representation and triplets of non-collinear points at various ranges are repeatedly randomly sampled to form a virtual plane with its corresponding virtual norm vector; the same process is performed on the ground truth depth map using the same triplets and the direction divergence between corresponding virtual norms is used as the loss. This approach also allows to obtain a large number of constraints (larger than the number of pixels in the full-resolution depth map), but can become cumbersome if too many virtual planes are needed to faithfully represent the underlying structure with too large sampling ranges.

1.2.9 Bhat et al. [3]

Bhat et al. [3] analyzed that previous methods employed a common structure with an encoder, a bottleneck module and a decoder in sequence, where the bottleneck often consisted in some attention mechanism performing global processing to model dependencies between feature regions; however, they argued that attention computation is much more effective when performed at high resolution and proposed to change the components ordering to a sequence of encoder, decoder and final attention. They proposed an attention module taking in input the decoded features, performing a global statistical analysis on them and consequently modifying the output depth distribution to match the ground truth, since different input

images can correspond to very different depth ranges and distributions. In particular, the proposed module is based on a visual transformer, which is fed with a patch embedding of the high-resolution decoded features and learns to predict a vector of depth-bin-widths (after a multi-layer perceptron head) and a set of 1×1 convolutional kernels; a normalization operator causes a competition among bins and pressures the transformer to focus on the most probable depth sub-intervals to adaptively model interesting regions in the specific input image. Applying the kernels and a softmax function on the decoded features produces the probability distribution of each pixel over the possible bins, which is aggregated in a single value by the corresponding weighted linear combination of estimated bin-centers; to enforce the learnt bins to follow the ground truth distribution, a bi-directional Chamfer loss term is also introduced.

1.3 Method

Our method is based on the work by Song et al. [52]. They argued that previous methods did not manage to fully exploit the underlying properties of encoded features in the decoding phase; this caused a difficulty in solving multi-scale ambiguities and accurately predicting depth boundaries at various scales with sufficient detail and without blurring artifacts. They then proposed a depth-residual decoder based on the Laplacian pyramid decomposition of the input image, to retain the higher-resolution spatial information lost in the repeated downsampling during the feature extraction in the encoder. While their contribution is highly valuable, as also proved by quantitative and qualitative evaluations, we claim that not all the information carried by Laplacian residuals is relevant to the task: on one hand, crucial information like object boundaries and edges is correctly captured at all scales, but, on the other hand, a significant amount of noisy information is also retained in all image regions, even in absence of rich texture or gradient. Furthermore, their approach of pixel-wise summing at all scales the predicted depth residual with the channel-wise average of the Laplacian residual causes the appearance of visual anomalies we termed 'Laplacian artifacts' both at and inside object boundaries. Taking inspiration from their approach and from the well-studied strong relationship between depth estimation and semantic segmentation, we explore in this work several decoder variations based on the use of instance contours with the aim of fully leveraging the relevant information, while ignoring the uninformative noise, which also has the effect of reducing the artifacts.

In this section, we first describe the overall architecture in common with [52] (illustrated in Figure 1.5), then we present our proposed decoder variations highlighting the main differences, and lastly define the loss function used during the training.

1.3.1 Overall architecture

Following the most common approach in previous works, the employed network consists in an encoder-decoder structure. The encoding backbone can vary, but we decide to use

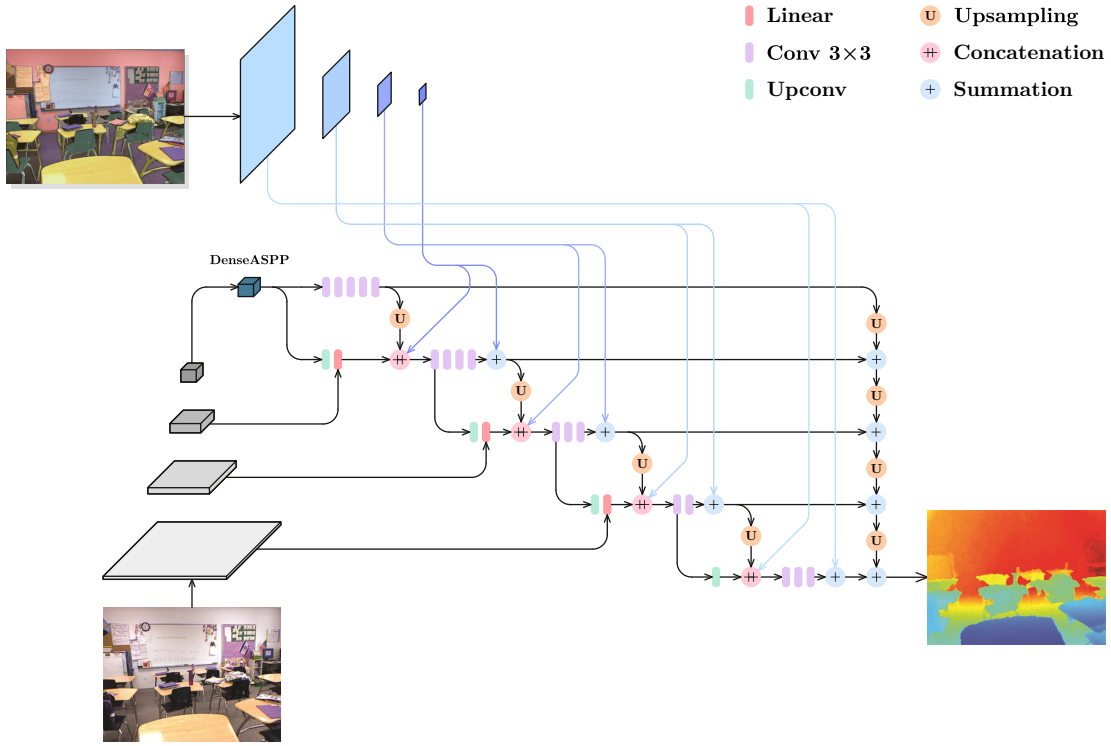


Figure 1.5: Overall architecture of $L_k \otimes C_k$: the decoding of the feature pyramid produced by the encoder is guided by the contour-filtered Laplacian pyramid to predict depth residuals and progressively combine them in a coarse-to-fine flow.

the ResNeXt-101 [68] explained in subsection 1.1.2. To avoid over-downsampling of the extracted features to 1/32 of the original resolution, the 4th encoder stage is replaced with a DenseASPP [71] with cascaded $\{3,6,12,18\}$-dilated convolutions, each of which takes in input all previous features and produces dense outputs at 1/16 of the original resolution. The outputs of the encoder stages at $\{1/2, 1/4, 1/8\}$ of the original resolution are also delivered to the decoder through lateral skip connections, while the bottleneck output is fed to the decoder as its initial input, containing the deepest and highest-level contextual information condensed in the color-depth embedding space.

The decoder proposed in [52] is divided in five branches, among which the upmost one predicts the global-scale coarse depth map of 1/16 of the original resolution solely from the bottleneck output features; the lower three branches receive the upsampled latent features and predicted depth map from the upper branch, and the Laplacian residual and skip connection at the respective resolution and predict progressively higher-resolution depth residuals; the lowest branch operates at full resolution, hence it does not have any corresponding skip connection. The outputs of the five branches are iteratively upsampled and combined through element-wise summation to gradually recover local-scale details in a coarse-to-fine flow. Notably, all convolution operations in the decoder are performed by pre-activation convolutional layers [22] with weight-standardization [42], to avoid the vanishing gradient problem when

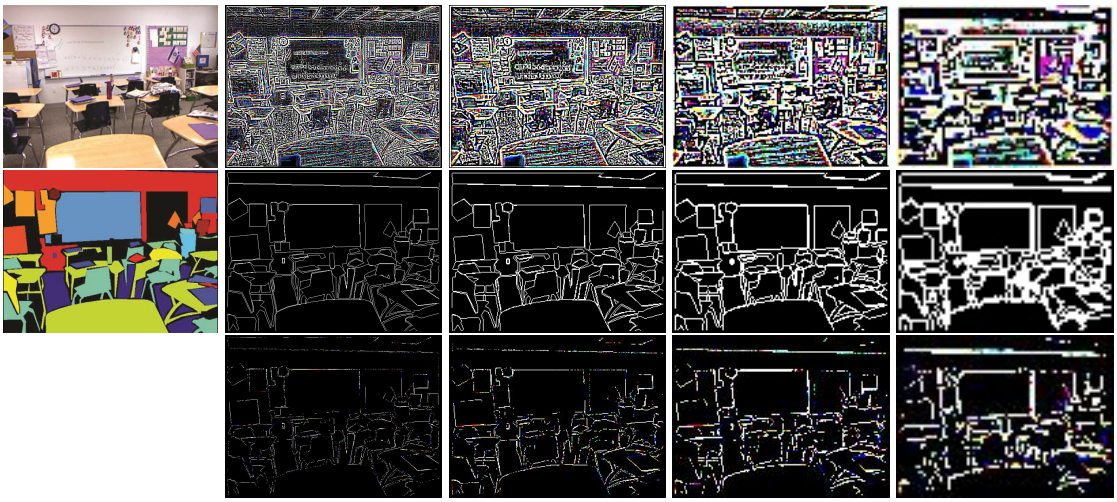


Figure 1.6: Top to bottom: Laplacian pyramid residuals [52], segmentation contour pyramid, our proposed filtering.

estimating sparse depth residual potentially containing both positive and negative values, but mostly close to zero.

1.3.2 Decoder variants

Denoting the input RGB image I_1, the bilinear upsampling ($\times 2$) $Up_B(\cdot)$ and downsampling ($\div 2$) $Down_B(\cdot)$, the five-level Gaussian pyramid is constructed as $I_k = Down_B(I_{k-1}),\ k \in \{2,3,4,5\}$, where I_k is a low-pass filtered image of I_{k-1}, hence also of I_1; from it, the five-level Laplacian pyramid is derived as $L_k = I_k - Up_B(I_{k+1}) = I_k - Up_B(Down_B(I_k)),\ k \in \{1,2,3,4\}$, where L_k is a high-pass filtered image of I_k, hence a band-pass filtered image of I_1. Denoting the corresponding instance segmentation map S_1, the nearest downsampling ($\div 2$) $Down_N(\cdot)$ and the contour extraction operator $Con(\cdot)$, the four-level binary 'contour pyramid' is computed as $C_k = Con(S_k),\ S_{k+1} = Down_N(S_k),\ k \in \{1,2,3,4\}$. A pixel is considered to be part of a contour if it belongs to a valid semantic class (not to the 'void' 0-class described in the Introduction) and at least one out of the eight neighbouring pixels belongs to a different class, or if at least one out of the eight neighbouring pixels belongs to the same class but to a different instance.

I First, we simply tried to replace the 3-channel Laplacian residual L_k with the single-channel contour map C_k. While this approach completely removes all the undesirable noise, it only provides the decoder information regarding the object boundaries at various scales, but not information regarding the loss of spatial detail due to the downsampling occurring during feature encoding.

II Second, we tried to provide the decoder both types of information, by simply concatenating them in the 4-channel $L_k +\!\!+ C_k$. With this approach we wanted to test the decoder

ability to autonomously leverage the two different types of guidance at various scales in distinguishing between uninformative noise and useful information and only retain the latter.

III Finally, we tried to explicitly filter out the misleading noise by computing the 3-channel element-wise product $L_k \otimes C_k$. This operation can be explained intuitively by the assumption that most spatial details relevant to the task of depth estimation are lost at and not inside the object boundaries; while this is not true in general, it models the image-depth distribution with reasonable approximation (as pointed out by Chen et al. [9] and discussed in subsection 1.2.1) and represents a favorable trade-off between noise reduction and information loss. The derivation of the contour-filtered Laplacian pyramid is shown in Figure 1.6.

1.3.3 Loss function

Following common practice, the loss function L_t used to optimize the network learnable parameters during training contains multiple terms, each enforcing a different constraint; in particular, we adopt a two-terms loss function

$$L_t\left(D, D^*\right) = \alpha L_d\left(D, D^*\right) + \beta L_g\left(D, D^*\right), \tag{1.1}$$

where $D, D^* \in \mathscr{D} \subset \mathbb{R}^{H \times W}$ are respectively the ground truth and predicted depth maps as defined in the Introduction. The loss terms weights are set to $\alpha = 10$ and $\beta = 0.1$ as in [52].

I Eigen et al. [14] introduced the scale-invariant error in three equivalent forms

$$E_d\left(D, D^*\right) = \frac{1}{|T|} \sum_{i \in T} \left(\log d_i^* - \log d_i - \frac{1}{|T|} \sum_{j \in T} \left(\log d_j^* - \log d_j \right) \right)^2 \tag{1.2}$$

$$= \frac{1}{|T|^2} \sum_{i,j \in T^2} \left(\left(\log d_i^* - \log d_j^* \right) - \left(\log d_i - \log d_j \right) \right)^2 \tag{1.3}$$

$$= \frac{1}{|T|} \sum_{i \in T} \left(\log d_i^* - \log d_i \right)^2 - \frac{1}{|T|^2} \left(\sum_{i \in T} \left(\log d_i^* - \log d_i \right) \right)^2, \tag{1.4}$$

where T is the set of pixels and $d_i^{(*)} \in \mathbb{R}$ is the depth value of pixel $i \in T$ of $D^{(*)}$ as defined in the Introduction. Equation 1.2 corresponds to the variance of $\log d_i^* - \log d_i$; Equation 1.3 computes the depth difference between pairs of pixels in the predicted map and explicitly compares it with the corresponding difference in the ground truth; Equation 1.4 relates E_d to the Mean Squared Error and is used to derive the scale-invariant loss

$$L_d\left(D, D^*\right) = \sqrt{\frac{1}{|T|} \sum_{i \in T} \left(\log d_i^* - \log d_i \right)^2 - \frac{\lambda}{|T|^2} \left(\sum_{i \in T} \left(\log d_i^* - \log d_i \right) \right)^2}, \tag{1.5}$$

where $(1-\lambda) \in [0,1]$ is the weight of the squared mean of the error in log space: when $\lambda = 0$ L_d corresponds to the root of MSE, when $\lambda = 1$ it corresponds to $\sqrt{E_d}$. We adopt the squared root and $\lambda = 0.85$ from [31] to put more weight on the minimization of the variance.

II Eigen and Fergus [13] introduced the gradient loss term

$$L_g\left(D, D^*\right) = \frac{1}{|T|} \sum_{i \in T} \left(\left| \nabla_x d_i^* - \nabla_x d_i \right| + \left| \nabla_y d_i^* - \nabla_y d_i \right| \right), \tag{1.6}$$

where $\nabla_x d_i^{(*)}$ and $\nabla_y d_i^{(*)}$ are respectively the horizontal and vertical gradient values in pixel $i \in T$ of $D^{(*)}$.

1.4 Experiments

In this section we compare the performance of our proposed method with previous works on the challenging NYU Depth V2 [50], thanks to the presence of densely labelled maps for semantic and instance segmentation in its official dataset.

1.4.1 Implementation details and training settings

The Pytorch [41] implementation is based on [52]. The encoding backbone corresponds to the first 3 stages (without 4th stage and fully connected layer) of the ResNeXt-101 [68], with the parameters initialized as the original network pre-trained on ILSVRC [45] for image classification. The first three convolutional blocks and all the batch normalization layers in the encoder are frozen. The number of parameters is 58M in the encoder and 15M in the decoder. The network training is 35 epochs long with a batch size of 16. The optimizer used is AdamW [37], with weight decay set to 10^{-2} and 0 respectively for the encoder and the decoder; the learning rate starts from 10^{-4} and decreases polynomially with power 0.5 to 10^{-5}.

During training, standard data augmentation is performed to increase learning robustness: the samples are rotated by a random angle in the range $[-2.5, 2.5]°$ (through bilinear resampling for the RGB image and nearest resampling for the ground truth and contours images), center cropped to 427×565, randomly cropped to 416×544, horizontally flipped with 0.5 probability, their brightness is multiplied by a random factor in the range $[0.75, 1.25]$ with 0.5 probability, their gamma is multiplied by a random factor in the range $[0.8, 1.2]$ with 0.5 probability, and their color channels are separately multiplied by a random factor in the range $[0.8, 1.2]$ and randomly permuted with 0.5 probability.

1.4.2 Performance evaluation

To ensure a fair comparison, in Table 1.3 we report only the performance of methods trained on the 795 frames of the official dataset of NYU Depth V2 [50]. In particular, the paper

Table 1.3: Quantitative evaluations on NYU Depth V2 [50] with 795 training frames. The performance of [72] is provided by their paper; the performance of [52] and of our proposed variations (ordered as in subsection 1.3.2) corresponds to the best one obtained over multiple training runs. For each metric, the best result is highlighted in bold and the second-best is underlined; the relative percentage difference between our $L_k \otimes C_k$ and [52] is also included in the last row.

Method	**AbsDiff** ↓	**AbsRel** ↓	**log10** ↓	δ_1 ↑	δ_2 ↑	δ_3 ↑	**RMSE** ↓	**RMSElog** ↓
Ye et al. [72]	-	-	0.063	0.784	0.948	0.986	**0.474**	**0.081**
Song et al. [52] (L_k)	**0.360**	<u>0.147</u>	<u>0.060</u>	<u>0.809</u>	<u>0.965</u>	0.992	<u>0.481</u>	0.175
Ours (C_k)	0.378	0.153	0.063	0.793	0.959	0.992	0.510	0.183
Ours ($L_k +\!\!+ C_k$)	0.369	0.149	0.062	0.802	0.962	<u>0.992</u>	0.498	0.180
Ours ($L_k \otimes C_k$)	<u>0.360</u>	**0.141**	**0.060**	**0.815**	**0.965**	**0.993**	0.485	<u>0.174</u>
$L_k \otimes C_k$ vs L_k	+0.08%	-4.15%	-0.66%	+0.69%	+0.06%	+0.08%	+0.91%	-0.69%

by Ye et al. [72] compares their proposed method with other works (not reported here) using the same training dataset and achieves the best result among them in all metrics, hence we use it as a literature baseline. To be able to compare our proposed variations with the original method by Song et al. [52], we train several times their model on the official dataset with the default settings provided in the published code and report the best result. Table 1.3 contains all three variations proposed in subsection 1.3.2 and the relative percentage difference between the best performing one ($L_k \otimes C_k$) and the original method [52] (L_k). For each metric, the best result is highlighted in bold and the second-best is underlined.

First of all, we note that the method by Song et al. [52] outperforms the baseline in every metric except for the Root Mean Square Error (both linear and logarithmic versions) and that in Tables 1.1, 1.2 it achieves highly competitive results in every metric when trained with more frames, making it part of the current state of the art. Analyzing the proposed variations in order, the one only using the contours C_k is the worst-performing among the three; this is probably due to the complete loss of residual information with respect to the Laplacian pyramid, which does not allow the decoder to restore the finer spatial details during upsampling. The one concatenating Laplacian residuals and contours $L_k +\!\!+ C_k$ performs significantly better, but still worse than the original decoder [52] (L_k), while the expectation could be to have comparable results; the reason may lie in the pixel-wise addition computed after the prediction of each depth residual, which in this case also involves the corresponding contour map besides the Laplacian residual. This interaction could also be a further reason for the low performance of the first variation. Finally, the desired behaviour is obtained with the third decoder design, which successfully reduces the uninformative information present in the Laplacian residuals without losing most of the desired and useful features contained in it, by filtering out any appearance residual located inside rather than at the object boundaries ($L_k \otimes C_k$); this decoder overall outperforms the original design: we improve all considered metrics except for AbsDiff (on which we obtain comparable results) and RMSE, in particular achieving a significant reduction of AbsRel by 4.15%. Furthermore, the performance increase is confirmed by the average of each metric over several training runs, on which the proposed decoder reaches larger improvement margins over all metrics (AbsDiff and RMSE included), proving to be more

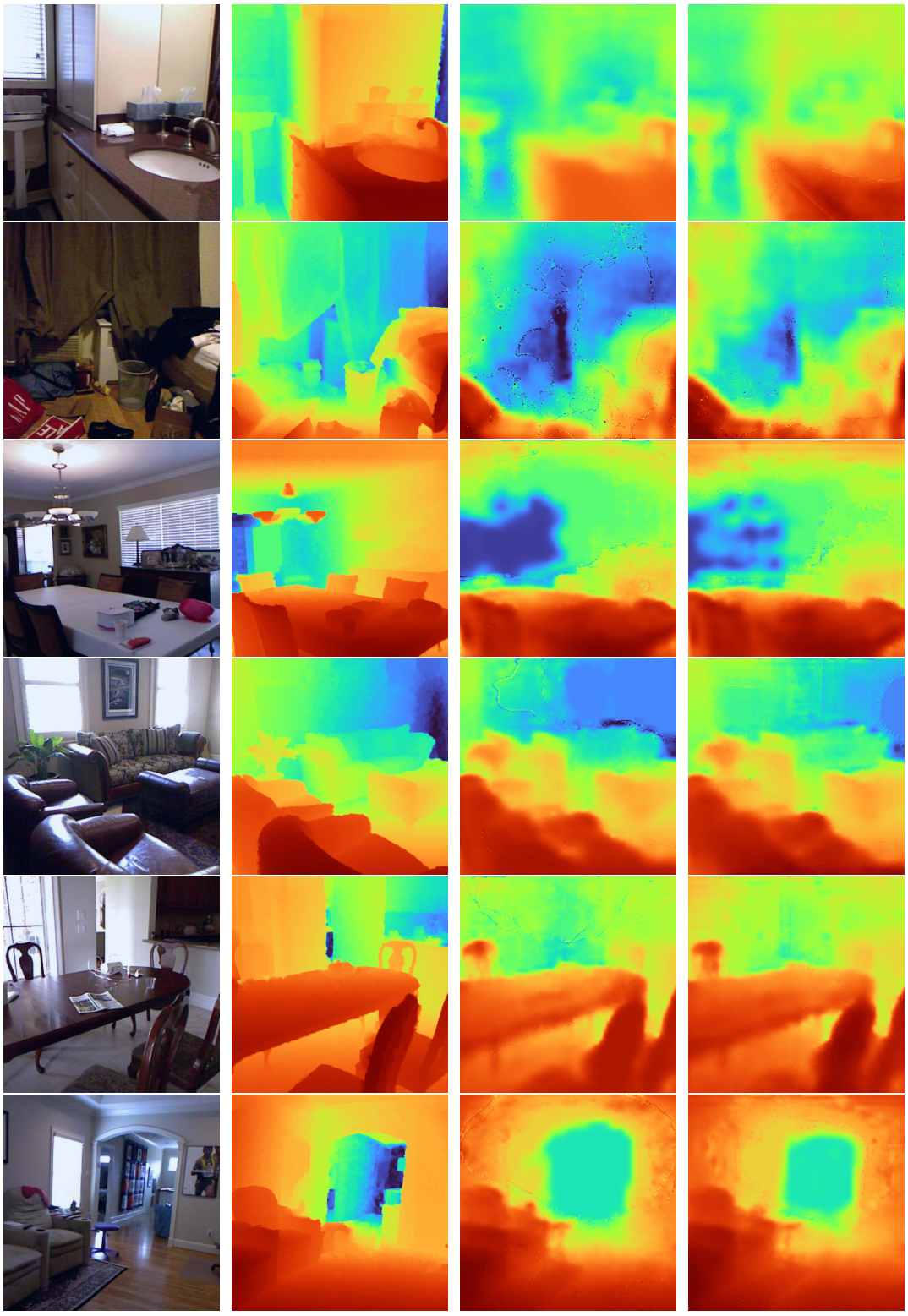
reliable and consistent.

Figure 1.7 shows a comparison of the qualitative results obtained with the method by Song et al. [52] (L_k) and our proposed variant $L_k \otimes C_k$ with respect to the ground truth: red represents a shorter depth, while blue represents a longer depth; all results are scaled according to the minimum and maximum values in the corresponding ground truth map for visualization purposes, hence the same color could correspond to different depth values in different scenes. It is possible to see that, in general, the proposed method produces depth maps more similar to the ground truth, not only considering the distance of objects from the camera, but also the distance between objects. For example, in the bathroom scene (first row), all objects are correctly predicted to be slightly closer to the camera than what output by the previous method; in the home office scene (fourth row), instead, the armchairs in the foreground are similarly predicted, while our method recognizes that the sofa in the background and the wall behind it are closer to them. Furthermore, in most scenes, but especially in the bedroom one (second row), the Laplacian artifacts are considerably reduced. Similar conclusions can be drawn if comparing the error maps, as shown in Figure 1.8: intense red represents overestimated depths, while intense blue represents underestimated depths, hence a whiter error map corresponds to a more accurate prediction. For example, in the living room scene (sixth row), the arch outline is much less visible in the map corresponding to the output of our variant, confirming that its shape has been better recognised.

