# Vision Transformers for Dense Prediction

René Ranftl Alexey Bochkovskiy Vladlen Koltun

Intel Labs

rene.ranftl@intel.com

## Abstract

*We introduce dense prediction transformers, an architecture that leverages vision transformers in place of convolutional networks as a backbone for dense prediction tasks. We assemble tokens from various stages of the vision transformer into image-like representations at various resolutions and progressively combine them into full-resolution predictions using a convolutional decoder. The transformer backbone processes representations at a constant and relatively high resolution and has a global receptive field at every stage. These properties allow the dense prediction transformer to provide finer-grained and more globally coherent predictions when compared to fully-convolutional networks. Our experiments show that this architecture yields substantial improvements on dense prediction tasks, especially when a large amount of training data is available. For monocular depth estimation, we observe an improvement of up to 28% in relative performance when compared to a state-of-the-art fully-convolutional network. When applied to semantic segmentation, dense prediction transformers set a new state of the art on ADE20K with 49.02% mIoU. We further show that the architecture can be fine-tuned on smaller datasets such as NYUv2, KITTI, and Pascal Context where it also sets the new state of the art. Our models are available at* `https://github.com/intel-isl/DPT`.

## 1. Introduction

Virtually all existing architectures for dense prediction are based on convolutional networks [6, 33, 36, 44, 51, 52, 55]. The design of dense prediction architectures commonly follows a pattern that logically separates the network into an encoder and a decoder. The encoder is frequently based on an image classification network, also called the backbone, that is pretrained on a large corpus such as ImageNet [9]. The decoder aggregates features from the encoder and converts them to the final dense predictions. Architectural research on dense prediction frequently focuses on the decoder and its aggregation strategy [6, 7, 52, 55]. However, it is widely recognized that the choice of backbone architecture has a large influence on the capabilities of the overall model, as any information that is lost in the encoder is impossible to recover in the decoder.

Convolutional backbones progressively downsample the input image to extract features at multiple scales. Downsampling enables a progressive increase of the receptive field, the grouping of low-level features into abstract high-level features, and simultaneously ensures that memory and computational requirements of the network remain tractable. However, downsampling has distinct drawbacks that are particularly salient in dense prediction tasks: feature resolution and granularity are lost in the deeper stages of the model and can thus be hard to recover in the decoder. While feature resolution and granularity may not matter for some tasks, such as image classification, they are critical for dense prediction, where the architecture should ideally be able to resolve features at or close to the resolution of the input image.

Various techniques to mitigate the loss of feature granularity have been proposed. These include training at higher input resolution (if the computational budget permits), dilated convolutions [51] to rapidly increase the receptive field without downsampling, appropriately-placed skip connections from multiple stages of the encoder to the decoder [33], or, more recently, by connecting multi-resolution representations in parallel throughout the network [44]. While these techniques can significantly improve prediction quality, the networks are still bottlenecked by their fundamental building block: the convolution. Convolutions together with non-linearities form the fundamental computational unit of image analysis networks. Convolutions, by definition, are linear operators that have a limited receptive field. The limited receptive field and the limited expressivity of an individual convolution necessitate sequential stacking into very deep architectures to acquire sufficiently broad context and sufficiently high representational power. This, however, requires the production of many intermediate representations that require a large amount of memory. Downsampling the intermediate representations

is necessary to keep memory consumption at levels that are feasible with existing computer architectures.

In this work, we introduce the dense prediction transformer (DPT). DPT is a dense prediction architecture that is based on an encoder-decoder design that leverages a transformer as the basic computational building block of the encoder. Specifically, we use the recently proposed vision transformer (ViT) [11] as a backbone architecture. We reassemble the bag-of-words representation that is provided by ViT into image-like feature representations at various resolutions and progressively combine the feature representations into the final dense prediction using a convolutional decoder. Unlike fully-convolutional networks, the vision transformer backbone foregoes explicit downsampling operations after an initial image embedding has been computed and maintains a representation with constant dimensionality throughout all processing stages. It furthermore has a global receptive field at every stage. We show that these properties are especially advantageous for dense prediction tasks as they naturally lead to fine-grained and globally coherent predictions.

We conduct experiments on monocular depth estimation and semantic segmentation. For the task of general-purpose monocular depth estimation [32], where large-scale training data is available, DPT provides a performance increase of more than 28% when compared to the top-performing fully-convolutional network for this task. The architecture can also be fine-tuned to small monocular depth prediction datasets, such as NYUv2 [37] and KITTI [15], where it also sets the new state of the art. We provide further evidence of the strong performance of DPT using experiments on semantics segmentation. For this task, DPT sets a new state of the art on the challenging ADE20K [56] and Pascal Context [28] datasets.

## 2. Related Work

Fully-convolutional networks [35, 36] are the prototypical architecture for pixel-level dense prediction tasks such as semantic segmentation [7, 25, 55], monocular depth estimation [12, 16, 32], and keypoint detection [21, 57]. Many variants of this pattern have been proposed over the years, however, all existing architectures adopt convolution and subsampling as their fundamental elements in order to learn multi-scale representations that can leverage an appropriately large context. Several works propose to progressively upsample representations that have been pooled at different stages [1, 25, 29, 33], while others use dilated convolutions [6, 7, 51] or parallel multi-scale feature aggregation at multiple scales [55] to recover fine-grained predictions while at the same time ensuring a sufficiently large context. More recent architectures maintain a high-resolution representation together with multiple lower-resolution representations throughout the network [39, 44].

Attention-based models [2] and in particular transformers [41] have been the architecture of choice for learning strong models for natural language processing (NLP) [4, 10, 26] in recent years. Transformers are set-to-set models that are based on the self-attention mechanism. Transformer models have been particularly successful when instantiated as high-capacity architectures and trained on very large datasets. There have been several works that adapt attention mechanisms to image analysis [3, 30, 31, 43, 54]. In particular, it has recently been demonstrated that a direct application of token-based transformer architectures that have been successful in NLP can yield competitive performance on image classification [11]. A key insight of this work was that, like transformer models in NLP, vision transformers need to be paired with a sufficient amount of training data to realize their potential.

