Cascaded Transposed Long-range Convolutions for Monocular Depth Estimation

Go Irie, Daiki Ikami, Takahito Kawanishi, and Kunio Kashino

NTT Corporation, Kanagawa 243-0124, Japan
goirie@ieee.org go.irie.nv@hco.ntt.co.jp

Abstract. We study the shape of the convolution kernels in the upsampling block for deep monocular depth estimation. First, our empirical analysis shows that the depth estimation accuracy can be improved consistently by only changing the shape of the two consecutive convolution layers with square kernels, e.g., $(5 \times 5) \rightarrow (5 \times 5)$, to two "long-range" kernels, one having the transposed shape of the other, e.g., $(1 \times 25) \rightarrow (25 \times 1)$. Second, based on this observation, we propose a new upsampling block called Cascaded Transposed Long-range Convolutions (CTLC) that uses parallel sequences of two long-range convolutions with different kernel shapes. Experiments with NYU Depth V2 and KITTI show that our CTLC offers higher accuracy with fewer parameters and FLOPs than state-of-the-art methods.

1 Introduction

Depth information often provides useful clues for recognizing the 3D structure of a scene, which is essential for many practical applications such as 3D scene reconstruction, navigation, and autonomous driving. While the performance of depth sensors has improved, obtaining dense and accurate depth information still requires high-end sensors. Unfortunately, such sensors cannot always be available due to limitations of device size or cost constraints. To overcome this issue, depth estimation from a single RGB image has received much attention in recent years.

Monocular depth estimation is an ill-posed problem; it is impossible to recover the depth of a scene from only a single RGB image without any assumptions or knowledge about the scene. Modern approaches rely on deep convolutional neural networks (CNNs) to learn a direct mapping from an RGB image to a corresponding depth map [1–3]. The majority of existing methods can be grouped into supervised [2–4] and self-supervised approaches [5]. The former uses ground truth depth maps measured by a depth sensor to train the prediction network, and the latter instead trains the network using a stereo pair or a sequence of moving frames that can be used to estimate depth. Although the self-supervised approach has a significant advantage of being trainable without explicit ground truth information, the supervised approach still tends to accurate. In this paper, we consider the supervised approach for monocular depth estimation.

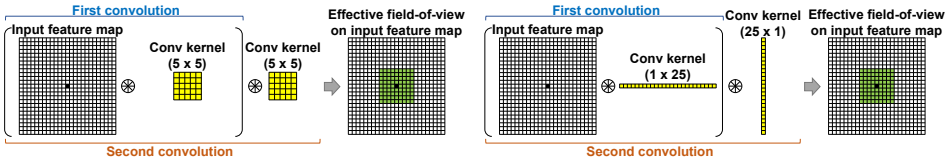

Fig. 1: **Kernel shape and effective field-of-view**. (a) Two convolution layers whose kernel shapes are (5×5) are applied to the input feature map. The size of the effective field of view, i.e., the number of the pixels on the input feature map used to compute the value of the output feature map at the position of the black pixel, is 9×9 (green pixels). (b) The shape of the two convolution kernels is changed to (1×25) and (25×1), respectively. Although the number of parameters and FLOPs are exactly the same as the case of (a), the area of the effective field-of-view (green) is dramatically increased to 25×25.

A common approach is to use an encoder-decoder type CNN. The encoder part is typically built by using a Fully Convolutional Network (FCN) to extract a low-resolution feature map, and the decoder part upsamples the feature map to recover the target depth map by applying several upsampling blocks. Previous studies have explored diverse aspects of this common strategy to improve the final prediction performance, including network architectures [1–3, 6], loss functions [1, 3, 4, 7], and usage of additional cues such as sparse depth [4, 8] or relative depth information [9], just to name a few.

We focus on the shape of the convolution kernels in the upsampling blocks. To the best of our knowledge, the only existing method that has focused on the shape of the kernels in the upsampling blocks is Whole Strip Masking (WSM) [10]. WSM uses a "long-range" convolution whose kernel is designed so that either of its vertical or horizontal length is equal to that of the input feature map. By stacking several WSM blocks, the receptive field spanned by the network as a whole effectively covers a wide area of the input RGB image. However, a single WSM layer only looks at pixels on approximately the same vertical or horizontal line to compute each output pixel value. This may overlook potential dependencies between vertical and horizontal directions on each intermediate feature map, which should be a powerful clue for accurate depth estimation. Another disadvantage would be that, because the kernel size depends on the size of the input feature map, it can only be applied to images of the same size (similar to a CNN that has fully connected layers), which may not always be desirable in practice.