1.5 Conclusions

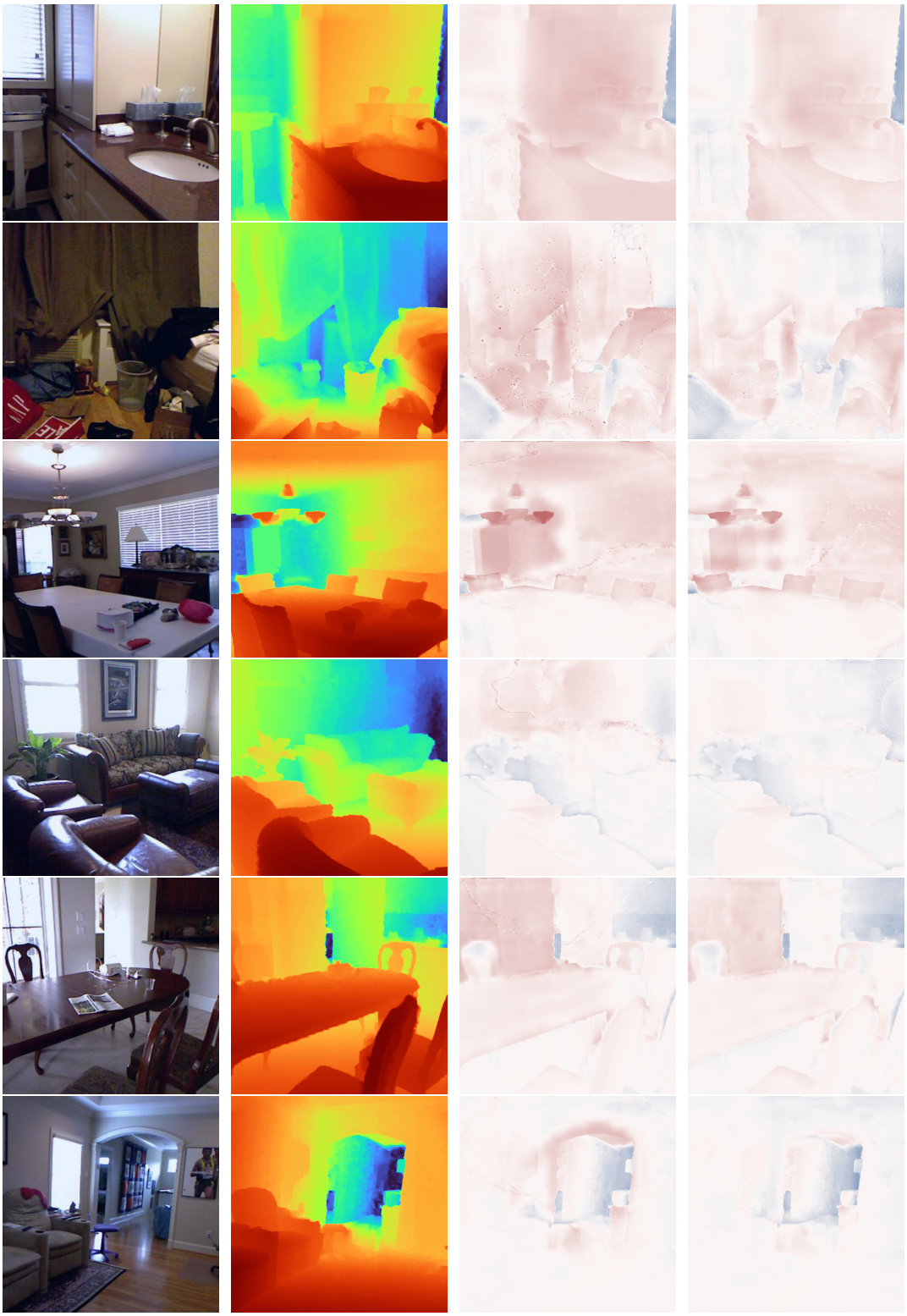
In this part, we reasoned about the relevance of the information carried by Laplacian residuals in the decoding of deep features for the monocular depth estimation task and proposed to substitute, integrate or filter it with the contours extracted from the corresponding instance segmentation mask. The results of experiments on the NYU Depth V2 dataset revealed that Laplacian residuals are more informative than contours alone and that a direct use of the latter as a replacement of the former or as additional information jointly delivered to the decoder reduces performance with respect to the original method. However, our intuition regarding the importance of Laplacian residuals mostly at the object boundaries and the potentially misleading information present inside object segments was empirically demonstrated: if the contours are used to exclude the small uninformative photometric variations acting as residual noise, the decoder is able to focus on the resolution of multi-scale boundary ambiguities, reducing the appearance of visual artifacts and producing more detailed and less blurred depth residuals, hence improving performance.

This usage of low-level residuals to help the decoder restore spatial details lost during the feature encoding mainly addresses the issue related to the over-downsampling of feature maps typical of convolutional backbones, which is needed to effectively model global image-level relationships through network layers with limited receptive fields. While the DenseASPP module at the bottleneck allows to produce dense features by significantly enlarging the



(a) RGB (b) GT (c) Song et al. [52] (L_k) (d) Ours $(L_k \otimes C_k)$

Figure 1.7: Qualitative comparison of depth estimation results on NYU Depth V2 [50].



(a) RGB (b) GT (c) Song et al. [52] (L_k) (d) Ours $(L_k \otimes C_k)$

Figure 1.8: Qualitative comparison of error maps on NYU Depth V2 [50].

virtual receptive field of its convolutional kernels through cascaded dilations of increasing rate, it is applied to features of higher resolution than the standard encoder output and does not perform global operations relating all pixels in the semantically high-level deep feature maps. On one hand, this approach avoids the reduction of feature resolution and further allows to retain more local details in the encoding backbone, which is desirable in dense prediction tasks requiring to restore the estimate to the full resolution of the input image; on the other hand, it does not fully exploit the image-level information as many other works do by explicitly modelling long-range dependencies on a global scale, typically through attention mechanisms.

Vision Transformers Part II

2 Multitask shared-attention encoding

In this chapter we will explore in depth the mechanism that has recently emerged in the computer vision field as a possible replacement of convolutional layers: self-attention; in particular, we consider it not from a stand-alone perspective but as a fundamental module in the transformer blocks ported from the NLP field. To this end, following the successful results obtained in Part I with the use of information derived from segmentation masks as a guidance for depth, we propose a multitask setting with joint learning of depth estimation and semantic segmentation of the same single input image. We here focus on the study of the encoding of multiple streams related to multiple tasks for effective independent decoding and dense prediction, by adopting a transformer-based columnar encoder and a double-head convolutional decoder. Self-attention is by construction very different from convolutions, since it is characterized by a global instead of local scope and by a content- instead of parameter-based interaction between input elements. We exploit the peculiar structure causing this latter property to design several different custom attention-sharing mechanisms; the resulting encoder consists in task-specific streams running in parallel and interacting with each other only by sharing a single attention map, computed by only one of the two or jointly by both, which identically controls the two information flows. Moreover, we further merge the two streams in a single one to encode task-invariant features, given the high correlation between the two tasks. Experiments on the official NYU Depth V2 dataset validate our proposed designs, all outperforming both the depth estimation and semantic segmentation baselines; in particular, the depth estimation task benefits from an increasing merging of the two streams, obtaining better results when both are involved in the attention computation and even more when a single stream encodes identical features for both decoder heads, achieving significant improvements on all metrics.

2.1 Introduction and background

From the previous works presented in section 1.2, as also analyzed by Bhat et al. [3], it is possible to notice a growing trend in the use of attention mechanisms in fully-convolutional architectures for monocular depth estimation and more in general for dense prediction tasks. The reason is that convolutional layers are intrinsically local operators, heavily relying on the spatial organization of information and on the consistent relationship between neighbouring pixels to learn translation-invariant filters capable of recognizing low-level patterns in the shallow layers and higher- and higher- level patterns in the deeper layers; regardless of the progressive increase of the virtual receptive field obtained by stacking more and more layers, convolutions lack by construction the ability to relate faraway spatial locations and to adapt their kernels to each specific input image. This lack of non-local interactions in convolutions and the dominant success of self-attention in the field of Natural Language Processing with Transformers [61] inspired many works to try integrating attention in CNN architectures, as we have seen, and some works [43, 62] to try completely replacing convolutional layers with pure self-attention layers, hence obtaining fully-attentional networks.

The breakthrough use of transformers in the vision field is due to the work by Dosovitskiy et al. [12], which first introduced the Vision Transformer, and transformer-based networks represent now the state of the art in most vision tasks, from image classification to dense prediction tasks like semantic segmentation. Focusing on the latter (similar in essence to depth estimation, but much more studied, hence already showing the very recent overtake of transformers on convolutions), the typical architecture structure is maintained from the fully-convolutional networks, with an encoder used to extract deep features from the RGB image in input and a decoder used to produce the corresponding dense prediction map at high resolution, possibly with the addition of bottleneck or post-processing modules; in particular, the general approach is to employ transformer-based encoding backbones borrowed from state-of-the-art works in the object detection and recognition fields with the corresponding pre-trained weights and to maintain convolution-based decoders.

2.1.1 Transformer blocks

Before reviewing the few previous works employing transformer-based networks for vision tasks, we describe here their building block: the transformer, illustrated in Figure 2.1. A basic transformer consists in the sequence of two main components: in order, a multi-head self-attention module and a multi-layer perceptron; it is important to mention that transformers operate on an 1D unordered set of tokens, hence some explicit positional encoding is needed (in NLP as in 2D vision tasks) to be able to successively reassemble the output tokens in a consistent way with respect to the input organization. Consider a tensor x of shape (C, H, W), where H is the height, W is the width and C is the number of channels, it is first reshaped to (L, C), where $L = H \times W$ is the number of tokens, and linearly projected to obtain the triplet

q, k, v, where q is the query, k is the key and v is the value, each of shape $\left(L, C_{qkv}\right)$

$$q = xW_q^T + b_q,\ k = xW_k^T + b_k,\ v = xW_v^T + b_v, \tag{2.1}$$

where W_q, W_k, W_v are weights of shape $\left(C_{qkv}, C\right)$ and b_q, b_k, b_v are bias of shape $\left(C_{qkv}\right)$. Query and key are the embedded representations of x used to compute the similarity between each pair of tokens $\left(i, j\right) \in L^2$, where i plays the role of the query and j plays the role of the key

$$S = qk^T, \text{ where } S_{ij} = q_i \left(k_j\right)^T; \tag{2.2}$$

note that the similarity matrix S is in general not symmetrical, because the query and key embeddings of the same token are obtained separately through different linear transformations. The similarity matrix S is then transformed via a row-wise softmax in the attention matrix A

$$A_i = \text{softmax}\left(\frac{S_i}{\sqrt{C_{qkv}}}\right),\ \forall i \in L, \tag{2.3}$$

to ensure that $A_{ij} \geq 0,\ \forall \left(i, j\right) \in L^2$ and that $\sum_{j \in L} A_{ij} = 1,\ \forall i \in L$; the division by $\sqrt{C_{qkv}}$ is done to avoid the softmax function reaching small-gradient regions for too large elements of the dot product S when C_{qkv} is large. The attention is then used to recompute every token $i \in L$ as a weighted average of the values of all tokens

$$x' = Av, \text{ where } x'_i = A_i v. \tag{2.4}$$

The information contained in the input x then flows to the output x' through the values v, with the attention A controlling this flow according to the feature similarity of tokens. This described process corresponds to a single head of the module and is usually repeated in M parallel branches, each operating on $C_{qkv} = C/M$ channels and whose outputs are concatenated in the channel dimension and merged together through a linear projection

$$x'' = \left[x'_{(m)}\right]_{m \in M} W_o^T + b_o, \tag{2.5}$$

where $[\cdot]$ is the channel-wise concatenation operator, W_o are weights of shape (C, C), b_o are bias of shape (C) and x'' has shape (L, C). The concept behind multi-head attention modules resembles grouped convolutions and the ResNeXt [68] residual block described in subsection 1.1.2, since it allows to separately relate tokens in different representation subspaces. The output x'' of the module MHA is then element-wise summed with the input x through a shortcut

$$y = x + x'' = x + MHA(x). \tag{2.6}$$

Each token in the resulting set is fed separately to a shared shallow multi-layer perceptron consisting in two linear transformations with an activation layer in between

$$y' = \text{act}\left(yW_1^T + b_1\right) W_2^T + b_2, \tag{2.7}$$

where W_1 has shape (C_h, C), b_1 has shape (C_h), W_2 has shape (C, C_h) and b_2 has shape (C). The output y' of MLP is then element-wise summed with y through a shortcut

$$z = y + y' = y + MLP\left(y\right). \tag{2.8}$$

In practice, a normalization layer is applied before both components

$$z = x + MHA\left(\text{norm}\left(x\right)\right) + MLP\left(\text{norm}\left(x + MHA\left(\text{norm}\left(x\right)\right)\right)\right). \tag{2.9}$$

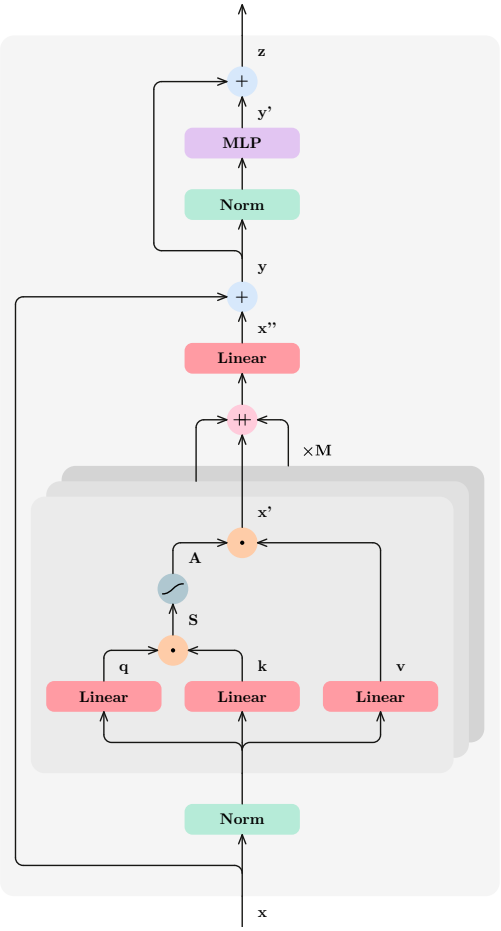


Figure 2.1: Transformer block.

As mentioned previously, the self-attention module operates on a global level, relating every pair of tokens in the input tensor regardless of their distance, hence can be thought as having an infinitely large receptive field. Furthermore, the linear projections used to compute q, k, v, to merge the features $x'_{(m)}$ extracted by multiple heads into x'', and inside the multi-layer perceptron are "local" and translation-invariant operations involving each token separately, making the parameter-less self-attention the only point in which tokens interact with one another.

The main issue in porting the transformer block from the NLP field to vision tasks is that self-attention is quadratic in complexity with respect to the number of tokens and the images in input are of much higher resolution than the sentence representations of textual inputs. The naive approach of flattening a RGB image of shape $(3, H_{RGB}, W_{RGB})$ into a set of pixel-tokens of size $H_{RGB} \times W_{RGB}$ would then result intractable both in terms of time and memory required. The first approach to solve this problem was to divide the image in non-overlapping patches of shape $\left(p_H, p_W\right)$ to obtain a set of tokens of shape $\left(L, 3 \times p_H \times p_W\right)$, where $L = H_{RGB}/p_H \times W_{RGB}/p_W = H \times W$, and to embed them in C dimensions through a linear projection. Alternatively, the input RGB image can be fed to a convolutional backbone to produce feature maps of shape $(C_{CNN}, H_{CNN}, W_{CNN})$, where $H \leq H_{CNN} \leq H_{RGB}$, $W \leq W_{CNN} \leq W_{RGB}$, which if needed can then undergo the patch embedding process described above. In both cases, this part of the model in charge of compression and flattening is called 'stem'.

Finally, we introduced before the set-to-set nature of transformers and the need of encoding positional (in the case of vision, spatial) information into the patch-embedded tokens to provide the network some insight about their mutual relationships. This is mainly done in two ways:

A) Absolute positional embedding:

   a) L C-dimensional parameters are learnt and added to the corresponding token;
   b) W $C/2$-dimensional parameters are learnt as X-embeddings for the horizontal dimension and H $C/2$-dimensional parameters are learnt as Y-embeddings for the vertical dimension, they are then channel-wise concatenated according to the coordinates of each token to obtain L C-dimensional parameters, which are added to the corresponding token.

   Moreover, this type of embeddings can be integrated at various levels in the network:

   a) The position parameters can be learnt once and added only to the output of the stem;
   b) The shared position parameters can be learnt once and added to the tokens before each transformer block;
   c) The position parameters can be separately learnt and added to the tokens before each transformer block.

B) Relative positional embedding: $(2H-1)\times(2W-1)$ 1-dimensional parameters are learnt for each attention head to represent the positional relationship between a generic query token and all possible key tokens; for each token $i \in L$, the corresponding slice of shape (H, W) is extracted from them to match its position in the input tokenized image and used as a bias term B_i (flattened to shape (L)) in the attention, changing Equation 2.3 into

$$A_i = \text{softmax}\left(\frac{S_i}{\sqrt{C_{qkv}}} + B_i\right), \ \forall i \in L. \tag{2.10}$$

While convolutional layers and transformer blocks are independent from their input size, be it a 2D image of shape (C, H, W) or a 1D set of tokens of shape (L, C), the positional embeddings learn and carry resolution-specific information; to circumvent this constraint, it is possible to generalize them to any resolution by using an appropriate interpolation.

2.2 Related work

In this section, some recent works employing transformer-based networks in the vision field are presented. Due to the lack of a large number of relevant works addressing the monocular depth estimation task, we also include the most important ones from related tasks. A comparison of the semantic segmentation performance of some related methods can be

Table 2.1: Semantic segmentation quantitative evaluations of related works on ADE20K [79]. The best results are highlighted in bold; the data reported are provided by the respective papers.

Method	mIoU ↑	aAcc ↑
Ranftl et al. [44]	0.4902	0.8311
Zheng et al. [78]	**0.5028**	**0.8346**

found in Table 2.1 for ADE20K [79]; the data reported are provided by the respective papers and the best results are highlighted in bold.

2.2.1 Dosovitskiy et al. [12]

Inspired by the success of transformers in the NLP field and by the increasing number of works trying to integrate their self-attention in the fully-convolutional framework that was dominating the vision field, Dosovitskiy et al. [12] introduced an extremely simple yet effective architecture. They proposed to employ the standard transformer blocks by Vaswani et al. [61] out of the box for image classification; the resulting network consisted in a stack of identical transformer encoders operating at fixed resolution and on a fixed number of channels. It was preceded by a basic patch embedding stem without convolutions to obtain the set of tokens, to which was added a 1D absolute positional embedding (type A.aa, as described in subsection 2.1.1). Furthermore, they proposed to prepend a learnable global context token to the flattened embedded patches, which attends to all spatial locations without any positional bias and is used at the last layer as the image-level representation: a shallow multi-layer perceptron constitutes the classification head, which is fed with this extra '[class]' embedding to produce the final confidence distribution. It is important to report that the alternative stem with a convolutional backbone and the positional embedding variants described in subsection 2.1.1 were also all analyzed in this work.

2.2.2 Ranftl et al. [44]

Ranftl et al. [44] proposed to follow the typical encoder-decoder structure for dense prediction, employing the network by Dosovitskiy et al. [12] as encoding backbone and focusing on the development of a convolutional decoder, as usually happens in architectural research. In particular, they adopted the hybrid version of the vision transformer, which produces patch embeddings of 1/16 of the original resolution by applying the first three stages of a residual-based backbone and considering each resulting pixel feature as a token. Even if their considered tasks do not explicitly require it for classification purposes, they decide to retain the additional '[class]' token, since it provides a global image representation. Their proposed decoder consists in the reassembling of intermediate tokens from several encoding stages at different resolution and their progressive fusion to generate the refined dense prediction. In particular, in each skip connection, tokens are reshaped to a 2D representation, individually combined with the global token either through summation or channel-wise concatenation,

and spatially resampled to a specific resolution; the approach adopted is to downsample the deeper features and to upsample the shallower ones to mimic the output of typical convolutional networks. Finally, the obtained feature maps are gradually refined through residual convolutional blocks, upsampled, summed and passed to a shallow convolutional head for prediction.

2.3 Method

Our method is built starting from the work by Zheng et al. [78], which concurrently applied concepts similar to [44], but for semantic segmentation. We first port their proposed model to the monocular depth estimation task and use the two as a draft to implement our proposed variations. Our goal is to jointly train a single model for both tasks to validate the intuition that object segments (image regions consisting of pixels belonging to them) are better characterized by the corresponding depth values and their changes, rather than by their low-level visual appearance; we believe that leveraging this strong relationship between the two types of information and respective tasks can be beneficial for both, as also partially demonstrated in Part I. To this end, while still maintaining the main focus on the monocular depth estimation task, we catch the opportunity provided by the promising transformer-based encoders recently introduced to propose several variations of their structure to satisfy our needs and better analyze their inner working. In particular, we recognize that the peculiar structure of the transformer blocks, of which the encoder is made, allows interesting modifications previously not fitting in the fully-convolutional framework.

In this section, we first describe the overall architecture with the decoder proposed by [78] and the adjustments needed to match our problem, then we present our proposed encoder variations, and lastly we discuss the adopted loss function and some possible alternatives in the form of additional terms.

2.3.1 Overall architecture

The employed network consists in an encoder-decoder structure. For the encoder, we adopt ViT-Large [12], which stacks $N = 24$ transformer blocks, each performing self-attention with $M = 16$ heads; the hidden layer of the MLP in each block has channel size $C_h = 4C$. Due to the quadratic complexity with respect to the token sequence length, as discussed in subsection 2.1.1, and considering that typically CNN encoders for semantic segmentation produce outputs of 1/16 of the original resolution, a basic patch embedding is performed with square patches of size $p_H = p_W = 16$ and with channel size $C = 1024$ (hence $C_h = 4096$); following [12], a 1D absolute positional encoding is only learnt and added to the stem output (type A.aa, as described in subsection 2.1.1), but the additional '[class]' token is not kept. The encoding backbone processes the tokens at fixed resolution and with a fixed number of channels.

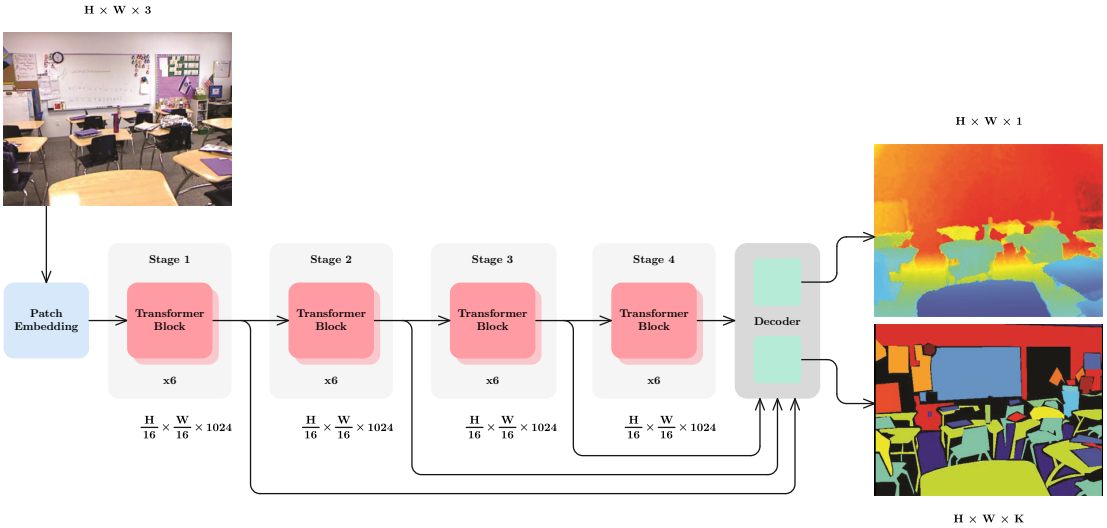


Figure 2.2: Overall architecture of METR: a single-stream vision transformer is used to extract task-invariant feature representations, which are shared by the two task-specific decoder heads as their identical input. DEPTR and SETR [78] have a single decoder head. A_{se}, A_{de}, A_{me} have a quasi-two-stream encoder with three different custom attention mechanisms.

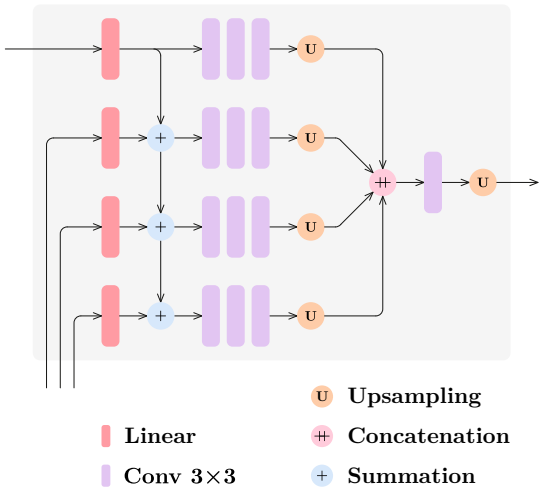


Figure 2.3: MLA decoder head.

The decoder employed is the Multi-Level feature Aggregation (MLA, Figure 2.3) proposed by Zheng et al. [78], which takes in input the intermediate tokens produced by the $\{6, 12, 18, 24\}$-th transformer block. These features are separately reshaped in a 2D representation, refined though a linear transformation reducing the number of channels to 256, aggregated in a top-down fashion via element-wise summation and separately fed to parallel 3-layer convolutional branches with channel size of 128; the feature maps are then upsampled $\times 4$ to $1/4$ of the original resolution, channel-wise concatenated and passed to the final prediction layer. Since our setup involves two different tasks, we instantiate two separate decoder heads, one predicting a single channel followed by a sigmoid for the depth estimation task and one predicting K channels followed by a softmax for the semantic segmentation task.

2.3.2 Encoder variants

Differently from the typical fully-convolutional network, in which the flow of information follows a single processing stream of stacked convolutional layers (plus possibly some short-

cuts or skip connections), in transformer-based networks the embedded features are split in three different representation in each block to apply the attention mechanism; in particular, as explained in subsection 2.1.1, the information flows through the value, but it is controlled by the pair-wise interaction of tokens through their query and key. This allows the possibility to customize the way attention is computed and, consequently, the way information is gated or flows.

We then propose four different architectures of the transformer encoder for our multi-task learning and, more specifically, four different variants of the attention mechanism performed in each transformer block of the encoder. They can be divided in two categories, since the first three consist in a quasi-two-stream encoder with custom attention, while the remaining one consists in a single-stream encoder with vanilla self-attention as in [12, 44, 78].