## 3. Architecture

This section introduces the dense prediction transformer. We maintain the overall encoder-decoder structure that has been successful for dense prediction in the past. We leverage vision transformers [11] as the backbone, show how the representation that is produced by this encoder can be effectively transformed into dense predictions, and provide intuition for the success of this strategy. An overview of the complete architecture is shown in Figure 1 (left).

**Transformer encoder.** On a high level, the vision transformer (ViT) [11] operates on a bag-of-words representation of the image [38]. Image patches that are individually embedded into a feature space, or alternatively deep features extracted from the image, take the role of "words". We will refer to embedded "words" as *tokens* throughout the rest of this work. Transformers transform the set of tokens using sequential blocks of multi-headed self-attention (MHSA) [41], which relate tokens to each other to transform the representation.

Importantly for our application, a transformer maintains the number of tokens throughout all computations. Since tokens have a one-to-one correspondence with image patches, this means that the ViT encoder maintains the spatial resolution of the initial embedding throughout all transformer stages. Additionally, MHSA is an inherently global operation, as every token can attend to and thus influence every other token. Consequently, the transformer has a global receptive field at every stage after the initial embedding. This is in stark contrast to convolutional networks, which progressively increase their receptive field as features pass through consecutive convolution and downsampling layers.

More specifically, ViT extracts a patch embedding from the image by processing all non-overlapping square patches of size $p^2$ pixels from the image. The patches are flattened into vectors and individually embedded using a linear pro-

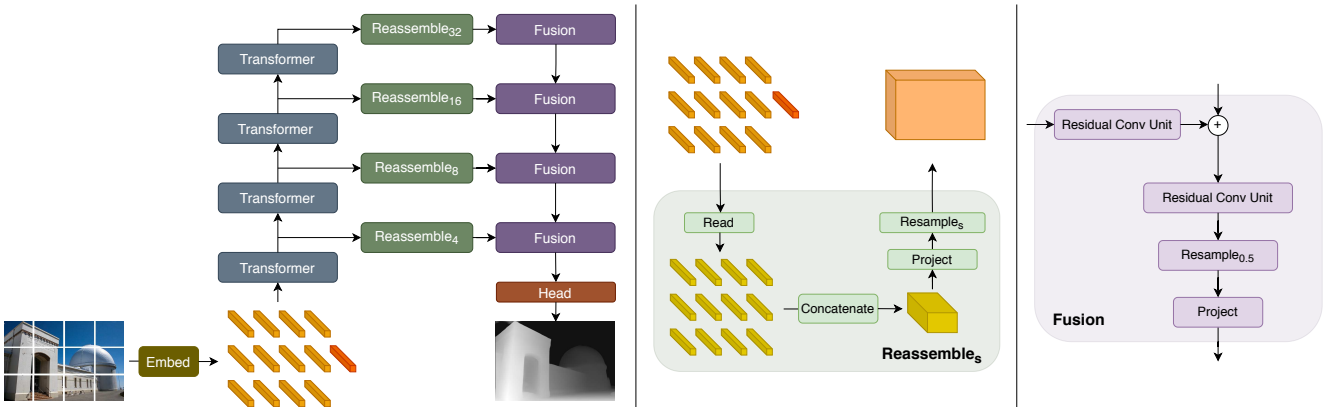


Figure 1. *Left*: Architecture overview. The input image is transformed into tokens (orange) either by extracting non-overlapping patches followed by a linear projection of their flattened representation (DPT-Base and DPT-Large) or by applying a ResNet-50 feature extractor (DPT-Hybrid). The image embedding is augmented with a positional embedding and a patch-independent readout token (red) is added. The tokens are passed through multiple transformer stages. We reassemble tokens from different stages into an image-like representation at multiple resolutions (green). Fusion modules (purple) progressively fuse and upsample the representations to generate a fine-grained prediction. *Center*: Overview of the Reassemble$_s$ operation. Tokens are assembled into feature maps with $\frac{1}{s}$ the spatial resolution of the input image. *Right*: Fusion blocks combine features using residual convolutional units [25] and upsample the feature maps.

jection. An alternative, more sample-efficient, variant of ViT extracts the embedding by applying a ResNet50 [17] to the image and uses the pixel features of the resulting feature maps as tokens. Since transformers are set-to-set functions, they do not intrinsically retain the information of the spatial positions of individual tokens. The image embeddings are thus concatenated with a learnable position embedding to add this information to the representation. Following work in NLP [10], the ViT additionally adds a special learned token that is not grounded in the input image. This token serves the purpose to aggregate information into a global image representation which is then used for classification. We refer to this token as the *readout* token. The result of applying the embedding procedure to an image of size $H \times W$ pixels is a a set of $t^0 = \{t_0^0, \ldots, t_{N_p}^0\},\ t_n^0 \in \mathbb{R}^D$ tokens, where $N_p = \frac{HW}{p^2}$, $t_0$ refers to the readout token, and $D$ is the feature dimension of each token.

The input tokens are transformed using $L$ transformer layers into new representations $t^l$, where $l$ refers to the output of the $l$-th transformer layer. Dosovitskiy *et al*. [11] define several variants of this basic blueprint. We use three variants in our work: ViT-Base, which uses the patch-based embedding procedure and features 12 transformer layers; ViT-Large, which uses the same embedding procedure and has 24 transformer layers and a wider feature size $D$; and ViT-Hybrid, which employs a ResNet50 to compute the image embedding followed by 12 transformer layers. We use patch size $p = 16$ for all experiments. We refer the interested reader to the original work [11] for additional details on these architectures.

The embedding procedure for ViT-Base and ViT-Large projects the flattened patches to dimension $D = 768$ and $D = 1024$, respectively. Since both feature dimensions are larger than the number of pixels in an input patch, this means that the embedding procedure can learn to retain information if it is beneficial for the task. Features from the input patches can in principle be resolved with pixel-level accuracy. Similarly, the ViT-Hybrid architecture extracts features at $\frac{1}{16}$ the input resolution, which is twice as high as the lowest-resolution features that are commonly used with convolutional backbones.