We propose a modified upsampling block called Cascaded Transposed Long-range Convolutions (CTLC) free from these problems. Unlike WSM that applies a single long-range convolution separately to the input feature map, the core idea of our CTLC is to sequentially apply two long-range convolutions, one having the spatially transposed shape of the other. More specifically, suppose there are two consecutive convolution layers with normal square kernels, $(5 \times 5) \rightarrow$

(5×5), as shown in Fig. 1a. Our CTLC changes their shapes to $(1 \times 25) \rightarrow (25 \times 1)$ as in Fig. 1b. A notable advantage of our CTLC block is that the resulting area of the effective field-of-view is dramatically increased, without changing the number of parameters and FLOPs. Moreover, unlike WSM, the kernel size of CTLC does not depend on the size of the input image, so it can be applied to images of any size. Our final CTLC block is configured to apply multiple convolutional sequences with different kernel shapes in parallel to capture different levels of contextual information of the input feature map by a single upsampling block. Experiments on two standard benchmark datasets for monocular depth estimation, namely NYU Depth V2 [11] and KITTI [12], demonstrate that our CTLC block yields consistently better accuracies than existing upsampling blocks and can outperform several state-of-the-art methods.

2 Related Work

We briefly review previous studies on supervised monocular depth estimation and upsampling block that are relevant to this work.

2.1 Supervised Monocular Depth Estimation

Supervised methods assume that ground truth depth maps, usually measured by a range imaging system such as an infrared camera or LiDAR, are available during the training and train a model so that the error between the model's output and the ground truth depth map is minimized. Early attempts addressed the problem using probabilistic structured prediction models, such as Markov Random Fields (MRFs) and Conditional Random Fields (CRFs) [13–15], or an external database to obtain a depth map with RGB content similar to the target scene [16]. These methods generally used hand-crafted features, but designing useful features has often been a difficult problem. Modern approaches use deep learning. The ability to learn features directly from the data has been shown to improve performance significantly, and since then, a variety of researches have been conducted on network architecture design, loss function, and so on.

Network architectures. One of the key requirements for the network architecture is to effectively model the multiscale nature of the depth estimation task, which has been a mainstream of the architecture study. Eigen et al. proposed using multiple FCNs, where each FCN aims to infer different scale levels of the depth map [1, 2]. Liu et al. [17] proposed a model that incorporates a CRF into the FCN to obtain a smoother depth map [17]. Xu et al. [6] also used a variant of CRFs to fuse multiscale outputs extracted from the intermediate layers. Lee et al. [18] fused multiple depth predictions at different cropping rates with the weights obtained by Fourier domain analysis. Fu et al. [7] proposed to use Atrous Spatial Pyramid Pooling (ASPP) that is originally proposed for semantic segmentation tasks [19] to capture the different levels of contextual information of the feature map. Lee et al. [20] also used an extended version of ASPP called Dense ASPP [21].

Loss functions. [4] investigated several standard loss functions, such as the mean absolute error (MAE) and the mean square error (MSE). Robust and "anti-robust" loss functions have also been explored in several studies [3, 22]. [1] introduced a scale-invariant loss to mitigate a harmful effect that the average scale of the scene dominates the overall prediction error. [23] proposed an attention-driven loss to take into account the long-tail distribution of depth values. Fu et al. proposed to solve the problem as an ordinal regression problem by quantizing depth values [7].

Other attempts. Several studies have focused on densifying sparse depth maps with the help of RGB images [4, 24]. Given the strong correlation between structured visual prediction tasks, such as depth estimation, semantic segmentation, and normal map estimation, some studies explored multi-task learning approaches of these tasks [2, 25]. [26] showed that the result could be improved by explicitly modeling the uncertainties of the depth information.

The focus of this work is on the network architecture, more specifically the upsampling block, which will be described in detail in the next subsection.

2.2 Upsampling Block

The majority of the existing architectures for monocular depth estimation have a feature extraction backbone to extract a low-resolution feature map and upsampling blocks to recover the target depth size. Hence, the performance of the upsampling block greatly affects the final depth estimation accuracy. Given a general architecture that has a feature extraction backbone (e.g., ResNet-50 [27]) and an upsampling network consisting of a sequence of four upsampling blocks, Laina et al. [3] explored several types of upsampling blocks including unpooling (UnPool), upconvolution (UpConv), deconvolution (DeConv) and upprojection (UpProj), and showed that UpProj outperforms the others. The most relevant research to our work would be WSM [10]. As discussed in the introduction, a major drawback of WSM is that it cannot capture dependencies between vertical and horizontal directions on each intermediate feature map, and our CTLC overcomes this weakness. Vertical pooling [28] that pools the input feature map along the vertical direction and RowColCNN [29] that uses long-range kernels for image distortion correction have also the same weakness as WSM. Our experimental results in Sec. 4 will demonstrate that our CTLC can achieve significantly better performance than Vertical Pooling and WSM.