I A segmentation-guided quasi-two-stream vision transformer is used to extract two task-specific encoded feature representations. Patch and positional embeddings are computed from the input image and shared by the semantic segmentation and depth estimation task-specific streams as their identical input, respectively $x_{se}^{(0)} = x_{de}^{(0)}$. In each block $n \in N$, query q_{se}, key k_{se} and value v_{se} representations are computed from the semantic segmentation stream $x_{se}^{(n-1)}$ as described in Equation 2.1 and a value representation $v_{de} = x_{de}^{(n-1)} W_{de}^{T} + b_{de}$ is separately computed from the depth estimation stream; the attention matrices A_{se} are then computed using q_{se} and k_{se} as described in Equations 2.2, 2.3 and the new tokens are produced as

$$x'^{(n-1)}_{se} = A_{se} v_{se},\ x'^{(n-1)}_{de} = A_{se} v_{de}. \tag{2.11}$$

The semantic segmentation stream is then performing self-attention and the depth estimation stream is being guided in such a way that its new tokens are produced as averages with the weights dependent on their pair-wise similarity in the semantic embedding space; intuitively, this means that tokens corresponding to the same class will be considered more likely to have similar depth values, while tokens corresponding to different classes will be considered less likely to have similar depth values, hence helping to define object boundaries. The described shared attention is shown in Figure 2.4a. The two output streams of the multi-head attention are summed with the respective input stream and delivered to two separate multi-layer perceptrons with side-shortcut. The task-specific output features $x_{se}^{(N)}$, $x_{de}^{(N)}$ of this transformer encoder are then fed to the corresponding task-specific decoder head.

II A segmentation-regularized quasi-two-stream vision transformer is used to extract two task-specific encoded feature representations. Patch and positional embeddings are computed from the input image and shared by the semantic segmentation and depth estimation task-specific streams as their identical input, respectively $x_{se}^{(0)} = x_{de}^{(0)}$. In each block $n \in N$, query q_{de}, key k_{de} and value v_{de} representations are computed from the depth estimation stream $x_{de}^{(n-1)}$ as described in Equation 2.1 and a value represen-

tation $v_{se} = x_{se}^{(n-1)} W_{se}^T + b_{se}$ is separately computed from the semantic segmentation stream; the attention matrices A_{de} are then computed using q_{de} and k_{de} as described in Equations 2.2, 2.3 and the new tokens are produced as

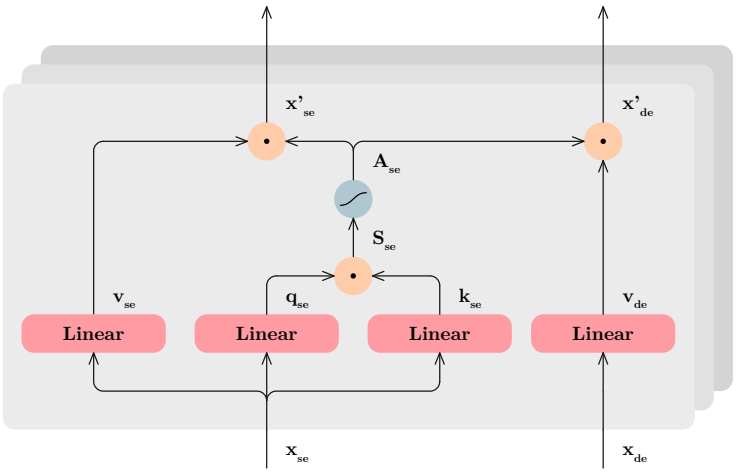
$$x'^{(n-1)}_{se} = A_{de} v_{se},\ x'^{(n-1)}_{de} = A_{de} v_{de}. \tag{2.12}$$

The depth estimation stream is then performing self-attention and the new tokens of the semantic segmentation stream are produced as averages with the weights dependent on their pair-wise similarity in the depth embedding space; intuitively, this means that the depth estimation stream must additionally learn to generate tokens carrying semantically consistent information so that the computed attention is relevant for the secondary task too. The described shared attention is shown in Figure 2.4b. The two output streams of the multi-head attention are summed with the respective input stream and delivered to two separate multi-layer perceptrons with side-shortcut. The task-specific output features $x_{se}^{(N)},\ x_{de}^{(N)}$ of this transformer encoder are then fed to the corresponding task-specific decoder head.

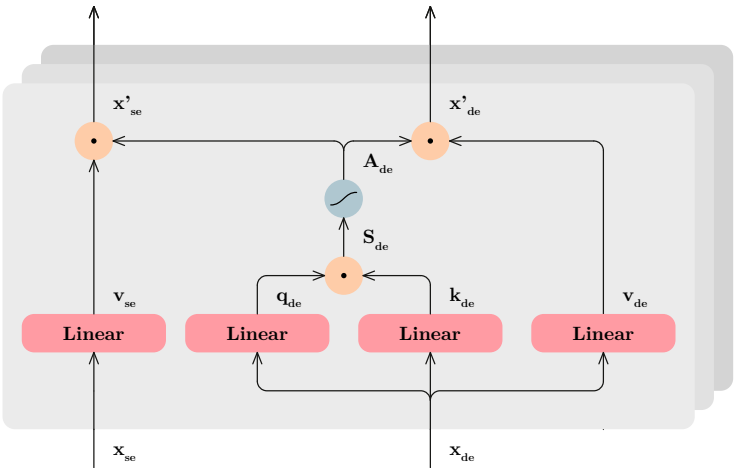
III A merged-attention quasi-two-stream vision transformer is used to extract two task-specific encoded feature representations. Patch and positional embeddings are computed from the input image and shared by the semantic segmentation and depth estimation task-specific streams as their identical input, respectively $x_{se}^{(0)} = x_{de}^{(0)}$. In each block $n \in N$, value representations $v_{se},\ v_{de}$ are separately computed from both streams as described in Equation 2.1, and query q_{me} and key k_{me} representations are computed from the channel-wise concatenation $x_{se}^{(n-1)} +\!\!+ x_{de}^{(n-1)}$ of the two streams (in this case, the weights $W_{q,me},\ W_{k,me}$ of the linear projections defined in Equation 2.1 are of shape $\left(C_{qkv}, 2C\right)$); the attention matrices A_{me} are then computed using q_{me} and k_{me} as described in Equations 2.2, 2.3 and the new tokens are produced as

$$x'^{(n-1)}_{se} = A_{me} v_{se},\ x'^{(n-1)}_{de} = A_{me} v_{de}. \tag{2.13}$$

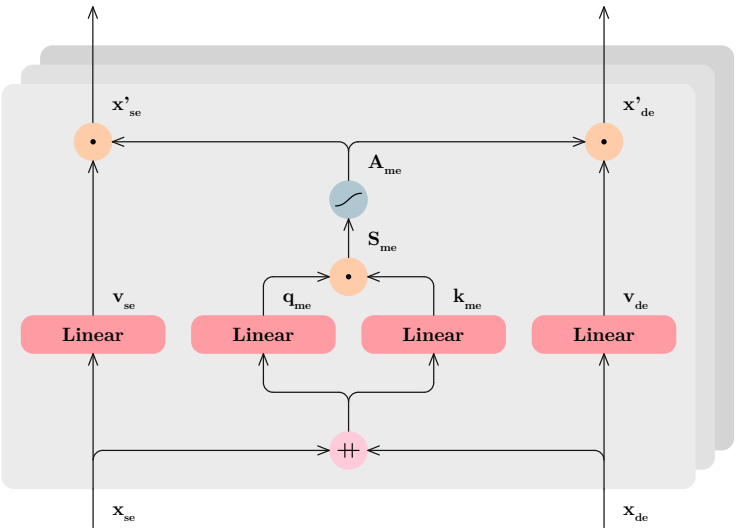
The new tokens are produced as averages with the weights dependent on their pair-wise similarity in both embedding spaces, since the merging of the two streams generates a joint-attention mechanism. In particular, each transformer block and each attention head in it is allowed to independently give different importance to either stream or even to each of their feature maps; intuitively, this means that the two streams are able to adaptively influence each other and could potentially provide more useful information at different stages in the model. This design is therefore a general case of the previous two, because the linear projections generating $q_{me},\ k_{me}$ could completely ignore one of the streams and focus only on the other. The described shared attention is shown in Figure 2.4c. The two output streams of the multi-head attention are summed with the respective input stream and delivered to two separate multi-layer perceptrons with side-shortcut. The task-specific output features $x_{se}^{(N)},\ x_{de}^{(N)}$ of this transformer encoder are then fed to the corresponding task-specific decoder head.



(a) A_{se}



(b) A_{de}



(c) A_{me}

Figure 2.4: Custom shared attention for multi-task quasi-two-stream encoders.

IV A vanilla vision transformer is used to extract encoded feature representations. Patch and positional embeddings are computed from the input image and fed to the stack of transformer blocks; in each block, the query, key and value representations are all computed from the single input stream as described in Equation 2.1 and from them the attention matrices are obtained and the new tokens are produced as global weighted averages; the single output stream of the multi-head self-attention is summed with the single input stream and delivered to a single multi-layer perceptron with side-shortcut. The output features of this transformer encoder are then shared by the two task-specific decoder heads as their identical input: this encourages the encoder to produce task-invariant features useful for both related tasks and acts as a form of regularization to improve the robustness of the learnt mapping to the high-level embedding space. The resulting architecture (METR) is shown in Figure 2.2.

2.3.3 Loss function

Due to the multi-task nature of our setting, the loss function L_t used to optimize the network learnable parameters during training contains multiple terms

$$\begin{aligned} L_t\left(D, D^*, C, C^{**}\right) &= \delta L_{main}\left(D, D^*\right) + \sigma L_{sec}\left(C, C^{**}\right) \\ &= \delta\left(\alpha L_d\left(D, D^*\right) + \beta L_g\left(D, D^*\right)\right) + \sigma L_{sec}\left(C, C^{**}\right) \end{aligned} \tag{2.14}$$

where $D, D^* \in \mathscr{D} \subset \mathbb{R}^{H \times W}$ are respectively the ground truth and predicted depth maps, $C \in \mathscr{C} \subset \mathbb{N}^{H \times W}$ is the ground truth segmentation mask and $C^{**} \in [0,1]^{K \times H \times W}$ is the predicted confidence distribution over the semantic classes as defined in the Introduction.

The loss term corresponding to the monocular depth estimation main task L_{main} is exactly the same described in Equation 1.1 in subsection 1.3.3, with a scale-invariant term L_d and a gradient term L_g respectively weighted $\alpha = 10,\ \beta = 0.1$. The loss term corresponding to the semantic segmentation secondary task is the negative log likelihood loss, applied on the logarithm of the predicted probabilities (hence, together with the softmax, it is the cross entropy loss)

$$L_{sec}\left(C, C^{**}\right) = \frac{1}{|T|} \sum_{i \in T} -\log\left(c^{**}_{i,c_i}\right), \tag{2.15}$$

where T is the set of pixels, $c^{**}_{i,k} \in [0,1]$ is the predicted probability that pixel $i \in T$ belongs to class $k \in \{0, \ldots, K-1\}$ and $c_i \in \{0, \ldots, K-1\}$ is the ground truth class of pixel $i \in T$. The pixels belonging to the 'void' 0-class are ignored in the computation of the semantic segmentation loss term.

After extensive experiments, the optimal weights for the two terms corresponding to the two tasks have been found to be respectively $\delta = 0.8$ for the depth estimation loss L_{main} and $\sigma = 1.2$ for the semantic segmentation loss L_{sec}; in the conducted experiments, many different combinations have been tried, by always keeping the balancing sum $\delta + \sigma = 2$ constant. Note that the optimality of these hyper-parameters refers to the monocular depth estimation

performance.

Furthermore, we explore the possibility of adding an edge consistency loss term L_{ec} between the two predicted outputs D^*, C^*, to enforce the intuitive constraint that the object contours defined by the segmentation mask obtained by assigning to each pixel the class with highest confidence and the object boundaries defined by changes in depth values should ideally match perfectly. Before implementing this additional term, we checked the goodness of the ground truth to this end, by computing the mean absolute error between the segmentation contours and the edges extracted from the corresponding depth map. A pixel is considered to be part of a segmentation contour if it belongs to a valid semantic class (not to the 'void' 0-class described in the Introduction) and at least one out of the eight neighbouring pixels belongs to a different class; the edges are extracted from depth maps using the Canny algorithm [4]: it smooths the map using a Gaussian filter, applies the Sobel filter horizontally and vertically to derive edge gradient and normal direction (rounded to horizontal 0°, diagonal 45°, vertical 90°, anti-diagonal 135°), suppresses pixels not having the greatest norm in their neighbourhood along the positive and negative direction of their gradient, and performs a hysteresis double-thresholding to keep only strong edges and weaker edges directly connected to strong edges (too weak edges are thrown even if connected). A grid search was conducted to find the optimal low and high quantile thresholds to minimize the MAE over all segmentation/depth pairs in the official NYU Depth V2 [50] dataset, which resulted in values for which no edge was retained. Since this means that there is no perfect correspondence, we tried to soften the minimization problem by applying all combinations of the following variations:

a) Replace the initial linear Gaussian filter with a non-linear median filter. While both filters have the aim of reducing noise to prevent the detection of false edges, the former also smooths the true edges and reduces their strength, possibly leading to the suppression of not-so-weak edges and the emergence of isolated edges. On the other hand, the median filter is well-known to be a preferable edge-preserving alternative.

b) Remove the non-maximum suppression step of the algorithm to obtain thicker edges and consequently increase the probability of pixel matching.

c) Apply the contour extraction operator and the edge detection algorithm respectively to segmentation masks and depth maps downsampled by a factor of $\{1/2, 1/4, 1/8, 1/16\}$ and compute the MAE between the output contours and edges upsampled to the the original resolution. This has the benefit of reducing the pixel-level noise and filtering out the weakest edges of the depth map before the algorithm is applied, but could negatively impact the precision of edge localization. The upsampling of the two binary masks produces thicker edges and transforms them to continuous maps with the pixels closest to the detected contours/edges having values close to 1, the neighbouring pixels having gradually decreasing values and the faraway pixels having values close to 0; this allows the pixel-wise absolute error to take values in the range $[0, 1]$ instead of in the set $\{0, 1\}$ and consequently increases the probability of at least partial pixel matching.

d) Apply a Gaussian filter to smooth the output contours and edges binary masks to continuous maps and consequently increase the probability of at least partial pixel matching. In the case of resampling presented above, the application of the Gaussian filter has been tried either before or after the upsampling to verify is it is more beneficial to our end to respectively interpolate a smoothed map or smooth an already continuous map.

In general, we found that the implementation of these variations positively impacted the matching between the two object boundaries maps, since the optimal low and high quantile thresholds to minimize the MAE outputted by the grid search were less restrictive and allowed a higher amount of edges to be retained. However, we noticed several drawbacks of this approach: in particular, the average MAE increases significantly as the maps get denser to soften the matching problem, but simultaneously become less informative regarding the precise location of boundaries and still contain many false edges even when missing many true edges. This behaviour and the lack of correspondence can be at least partially explained by the way in which the ground truth semantic segmentation masks have been annotated: from a qualitative analysis, most depicted objects are surrounded by strips of unlabeled pixels of varying thickness, which introduce a second shifted boundary between them and their background they occlude, instead of reporting the adjacency of the two classes. Furthermore, the time needed to process a single segmentation/depth pair is not negligible, especially when applying several filters at high resolution, and would considerably increase the training time. For these reasons, we decided to avoid introducing this additional cross-task constraint, also because it is already weakly implicitly enforced by the task-specific loss terms penalizing wrong predictions at the object boundaries.

2.4 Experiments

In this section we compare the performance of our proposed method with previous works on the challenging NYU Depth V2 [50], thanks to the presence of densely labelled maps for semantic segmentation. The main focus remains on the monocular depth estimation task, but we also report and discuss the results of the secondary semantic segmentation task.

2.4.1 Implementation details and training settings

The Pytorch [41] implementation is based on [78] on the 'mmsegmentation' [39, 40] toolbox. As previously mentioned, the various encoding backbones proposed are based on ViT-Large [12]; the single-stream version and both streams in the quasi-two-stream versions are initialized with the parameters pre-trained on ILSVRC [45] for image classification: specifically, all relevant parameters are duplicated to identically initialize the two streams, hence $W_{se} = W_{de} = W_v$, $b_{se} = b_{de} = b_v$ for all three variations and $W_{q,me} = \left(W_q + + W_q\right)/2$, $W_{k,me} = (W_k + + W_k)/2$, $b_{q,me} = b_q$, $b_{k,me} = b_k$ for the merged-attention variation described in subsec-

tion 2.3.2. Synchronized batch norm is used in the decoder. The number of parameters is 555M, 555M, 606M, 303M respectively in the A_{se}, A_{de}, A_{me}, A (used in DEPTR, SETR and METR) encoders, and 5M both in the semantic segmentation and depth estimation decoder heads. The network training is 40k iterations long with a batch size of 8, corresponding to about 402 epochs. The optimizer used is SGD (stochastic gradient descent), with momentum set to 0.9 and weight decay set to $5 \cdot 10^{-4}$ apart from the positional embedding, for which it is set to 0; the learning rate starts from 10^{-3} and decreases polynomially with power 0.9 to 10^{-4}.

During training, standard data augmentation is performed to increase learning robustness: the samples are rotated by a random angle in the range $[-2.5, 2.5]°$ (through bilinear resampling for the RGB image and nearest resampling for the semantic segmentation and depth ground truths), center cropped to 464×565, randomly cropped to 464×464, horizontally flipped with 0.5 probability, their brightness is multiplied by a random factor in the range $[0.75, 1.25]$ with 0.5 probability, their gamma is multiplied by a random factor in the range $[0.8, 1.2]$ with 0.5 probability, and their color channels are separately multiplied by a random factor in the range $[0.8, 1.2]$ and randomly permuted with 0.5 probability.

2.4.2 Performance evaluation

Table 2.2 reports the depth estimation performance of the previous methods considered for comparison in subsection 1.4.2 and our best performing variation proposed in Part I, which we use as a first baseline. To be able to compare our proposed variations with the original architecture by Zheng et al. [78], we ported their proposed model to the monocular depth estimation task (which we called depth transformer, or DEPTR), trained it several times with the same settings described in subsection 2.4.1 and report the best result. Table 2.2 contains all four variations proposed in subsection 2.3.2 and the relative percentage difference between the best performing one (METR, from MErged TRansformer) and the DEPTR. To ensure a fair comparison, the performance of all methods mentioned above is obtained from the models being trained on the 795 frames of the official dataset of NYU Depth V2 [50]. For each metric, the best result is highlighted in bold and the second-best is underlined. Table 2.2 also includes the results of [44] for completeness, but they cannot be fairly considered in the comparison (hence are indicated in italic), because their network is trained on the MIX6 [44] meta-dataset containing 1.4M images and fine-tuned on the full NYU Depth V2 [50]; moreover, since the architecture by Zheng et al. [78] convincingly outperforms the one by Ranftl et al. [44] in the semantic segmentation task as can be seen in Table 2.1, we can reasonably assume the same would happen in the monocular depth estimation task, considering both the close relationship between the two tasks and the fact that the adjustments needed to port the architectures to the monocular depth estimation task can be safely assumed to not influence the performance. As a side note, comparing [44] with the related work in Table 1.3 in section 1.2 suggests that the use of transformers has a major impact on results, since [44] outperforms all convolutional networks, even if employing attention modules, and in particular the previous state of the art [3], which already proposed a post-processing transformer block.

Table 2.2: Depth estimation quantitative evaluations on NYU Depth V2 [50].
The top part contains results discussed in Part I. The performance of DEPTR [78] and of our proposed variations (ordered as in subsection 2.3.2) corresponds to the best one obtained over multiple training runs. For each metric, the best result is highlighted in bold and the second-best is underlined; the relative percentage difference between our METR and DEPTR [78] is also included in the last row.
* [44] cannot be fairly considered in the comparison, because of their training protocol.

Method	**AbsDiff** ↓	**AbsRel** ↓	**log10** ↓	δ_1 ↑	δ_2 ↑	δ_3 ↑	**RMSE** ↓	**RMSElog** ↓
Ye et al. [72]	-	-	0.063	0.784	0.948	0.986	**0.474**	**0.081**
Song et al. [52] (L_k)	0.360	0.147	0.060	0.809	0.965	0.992	0.481	0.175
Ours ($L_k \otimes C_k$)	0.360	0.141	0.060	0.815	0.965	**0.993**	0.485	0.174
Ranftl et al. [44] *	-	*0.110*	*0.045*	*0.904*	*0.988*	*0.998*	*0.357*	-
Zheng et al. [78] (DEPTR)	0.361	0.129	0.055	0.850	0.971	0.991	0.485	0.161
Ours (A_{se})	0.361	0.129	0.055	0.850	0.971	0.992	0.484	0.160
Ours (A_{de})	0.359	0.128	0.055	0.852	0.972	0.992	0.481	0.160
Ours (A_{me})	<u>0.357</u>	<u>0.128</u>	<u>0.055</u>	<u>0.852</u>	<u>0.973</u>	0.992	0.479	0.159
Ours (METR)	**0.353**	**0.126**	**0.054**	**0.855**	**0.973**	<u>0.993</u>	<u>0.475</u>	<u>0.157</u>
METR vs DEPTR	-2.27%	-3.02%	-2.55%	+0.52%	+0.25%	+0.12%	-1.93%	-2.30%

Table 2.3: Semantic segmentation quantitative evaluations on NYU Depth V2 [50].
The performance of [78] corresponds to the best one obtained over multiple training runs, while of our proposed variations (ordered as in subsection 2.3.2) corresponds to the same run reported in Table 2.2. For each metric, the best result is highlighted in bold and the second-best is underlined; the relative percentage difference between our METR and [78] is also included in the last row.

Method	**mIoU** ↑	**mAcc** ↑	**aAcc** ↑
Zheng et al. [78] (SETR)	0.4754	0.5866	0.7150
Ours (A_{se})	0.4849	0.6134	0.7549
Ours (A_{de})	**0.4897**	<u>0.6174</u>	<u>0.7563</u>
Ours (A_{me})	0.4874	**0.6182**	0.7535
Ours (METR)	<u>0.4878</u>	0.6139	**0.7569**
METR vs SETR	+2.61%	+4.65%	+5.86%

Table 2.3 reports the semantic segmentation performance of the original architecture by Zheng et al. [78], of all four variations proposed in subsection 2.3.2 and the relative percentage difference between the best performing one on the depth estimation task (METR) and the original method [78]. Note that the best single-task performance of [78] over several runs with the same settings described in subsection 2.4.1 is reported, while the semantic segmentation performance of each of our multi-task variations corresponds to the same run selected for its best performance on the depth estimation task and reported in Table 2.2; for each metric, the best result is highlighted in bold and the second-best is underlined.

First of all, we note that porting the method by Zheng et al. [78] to the monocular depth estimation task (DEPTR) outperforms our best variation from Part I in most metrics and reaches comparable results in the remaining ones (AbsDiff and δ_3); in particular, it achieves an improvement of -8.20% in AbsRel, -7.42% in log10, $+4.35\%$ in δ_1 and -7.34% in RMSElog.

The proposed variations are analyzed in the following.

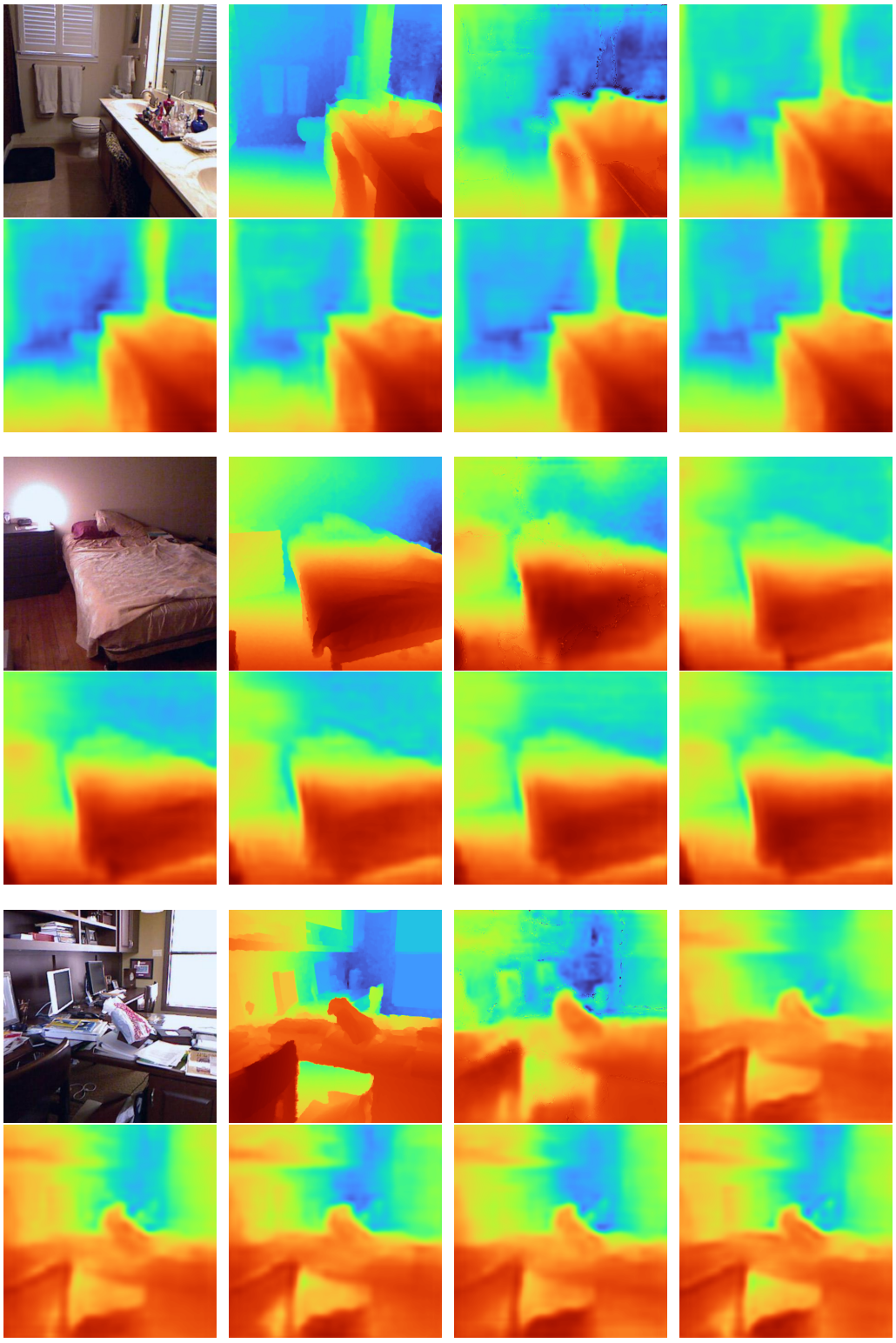
From the depth estimation point of view, the segmentation-guided quasi-two-stream variant (A_{se}) slightly outperforms the original method (DEPTR) in all metrics except for δ_1, but the improvement is negligible for most of them and is in general the smallest among the four models; this means that the guidance provided by the semantic segmentation task still has a small positive impact on the results, but the intuition that semantic similarity alone corresponds to similarity in depth values does not always hold: in fact, tokens belonging to the same class can be part of distinct objects at different distances from the camera, hence the depth estimation stream receives useful but sub-optimal information. The segmentation-regularized quasi-two-stream variant (A_{de}) achieves more relevant improvements on most metrics, proving that forcing the depth estimation stream to integrate in its tokens representation semantically consistent information helps to better model depth gradients inside object boundaries and steeper depth changes at object boundaries. The merged-attention quasi-two-stream variant (A_{me}) outperforms both the previous ones, since it allows the segmentation stream to provide high-quality semantic information to the depth estimation stream, without ignoring the token relationships in the latter embedding space; moreover, the network is able to adaptively scale the cross-stream influence in the different encoding stages according to which type of information is more relevant at the considered block position in the stack. The best performing variant is the single-stream one (METR), which convincingly achieves better results than all previous ones and than the baseline DEPTR: it improves in all metrics, in particular by -2.27% in AbsDiff, -3.02% in AbsRel, -2.55% in log10, -1.93% in RMSE and -2.30% in RMSElog, and it is the only one to reach a comparable performance on δ_3 with respect to our best variation from Part I (-0.04%) and to close the gap on RMSE with respect to [72] ($+0.21\%$). A possible reason why the learning of task-invariant features in the encoder outperforms the learning of task-specific features is the very high correlation between the two considered tasks, which makes a shared representation more efficient to learn and more robust; the resulting encoded output is then forced to be semantically consistent and depth consistent at the same time to allow the task-specific modelling to effectively take place during the decoding, by appropriately leveraging both types of information.