**Convolutional decoder.** Our decoder assembles the set of tokens into image-like feature representations at various resolutions. The feature representations are progressively fused into the final dense prediction. We propose a simple three-stage *Reassemble* operation to recover image-like representations from the output tokens of arbitrary layers of the transformer encoder:

$$\text{Reassemble}_s^{\hat{D}}(t) = (\text{Resample}_s \circ \text{Concatenate} \circ \text{Read})(t),$$

where $s$ denotes the output size ratio of the recovered representation with respect to the input image, and $\hat{D}$ denotes the output feature dimension.

We first map the $N_p + 1$ tokens to a set of $N_p$ tokens that is amenable to spatial concatenation into an image-like representation:

$$\text{Read} : \mathbb{R}^{N_p+1 \times D} \rightarrow \mathbb{R}^{N_p \times D}. \tag{1}$$

This operation is essentially responsible for appropriately handling the readout token. Since the readout token doesn't serve a clear purpose for the task of dense prediction, but could potentially still be useful to capture and distribute global information, we evaluate three different variants of

this mapping:

$$\text{Read}_{ignore}(t) = \{t_1, \ldots, t_{N_p}\} \tag{2}$$

simply ignores the readout token,

$$\text{Read}_{add}(t) = \{t_1 + t_0, \ldots, t_{N_p} + t_0\} \tag{3}$$

passes the information from the readout token to all other tokens by adding the representations, and

$$\text{Read}_{proj}(t) = \{\text{mlp}(\text{cat}(t_1, t_0)), \ldots, \text{mlp}(\text{cat}(t_{N_p}, t_0))\} \tag{4}$$

passes information to the other tokens by concatenating the readout to all other tokens before projecting the representation to the original feature dimension $D$ using a linear layer followed by a GELU non-linearity [18].

After a Read block, the resulting $N_p$ tokens can be reshaped into an image-like representation by placing each token according to the position of the initial patch in the image. Formally, we apply a spatial concatenation operation that results in a feature map of size $\frac{H}{p} \times \frac{W}{p}$ with $D$ channels:

$$\text{Concatenate} : \mathbb{R}^{N_p \times D} \rightarrow \mathbb{R}^{\frac{H}{p} \times \frac{W}{p} \times D}. \tag{5}$$

We finally pass this representation to a spatial resampling layer that scales the representation to size $\frac{H}{s} \times \frac{W}{s}$ with $\hat{D}$ features per pixel:

$$\text{Resample}_s : \mathbb{R}^{\frac{H}{p} \times \frac{W}{p} \times D} \rightarrow \mathbb{R}^{\frac{H}{s} \times \frac{W}{s} \times \hat{D}}. \tag{6}$$

We implement this operation by first using $1 \times 1$ convolutions to project the input representation to $\hat{D}$, followed by a (strided) $3 \times 3$ convolution when $s \geq p$, or a strided $3 \times 3$ transpose convolution when $s < p$, to implement spatial downsampling and upsampling operations, respectively.

Irrespective of the exact transformer backbone, we reassemble features at four different stages and four different resolutions. We assemble features from deeper layers of the transformer at lower resolution, whereas features from early layers are assembled at higher resolution. When using ViT-Large, we reassemble tokens from layers $l = \{6, 12, 18, 24\}$, whereas with ViT-Base we use layers $l = \{3, 6, 9, 12\}$. We use features from the first and second ResNet block from the embedding network and stages $l = \{9, 12\}$ when using ViT-Hybrid. Our default architecture uses projection as the readout operation and produces feature maps with $\hat{D} = 256$ dimensions. We will refer to these architectures as DPT-Base, DPT-Large, and DPT-Hybrid, respectively.

We finally combine the extracted feature maps from consecutive stages using a RefineNet-based feature fusion block [25, 47] (see Figure1 (right)) and progressively upsample the representation by a factor of two in each fusion stage. The final representation size has half the resolution of the input image. We attach a task-specific output head to produce the final prediction. A schematic overview of the complete architecture is shown in Figure 1.

**Handling varying image sizes.** Akin to fully-convolutional networks, DPT can handle varying image sizes. As long as the image size is divisible by $p$, the embedding procedure can be applied and will produce a varying number of image tokens $N_p$. As a set-to-set architecture, the transformer encoder can trivially handle a varying number of tokens. However, the position embedding has a dependency on the image size as it encodes the locations of the patches in the input image. We follow the approach proposed in [11] and linearly interpolate the position embeddings to the appropriate size. Note that this can be done on the fly for every image. After the embedding procedure and the transformer stages, both the reassemble and fusion modules can trivially handle a varying number of tokens, provided that the input image is aligned to the stride of the convolutional decoder (32 pixels).

## 4. Experiments

We apply DPT to two dense prediction tasks: monocular depth estimation and semantic segmentation. For both tasks, we show that DPT can significantly improve accuracy when compared to convolutional networks with a similar capacity, especially if a large training dataset is available. We first present our main results using the default configuration and show comprehensive ablations of different DPT configurations at the end of this section.

### 4.1. Monocular Depth Estimation

Monocular depth estimation is typically cast as a dense regression problem. It has been shown that massive meta-datasets can be constructed from existing sources of data, provided that some care is taken in how different representations of depth are unified into a common representation and that common ambiguities (such as scale ambiguity) are appropriately handled in the training loss [32]. Since transformers are known to realize their full potential only when an abundance of training data is available, monocular depth estimation is an ideal task to test the capabilities of DPT.