From the point of view of the secondary semantic segmentation task, the segmentation-guided quasi-two-stream variation (A_{se}) already significantly outperforms the single-task baseline [78], with an improvement of $+2.00\%$ in mIoU, $+4.57\%$ in mAcc and $+5.58\%$ in aAcc, proving that the proposed multi-task setting is beneficial for both tasks, even if it was designed to boost the main one. Differently from the previous discussion, in which better depth estimation results were obtained with the configuration with the depth stream computing the shared attention, in the case of semantic segmentation, the configuration with the segmentation stream computing the shared attention is outperformed by the segmentation-regularized quasi-two-stream variant (A_{de}), which in particular achieves the top $+3.01\%$ in mIoU; this outcome confirms the claim that object segments are better defined by depth changes rather than by low-level features, since the depth similarity provides information more relevant for the task than the segmentation similarity itself. Further validation comes from the perfor-

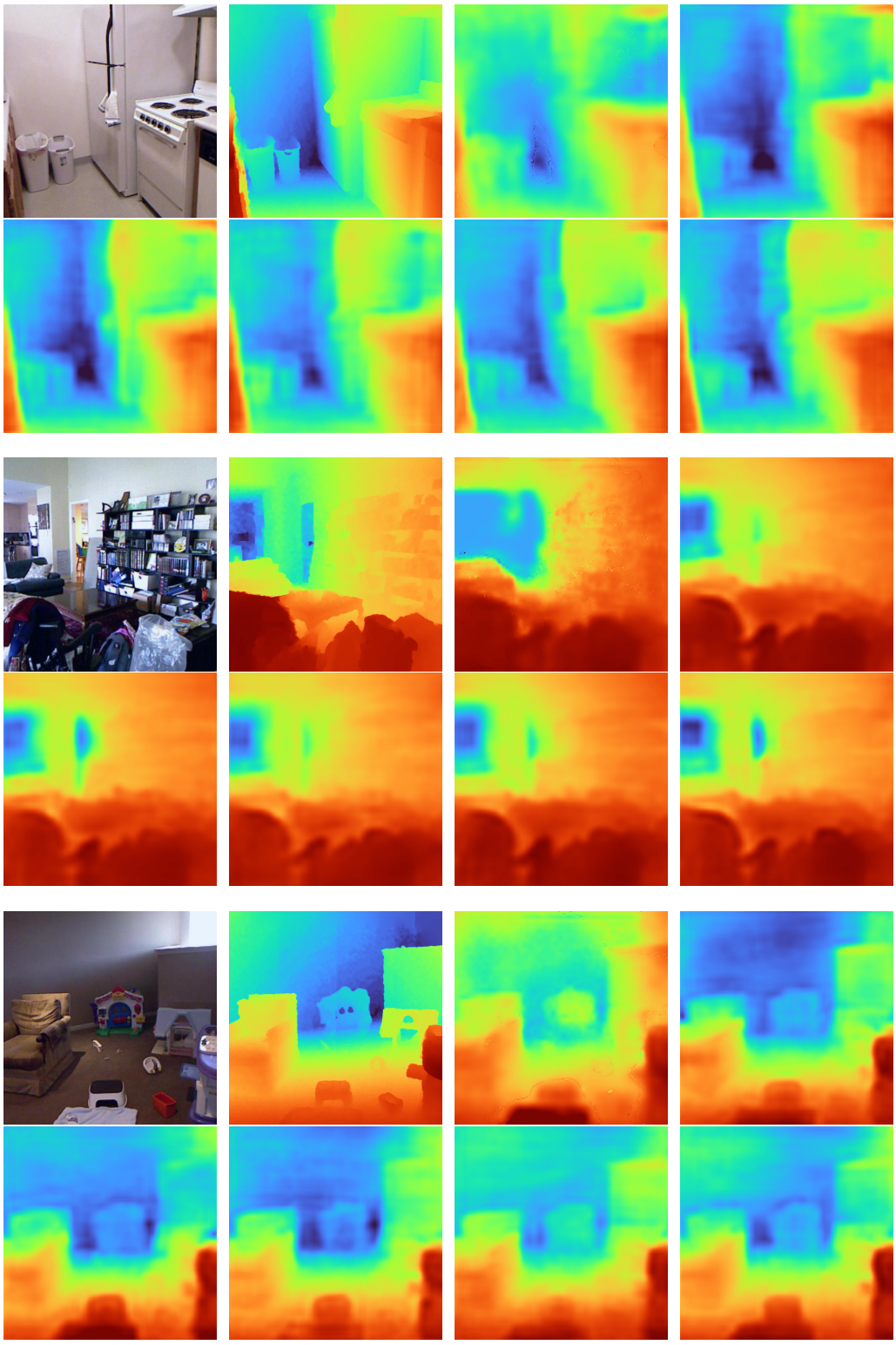
mance of the merged-attention quasi-two-stream (A_{me}) and single-stream (METR) variants, which respectively achieve the top +5.39% in mAcc and +5.86% in aAcc, but obtain overall worse results than A_{de}. However, both these last two models outperform A_{se} and in particular METR shows an overall smaller gap in the results with respect to A_{de}.

Considering the performance of the four proposed variants on both tasks, with a focus on the monocular depth estimation one, the single-stream model (METR) results being the best one, even more if taking into account that the number of parameters in the encoder is not increased; nevertheless, it is worth mentioning that A_{de} allows to completely remove at test time the semantic segmentation stream in the encoder (252M parameters) in addition to the corresponding decoder head (which can be removed in all variations), almost halving the network complexity and almost doubling its processing speed with respect to the training; conversely, if desired, A_{se} allows to completely remove at test time the depth estimation stream in the encoder.

Figure 2.5 shows a comparison of the qualitative depth estimation results obtained with our best method from Part I ($L_k \otimes C_k$), the method by Zheng et al. [78] (DEPTR) and our proposed variants with respect to the ground truth: red represents a shorter depth, while blue represents a longer depth; all results are scaled according to the minimum and maximum values in the corresponding ground truth map for visualization purposes, hence the same color could correspond to different depth values in different scenes. Figure 2.6 shows a comparison of the respective error maps: intense red represents overestimated depths, while intense blue represents underestimated depths, hence a whiter error map corresponds to a more accurate prediction. First, we can notice that all the models discussed in this chapter produce better results than $L_k \otimes C_k$: for example, in the bathroom scene (first group), it is evident that object shapes are sharper and the structural element at the end of the mirror is correctly understood as separate from the background wall. Considering our four multi-task methods (A_{se}, A_{de}, A_{me}, METR), they in general produce progressively better and better depth maps, all more similar to the ground truth than DEPTR [78]. This can be seen in the playroom scene (sixth group), in which each error map is overall whiter than the previous, with the one corresponding to METR being the best one among the considered six; however, there obviously are some fluctuations around this behaviour: for example, in the kitchen scene (fourth group), A_{de} produces a slightly better depth map than A_{me}, by more precisely recognising corners and edges. Focusing on DEPTR and METR, the improvement of the latter on the former is visible both on the planar surface of bigger elements, like the bed and night table in the bedroom scene (second group), and on the boundaries of smaller objects, like the computer screen in the home office scene (third group). Figure 2.7 shows a comparison of the qualitative semantic segmentation results obtained with the method by Zheng et al. [78] (SETR) and our proposed variants with respect to the ground truth. The improvement of our multi-task models on the baseline is easy to assess in the segmentation masks, for example in the kitchen scene (fourth group). However, the superiority of A_{de} among them, certified by the quantitative evaluations, is less visually evident; in particular, METR often seems to produce outputs overall more similar to the ground truth. Figure 2.8 compares the

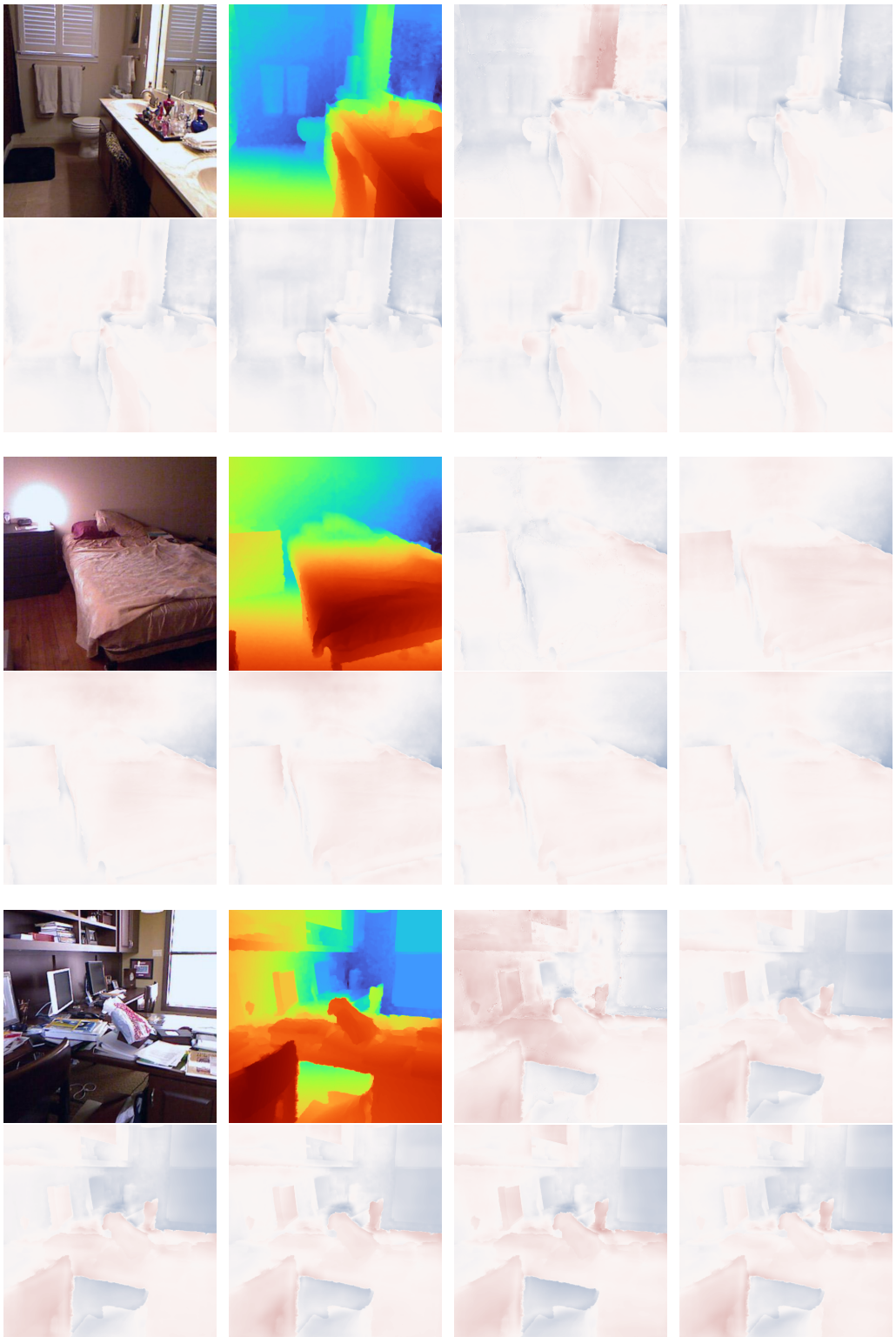


(a) RGB (b) GT (c) Ours $\left(L_k \otimes C_k\right)$ (d) Zheng et al. [78] (DEPTR)
(e) Ours (A_{se}) (f) Ours (A_{de}) (g) Ours (A_{me}) (h) Ours (METR)

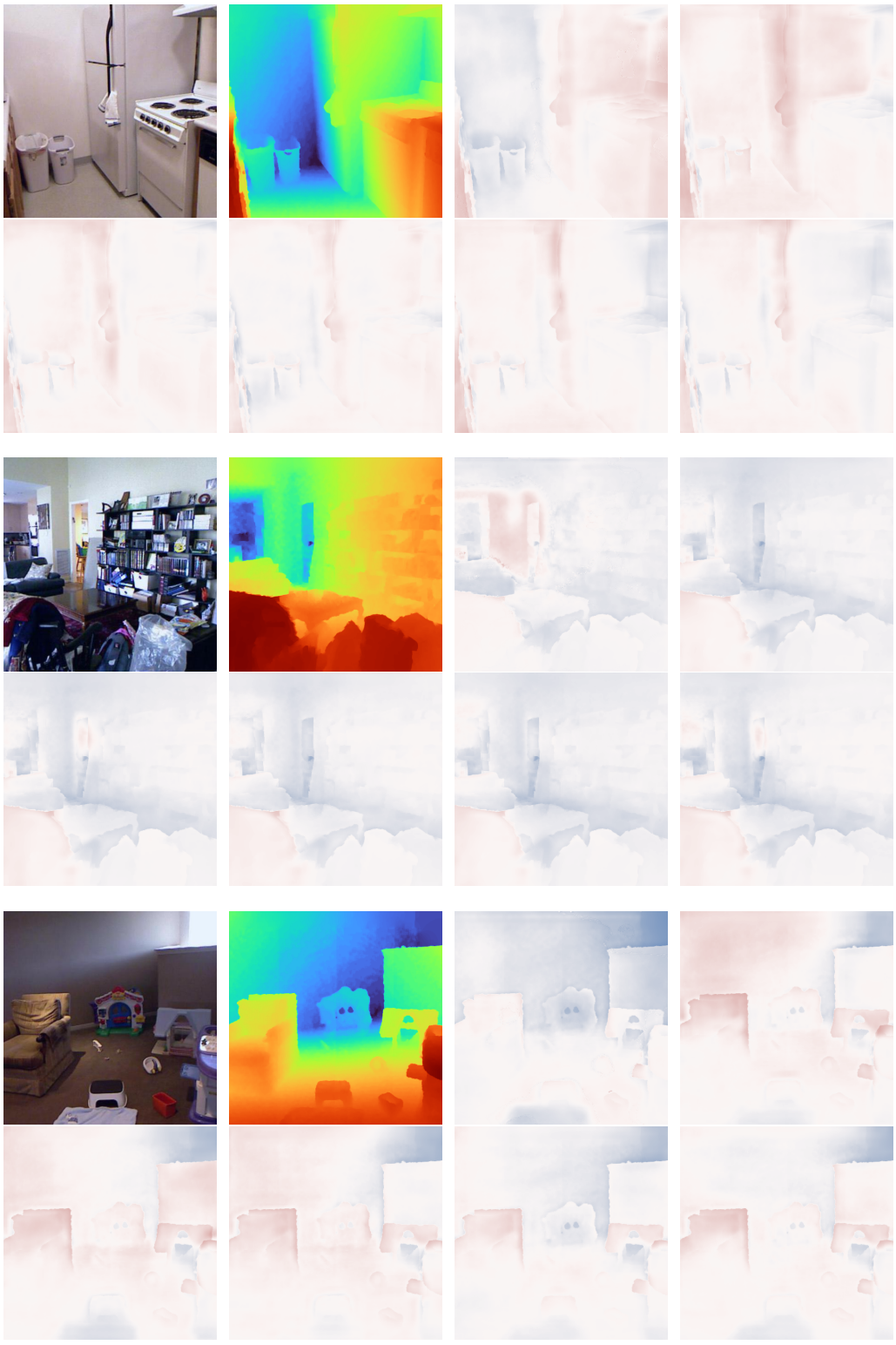


(a) RGB (b) GT (c) Ours $\left(L_k \otimes C_k\right)$ (d) Zheng et al. [78] (DEPTR)
(e) Ours (A_{se}) (f) Ours (A_{de}) (g) Ours (A_{me}) (h) Ours (METR)

Figure 2.5: Qualitative comparison of depth estimation results on NYU Depth V2 [50].

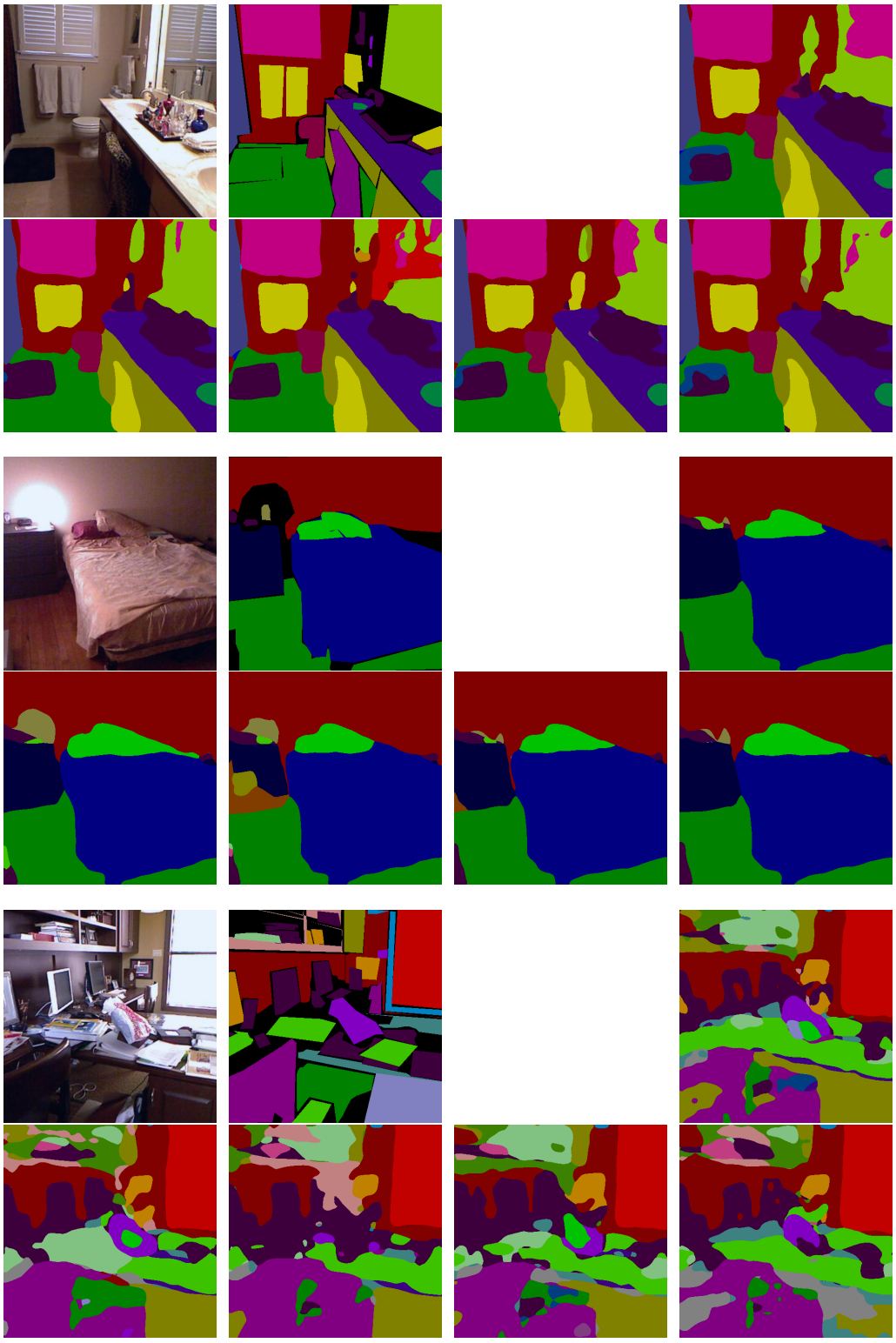


(a) RGB (b) GT (c) Ours $\left(L_k \otimes C_k\right)$ (d) Zheng et al. [78] (DEPTR)
(e) Ours (A_{se}) (f) Ours (A_{de}) (g) Ours (A_{me}) (h) Ours (METR)

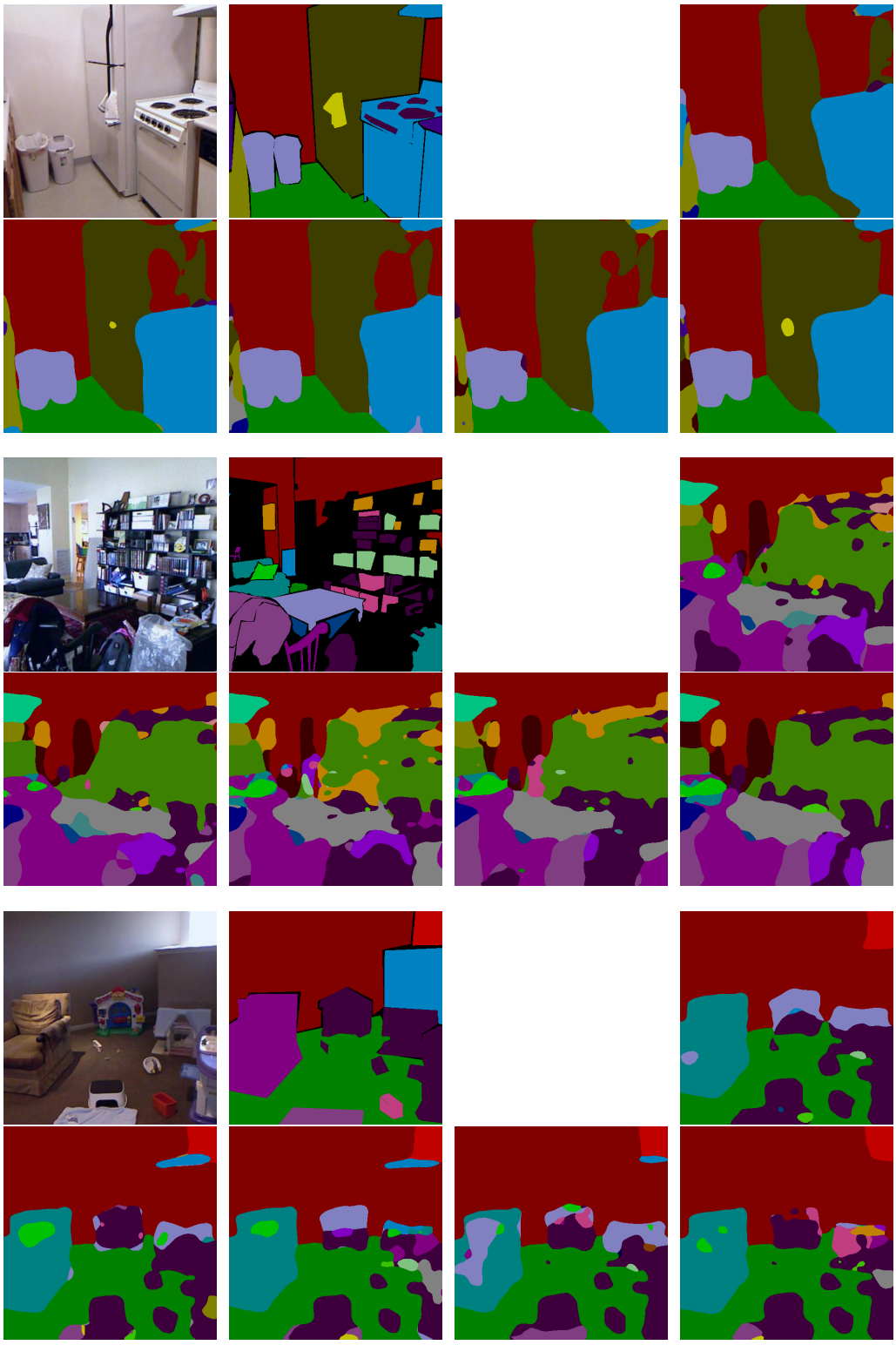


(a) RGB (b) GT (c) Ours $\left(L_k \otimes C_k\right)$ (d) Zheng et al. [78] (DEPTR)
(e) Ours (A_{se}) (f) Ours (A_{de}) (g) Ours (A_{me}) (h) Ours (METR)

Figure 2.6: Qualitative comparison of error maps on NYU Depth V2 [50].