**Experimental protocol.** We closely follow the protocol of Ranftl *et al*. [32]. We learn a monocular depth prediction network using a scale- and shift-invariant trimmed loss that operates on an inverse depth representation, together with the gradient-matching loss proposed in [24]. We construct a meta-dataset that includes the original datasets that were used in [32] (referred to as *MIX 5* in that work) and extend it with with five additional datasets ([19, 45, 46, 48, 49]).

| | Training set | DIW WHDR | ETH3D AbsRel | Sintel AbsRel | KITTI $\delta$>1.25 | NYU $\delta$>1.25 | TUM $\delta$>1.25 |
|---|---|---|---|---|---|---|---|
| DPT - Large | MIX 6 | **10.82** (-13.2%) | **0.089** (-31.2%) | **0.270** (-17.5%) | **8.46** (-64.6%) | **8.32** (-12.9%) | **9.97** (-30.3%) |
| DPT - Hybrid | MIX 6 | 11.06 (-11.2%) | 0.093 (-27.6%) | 0.274 (-16.2%) | 11.56 (-51.6%) | 8.69 (-9.0%) | 10.89 (-23.2%) |
| MiDaS | MIX 6 | 12.95 (+3.9%) | 0.116 (-10.5%) | 0.329 (+0.5%) | 16.08 (-32.7%) | 8.71 (-8.8%) | 12.51 (-12.5%) |
| MiDaS [32] | MIX 5 | 12.46 | 0.129 | 0.327 | 23.90 | 9.55 | 14.29 |
| Li [24] | MD [24] | 23.15 | 0.181 | 0.385 | 36.29 | 27.52 | 29.54 |
| Li [23] | MC [23] | 26.52 | 0.183 | 0.405 | 47.94 | 18.57 | 17.71 |
| Wang [42] | WS [42] | 19.09 | 0.205 | 0.390 | 31.92 | 29.57 | 20.18 |
| Xian [47] | RW [47] | 14.59 | 0.186 | 0.422 | 34.08 | 27.00 | 25.02 |
| Casser [5] | CS [8] | 32.80 | 0.235 | 0.422 | 21.15 | 39.58 | 37.18 |

Table 1. Comparison to the state of the art on monocular depth estimation. We evaluate zero-shot cross-dataset transfer according to the protocol defined in [32]. Relative performance is computed with respect to the original MiDaS model [32]. Lower is better for all metrics.

We refer to this meta-dataset as *MIX 6*. It contains about 1.4 million images and is, to the best of our knowledge, the largest training set for monocular depth estimation that has ever been compiled.

We use multi-objective optimization [34] together with Adam [20] and set a learning rate of $1e{-}5$ for the backbone and $1e{-}4$ for the decoder weights. The encoder is initialized with ImageNet-pretrained weights, whereas the decoder is initialized randomly. We use an output head that consists of 3 convolutional layers. The output head progressively halves the feature dimension and upsamples the predictions to the input resolution after the first convolutional layer (details in supplementary material). We disable batch normalization in the decoder, as we found it to negatively influence results for regression tasks. We resize the image such that the longer side is 384 pixels and train on random square crops of size 384. We train for 60 epochs, where one epoch consists of 72,000 steps with a batch size of 16. As the batch size is not divisible by the number of datasets, we construct a mini-batch by first drawing datasets uniformly at random before sampling from the respective datasets. We perform random horizontal flips for data augmentation. Similar to [32], we first pretrain on a well-curated subset of the data [47, 48, 49] for 60 epochs before training on the full dataset.

| | $\delta$<1.25 | $\delta$<1.25$^2$ | $\delta$<1.25$^3$ | AbsRel | RMSE | log10 |
|---|---|---|---|---|---|---|
| DORN [13] | 0.828 | 0.965 | 0.992 | 0.115 | 0.509 | 0.051 |
| VNL [50] | 0.875 | 0.976 | 0.994 | 0.111 | 0.416 | 0.048 |
| BTS [22] | 0.885 | 0.978 | 0.994 | **0.110** | 0.392 | 0.047 |
| DPT-Hybrid | **0.904** | **0.988** | **0.998** | **0.110** | **0.357** | **0.045** |

Table 2. Evaluation on NYUv2 depth.

| | $\delta$<1.25 | $\delta$<1.25$^2$ | $\delta$<1.25$^3$ | AbsRel | RMSE | RMSE log |
|---|---|---|---|---|---|---|
| DORN [13] | 0.932 | 0.984 | 0.994 | 0.072 | 2.626 | 0.120 |
| VNL [50] | 0.938 | 0.990 | 0.998 | 0.072 | 3.258 | 0.117 |
| BTS [22] | 0.956 | 0.993 | 0.998 | **0.059** | 2.756 | 0.096 |
| DPT-Hybrid | **0.959** | **0.995** | **0.999** | 0.062 | **2.573** | **0.092** |

Table 3. Evaluation on KITTI (Eigen split).

**Zero-shot cross-dataset transfer.** Table 1 shows the results of zero-shot transfer to different datasets that were not seen during training. We refer the interested reader to Ranftl *et al*. [32] for details of the evaluation procedure and error metrics. For all metrics, lower is better. Both DPT variants significantly outperform the state of the art. The average relative improvement over the best published architecture, MiDaS, is more than 23% for DPT-Hybrid and 28% for DPT-Large. DPT-Hybrid achieves this with a comparable network capacity (Table 9), while DPT-Large is about 3 times larger than MiDaS. Note that both architectures have similar latency to MiDaS (Table 9).

To ensure that the observed improvements are not only due to the enlarged training set, we retrain the fully-convolutional network used by MiDaS on our larger meta-dataset *MIX 6*. While the fully-convolutional network indeed benefits from the larger training set, we observe that both DPT variants still strongly outperform this network. This shows that DPT can better benefit from increased training set size, an observation that matches previous findings on transformer-based architectures [11].

The quantitative results are supported by visual comparisons in Figure 2. DPT can better reconstruct fine details while also improving global coherence in areas that are challenging for the convolutional architecture (for example, large homogeneous regions or relative depth arrangement across the image).