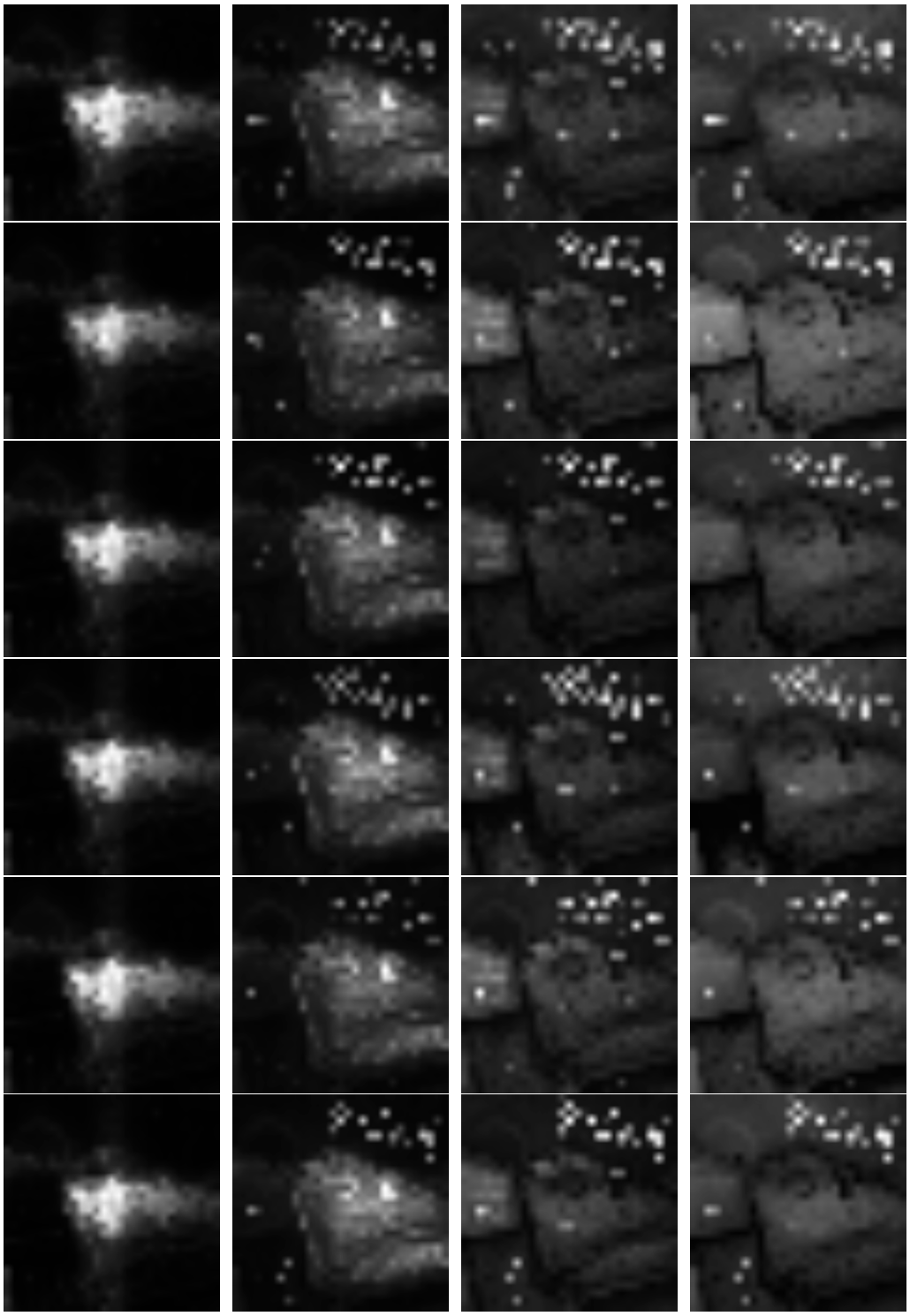


(a) RGB (b) GT (c) Zheng et al. [78] (SETR)
(d) Ours (A_{se}) (e) Ours (A_{de}) (f) Ours (A_{me}) (g) Ours (METR)



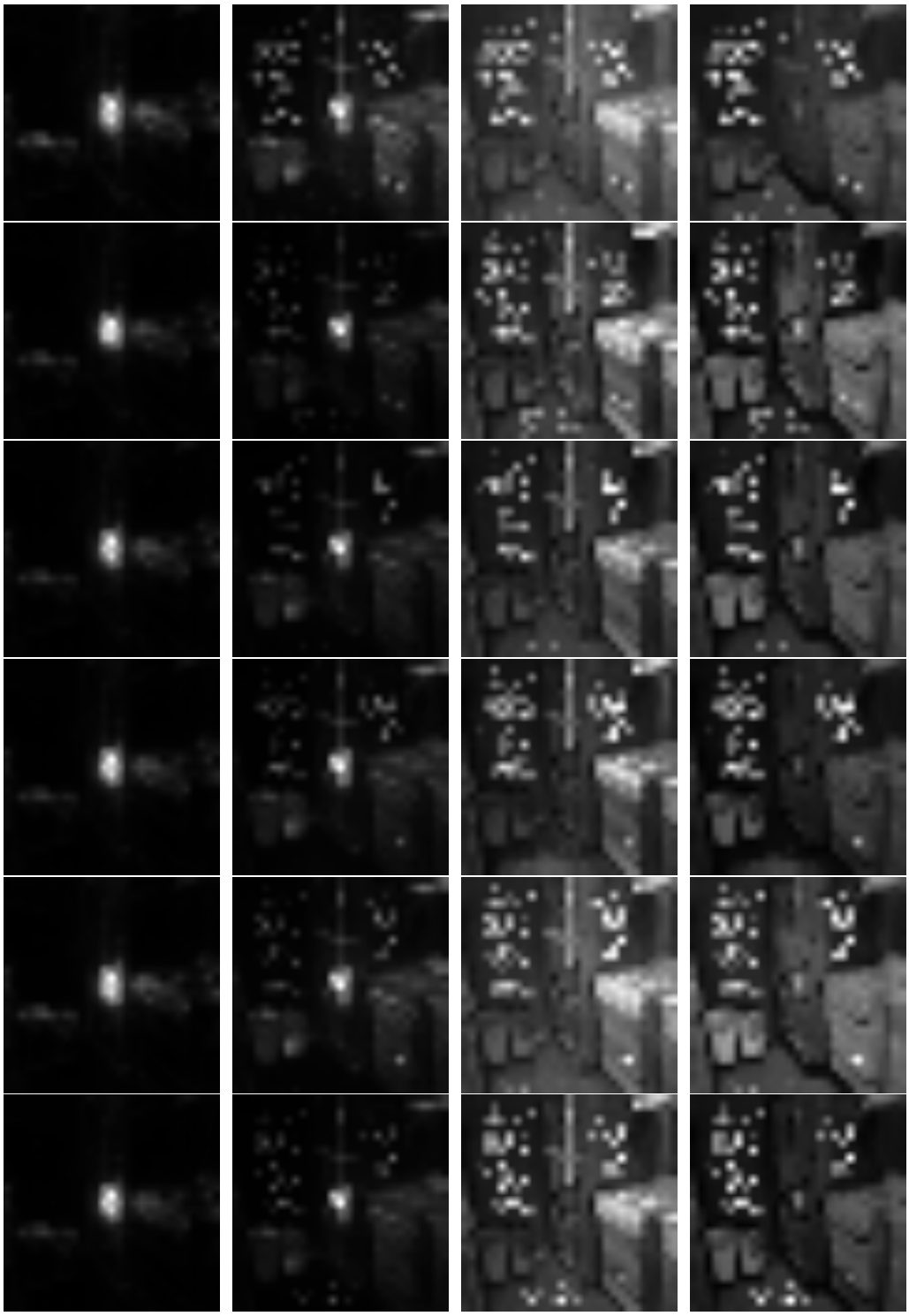
(a) RGB (b) GT (c) Zheng et al. [78] (SETR)
(d) Ours (A_{se}) (e) Ours (A_{de}) (f) Ours (A_{me}) (g) Ours (METR)

Figure 2.7: Qualitative comparison of semantic segmentation results on NYU Depth V2 [50].



(a) $A^{(5)}$ (b) $A^{(11)}$ (c) $A^{(17)}$ (d) $A^{(23)}$

Top to bottom: DEPTR [78], SETR [78], A_{se}, A_{de}, A_{me}, METR.



(a) $A^{(5)}$ (b) $A^{(11)}$ (c) $A^{(17)}$ (d) $A^{(23)}$

Figure 2.8: Comparison of attention maps in the last transformer block of each encoding stage. Top to bottom: DEPTR [78], SETR [78], A_{se}, A_{de}, A_{me}, METR.

attention maps (bilinearly upsampled to full resolution) related to the central 16 × 16 patch in the last transformer block of each encoding stage. Both in the bedroom (first page) and kitchen (second page) scenes, the attention maps are very similar in all single- and multi- task methods. Specifically, at the end of the first encoding stage, attention is focused very closely around the considered patch and on tokens similar in appearance; at the end of the second encoding stage, attention starts spreading more over the shape of the object depicted in the considered patch; at the end of the third and fourth encoding stages, attention focuses more on the global scene layout. Interestingly, the attention computed in the two single-task models, DEPTR for depth estimation (first row) and SETR for semantic segmentation (second row), already show little significative differences; therefore, it is not surprising that the multi-task models display an intermediate behaviour, without appreciable dissimilarities traceable to which stream was used in the computation. Intuitively, this confirms the close relationship between the two tasks and suggests that most task-specific processing occurs in the decoder heads, regardless of the encoder's structure (at least with such a small training set).

2.5 Conclusions

In this chapter, we explored the role of different shared-attention mechanisms in the interaction between jointly-learnt task-specific encoding streams related to depth estimation and semantic segmentation. While the experiments on the NYU Depth V2 dataset demonstrate the overall potential of the proposed multitask approach in mutually boosting the performance of the involved related tasks with respect to the respective individual monotask settings, some of the tested designs clearly produce better results. Specifically, the attention computed by the semantic segmentation stream provides the least effective information for both tasks: worse than the one produced by the depth estimation stream, but still better than the monotask self-attentions. The jointly-computed attention achieves higher performance than both the ones obtained by only one of the streams, but lower than the fully-merged task-invariant single stream. This could be due both to the strong relationship between the two tasks and to the small size of the training set, which does not allow the two streams to properly learn specific feature embeddings; in fact, is has been empirically demonstrated both in the NLP and vision fields that the performance of transformers considerably improves when trained on sufficiently large datasets, without reaching evident saturation thresholds. As a side benefit, the use of global attention mechanisms instead of convolutional layers also increases the interpretability of feature encoding.

The presented transformer-based encoders allow to spread the application of global-scale attention mechanisms to all stages of feature embedding instead of only at the bottleneck on the output of a convolutional backbone; this approach effectively solves the problem of locality intrinsically present in all fully-convolutional networks because of their limited receptive filed, in particular in the shallow stages, which makes the learning of long-range dependencies difficult. On the other hand, the computation and memory constraints and the consequent specific implementation choices discussed in this chapter do not address

other two important issues. Firstly, the main limitation highlighted by almost all works in the field is the need to downsample the feature maps: while it can be advantageous for some tasks focused on the learning of a image-level representation (e.g., image classification) and in general can provide useful information from a global understanding perspective, it also represents a significant downside for dense prediction tasks, which aim at producing precise estimates with high-resolution details. Thanks to the use of attention, the image context is taken into account by construction at all stages, hence depriving downsampling of its role. Nevertheless, the quadratic complexity of global attention prevents the elimination of downsampling mechanisms and possibly even worsen them: the discussed methods operate on feature maps of fixed resolution during the whole encoding process and this resolution is the same as the typical output of a convolutional backbone, hence most of the local spatial information is immediately lost at the stem and processing of higher-resolution features is never performed. This introduces the second unresolved issue: previous works have shown that attention mechanisms are more effective when applied at high resolution in or after the decoder rather than at low resolution at the bottleneck. While this could be related to the presence of semantically high-level information in both cases, it does not exclude the possibility that high-resolution attention is beneficial in general, even when applied to learn dependencies between shallow low-level features.

3 Pyramid fully-transformer network

In this chapter we will continue the work started in the previous one with the adoption of vision transformers in the encoding part of the architecture. First, we address the issue caused by the use of a columnar backbone operating exclusively on low-resolution features, which is able to provide only single-scale information to the decoder. To do so, we replace it with a transformer-based pyramidal encoder capable of processing feature embeddings at gradually decreasing resolutions, allowing to retain more fine-grained details in the early stages and to feed them to the decoder through hierarchical multi-scale connections; this is possible thanks to shifted-windows attention which reduces the operation's complexity. After integrating this approach in our multi-task learning framework with semantic segmentation as a secondary task to monocular depth estimation, which proved to successfully improve performance in chapter 2, we extend the same approach to the design of transformer-based vision decoders. We propose several single-task variants of increasing complexity, both inspired by convolutional decoders and novel modules leveraging the information coming from the skip connections in an attentive way. For the latter, we also propose a quasi-two-stream with shared attention strategy to port them to the multi-task setting, as opposed to the simple instantiation of separate identical decoding heads. The architecture resulting from the combination of the adopted pyramidal encoder and the proposed transformer-based decoders is completely convolution-free. Experiments on the official NYU Depth V2 dataset show that carefully designed fully-transformer models can compete with methods making use of convolutional layers and reach at least comparable results. In particular, our novel attentive use of skip connections in the decoder obtains an improvement on the baseline in the depth estimation task of -2.27% in AbsRel and in the semantic segmentation task of +1.30% in mIoU.

3.1 Introduction and background

The works discussed in chapter 2 demonstrated that transformer-based encoders are able not only to compete with the dominating fully-connected networks, but also to surpass them, thanks to the particularly desirable properties of their building block, the vision transformer, and specifically of its integrated attention mechanism, which solves the fundamental limit of convolutional layers. This transition from a local- to a global- based focus and from a parameter- to a content- based approach is revolutionizing the vision field as it did with NLP. However, some issues remain unaddressed in their practical implementation, possibly containing their full potential.

As previously mentioned, the current approach in architecture design is to follow the encoder-decoder structure inherited from the many studies on fully-convolutional networks and to employ columnar transformer-based encoders and convolutional decoders. Here columnar signifies that, after the patch-embedding step in the stem, the resolution is kept constant for all transformer blocks in the stack and the only high-resolution processing occurs in the decoder after upsampling. The reason behind this design is that global attention has quadratic complexity with respect to the number of tokens, hence to the number of patches, making the use of too small patch sizes intractable. This loss of details at the very beginning of the processing, before even the shallowest blocks can learn their finer high-resolution information, is a problematic effect, because the difficulty of recovering in the decoder what is lost in the encoder is well known and studied, since downsampling also afflicts the convolutional encoders. Among the various tentative solutions proposed in the past to deliver high-resolution detail-rich features to the decoder, skip connections are the most successful and widely adopted; their use, in fact, has been seamlessly transferred to the transformer-based architectures during the change of paradigm in encoding. The main difference is that in this last case skip connections are only able to forward low-level appearance features, which still provide useful information to solve high-level semantic ambiguities, but constrained in a lower resolution with respect to their analogous in convolutional architectures, hence losing most of their positive impact. The coupling of the columnar structure of encoders and of the consequent lack of high-resolution information in the skip connections is the main downside of the previously discussed methods.

Moreover, we noticed a reticence in the exploration and adoption of transformer-based decoders for dense prediction together with transformer encoders, which would result in a fully-transformer architecture equipped with long-range attention from the first encoding stage to the final estimation stage.

3.2 Related work

In this section, some recent works employing transformer-based networks and addressing the issues present in their early designs in the vision field are presented. Due to the lack

Table 3.1: Semantic segmentation quantitative evaluations of related works on ADE20K [79]. For each metric, the best result is highlighted in bold and the second-best is underlined; the data reported are provided by the respective papers.

Method	**mIoU ↑**	**aAcc ↑**
Ranftl et al. [44]	0.4902	<u>0.8311</u>
Zheng et al. [78]	<u>0.5028</u>	**0.8346**
Wang et al. [65]	0.4260	-
Liu et al. [36]	**0.5350**	-

of a large number of relevant works considering the monocular depth estimation task, we also include the most important ones from related tasks. A comparison of the semantic segmentation performance of some related methods can be found in Table 3.1 for ADE20K [79]; the data reported are provided by the respective papers, with the best result highlighted in bold and the second-best underlined for each metric.

3.2.1 Wang et al. [64]

Wang et al. [64] studied the self-attention mechanism in the NLP field and argued that previous efforts to reduce its time and memory complexity from the point of view of the trade-off between efficiency and performance were insufficient and unsatisfactory. In particular, the most common techniques were to either perform sparse attention on a subset of tokens, which reduces complexity to sub-quadratic, or to use locally-sensitive hashing, which reduces complexity to linearithmic but increases the number of sequential operations. They then performed a spectrum analysis of the attention matrices produced inside two variations of a popular language-modeling network and discovered that they have long-tail spectrum distributions, hence they could be reasonably approximated by a low-rank matrix. Therefore, they proposed to leverage this behaviour to reduce the complexity of the overall mechanism, by adding linear transformation layers to project the computed key and value to a lower-dimensional space with a fixed number of representative tokens; this would reduce the complexity to linear. Moreover, they proposed additional optimization techniques, such as the sharing of projection parameters between heads and layers, or the possibility to tune the projected dimension across the different layers (since deeper layers show a more skewed spectrum distribution and can be approximated with a lower-rank matrix with respect to shallower layers).

3.2.2 Ramachandran et al. [43]

Inspired by the many attempts to augment convolutional networks with self-attention modules, Ramachandran et al. [43] proposed to transform content-based interactions in a stand-alone vision primitive, capable of being applied at any stage and at any resolution.

To achieve this goal, they had to solve the problem of quadratic complexity, that prevents to naively use attention on shallow feature maps without considerable prior downsamplig. They developed a basic layer designed to replace spatial convolutions in a parameter- and computation- efficient way, without losing their main strengths. In particular, local attention is performed for every query pixel using keys and values derived only from the pixels in the neighbourhood centered in the considered pixel and with a limited spatial extent. The resulting behaviour is similar to the application of strided convolutions, but output pixels are computed as convex combinations weighted by local content interactions, instead of parameter-shared linear combinations; moreover, the use of relative position embedding provides translation invariance. The theoretical complexity of this approach is linear in the input resolution, which allows a less constrained use; in fact, they proposed to replace all convolutions in several architectures of the ResNet [21] family and achieved comparable or slightly better performance with less learnable parameters and a gain in time and memory consumption. They also introduced a variation even more similar to convolutional layers, using spatially-varying linear transformations in the computation of value features to be able to learn and leverage distance-based information, which is particularly relevant for low-level pattern detection in shallow layers.

3.2.3 Wang et al. [62]

Working in the same improvement direction, Wang et al. [62] designed a stand-alone attentive layer with reduced complexity. They argued that following the convolutional approach of restricting computation in local regions for efficiency purposes significantly limits the characteristic property of attention mechanisms, which is their ability to effectively model long-range dependencies. Instead, they proposed to factor vanilla 2D self-attention in two 1D self-attentions applied in sequence along the two axis of the input image (specifically, height-wise and width-wise). This approach trades off content expressiveness for computation efficiency, by approximating the token-level pair-wise relationships with axial attention and achieving a sub-quadratic complexity (linear if the 1D self-attentions are constrained in a neighbouring span, hence obtaining an efficient approximation of local attention). They also proposed a different approach to relative position embedding, by keeping the query-dependent bias, but also adding a key-dependent bias in the similarity computation and augmenting the values with positional information. Furthermore, they introduced a simple attention-based decoder block, in which the output of their proposed axial mechanism is summed with its upsampled input and with encoder features at the corresponding resolution.

3.2.4 Wang et al. [65]

Wang et al. [65] was the first work to highlight the differences between popular convolutional neural networks and vision transformers that make the previous designs of the latter inadequate (or at least sub-optimal) for dense vision tasks. They underlined that the columnar structure of vision transformers [12] was specifically tailored for image classification,

which is not negatively affected by single-scale low-resolution outputs obtained from coarse image patches. To solve this inefficiency, they adopted some design choices typical of fully-convolutional architectures and applied them to transformers to take advantage of different characteristics of the two paradigms. In particular, they proposed a pyramidal structure inspired by ResNet [21] rules: it is divided in four stages, inside which all transformer blocks are identical, such that the shallower blocks operate at higher resolution on a smaller number of channels, the deeper blocks operate at lower resolution on a larger number of channels, and most computation is concentrated in the intermediate stages (mainly in the third stage). The initial patch embedding produces input feature maps of $1/4$ of the original resolution, as opposed to the $1/16$ of previous approaches, while the output consists of multi-scale feature maps. Even if the progressive shrinking of the resolution between stages already considerably reduces the total computation, the quadratic complexity of self-attention still poses serious consequences on time and memory consumption. They then proposed a spatial-reduction attention, which temporarily further shrinks the feature maps to super-patches before computing key and value (query is not affected); in this way, the similarity between each query patch and each key super-patch is computed, reducing the complexity by a factor equal to the super-patch token-resolution. The spatial-reduction ratio exponentially decreases in every stage to reach vanilla global self-attention in the fourth and last one.

3.3 Method

Our method is built starting from the work by Liu et al. [36], which concurrently applied concepts similar to [65], but focusing on general-purpose performance instead of dense prediction and achieving linear complexity with respect to input resolution in the attention computation instead of quadratic. In particular, its considered tasks were image classification, object detection and semantic segmentation. First, we port their method to the monocular depth estimation task and apply it both in a single- and multi- task framework; in the latter, we follow our previous work in chapter 2 and directly consider only the best performing version among the proposed and tested ones. We then adopt their pyramidal transformer encoder as a backbone and propose to replace the decoder with several different designs based on their transformer variant, hence achieving a fully-transformer architecture without convolutions. This approach is particularly interesting in the multi-task setting, because it allows us to continue exploiting the peculiar structure of transformer blocks in exploring the interactions between distinct streams and their integration with the multi-scale features coming from the encoder. In addition, this analysis can be seen as the intersection between the concepts developed in chapters 1 and 2, since the mutual influence of the two different tasks mainly takes place at the decoder through the application of custom attention mechanisms; moreover, the joint learning of task invariant features in the encoder directly affects the decoding process through an attentive use of skip connections.

In this section, we first describe the overall architecture with the encoder-decoder proposed by [36] and the adjustments needed to match our multi-task problem, then we present

our proposed transformer-based decoders for dense prediction and finally detail the multi-task version of the most promising one.

3.3.1 Overall architecture

The employed network consists in an encoder-decoder structure. For the encoder, we adopt Swin-L [36], which applies stacked transformers to features of gradually decreasing resolution in a pyramidal manner, hence producing hierarchical multi-scale encoded features as shown in Figure 3.1. In particular, following ResNet [21] structure and design rules, four stages are defined in succession: each of them contains a patch embedding step, which reduces the spatial resolution and increases the channel dimension, and a columnar sequence of transformer blocks. The initial basic patch embedding in the first stage is performed with square patches of size $p_H = p_W = 4$ and with channel size $C = 192$, without the addition of the 'class' token; the patch merging in all three subsequent stages takes the output tokens of the previous stage, reshapes them in a 2D representation and aggregates neighbouring tokens in non-overlapping patches of size $p_H = p_W = 2$ through channel-wise concatenation and a linear transformation that halves the resulting number of channels (hence doubles the number of channels with respect to the input tokens). This approach halves the resolution and doubles the channel dimension at every intermediate stage, matching the behaviour of typical fully-convolutional backbones and producing a feature pyramid (with outputs of $\{1/4, 1/8, 1/16, 1/32\}$ of the original resolution) compatible with most previous methods for vision tasks. Following [21], most computation is concentrated in the third stage: out of a total of $N = 24$ transformer encoders, 2 blocks are in the first, second and fourth stage and 18 are in the third stage; for comparison, note that in [78] and chapter 2 the same total number of blocks is employed, but they are all applied on features of $1/16$ of the original resolution (corresponding to the third stage in this case). In each block, the attention is performed with an increasing number of heads depending on the stage of belonging to match the channel dimension increase, with $M \in \{6, 12, 24, 48\}$ respectively in the first, second, third and fourth stage, and is augmented with relative position bias (type B, as described in subsection 2.1.1); the hidden layer of the MLP in each block has a channel expansion factor of 4.

However, the high resolution in the first two stages does not allow the use of global self-attention, due to its quadratic complexity with respect to the token sequence length. To solve this issue, in all stages, the tokens reshaped in a 2D representation are divided in non-overlapping square windows of size $h = w = 7$ and intra-window self-attention is independently computed for each of them; this means that each token attends to all and only the tokens in its own window, both as a query and as a key/value. A possible downside of this approach could be that the restriction to fixed local windows completely stops any type of global-like long-range interaction; this undesirable behaviour is prevented in two ways: while the fixed window size in a context of progressive resolution reduction allows by construction farther and farther tokens to interact in the deeper stages, explicit inter-window connections can be introduced to facilitate the learning of non-local dependencies.