**Fine-tuning on small datasets.** We fine-tune DPT-Hybrid on the KITTI [15] and NYUv2 [37] datasets to further compare the representational power of DPT to existing work. Since the network was trained with an affine-invariant loss, its predictions are arbitrarily scaled and shifted and can have large magnitudes. Direct fine-tuning would thus be challenging, as the global mismatch in the magnitude of the predictions to the ground truth would dominate the loss. We thus first align predictions of the initial network to each training sample using the robust alignment procedure described in [32]. We then average the resulting scales and shifts across the training set and apply the average scale and

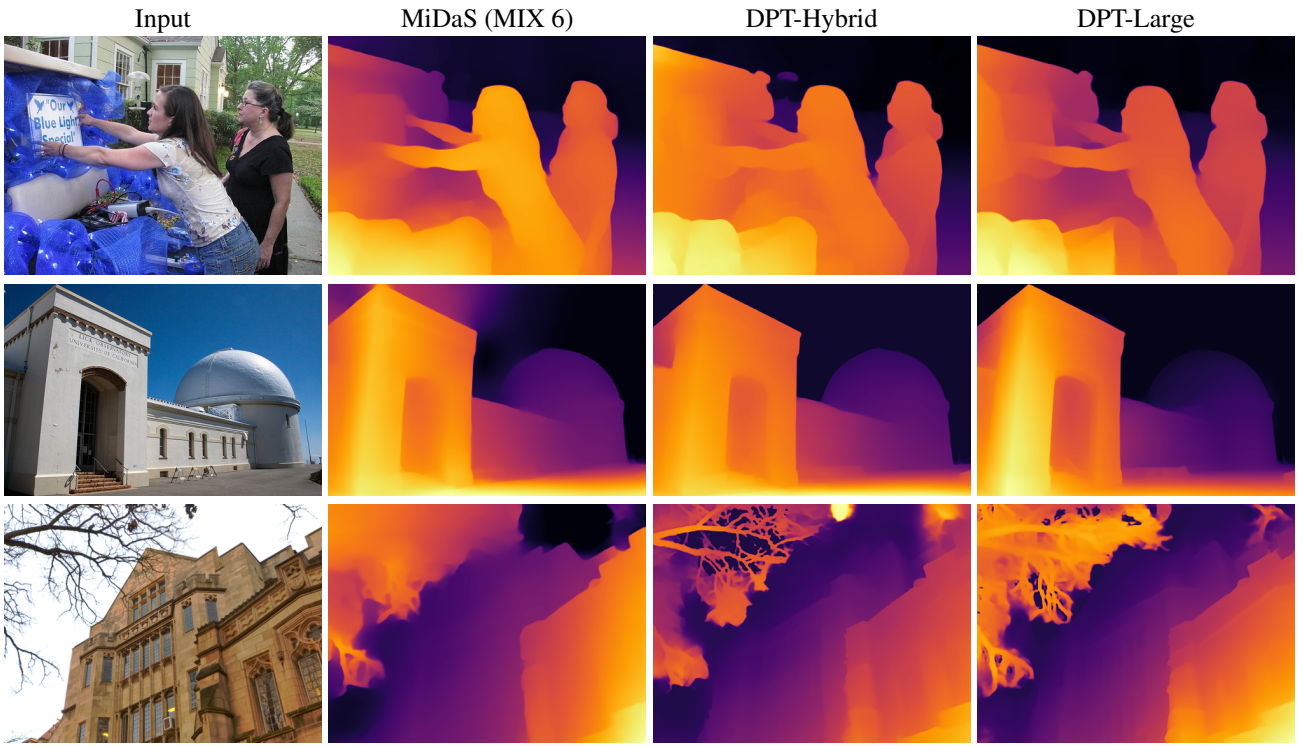


Figure 2. Sample results for monocular depth estimation. Compared to the fully-convolutional network used by MiDaS, DPT shows better global coherence (e.g., sky, second row) and finer-grained details (e.g., tree branches, last row).

shift to the predictions before passing the result to the loss. We fine-tune with the loss proposed by Eigen *et al*. [12]. We disable the gradient-matching loss for KITTI since this dataset only provides sparse ground truth.

Tables 2 and 3 summarize the results. Our architecture matches or improves state-of-the-art performance on both datasets in all metrics. This indicates that DPT can also be usefully applied to smaller datasets.

### 4.2. Semantic Segmentation

We choose semantic segmentation as our second task since it is representative of discrete labeling tasks and is a very competitive proving ground for dense prediction architectures. We employ the same backbone and decoder structure as in previous experiments. We use an output head that predicts at half resolution and upsamples the logits to full resolution using bilinear interpolation (details in supplementary material). The encoder is again initialized from ImageNet-pretrained weights, and the decoder is initialized randomly.

**Experimental protocol.** We closely follow the protocol established by Zhang *et al*. [53]. We employ a cross-entropy loss and add an auxiliary output head together with an auxiliary loss to the output of the penultimate fusion layer. We set the weight of the auxiliary loss to $0.2$. Dropout with a rate of $0.1$ is used before the final classification layer in both heads. We use SGD with momentum $0.9$ and a polynomial learning rate scheduler with decay factor $0.9$. We use batch normalization in the fusion layers and train with batch size $48$. Images are resized to $520$ pixels side length. We use random horizontal flipping and random rescaling in the range $\in (0.5, 2.0)$ for data augmentation. We train on square random crops of size $480$. We set the learning rate to $0.002$. We use multi-scale inference at test time and report both pixel accuracy (pixAcc) as well as mean Intersection-over-Union (mIoU).

**ADE20K.** We train the DPT on the ADE20K semantic segmentation dataset [56] for $240$ epochs. Table 4 summarizes our results on the validation set. DPT-Hybrid outperforms all existing fully-convolutional architectures. DPT-Large performs slightly worse, likely because of the significantly smaller dataset compared to our previous experiments. Figure 3 provides visual comparisons. We observe that the DPT tends to produce cleaner and finer-grained delineations of object boundaries and that the predictions are also in some cases less cluttered.

**Fine-tuning on smaller datasets.** We fine-tune DPT-Hybrid on the Pascal Context dataset [28] for 50 epochs. All other hyper-parameters remain the same. Table 5 shows results on the validation set for this experiment. We again see that DPT can provide strong performance even on smaller datasets.

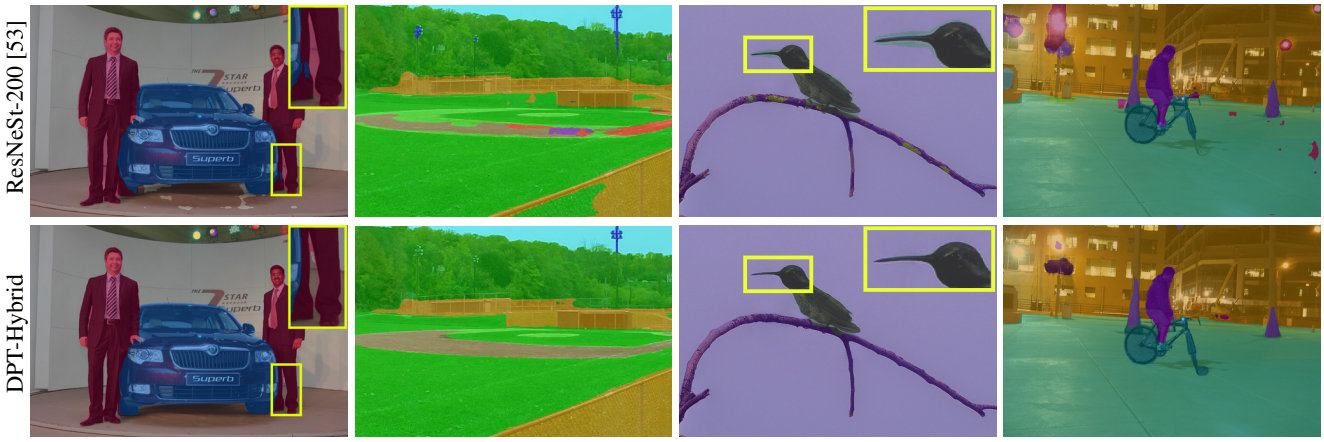


Figure 3. Sample results for semantic segmentation on ADE20K (first and second column) and Pascal Context (third and fourth column). Predictions are frequently better aligned to object edges and less cluttered.

### 4.3. Ablations

We examine a number of aspects and technical choices in DPT via ablation studies. We choose monocular depth estimation as the task for our ablations and follow the same protocol and hyper-parameter settings as previously described. We use a reduced meta-dataset that is composed of three datasets [47, 48, 49] and consists of about 41,000 images. We choose these datasets since they provide high-quality ground truth. We split each dataset into a training set and a small validation set of about 1,000 images total. We report results on the validation sets in terms of relative absolute deviation after affine alignment of the predictions to the ground truth [32]. Unless specified otherwise, we use ViT-Base as the backbone architecture.

**Skip connections.** Convolutional architectures offer natural points of interest for passing features from the encoder to the decoder, namely before or after downsampling of the representation. Since the transformer backbone maintains a constant feature resolution, it is not clear at which points in the backbone features should be tapped. We evaluate several possible choices in Table 6 (top). We observe that it is beneficial to tap features from layers that contain low-level features as well as deeper layers that contain higher-level features. We adopt the best setting for all further experiments.

We perform a similar experiment with the hybrid architecture in Table 6 (bottom), where R0 and R1 refer to using features from the first and second downsampling stages of the ResNet50 embedding network. We observe that using low-level features from the embedding network leads to better performance than using features solely from the transformer stages. We use this setting for all further experiments that involve the hybrid architecture.

**Readout token.** Table 7 examines various choices for implementing the first stage of the *Reassemble* block to handle the readout token. While ignoring the token yields good performance, projection provides slightly better performance on average. Adding the token, on the other hand, yields worse performance than simply ignoring it. We use projection for all further experiments.

**Backbones.** The performance of different backbones is

| | Backbone | | pixAcc [%] | mIoU [%] |
|---|---|---|---|---|
| OCNet | ResNet101 | [52] | – | 45.45 |
| ACNet | ResNet101 | [14] | 81.96 | 45.90 |
| DeeplabV3 | ResNeSt-101 | [7, 53] | 82.07 | 46.91 |
| DeeplabV3 | ResNeSt-200 | [7, 53] | 82.45 | 48.36 |
| DPT-Hybrid | ViT-Hybrid | | **83.11** | **49.02** |
| DPT-Large | ViT-Large | | 82.70 | 47.63 |

Table 4. Semantic segmentation results on the ADE20K validation set.

| | Backbone | | pixAcc [%] | mIoU [%] |
|---|---|---|---|---|
| OCNet | HRNet-W48 | [44, 52] | – | 56.2 |
| DeeplabV3 | ResNeSt-200 | [7, 53] | 82.50 | 58.37 |
| DeeplabV3 | ResNeSt-269 | [7, 53] | 83.06 | 58.92 |
| DPT-Hybrid | ViT-Hybrid | | **84.83** | **60.46** |

Table 5. Finetuning results on the Pascal Context validation set.

| | Layer $l$ | HRWSI | BlendedMVS | ReDWeb | Mean |
|---|---|---|---|---|---|
| Base | {3, 6, 9, 12} | 0.0793 | 0.0780 | 0.0892 | 0.0822 |
| | {6, 8, 10, 12} | 0.0801 | 0.0789 | 0.0904 | 0.0831 |
| | {9, 10, 11, 12} | 0.0805 | 0.0766 | 0.0912 | 0.0828 |
| Hybrid | {3, 6, 9, 12} | 0.0747 | **0.0748** | 0.0865 | 0.0787 |
| | {R0, R1, 9, 12} | **0.0742** | 0.0751 | **0.0857** | **0.0733** |

Table 6. Performance of attaching skip connections to different encoder layers. Best results are achieved with a combination of skip connections from shallow and deep layers.

| | HRWSI | BlendedMVS | ReDWeb | Mean |
|---|---|---|---|---|
| Ignore | **0.0793** | 0.0780 | **0.0892** | 0.0822 |
| Add | 0.0799 | 0.0789 | 0.0904 | 0.0831 |
| Project | 0.0797 | **0.0764** | 0.0895 | **0.0819** |

Table 7. Performance of approaches to handle the readout token. Fusing the readout token to the individual input tokens using a projection layer yields the best performance.

shown in Table 8. ViT-Large outperforms all other backbones but is also almost three times larger than ViT-Base and ViT-Hybrid. ViT-Hybrid outperforms ViT-Base with a similar number of parameters and has comparable performance to the large backbone. As such it provides a good trade-off between accuracy and capacity.

ViT-Base has comparable performance to ResNext101-WSL, while ViT-Hybrid and ViT-Large improve performance even though they have been pretrained on significantly less data. Note that ResNext101-WSL was pretrained on a billion-scale corpus of weakly supervised data [27] in addition to ImageNet pretraining. It has been observed that this pretraining boosts the performance of monocular depth prediction [32]. This architecture corresponds to the original MiDaS architecture.