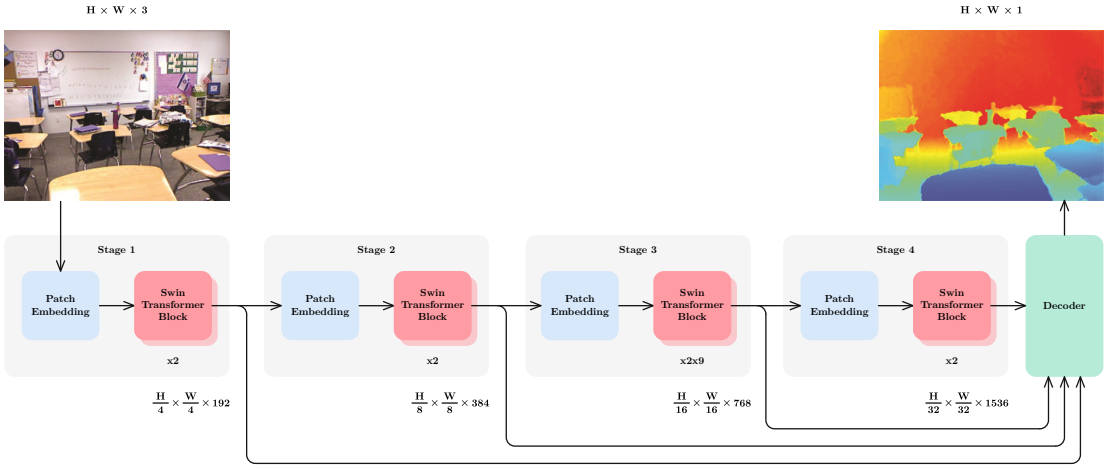


Figure 3.1: Overall architecture of monocular depth estimation networks with Swin backbone (DeSwin [36], PUPSwin, MLASwin, COASwin, SCASwin): the vision transformer based on shifted windows extracts hierarchical multi-scale features and feeds them to the decoder. In the multi-task versions, the two task-specific decoder heads in MeSwin and the two streams in DEASwin share the task-invariant encoded features as their identical input.

Specifically, the adopted solution is to alternate regular window partitioning with another non-overlapping partitioning in which the windows are shifted by half their size $\lfloor h/2 \rfloor = \lfloor w/2 \rfloor = 3$ both in the height and width dimensions. This can be seen as having the effect of gradually increasing the virtual receptive field of subsequent attention computations. The advantage with respect to the attention based on sliding windows [43] is that all query pixels in a window share the same key set, which allows efficient memory access and low latency, while still maintaining comparable expressiveness; furthermore, the locality characteristic of this type of attention mechanisms does not negatively impact performance due to the high correlation present in visual inputs.

When employed in the multi-task setting, we follow the best results previously obtained by our analysis in chapter 2 and adopt a single-stream encoder, leaving the task-specific modeling to the decoding part of the network; the task-invariant output features of the encoder are then shared by the decoder as identical inputs of separate heads or of a single quasi-two-stream head.

The loss function used to optimize the network learnable parameters during training is exactly the same described in Equation 1.1 for the single-task setting and in Equation 2.14 for the multi-task setting. The scale-invariant L_d and gradient L_g terms are respectively weighted $\alpha = 10$, $\beta = 0.1$; the optimal weights for the two terms corresponding to the depth estimation L_{main} and semantic segmentation L_{sec} tasks have been found to be $\delta = \sigma = 1$ through the same procedure defined in subsection 2.3.3.

3.3.2 Single-task decoders

First, we describe the UPer convolutional decoder proposed by [67] and adopted in [36] for semantic segmentation, since we initially used it after porting the method to the monocular depth estimation task (DeSwin); it is shown in Figure 3.2.

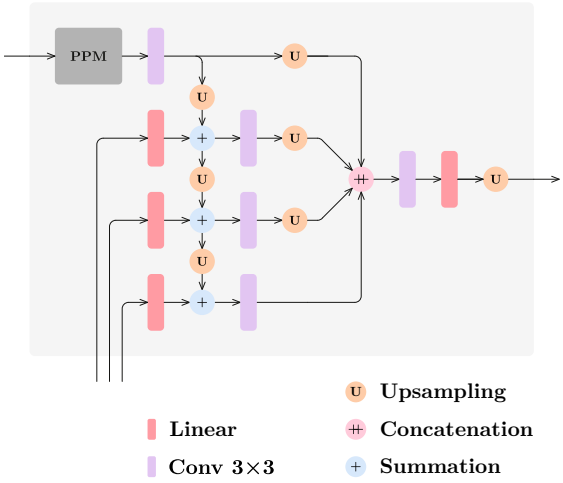


Figure 3.2: UPer decoder head.

I It takes in input the hierarchical multi-scale tokens produced by the four encoding stages, reshaped in 2D. The deepest feature, which is the lowest resolution and semantically highest-level one, is fed into a Pyramid Pooling Module (PPM) [77], which applies $\{1,2,3,6\}$ average pooling followed by a linear transformation in four parallel branches, concatenates their upsampled representation together with the input feature map and merges them through a convolutional layer. The other three skip connections are linearly refined and all four resulting feature maps are aggregated in a top-down manner via element-wise summation. They are then separately fed to a convolutional layer, upsampled, channel-wise concatenated and passed to the final prediction layer.

Then, inspired by the two decoder architectures proposed in [78] (one of which we used in chapter 2), we develop corresponding conceptually similar transformer-based versions. The general idea is to replace convolutional layers with windowed transformer blocks; in all designs described below, we adopt in each stage a number of sequential transformer blocks which is multiple of 2 to be able to leverage inter-window connectivity by alternating regular and shifted window configurations as in the encoder.

II The simplest decoder consists in taking in input only the output of the deepest stage of $1/32$ of the original resolution and Progressively UPsampling (PUPSwin, Figure 3.3) it, applying stacked transformer blocks between each interpolation. This decoder structure mimics the encoder one, with four stages respectively operating at $\{1/32, 1/16, 1/8, 1/4\}$ of the original resolution, each containing a sequence of 2 transformer blocks for a total of 8. The upsampling doubles the spatial resolution and halves the channel dimension; for this reason, the number of attention heads is accordingly adjusted to $\{48,24,12,6\}$ respectively in the first, second, third and fourth stage. This design does not leverage in any way the lower-level information provided by the shallow encoder stages through skip connections.

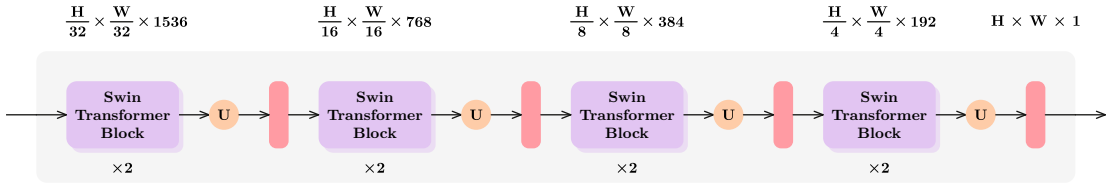


Figure 3.3: PUPSwin decoder.

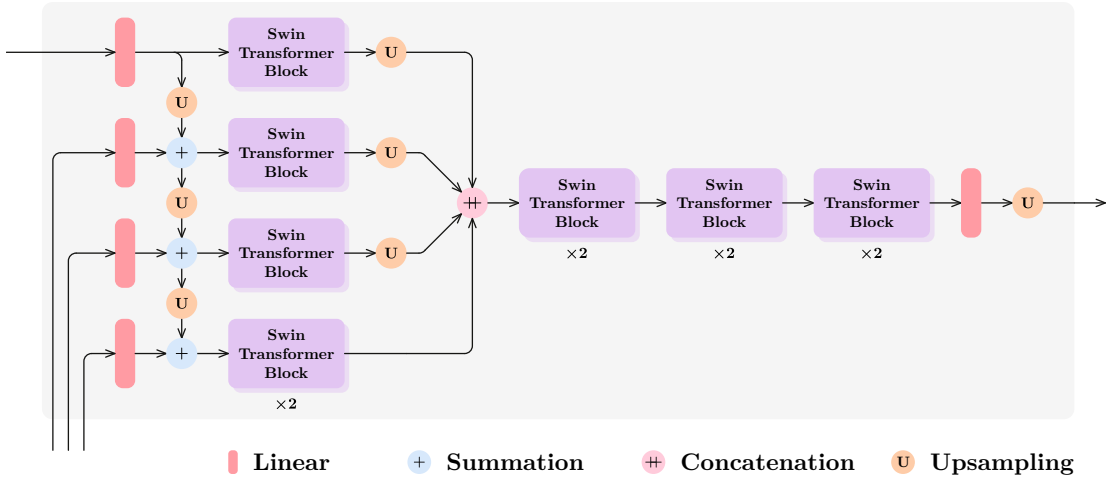


Figure 3.4: MLASwin decoder.

III The second adapted design is the Multi-Level feature Aggregation (MLASwin, Figure 3.4) described in subsection 2.3.1, which is also remarkably similar (apart from the PPM at the bottleneck) to the UPer previously described in this subsection. It takes in input the output of all four encoding stages, which are separately refined though a linear transformation, aggregated in a top-down fashion via element-wise summation and separately fed to parallel transformer branches; the feature maps are then upsampled, channel-wise concatenated, passed through a stack of transformer blocks and to the final prediction layer. Following the design by [78], the four parallel branches before upsampling and concatenation would contain distinct sequences of 6 transformer blocks each, while there would be a single sequence of 2 transformer blocks afterwards; this sums up to a total of 26, which exceeds the number of transformer encoders. We then experiment with distinct sequences of 2 transformer blocks in each parallel branch before upsampling and concatenation and a single sequence of 6 transformer blocks afterwards, for a more reasonable total of 14. However, it is important to underline that each parallel branch operates at the resolution of the corresponding encoded output, hence $\{1/32, 1/16, 1/8, 1/4\}$ of the original resolution in top-down order, while the stack after upsampling and concatenation operates at $1/4$ of the original resolution. The lower-resolution features are maintained by temporarily upsampling them when needed for the aggregation, but using their non-upsampled representation downstream until the concatenation.

Finally, we propose novel vision transformer-based decoders leveraging the finer information coming from the skip connections in an attentive way and in a more transformer-friendly style. The decoder template consists in four stages, each containing a sequence of 2 transformer blocks for a total of 8. Similarly to PUP, between consecutive stages, an upsampling layer doubles the spatial resolution and halves the channel dimension; for this reason, the number of attention heads is accordingly adjusted to $\{48, 24, 12, 6\}$ respectively in the first, second, third and fourth stage. The spatial/channel shape of the resulting feature maps matches the outputs of the encoder stages, which are delivered to the corresponding decoder stages by the skip connections. In this way we obtain an hourglass structure with mirrored encoder-decoder communication: the lower-resolution stages of the decoder are guided by the higher-level deeper encoded features and the higher-resolutions stages of the decoder are guided by the lower-level shallower encoded features, allowing to gradually recover information in a coarse to fine manner and to exploit the different semantic levels where they are more relevant. The difference between the various designs built on this template is the type of attention mechanism used to integrate the information contained in the encoded features into the decoding stream, while the overall structure of the windowed transformer block and its other components are unchanged. Each decoder stage receives in input the upsampled output of the previous stage x and the output of the encoding stage operating at the same resolution x_{sc}, through a skip connection. The first transformer block uses an even window partitioning and the second uses a shifted window partitioning; this is extendable to larger multiples of 2 by simply alternating the two configurations.

IV Independently by the number of stacked blocks in each considered stage, only the first one employs custom attention between x and x_{sc}, while all the subsequent ones apply windowed attention only on the output of the previous block. The CO-Attention (COA) mechanism is inspired by the transformer decoders used in NLP [61]. It computes a query q representation from the decoding stream x, and key k_{sc} and value v_{sc} representations from the skip connection x_{sc}, as described in Equation 2.1; the attention matrices A_{co} are then computed using q and k_{sc} as described in Equations 2.2, 2.3 and the new tokens are produced as averages of the skip connection tokens with the weights dependent on their pair-wise similarity with the decoder tokens:

$$x' = A_{co} v_{sc}. \tag{3.1}$$

The stream used for the residual shortcut is the decoder one, hence Equation 2.6 becomes

$$y = x + COA(x, x_{sc}). \tag{3.2}$$

Notice that, in this way, the information contained in the skip connection x_{sc} flows to y through the values v_{sc} controlled by the co-attention A_{co}, while the information contained in the decoding stream x flows to y through the shortcut. The described changes to a vanilla transformer block are shown in Figure 3.5.

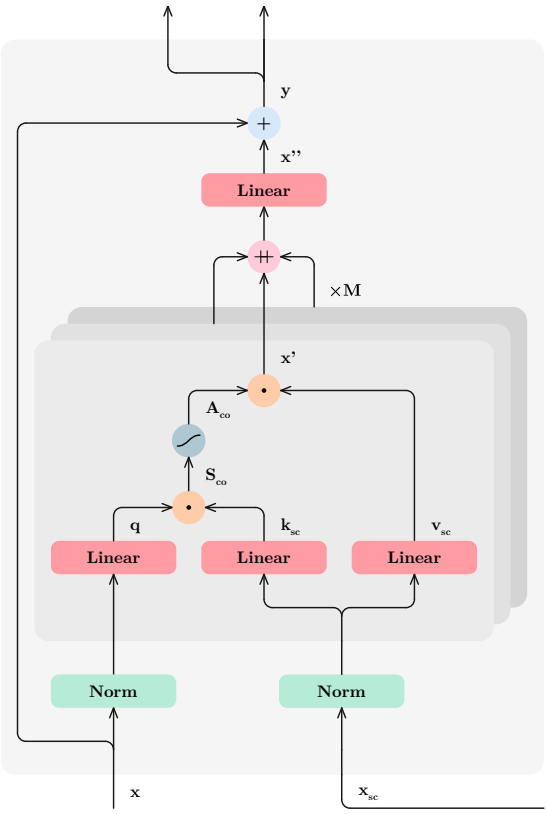


Figure 3.5: CO-Attention.

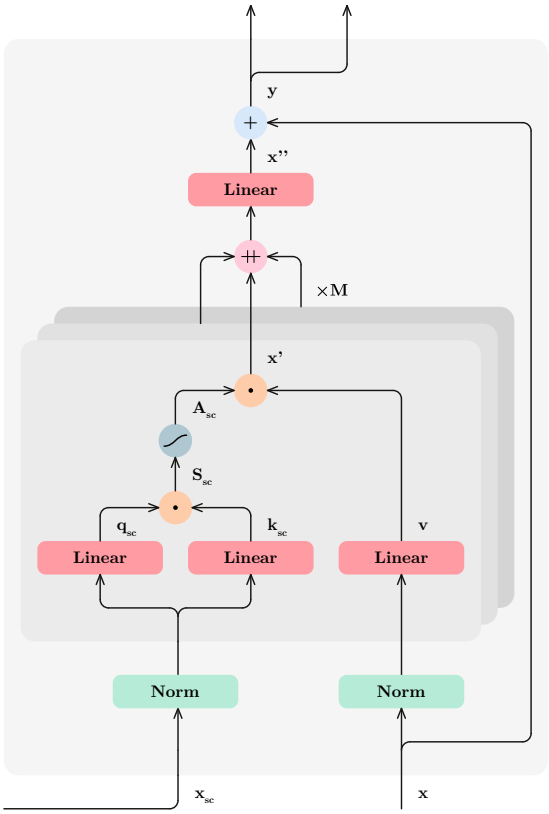


Figure 3.6: Skip-Connection-Attention v1.

V 1) Independently by the number of stacked blocks in each considered stage, only the first one employs custom attention between x and x_{sc} as shown in Figure 3.6, while all the subsequent ones apply windowed attention only on the output of the previous block. The Skip-Connection-Attention (SCA) mechanism is inspired by our previous work in subsection 2.3.2 and in particular by the variants that use only one of the two streams to calculate the attention. It computes query q_{sc} and key k_{sc} representations from the skip connection x_{sc} and a value representation v from the decoding stream x; the attention matrices A_{sc} are then computed using q_{sc} and k_{sc} as described in Equations 2.2, 2.3 and the new tokens are produced as averages of the decoded tokens with the weights dependent of the pair-wise similarity of the encoded tokens in their embedding space

$$x' = A_{sc} v. \tag{3.3}$$

Intuitively, this means that decoded tokens corresponding to similar tokens in the lower-level higher-resolution encoded feature maps will be considered more likely to have similar depth values. The stream used for the residual shortcut is the decoder one, hence Equation 2.6 becomes

$$y = x + SCA(x, x_{sc}). \tag{3.4}$$

Notice that the information contained in the skip connection x_{sc} cannot flow to y, while the information contained in the decoding stream x flows to y both through the values v controlled by the skip-connection-attention A_{sc} and through the shortcut.

2) To solve the lack of information flow from the skip connection x_{sc} to y, the value $v_{cat} = (x \mathbin{+\!\!+} x_{sc})\, W_{v,cat}^{T} + b_v$ representation is computed from the channel-wise concatenation of the two streams (in this case, the weights $W_{v,cat}$ are of shape $\big(C_{qkv}, 2C\big)$). This concatenation is performed only to compute the value, after the shortcut x is saved for summation in Equation 3.4.

3) While the skip connections provide very useful information to progressively recover spatial details as the resolution is increased, using it as the only flow controller in the attention computation has the effect of completely ignoring the semantically high-level relationships present in the decoder stream; this issue is particularly relevant in the last decoder stages, which receive shallow low-level features that do not necessarily match enough the actual complex context of the scene (two tokens similar in appearance can belong to distinct objects at different depths or, vice-versa, two tokens dissimilar in appearance can belong to the same object or to objects at the same depth). To improve the design in this direction and inspired by the merged variant in subsection 2.3.2, also query q_{cat} and key k_{cat} representations are computed from the channel-wise concatenation $x \mathbin{+\!\!+} x_{sc}$ of the two streams (in this case, the weights $W_{q,cat}$, $W_{k,cat}$ of the linear projections defined in Equation 2.1 are of shape $\big(C_{qkv}, 2C\big)$). Again, this concatenation is performed after the shortcut x is saved for summation in Equation 3.4.

4) Finally, we explored a denser encoder-decoder connectivity, by applying the skip-connection-attention between x and x_{sc} not only in the first transformer block of each stage, but to all stacked blocks, independently by their use of regular or shifted windowing. This is allowed by the improvement in the previous step, which, using the decoding stream also in the attention computation, does not oblige the following blocks to use self-attention to model relationships among the decoded tokens.

3.3.3 Multi-task decoders

The multi-task implementation of the decoder differs according to the type adopted. The two possible approaches are to either instantiate two separate identical heads as done in chapter 2 (one predicting a single channel followed by a sigmoid for the depth estimation task and one predicting K channels followed by a softmax for the semantic segmentation task) or to follow the philosophy of the quasi-two-stream encoder variants in subsection 2.3.2 and design a single head with custom attention to allow the interaction between the two task-specific decoding streams. The former strategy was first adopted to obtain MeSwin, whose decoder contains two convolutional UPer heads. Regarding the transformer-based decoders,

instead, here we focus on the latter strategy and, in particular, apply it to the SCASwin decoder presented in the previous section.

In version 1, the first transformer block in each stage already performs skip-connection-attention, hence it is left unchanged, with

$$x'_{se} = A_{sc} v_{se},\ x'_{de} = A_{sc} v_{de}, \tag{3.5}$$

$$y_{se} = x_{se} + SCA(x_{se}, x_{sc}),\ y_{de} = x_{de} + SCA(x_{de}, x_{sc}), \tag{3.6}$$

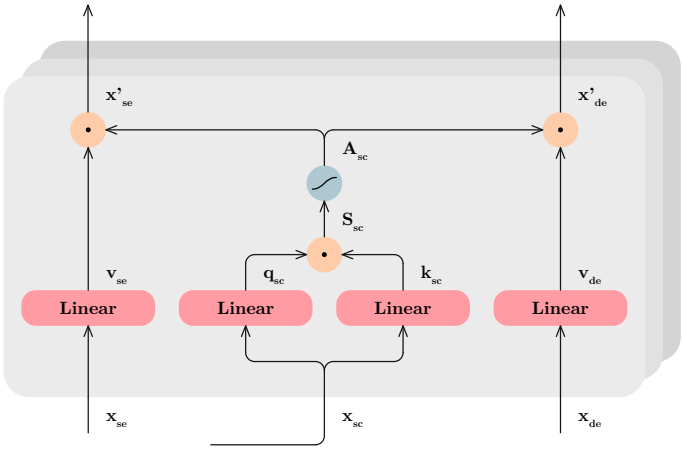
where value representations $v_{se},\ v_{de}$ are separately computed respectively from the semantic segmentation stream x_{se} and depth estimation stream x_{de} as described in Equation 2.1 and shown in Figure 3.7a. All other subsequent blocks (by default, 1) are modified to perform DEpth-Attention (for this reason, the resulting multi-task implementation of the SCASwin decoder is termed DEASwin): query q_{de}, key k_{de} and value v_{de} representations are computed from the depth estimation stream x_{de} as described in Equation 2.1 and a value representation $v_{se} = x_{se} W_{se}^T + b_{se}$ is separately computed from the semantic segmentation stream; the attention matrices A_{de} are then computed using q_{de} and k_{de} as described in Equations 2.2, 2.3 and the new tokens are produced as

$$x'_{se} = A_{de} v_{se},\ x'_{de} = A_{de} v_{de}. \tag{3.7}$$

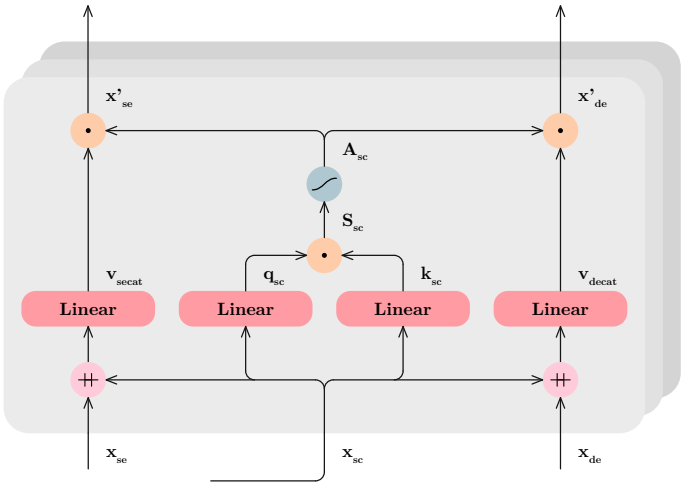
In the skip-connection-attention of version 2, value representations $v_{secat},\ v_{decat}$ are separately computed respectively from the channel-wise concatenation $x_{se} +\!\!+ x_{sc}$ of semantic segmentation stream and skip connection, and from the channel-wise concatenation $x_{de} +\!\!+ x_{sc}$ of depth estimation stream and skip connection, as shown in Figure 3.7b (in this case, the weights $W_{v,secat},\ W_{v,decat}$ of the linear projections defined in Equation 2.1 are of shape $(C_{qkv}, 2C)$).

In versions 3-4, we merge skip-connection- and depth- attention: query q_{decat} and key k_{decat} representations are computed from the channel-wise concatenation $x_{de} +\!\!+ x_{sc}$ of depth estimation stream and skip connection, as shown in Figure 3.7c (in this case, the weights $W_{q,decat},\ W_{k,decat}$ of the linear projections defined in Equation 2.1 are of shape $(C_{qkv}, 2C)$).

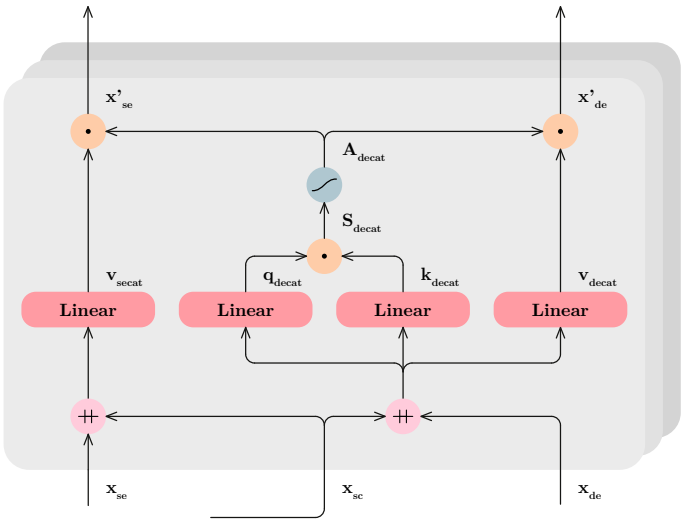
While adopting this type of attention in the encoder still allows the independent semantic segmentation head to effectively infer task-specific information to predict pixel classes, we expect that the combination of the task-invariant features learnt in a single-stream encoder and the lack of self-attention in the decoder (skip-connection- and depth- attention are alternated or merged) might result in a degradation of semantic segmentation performance. However, the main focus of this work is the monocular depth estimation task and not the secondary task, hence we choose this approach because it allows to completely remove at test time the semantic segmentation stream in the decoder; conversely, if desired, replacing depth-attention with segmentation-attention would allow to completely remove at test time the depth estimation stream in the decoder.



(a) Version 1



(b) Version 2



(c) Versions 3-4

Figure 3.7: Custom Skip-Connection-Attention for multi-task quasi-two-stream decoders.

3.4 Experiments

In this section we compare the performance of our proposed method with previous works on the challenging NYU Depth V2 [50], thanks to the presence of densely labelled maps for semantic segmentation. The main focus remains on the monocular depth estimation task, but we also report and discuss the results of the secondary semantic segmentation task.

3.4.1 Implementation details and training settings

The Pytorch [41] implementation is based on [36] on the 'mmsegmentation' [39, 40] toolbox. As previously mentioned, the adopted encoding backbone is Swin-L [36], initialized with the parameters pre-trained on ILSVRC [45] for image classification. The number of parameters is 197M in the encoder, and 36M, 77M, 3M, 77M, 95M respectively in the single-task UPer (used in DeSwin, SeSwin and MeSwin), PUPSwin, MLASwin, COASwin, SCA_4Swin (77M, 80M, 86M for versions 1-3) decoders; the total number of parameters is 166M in the multi-task DEA_4Swin (142M, 148M, 154M for versions 1-3) decoder, of which 71M (65M, 68M, 68M for versions 1-3) in the secondary task stream which can be removed at test time. The network training is 40k iterations long with a batch size of 16, corresponding to about 805 epochs; when the multi-task DEASwin (versions 1-4) decoders are used, the batch size is lowered to 8 due to memory constraints, hence the corresponding number of epochs is about 402. The optimizer used is AdamW [37], with weight decay set to 10^{-2} apart from the positional embedding and the norm layers, for which it is set to 0; the learning rate starts from $6 \cdot 10^{-6}$ and decreases linearly after a linear warmup of 1500 iterations.

During training, standard data augmentation is performed to increase learning robustness exactly as described in subsection 2.4.1, with random rotation, random cropping to 464×464, random horizontal flipping and random photometric distortion.

3.4.2 Performance evaluation

Table 3.2 reports the depth estimation performance of the previous works considered for comparison in subsection 2.4.2 and our best performing variation proposed in chapter 2, which we use as a first baseline. To be able to compare our proposed methods with the architecture by Liu et al. [36] (from which we started building), we ported their proposed model to the monocular depth estimation task (which we called depth shifted windows, or DeSwin), trained it several times with the same settings described in subsection 3.4.1 and report the best result. Table 3.2 contains the performance of all the single-task and multi-task decoders proposed in subsections 3.3.2, 3.3.3 and the relative percentage differences between the best performing ones (MeSwin and DEA_4Swin) and the DeSwin. To ensure a fair comparison, the performance of all methods mentioned above is obtained from the models being trained on the 795 frames of the official dataset of NYU Depth V2 [50]. For each metric, the best result is highlighted in bold and the second-best is underlined.

Table 3.2: Depth estimation quantitative evaluations on NYU Depth V2 [50].
The top parts contain results discussed in chapters 1, 2. The performance of DeSwin [36] and of our proposed variations (ordered as in section 3.3) corresponds to the best one obtained over multiple training runs. For each metric, the best result is highlighted in bold and the second-best is underlined; the relative percentage differences between our MeSwin, DEA_4Swin and DeSwin [36] are also included in the last two rows.

Method	**AbsDiff** ↓	**AbsRel** ↓	**log10** ↓	δ_1 ↑	δ_2 ↑	δ_3 ↑	**RMSE** ↓	**RMSElog** ↓
Ye et al. [72]	-	-	0.063	0.784	0.948	0.986	0.474	**0.081**
Song et al. [52] (L_k)	0.360	0.147	0.060	0.809	0.965	0.992	0.481	0.175
Ours ($L_k \otimes C_k$)	0.360	0.141	0.060	0.815	0.965	0.993	0.485	0.174
Zheng et al. [78] (DEPTR)	0.361	0.129	0.055	0.850	0.971	0.991	0.485	0.161
Ours (METR)	0.353	0.126	0.054	0.855	0.973	0.993	0.475	0.157
Liu et al. [36] (DeSwin)	<u>0.334</u>	0.125	0.052	0.861	<u>0.977</u>	0.994	<u>0.443</u>	0.151
Ours (MeSwin)	**0.331**	<u>0.123</u>	**0.051**	**0.865**	**0.977**	**0.995**	**0.440**	<u>0.150</u>
Ours (PUPSwin)	0.344	0.127	0.053	0.855	0.975	0.993	0.460	0.156
Ours (MLASwin)	0.342	0.128	0.053	0.856	0.976	0.994	0.454	0.155
Ours (COASwin)	0.341	0.126	0.053	0.859	0.974	0.993	0.457	0.155
Ours (SCA_1Swin)	0.339	0.126	0.053	0.860	0.975	0.994	0.453	0.154
Ours (SCA_2Swin)	0.339	0.126	0.053	0.860	0.976	0.994	0.452	0.154
Ours (SCA_3Swin)	0.338	0.124	0.052	0.861	0.976	0.994	0.452	0.153
Ours (SCA_4Swin)	0.337	0.125	0.052	0.862	0.976	0.994	0.449	0.153
Ours (DEA_1Swin)	0.336	0.125	0.052	0.864	0.977	0.994	0.449	0.152
Ours (DEA_2Swin)	0.335	0.125	0.052	0.861	0.976	0.994	0.448	0.152
Ours (DEA_3Swin)	0.336	0.124	0.052	0.863	0.976	0.994	0.449	0.152
Ours (DEA_4Swin)	0.335	**0.122**	<u>0.052</u>	<u>0.865</u>	0.976	<u>0.994</u>	0.448	0.152
MeSwin vs DeSwin	-1.15%	-1.59%	-1.32%	+0.50%	+0.01%	+0.03%	-0.73%	-0.96%
DEA_4Swin vs DeSwin	+0.19%	-2.27%	-0.29%	+0.40%	-0.07%	+0.01%	+1.15%	+0.29%

Table 3.3 reports the semantic segmentation performance of the previous works considered for comparison in subsection 2.4.2, our best performing variation proposed in chapter 2 and the original architecture by Liu et al. [36]. It also contains the performance of all the multi-task decoders proposed in subsection 3.3.3 and, as a supplementary baseline, of the single-task SCA_4Swin; the relative percentage differences between the best performing ones (MeSwin and SCA_4Swin) and the original method [36] can be found in the last two rows. Note that the best performance over several runs with the same settings described in subsection 3.4.1 is reported for each single-task decoder (among which [36]), while the semantic segmentation performance of each of our multi-task decoders corresponds to the same run selected for its best performance on the depth estimation task and reported in Table 3.2; for each metric, the best result is highlighted in bold and the second-best is underlined.

Starting from the convolutional decoders, we first note that porting the method by Liu et al. [36] to the monocular depth estimation task (DeSwin) outperforms in all metrics our previously best method proposed in chapter 2 (METR); in particular, it achieves considerable improvements of -5.27% in AbsDiff, -3.28% in log10, -6.78% in RMSE and -3.69% in RMSElog. Considering the similarity of the decoders used in the two architectures, this increase in performance is most likely due to the pyramidal structure of the transformer-based encoder,

Table 3.3: Semantic segmentation quantitative evaluations on NYU Depth V2 [50]. The top part contains results discussed in chapter 2. The performance of [36] and our SCA_4Swin corresponds to the best one obtained over multiple training runs, while of our other proposed variations (ordered as in section 3.3) corresponds to the same run reported in Table 3.2. For each metric, the best result is highlighted in bold and the second-best is underlined; the relative percentage differences between our MeSwin, SCA_4Swin and [36] are also included in the last two rows.

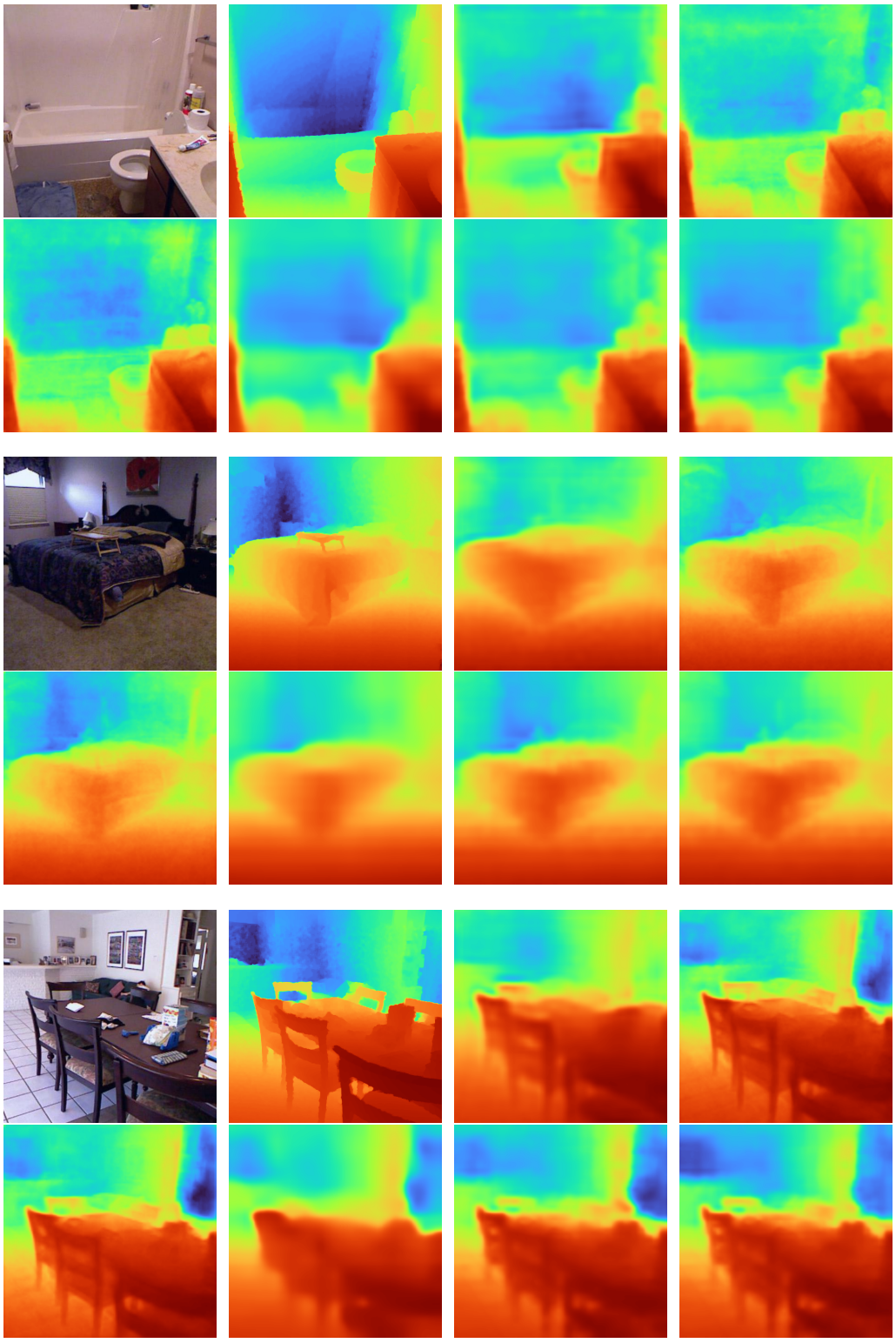
Method	**mIoU ↑**	**mAcc ↑**	**aAcc ↑**
Zheng et al. [78] (SETR)	0.4754	0.5866	0.7150
Ours (METR)	0.4878	0.6139	0.7569
Liu et al. [36] (SeSwin)	0.5405	0.6841	0.7609
Ours (MeSwin)	**0.5572**	**0.6888**	**0.7916**
Ours (SCA_4Swin)	<u>0.5475</u>	<u>0.6844</u>	<u>0.7876</u>
Ours (DEA_1Swin)	0.5281	0.6562	0.7787
Ours (DEA_2Swin)	0.5261	0.6544	0.7761
Ours (DEA_3Swin)	0.5276	0.6573	0.7764
Ours (DEA_4Swin)	0.5230	0.6563	0.7750
MeSwin vs SeSwin	+3.09%	+0.69%	+4.03%
SCA_4Swin vs SeSwin	+1.30%	+0.04%	+3.51%

which improves the quality of the encoded features delivered to the decoder with respect to the columnar structure previously adopted. Moreover, we further validate the results obtained in chapter 2, by applying to DeSwin the best multi-task strategy among the proposed and tested ones: the use of a shared single-stream encoder and of two separate task-specific heads provides an even better performing model (MeSwin), which further improves results in all metrics; in particular, by -1.15% in AbsDiff, -1.59% in AbsRel and -1.32% in log10. Analysing the transformer-based decoders in order, PUPSwin is the worst performing one as expected, but still achieves better results than METR; this outcome is interesting, because it confirms that the pyramidal structure of the backbone allows the learning of better deep features in addition to producing higher-resolution skip connections (that are not leveraged in this decoder). MLASwin performs better, by taking advantage of the high-quality skip connections coming from the encoder, but still worse than DeSwin and its convolutional decoder. To reach comparable performance, a change of paradigm is proposed in the design of the decoder, which makes use of the information contained in the skip connections in an attentive way. The first attempt, COASwin, performs comparably to MLASwin; this is probably due to the sub-optimal computation of attention during the integration of the shortcuts with the decoding stream, since they contain semantically different information difficult to effectively compare with each other. This supposition is confirmed by the better performance of SCA_1Swin, in which the attention matrix is computed solely by the skip connection and used to gate the information flow of the decoding stream. This design is progressively refined in SCA_2Swin, SCA_3Swin and SCA_4Swin, which show that allowing also the flow of the information contained in the skip connections and using also the decoder stream in the computation of the attention

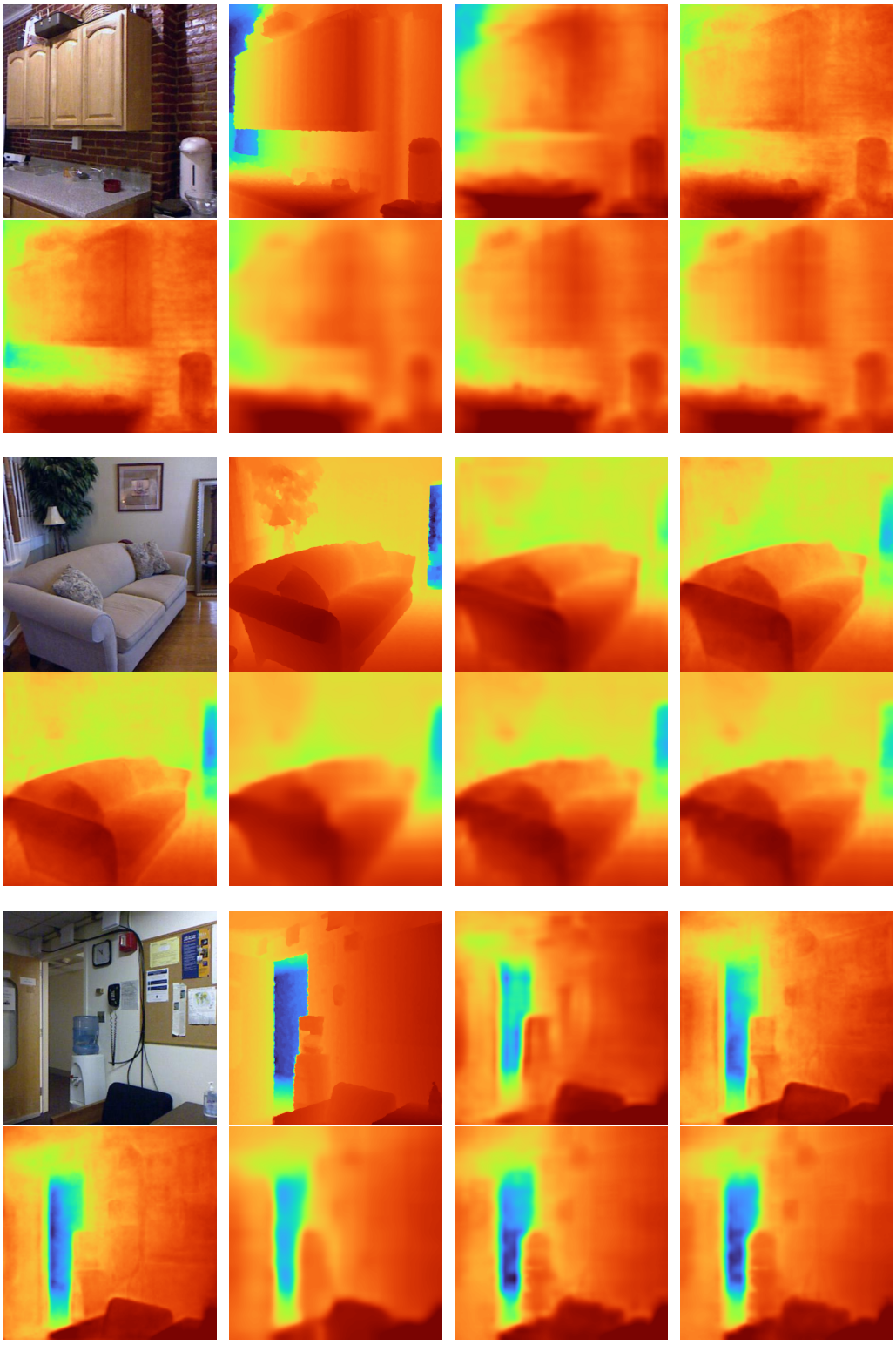
matrix improves results. Finally, the multi-task quasi-two-stream decoders (DEASwin) achieve higher performance than the respective single-task variants, but the increase is lower than the one between DeSwin and MeSwin; part of the reason may lay in the design choice of never using the semantic segmentation stream in the computation of the attention matrix, which could have caused a sub-optimal learning of the secondary task and the consequent decrease of benefit for the main task from the interaction with it. However, DEA_4Swin achieves the best performance in AbsRel, with an improvement of -2.27% on DeSwin and -0.70% on MeSwin.

From the semantic segmentation perspective, the method by Liu et al. [36] (SeSwin) outperforms in all metrics our methods proposed in chapter 2; in particular, it achieves improvements of +10.37% in mIoU on A_{de}, +10.66% in mAcc on A_{me} and +0.53% in aAcc on METR. As found in chapter 2, the multi-task setting benefits the semantic segmentation task too: MeSwin results being the best performing method among the analyzed and proposed ones, with improvements on SeSwin of +3.09% in mIoU, +0.69% in mAcc and +4.03% in aAcc. The final version of our novel transformer-based single-task decoder SCASwin also outperforms SeSwin in all metrics, but with smaller margins; in particular, the increase of +1.30% in mIoU and +3.51% in aAcc proves that the proposed decoder and the resulting fully-transformer architecture can compete with and surpass convolutional alternatives. Finally, the various versions of the multi-task quasi-two-stream decoder (DEASwin) overall perform slightly worse than SeSwin, all achieveing better results only in aAcc; this outcome exceeds our expectations, given the design focused on the monocular depth estimation main task.

Figure 3.8 shows a comparison of the qualitative depth estimation results obtained with our best method from chapter 2 (METR), the method by Liu et al. [36] (DeSwin) and our proposed models with respect to the ground truth: red represents a shorter depth, while blue represents a longer depth; all results are scaled according to the minimum and maximum values in the corresponding ground truth map for visualization purposes, hence the same color could correspond to different depth values in different scenes. Figure 3.9 shows a comparison of the respective error maps: intense red represents overestimated depths, while intense blue represents underestimated depths, hence a whiter error map corresponds to a more accurate prediction. Starting from the worst performing model among those compared here, COASwin already improves on the baseline set by the previous chapter (METR); for example, in the living room scene (fifth group), the sofa in the foreground is similarly predicted, but the plant in the background, the wall behind it and the reflection in the mirror are much closer to the ground truth. Next, the results output by SCA_4Swin show a more detailed understanding of object shapes and structural geometry, as can be seen in the dining room scene (third group), in which the chairs in the foreground are better defined and the distances from the various parts of the wall in the background are more accurate. DEA_4Swin and DeSwin have comparable performances: in particular, DEA_4Swin produces overall more correct predictions, like in the bathroom scene (first group), while DeSwin shows less blurred details; however, these finer changes in the prediction sometimes seem to correspond more to changes in appearance in the RGB image rather than to actual changes in depth in the ground truth, such as in the kitchen scene (fourth group) on the wall. Moreover, DEA_4Swin proves capable of

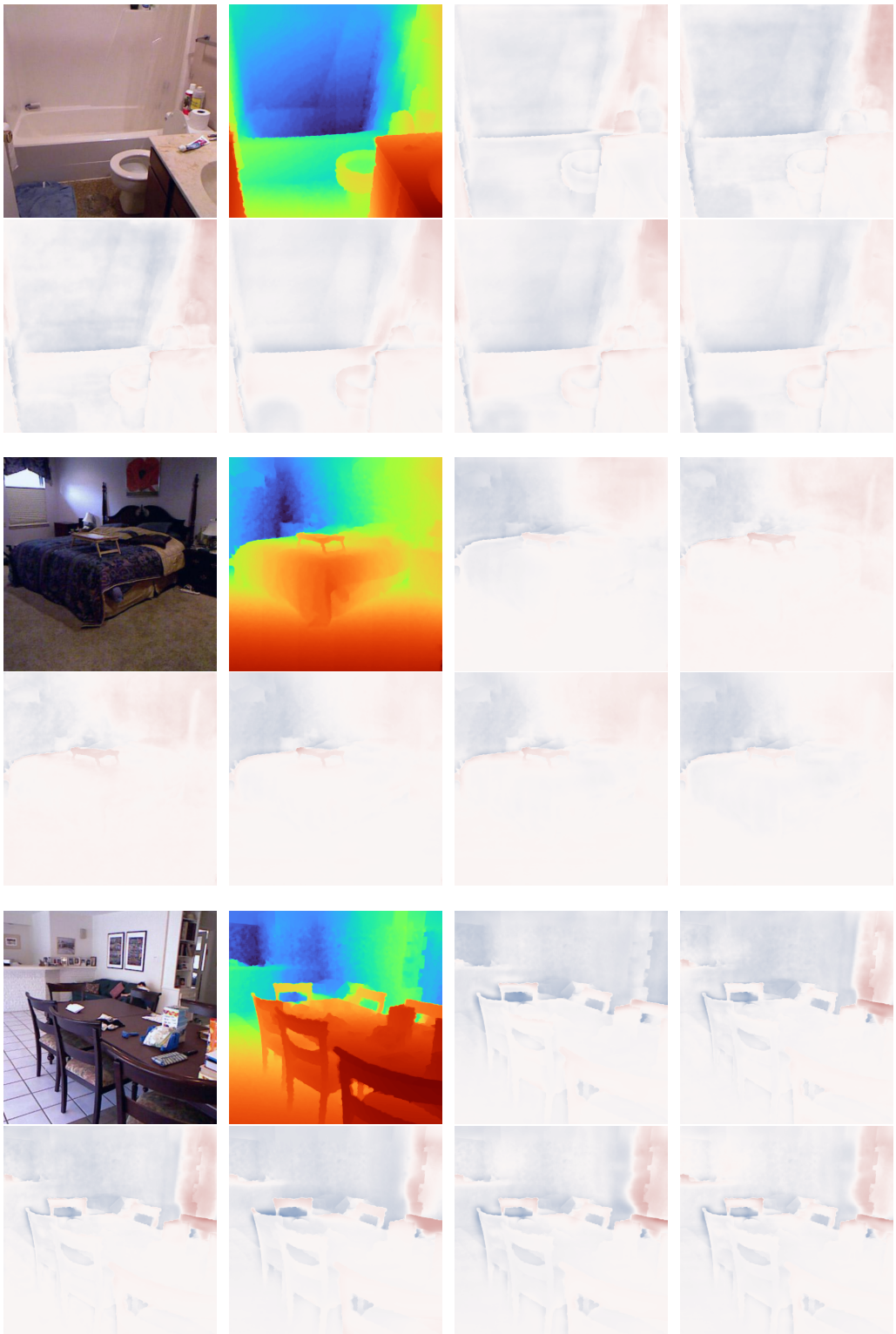


(a) RGB (b) GT (c) Ours (METR) (d) Liu et al. [36] (DeSwin)
(e) Ours (MeSwin) (f) Ours (COASwin) (g) Ours (SCA_4Swin) (h) Ours (DEA_4Swin)

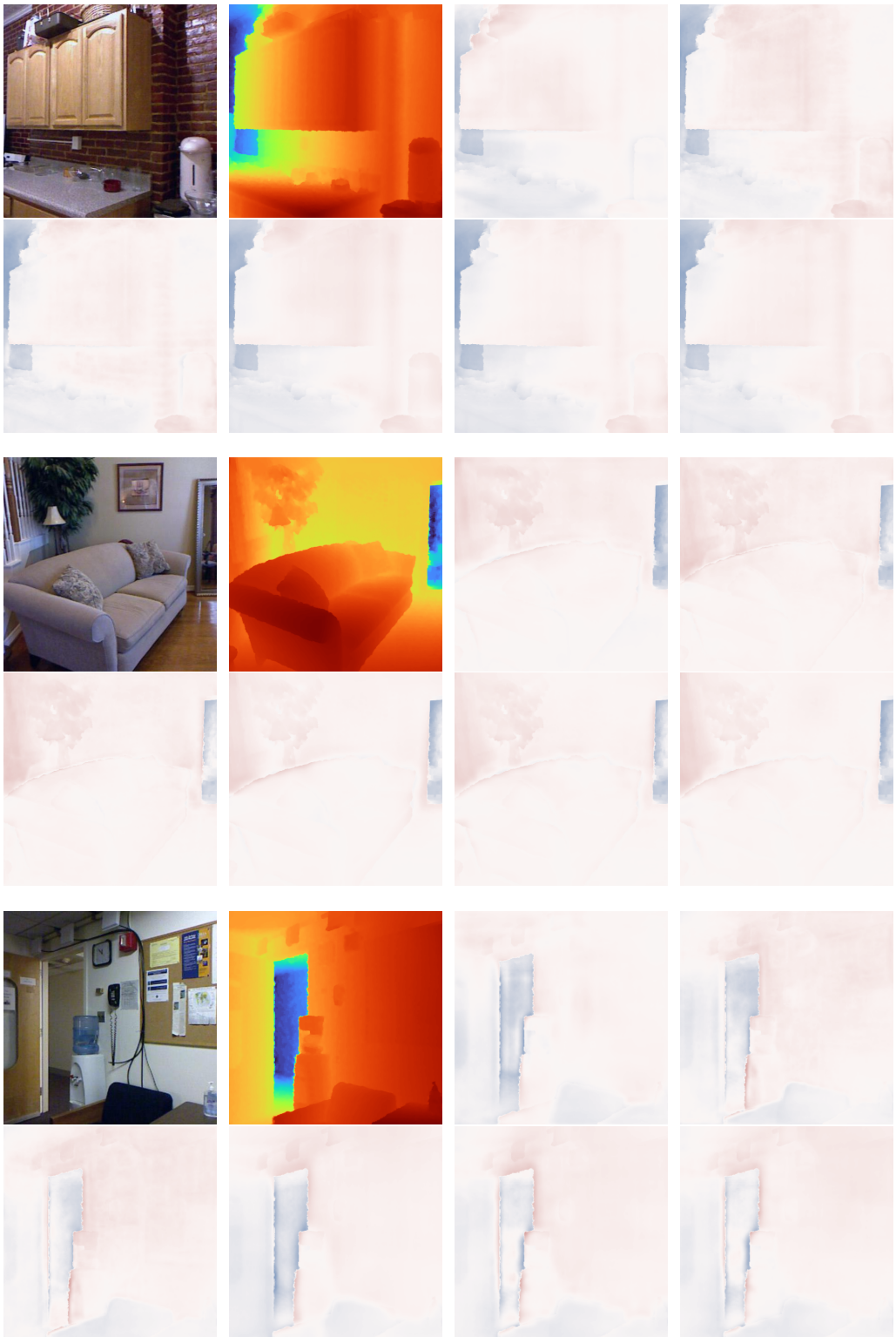


(a) RGB (b) GT (c) Ours (METR) (d) Liu et al. [36] (DeSwin)
(e) Ours (MeSwin) (f) Ours (COASwin) (g) Ours (SCA_4Swin) (h) Ours (DEA_4Swin)

Figure 3.8: Qualitative comparison of depth estimation results on NYU Depth V2 [50].