We finally compare to a recent variant of ViT called DeIT [40]. DeIT trains the ViT architecture with a more data-efficient pretraining procedure. Note that the DeIT-Base architecture is identical to ViT-Base, while DeIT-Base-Dist introduces an additional *distillation* token, which we ignore in the Reassemble operation. We observe that DeIT-Base-Dist improves performance when compared to ViT-Base. This indicates that similarly to convolutional architectures, improvements in pretraining procedures for image classification can benefit dense prediction tasks.

**Inference resolution.** While fully-convolutional architectures can have large effective receptive fields in their deepest layers, the layers close to the input are local and have small receptive fields. Performance thus suffers heavily when performing inference at an input resolution that is significantly different from the training resolution. Transformer encoders, on the other hand, have a global receptive field in every layer. We conjecture that this makes DPT less dependent on inference resolution. To test this hypothesis, we plot the loss in performance of different architectures when performing inference at resolutions higher than the training resolution of $384 \times 384$ pixels. We plot the relative decrease in performance with respect to the performance of performing inference at the training resolution in Figure 4. We observe that the performance of DPT variants indeed degrades more gracefully as inference resolution increases.

| | HRWSI | BlendedMVS | ReDWeb | Mean |
|---|---|---|---|---|
| ResNet50 | 0.0890 | 0.0887 | 0.1029 | 0.0935 |
| ResNext101-WSL | 0.0780 | 0.0751 | 0.0886 | 0.0806 |
| DeIT-Base | 0.0798 | 0.0804 | 0.0925 | 0.0842 |
| DeIT-Base-Dist | 0.0758 | 0.0758 | 0.0871 | 0.0796 |
| ViT-Base | 0.0797 | 0.0764 | 0.0895 | 0.0819 |
| ViT-Large | 0.0740 | 0.0747 | **0.0846** | **0.0778** |
| ViT-Hybrid | **0.0738** | **0.0746** | 0.0864 | 0.0783 |

Table 8. Ablation of backbones. The hybrid and large backbones consistently outperform the convolutional baselines. The base architecture can outperform the convolutional baseline with better pretraining (DeIT-Base-Dist).

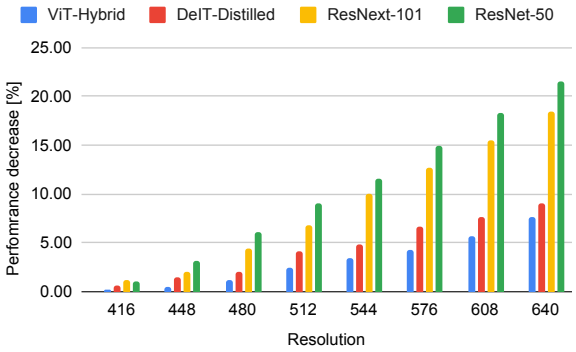


Figure 4. Relative loss in performance for different inference resolutions (lower is better).

**Inference speed.** Table 9 shows inference time for different network architectures. Timings were conducted on an Intel Xeon Platinum 8280 CPU @ 2.70GHz with 8 physical cores and an Nvidia RTX 2080 GPU. We use square images with a width of 384 pixels and report the average over 400 runs. DPT-Hybrid and DPT-Large show comparable latency to the fully-convolutional architecture used by MiDaS. Interestingly, while DPT-Large is substantially larger than the other architectures in terms of parameter count and multiply-accumulate operations, it has competitive latency since it exposes a high degree of parallelism through its wide and comparatively shallow structure.

| | MiDaS | DPT-Base | DPT-Hybrid | DPT-Large |
|---|---|---|---|---|
| Parameters [million] | 105 | 112 | 123 | 343 |
| Time [ms] | 32 | 17 | 38 | 35 |
| MACs [G] | 104 | 107 | 110 | 253 |

Table 9. Model statistics. DPT has comparable inference speed to the state of the art.

## 5. Conclusion

We have introduced the dense prediction transformer, DPT, a neural network architecture that effectively leverages vision transformers for dense prediction tasks. Our experiments on monocular depth estimation and semantic segmentation show that the presented architecture produces more fine-grained and globally coherent predictions when compared to fully-convolutional architectures. Similar to prior work on transformers, DPT unfolds its full potential when trained on large-scale datasets.

## References


[1] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. SegNet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE TIP*, 39(12):2481–2495, 2017.

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *ICLR*, 2015.

[3] Irwan Bello, Barret Zoph, Ashish Vaswani, Jonathon Shlens, and Quoc V Le. Attention augmented convolutional networks. In *ICCV*, 2019.

[4] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. In *NeurIPS*, 2020.

[5] Vincent Casser, Soeren Pirk, Reza Mahjourian, and Anelia Angelova. Unsupervised learning of depth and ego-motion: A structured approach. In *AAAI*, 2019.

[6] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille. DeepLab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *TPAMI*, 40(4):834–848, 2018.

[7] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*, 2017.

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

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

[10] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *ACL*, 2019.

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

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

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

[14] Jun Fu, Jing Liu, Yuhang Wang, Yong Li, Yongjun Bao, Jinhui Tang, and Hanqing Lu. Adaptive context network for scene parsing. In *ICCV*, 2019.

[15] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? The KITTI vision benchmark suite. In *CVPR*, 2012.

[16] Clément Godard, Oisin Mac Aodha, and Gabriel J. Brostow. Unsupervised monocular depth estimation with left-right consistency. In *CVPR*, 2017.

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

[18] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (GELUs). *arXiv preprint arXiv:1606.08415*, 2016.

[19] Xinyu Huang, Peng Wang, Xinjing Cheng, Dingfu Zhou, Qichuan Geng, and Ruigang Yang. The ApolloScape open dataset for autonomous driving and its application. *TPAMI*, 42(10):2702–2719, 2020.

[20] Diederik P. Kingma and Jimmy Lei Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

[21] Hei Law and Jia Deng. Cornernet: Detecting objects as paired keypoints. In *ECCV*, 2018.

[22] Jin Han Lee, Myung-Kyu Han, Dong Wook Ko, and Il Hong Suh. From big to small: Multi-scale local planar guidance for monocular depth estimation. *arXiv preprint arXiv:1907.10326*, 2019.

[23] Zhengqi Li, Tali Dekel, Forrester Cole, Richard Tucker, Noah Snavely, Ce Liu, and William T. Freeman. Learning the depths of moving people by watching frozen people. In *CVPR*, 2019.

[24] Zhengqi Li and Noah Snavely. MegaDepth: Learning single-view depth prediction from Internet photos. In *CVPR*, 2018.

[25] Guosheng Lin, Anton Milan, Chunhua Shen, and Ian D. Reid. RefineNet: Multi-path refinement networks for high-resolution semantic segmentation. In *CVPR*, 2017.

[26] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. RoBERTa: A robustly optimized BERT pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[27] Dhruv Mahajan, Ross Girshick, Vignesh Ramanathan, Kaiming He, Manohar Paluri, Yixuan Li, Ashwin Bharambe, and Laurens van der Maaten. Exploring the limits of weakly supervised pretraining. In *ECCV*, 2018.

[28] Roozbeh Mottaghi, Xianjie Chen, Xiaobai Liu, Nam-Gyu Cho, Seong-Whan Lee, Sanja Fidler, Raquel Urtasun, and Alan L. Yuille. The role of context for object detection and semantic segmentation in the wild. In *CVPR*, 2014.

[29] Hyeonwoo Noh, Seunghoon Hong, and Bohyung Han. Learning deconvolution network for semantic segmentation. In *ICCV*, 2015.

[30] Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Noam Shazeer, Alexander Ku, and Dustin Tran. Image transformer. In *ICML*, 2018.

[31] Prajit Ramachandran, Niki Parmar, Ashish Vaswani, Irwan Bello, Anselm Levskaya, and Jonathon Shlens. Stand-alone self-attention in vision models. In *NeurIPS*, 2019.

[32] René Ranftl, Katrin Lasinger, David Hafner, Konrad Schindler, and Vladlen Koltun. Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. *TPAMI*, 2020.

[33] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional networks for biomedical image segmentation. In *MICCAI*, 2015.

[34] Ozan Sener and Vladlen Koltun. Multi-task learning as multi-objective optimization. In *NeurIPS*, 2018.

[35] Pierre Sermanet, David Eigen, Xiang Zhang, Michaël Mathieu, Rob Fergus, and Yann LeCun. OverFeat: Integrated recognition, localization and detection using convolutional

networks. In *ICLR*, 2014.

[36] Evan Shelhamer, Jonathan Long, and Trevor Darrell. Fully convolutional networks for semantic segmentation. *CVPR*, 2015.

[37] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support inference from RGBD images. In *ECCV*, 2012.

[38] Josef Sivic and Andrew Zisserman. Efficient visual search of videos cast as text retrieval. *TPAMI*, 31(4):591–606, 2009.

[39] Ke Sun, Bin Xiao, Dong Liu, and Jingdong Wang. Deep high-resolution representation learning for human pose estimation. In *CVPR*, 2019.

[40] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Herve Jegou. Training data-efficient image transformers & distillation through attention. In *ICML*, 2021.

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

[42] Chaoyang Wang, Oliver Wang, Federico Perazzi, and Simon Lucey. Web stereo video supervision for depth prediction from dynamic scenes. In *3DV*, 2019.

[43] Huiyu Wang, Yukun Zhu, Bradley Green, Hartwig Adam, Alan L. Yuille, and Liang-Chieh Chen. Axial-DeepLab: Stand-alone axial-attention for panoptic segmentation. In *ECCV*, 2020.

[44] Jingdong Wang, Ke Sun, Tianheng Cheng, Borui Jiang, Chaorui Deng, Yang Zhao, Dong Liu, Yadong Mu, Mingkui Tan, Xinggang Wang, Wenyu Liu, and Bin Xiao. Deep high-resolution representation learning for visual recognition. *TPAMI*, 2020.

[45] Qiang Wang, Shizhen Zheng, Qingsong Yan, Fei Deng, Kaiyong Zhao, and Xiaowen Chu. IRS: A large synthetic indoor robotics stereo dataset for disparity and surface normal estimation. *arXiv preprint arXiv:1912.09678*, 2019.

[46] Wenshan Wang, Delong Zhu, Xiangwei Wang, Yaoyu Hu, Yuheng Qiu, Chen Wang, Yafei Hu, Ashish Kapoor, and Sebastian Scherer. TartanAir: A dataset to push the limits of visual slam. In *IROS*, 2020.

[47] Ke Xian, Chunhua Shen, Zhiguo Cao, Hao Lu, Yang Xiao, Ruibo Li, and Zhenbo Luo. Monocular relative depth perception with web stereo data supervision. In *CVPR*, 2018.

[48] Ke Xian, Jianming Zhang, Oliver Wang, Long Mai, Zhe Lin, and Zhiguo Cao. Structure-guided ranking loss for single image depth prediction. In *CVPR*, 2020.

[49] Yao Yao, Zixin Luo, Shiwei Li, Jingyang Zhang, Yufan Ren, Lei Zhou, Tian Fang, and Long Quan. BlendedMVS: A large-scale dataset for generalized multi-view stereo networks. *CVPR*, 2020.

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

[51] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. In *ICLR*, 2016.

[52] Yuhui Yuan, Xilin Chen, and Jingdong Wang. Object-contextual representations for semantic segmentation. In *ECCV*, 2020.

[53] Hang Zhang, Chongruo Wu, Zhongyue Zhang, Yi Zhu, Zhi Zhang, Haibin Lin, Yue Sun, Tong He, Jonas Muller, R. Manmatha, Mu Li, and Alexander Smola. ResNeSt: Split-attention networks. *arXiv preprint arXiv:2004.08955*, 2020.

[54] Hengshuang Zhao, Jiaya Jia, and Vladlen Koltun. Exploring self-attention for image recognition. In *CVPR*, 2020.

[55] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *CVPR*, 2017.

[56] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ADE20K dataset. In *CVPR*, 2017.

[57] Xingyi Zhou, Dequan Wang, and Philipp Krähenbühl. Objects as points. *arXiv preprint arXiv:1904.07850*, 2019.