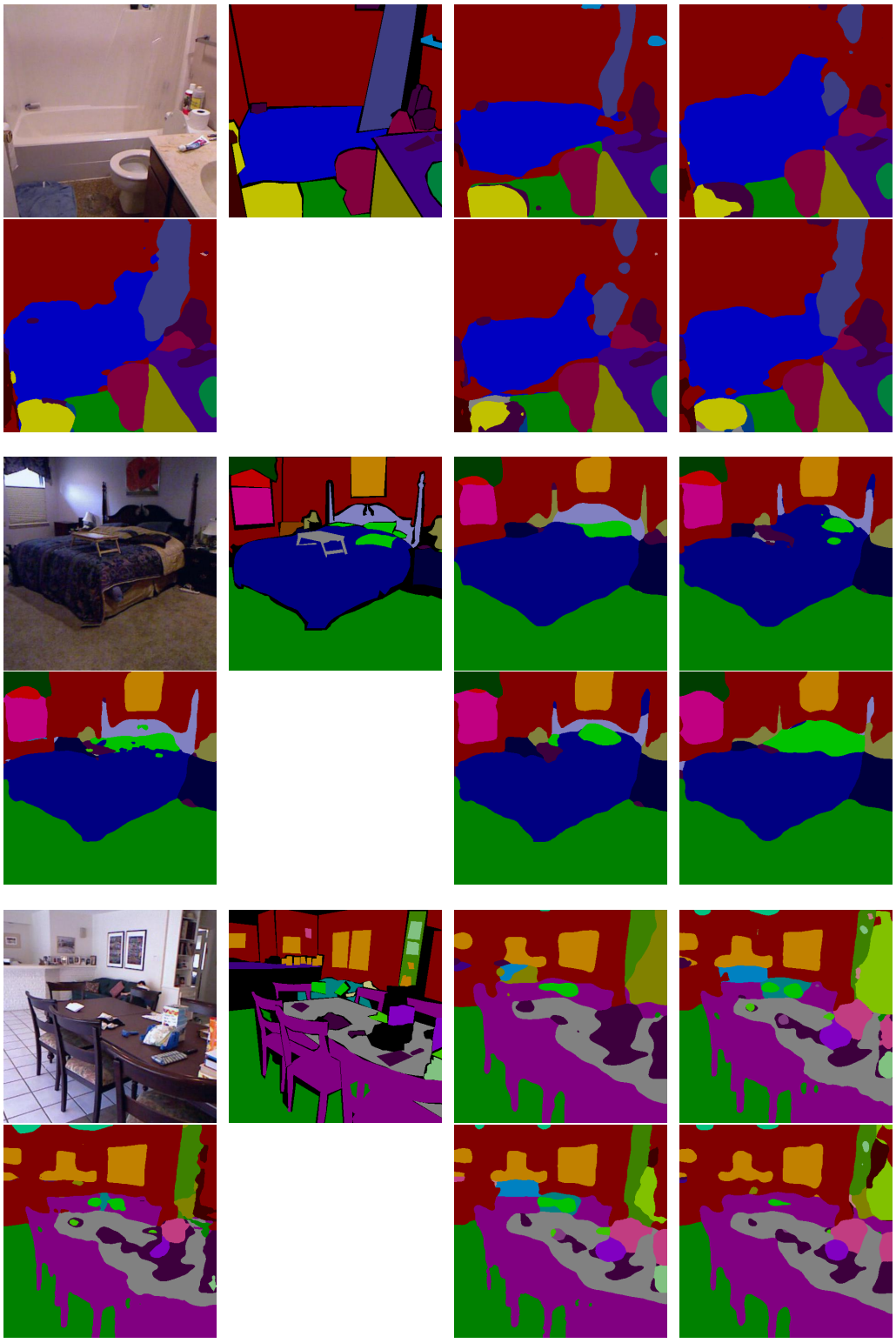


(a) RGB (b) GT (c) Ours (METR) (d) Liu et al. [36] (DeSwin)
(e) Ours (MeSwin) (f) Ours (COASwin) (g) Ours (SCA_4Swin) (h) Ours (DEA_4Swin)

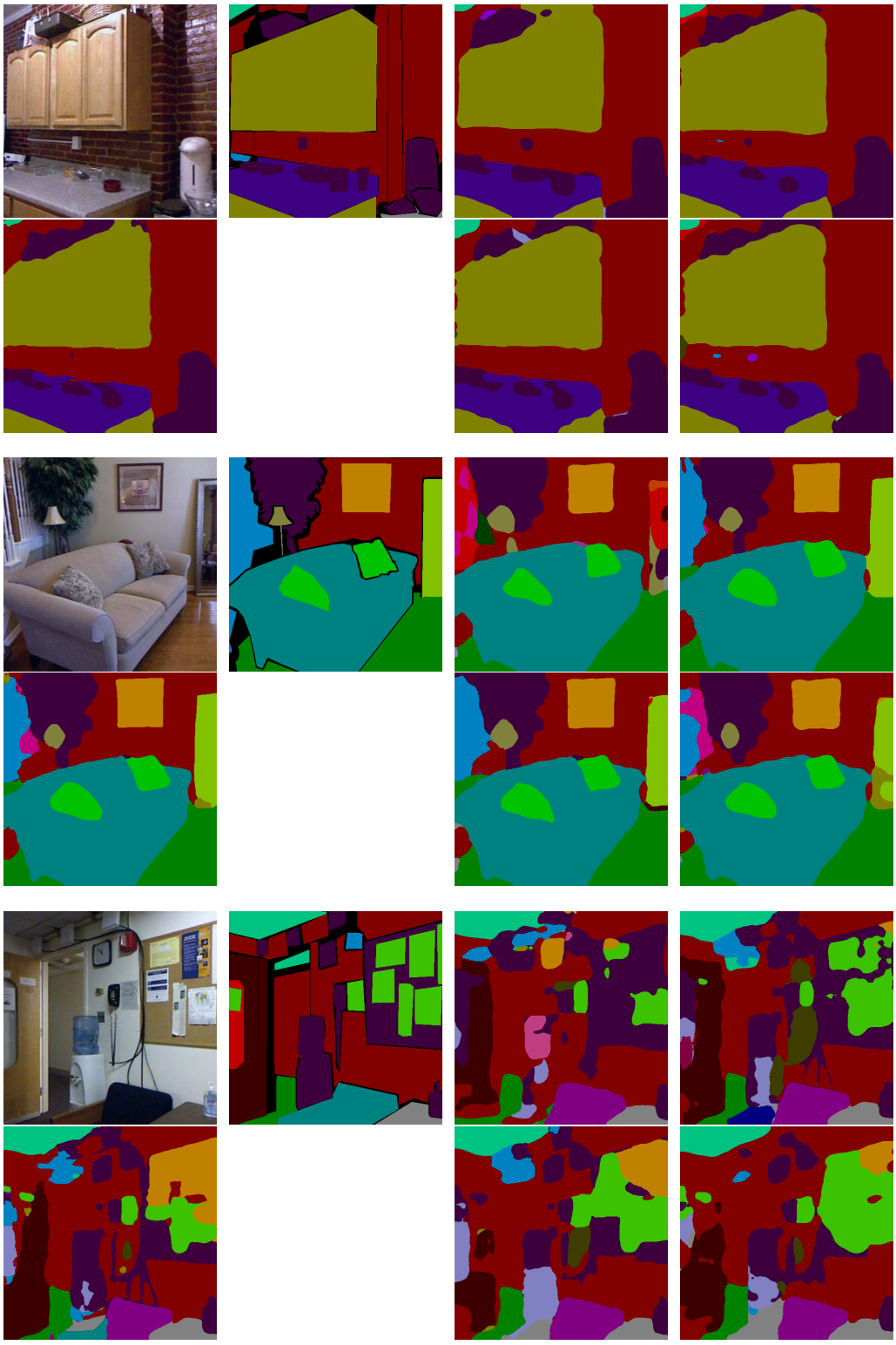


(a) RGB (b) GT (c) Ours (METR) (d) Liu et al. [36] (DeSwin)
(e) Ours (MeSwin) (f) Ours (COASwin) (g) Ours (SCA_4Swin) (h) Ours (DEA_4Swin)

Figure 3.9: Qualitative comparison of error maps on NYU Depth V2 [50].



(a) RGB (b) GT (c) Ours (METR) (d) Liu et al. [36] (SeSwin)
(e) Ours (MeSwin) (f) Ours (SCA_4Swin) (g) Ours (DEA_4Swin)



(a) RGB (b) GT (c) Ours (METR) (d) Liu et al. [36] (SeSwin)
(e) Ours (MeSwin) (f) Ours (SCA_4Swin) (g) Ours (DEA_4Swin)

Figure 3.10: Qualitative comparison of semantic segmentation results on NYU Depth V2 [50].

better recognising small objects, as the bed tray in the bedroom scene (second group). Finally, MeSwin often produces the best depth maps, but suffers from the same issue as DeSwin. Figure 3.10 shows a comparison of the qualitative semantic segmentation results obtained with our method from chapter 2 (METR), the method by Liu et al. [36] (SeSwin) and our proposed models with respect to the ground truth. First, the significative improvement of SeSwin over METR is confirmed by the produced segmentation masks, as can be clearly seen in the living room (fifth group) and office (sixth group) scenes. SCA_4Swin further improves the shape of the predicted object masks, as shown for example in the bedroom scene (second group). Overall, MeSwin produces the best outputs, accurately classifying both large structural elements and smaller objects, like the doorway and the items on the table and on the shelf in the dining room scene (third group). Finally, DEA_4Swin exceeds our expectations, by producing segmentation masks visually competitive with the quantitatively better methods, both in scenes with few elements, such as the kitchen one (fourth group), and in more complex ones, such as the bathroom one (first group).

3.5 Conclusions

In this chapter, we modified the structure of both the encoder and the decoder. First, we replaced the columnar transformer encoder with a pyramidal backbone. The difference between the transformer blocks stacked in the two models is that the former computes global attention on the input features, while the latter computes closer range intra-window attention; this is a trade-off between the distance at which dependencies can be learnt and computational and memory complexity: limiting the attention computation inside windows of fixed size is linear with respect to the input resolution instead of quadratic, but does not allow tokens to attend to other tokens outside their window of belonging. For this reason, stacks of a window and a shifted-window transformer blocks sequentially applied are the basic unit employed to allow inter-window interactions. Extensive experiments in semantic segmentation and depth estimation confirm that the lack of global attention in each block does not negatively affect the quality of the produced features, because of the high correlation present in visual inputs, of the larger receptive field with respect to common convolutional layers and of the faster enlargement of the virtual receptive field. Most importantly, the reduction of complexity allows to apply this modules on higher-resolution feature maps and to consequently extract higher-quality multi-scale information, especially relevant for dense prediction tasks as demonstrated in the obtained results. However, the use of windows worsens interpretability with respect to global attention.

Additionally, we proposed several single- and multi- task decoders based on the same windowed transformer block adopted in the encoder, since it allows to apply attention at every decoding stage, regardless of the resolution obtained after multiple upsampling operations. First, we modified the implementation of some existing decoders, by replacing the convolutional layers with transformer blocks; however, while we still surpassed the baseline set by the previous chapter, the resulting modules could not reach the performance of the convolutional

decoder considered here for comparison. Therefore, we designed novel transformer-based decoders able to better leverage the information contained in the skip connections, by effectively integrating them with the decoding stream through custom attention mechanisms. In particular, the design inspired by our shared attention proposed in chapter 2, with the attention matrix computed by the skip connection, and its following versions (SCASwin) manage to achieve comparable or only slightly worse results without the use of any convolutional layer in the whole architecture, proving the potential of carefully designed transformer-based methods in competing with the dominating paradigm in both parts of the typical encoder-decoder structure. This performance is further improved by the proposed multi-task quasi-two-stream decoder, whose final version (DEA_4Swin) defines our overall best fully-transformer model. Moreover, the results obtained in our previous work, regarding the benefits in the joint learning of depth estimation and semantic segmentation, are further validated by it and our best multi-task model with convolutional decoder (MeSwin).

Bibliography

[1] Aleotti, F., Tosi, F., Poggi, M., and Mattoccia, S. (2018). Generative adversarial networks for unsupervised monocular depth prediction. In *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*.

[2] Benosman, R., Manière, T., and Devars, J. (1996). Multidirectional stereovision sensor, calibration and scenes reconstruction. In *Proceedings of 13th International Conference on Pattern Recognition*, volume 1, pages 161–165. IEEE.

[3] Bhat, S. F., Alhashim, I., and Wonka, P. (2021). Adabins: Depth estimation using adaptive bins. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4009–4018.

[4] Canny, J. (1986). A computational approach to edge detection. *IEEE Transactions on pattern analysis and machine intelligence*, 8(6):679–698.

[5] Cao, Y., Wu, Z., and Shen, C. (2017). Estimating depth from monocular images as classification using deep fully convolutional residual networks. *IEEE Transactions on Circuits and Systems for Video Technology*, 28(11):3174–3182.

[6] Casser, V., Pirk, S., Mahjourian, R., and Angelova, A. (2019a). Depth prediction without the sensors: Leveraging structure for unsupervised learning from monocular videos. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33(01), pages 8001–8008.

[7] Casser, V., Pirk, S., Mahjourian, R., and Angelova, A. (2019b). Unsupervised monocular depth and ego-motion learning with structure and semantics. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*.

[8] Chen, X., Chen, X., and Zha, Z. (2019). Structure-aware residual pyramid network for monocular depth estimation. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, pages 694–700.

[9] Chen, Y., Zhao, H., Hu, Z., and Peng, J. (2021). Attention-based context aggregation network for monocular depth estimation. *International Journal of Machine Learning and Cybernetics*, 12(6):1583–1596.

[10] Das, S. and Ahuja, N. (1995). Performance analysis of stereo, vergence, and focus as depth cues for active vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(12):1213–1219.

[11] Delage, E., Lee, H., and Ng, A. Y. (2006). A dynamic bayesian network model for autonomous 3d reconstruction from a single indoor image. In *2006 IEEE computer society conference on computer vision and pattern recognition (CVPR'06)*, volume 2, pages 2418–2428. IEEE.

[12] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*.

[13] Eigen, D. and Fergus, R. (2015). Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *Proceedings of the IEEE international conference on computer vision*, pages 2650–2658.

[14] Eigen, D., Puhrsch, C., and Fergus, R. (2014). Depth map prediction from a single image using a multi-scale deep network. *Advances in Neural Information Processing Systems*, 27:2366–2374.

[15] Fu, H., Gong, M., Wang, C., Batmanghelich, K., and Tao, D. (2018). Deep ordinal regression network for monocular depth estimation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2002–2011.

[16] Garg, R., Bg, V. K., Carneiro, G., and Reid, I. (2016). Unsupervised cnn for single view depth estimation: Geometry to the rescue. In *European conference on computer vision*, pages 740–756. Springer.

[17] Geiger, A., Lenz, P., Stiller, C., and Urtasun, R. (2013). Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237.

[18] Geiger, A., Lenz, P., and Urtasun, R. (2012). Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE conference on computer vision and pattern recognition*, pages 3354–3361. IEEE.

[19] Godard, C., Mac Aodha, O., and Brostow, G. J. (2017). Unsupervised monocular depth estimation with left-right consistency. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 270–279.

[20] Godard, C., Mac Aodha, O., Firman, M., and Brostow, G. J. (2019). Digging into self-supervised monocular depth estimation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3828–3838.

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

[22] He, K., Zhang, X., Ren, S., and Sun, J. (2016b). Identity mappings in deep residual networks. In *European conference on computer vision*, pages 630–645. Springer.

[23] Hoiem, D., Efros, A. A., and Hebert, M. (2005). Geometric context from a single image. In *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, volume 1, pages 654–661. IEEE.

[24] Horn, B. K. and Brooks, M. J. (1986). The variational approach to shape from shading. *Computer Vision, Graphics, and Image Processing*, 33(2):174–208.

[25] Horn, B. K. P., Szeliski, R. S., and Yuille, A. L. (1993). Impossible shaded images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(2):166–170.

[26] Javidnia, H. and Corcoran, P. (2017). Accurate depth map estimation from small motions. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 2453–2461.

[27] Karsch, K., Liu, C., and Kang, S. B. (2012). Depth extraction from video using non-parametric sampling. In *European conference on computer vision*, pages 775–788. Springer.

[28] Kim, D., Lee, S., Lee, J., and Kim, J. (2020). Leveraging contextual information for monocular depth estimation. *IEEE Access*, 8:147808–147817.

[29] Konrad, J., Wang, M., and Ishwar, P. (2012). 2d-to-3d image conversion by learning depth from examples. In *2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pages 16–22. IEEE.

[30] Laina, I., Rupprecht, C., Belagiannis, V., Tombari, F., and Navab, N. (2016). Deeper depth prediction with fully convolutional residual networks. In *2016 Fourth international conference on 3D vision (3DV)*, pages 239–248. IEEE.

[31] Lee, J. H., Han, M.-K., Ko, D. W., and Suh, I. H. (2019). From big to small: Multi-scale local planar guidance for monocular depth estimation. *arXiv preprint arXiv:1907.10326*.

[32] Levin, A., Lischinski, D., and Weiss, Y. (2004). Colorization using optimization. In *ACM SIGGRAPH 2004 Papers*, pages 689–694. Association for Computing Machinery.

[33] Liu, F., Shen, C., Lin, G., and Reid, I. (2015). Learning depth from single monocular images using deep convolutional neural fields. *IEEE transactions on pattern analysis and machine intelligence*, 38(10):2024–2039.

[34] Liu, M., Salzmann, M., and He, X. (2014). Discrete-continuous depth estimation from a single image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 716–723.

[35] Liu, P., Zhang, Z., Meng, Z., and Gao, N. (2021a). Monocular depth estimation with joint attention feature distillation and wavelet-based loss function. *Sensors*, 21(1):54.

[36] Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., and Guo, B. (2021b). Swin transformer: Hierarchical vision transformer using shifted windows. *arXiv preprint arXiv:2103.14030*.

[37] Loshchilov, I. and Hutter, F. (2018). Decoupled weight decay regularization. In *International Conference on Learning Representations*.

[38] Mahjourian, R., Wicke, M., and Angelova, A. (2018). Unsupervised learning of depth and ego-motion from monocular video using 3d geometric constraints. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5667–5675.

[39] MMCV Contributors (2018). MMCV: OpenMMLab computer vision foundation. https://github.com/open-mmlab/mmcv.

[40] MMSegmentation Contributors (2020). MMSegmentation: OpenMMLab semantic segmentation toolbox and benchmark. https://github.com/open-mmlab/mmsegmentation.

[41] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

[42] Qiao, S., Wang, H., Liu, C., Shen, W., and Yuille, A. (2019). Micro-batch training with batch-channel normalization and weight standardization. *arXiv preprint arXiv:1903.10520*.

[43] Ramachandran, P., Parmar, N., Vaswani, A., Bello, I., Levskaya, A., and Shlens, J. (2019). Stand-alone self-attention in vision models. *Advances in Neural Information Processing Systems*, 32.

[44] Ranftl, R., Bochkovskiy, A., and Koltun, V. (2021). Vision transformers for dense prediction. *arXiv preprint arXiv:2103.13413*.

[45] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al. (2015). Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252.

[46] Saxena, A., Chung, S. H., Ng, A. Y., et al. (2005). Learning depth from single monocular images. In *NIPS*, volume 18, pages 1–8.

[47] Scharstein, D. and Pal, C. (2007). Learning conditional random fields for stereo. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE.

[48] Scharstein, D. and Szeliski, R. (2002). A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International journal of computer vision*, 47(1):7–42.

[49] Silberman, N. and Fergus, R. (2011). Indoor scene segmentation using a structured light sensor. In *2011 IEEE international conference on computer vision workshops (ICCV workshops)*, pages 601–608. IEEE.

[50] Silberman, N., Hoiem, D., Kohli, P., and Fergus, R. (2012). Indoor segmentation and support inference from rgbd images. In *European conference on computer vision*, pages 746–760. Springer.

[51] Simonyan, K. and Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

[52] Song, M., Lim, S., and Kim, W. (2021). Monocular depth estimation using laplacian pyramid-based depth residuals. *IEEE Transactions on Circuits and Systems for Video Technology*.

[53] Subbarao, M., Choi, T.-S., and Nikzad, A. (1993). Focusing techniques. *Optical Engineering*, 32(11):2824–2836.

[54] Subbarao, M. and Surya, G. (1994). Depth from defocus: A spatial domain approach. *International Journal of Computer Vision*, 13(3):271–294.

[55] Szegedy, C., Ioffe, S., Vanhoucke, V., and Alemi, A. A. (2017). Inception-v4, inception-resnet and the impact of residual connections on learning. In *Thirty-first AAAI conference on artificial intelligence*.

[56] Szeliski, R. and Kang, S. B. (1997). Shape ambiguities in structure from motion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(5):506–512.

[57] Szeliski, R. and Torr, P. H. (1998). Geometrically constrained structure from motion: Points on planes. In *European Workshop on 3D Structure from Multiple Images of Large-Scale Environments*, pages 171–186. Springer.

[58] Tosi, F., Aleotti, F., Poggi, M., and Mattoccia, S. (2019). Learning monocular depth estimation infusing traditional stereo knowledge. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9799–9809.

[59] Uhrig, J., Schneider, N., Schneider, L., Franke, U., Brox, T., and Geiger, A. (2017). Sparsity invariant cnns. In *2017 international conference on 3D Vision (3DV)*, pages 11–20. IEEE.

[60] Ullman, S. (1979). The interpretation of structure from motion. *Proceedings of the Royal Society of London. Series B. Biological Sciences*, 203(1153):405–426.

[61] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

[62] Wang, H., Zhu, Y., Green, B., Adam, H., Yuille, A., and Chen, L.-C. (2020a). Axial-deeplab: Stand-alone axial-attention for panoptic segmentation. In *European Conference on Computer Vision*, pages 108–126. Springer.

[63] Wang, J., Zhang, G., Yu, M., Xu, T., and Luo, T. (2020b). Attention-based dense decoding network for monocular depth estimation. *IEEE Access*, 8:85802–85812.

[64] Wang, S., Li, B. Z., Khabsa, M., Fang, H., and Ma, H. (2020c). Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*.

[65] Wang, W., Xie, E., Li, X., Fan, D.-P., Song, K., Liang, D., Lu, T., Luo, P., and Shao, L. (2021). Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. *arXiv preprint arXiv:2102.12122*.

[66] Wang, Z., Bovik, A. C., Sheikh, H. R., and Simoncelli, E. P. (2004). Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612.

[67] Xiao, T., Liu, Y., Zhou, B., Jiang, Y., and Sun, J. (2018). Unified perceptual parsing for scene understanding. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 418–434.

[68] Xie, S., Girshick, R., Dollár, P., Tu, Z., and He, K. (2017). Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1492–1500.

[69] Xu, D., Ricci, E., Ouyang, W., Wang, X., and Sebe, N. (2017). Multi-scale continuous crfs as sequential deep networks for monocular depth estimation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5354–5362.

[70] Xu, D., Wang, W., Tang, H., Liu, H., Sebe, N., and Ricci, E. (2018). Structured attention guided convolutional neural fields for monocular depth estimation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3917–3925.

[71] Yang, M., Yu, K., Zhang, C., Li, Z., and Yang, K. (2018). Denseaspp for semantic segmentation in street scenes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3684–3692.

[72] Ye, X., Chen, S., and Xu, R. (2021). Dpnet: Detail-preserving network for high quality monocular depth estimation. *Pattern Recognition*, 109:107578.

[73] Yin, W., Liu, Y., Shen, C., and Yan, Y. (2019). Enforcing geometric constraints of virtual normal for depth prediction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5684–5693.

[74] Yin, Z. and Shi, J. (2018). Geonet: Unsupervised learning of dense depth, optical flow and camera pose. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1983–1992.

[75] Zhang, R., Tsai, P.-S., Cryer, J. E., and Shah, M. (1999). Shape-from-shading: a survey. *IEEE transactions on pattern analysis and machine intelligence*, 21(8):690–706.

[76] Zhang, Z. (2000). A flexible new technique for camera calibration. *IEEE Transactions on pattern analysis and machine intelligence*, 22(11):1330–1334.

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

[78] Zheng, S., Lu, J., Zhao, H., Zhu, X., Luo, Z., Wang, Y., Fu, Y., Feng, J., Xiang, T., Torr, P. H., et al. (2021). Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6881–6890.

[79] Zhou, B., Zhao, H., Puig, X., Fidler, S., Barriuso, A., and Torralba, A. (2017a). Scene parsing through ade20k dataset. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 633–641.

[80] Zhou, T., Brown, M., Snavely, N., and Lowe, D. G. (2017b). Unsupervised learning of depth and ego-motion from video. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1851–1858.

[81] Zou, L. and Li, Y. (2010). A method of stereo vision matching based on opencv. In *2010 International Conference on Audio, Language and Image Processing*, pages 185–190. IEEE.