Several upsampling networks have also been explored in some neighboring fields, such as image classification and semantic segmentation. Atrous convolution [19] was originally proposed for semantic segmentation, performing convolutions at "wider intervals" to ensure larger receptive fields. DUpsampling [30] is a trainable upsampling layer that has been proposed to generate finer segmentation results from low-resolution images. More recently, several methods[31, 32] proposed using self-attention for CNNs [33] to integrate information over the entire image. Inception V3, which is a well-known architecture proposed by

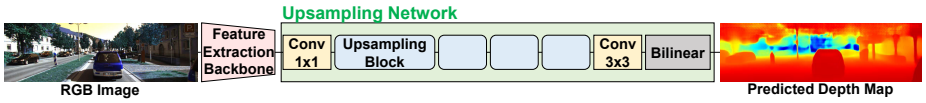

Fig. 2: **Our architecture of entire depth estimation network**. The entire structure follows [3] and consists of two major parts. One is a feature extraction backbone (encoder) that is typically constructed by using an image classification network such as ResNet-50 and DenseNet-161. The other is an upsampling network (decoder) which has one (1×1) convolution layer to reduce the number of channels by half, four upsampling blocks, one (3×3) convolution layer, and one bilinear upsampling layer to predict the target depth map.

Szegedy et al. for image classification, uses an idea of decomposing a square convolution kernel into two vertical and horizontal kernels [34]. Unlike their method that reduces the number of parameters for efficient convolution operations, our CTLC changes only the shape of the kernel without changing the number of parameters, which has not been considered in [34].

3 Method

We describe the details of our CTLC upsampling block in this section. We first present the entire architecture of our depth estimation network used in this paper. We next explain the core idea of our CTLC upsampling block and prove its effectiveness by showing empirical results of how the shape of the convolution kernel affects the final depth estimation performance. Finally, we introduce the final architecture of our CTLC upsampling block.

3.1 Entire Architecture of Depth Estimation Network

The entire architecture of our depth estimation network is illustrated in Fig 2. Overall, this has the same structure proposed in some previous work [3, 4] and consists of a feature extraction backbone and an upsampling network. The feature extraction backbone takes an RGB scene image and outputs a low-resolution feature map. Following [3, 4], we use popular CNN architectures for image classification (e.g., ResNet-50) pre-trained on ImageNet and adapt them to our depth estimation task by discarding a few top layers. We will detail this adaptation process later in Sec. 4. The upsampling network is built by stacking several upsampling blocks in which each doubles the spatial resolution of the input feature map while reducing the number of channels into a half [3]. Typically, four blocks are involved, resulting in a 16 times larger feature map than that generated by the feature extraction backbone.

3.2 Core Concept of CTLC

We start from UpProj that is a simple and effective upsampling block proposed in [3] and introduce the core concept of our CTLC.

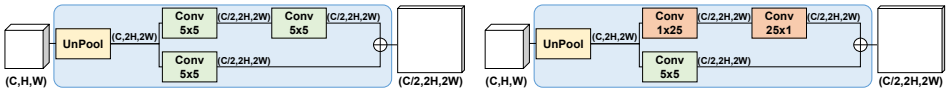

Fig. 3: **Architecture of upsampling block**. (a) UpProj and (b) our CTLC blocks with $(1 \times 25) \rightarrow (25 \times 1)$ convolutions. (C, H, W) means the number of channels, height, and width of the input feature map. A batch normalization layer and a ReLU activation function follow each convolution layer. $\oplus$ means element-wise addition.

The architecture of the UpProj block is given in Fig. 3a. After the input feature map is doubled in size by the UnPool layer, it passes through two separate branches and then fused into a single feature map with an element-wise addition. While the bottom branch in Fig. 3a simply applies a single (5×5) convolution to the input feature map, the upper branch uses a sequence of two convolutions which we denote by $(5 \times 5) \rightarrow (5 \times 5)$ for brevity[1].

The goal of our CTLC is to achieve a larger effective field of view both vertically and horizontally than the square convolutions while keeping the computational costs the same. The key idea is to modify the kernel shape of the two consecutive convolutions in the upper branch, such that (1) each is longer in either direction and (2) one is to be the transpose of the other. Such a modification can be done systematically as follows. Suppose we have two consecutive convolutions with size k, i.e., $(k \times k) \rightarrow (k \times k)$. We can obtain the CTLC versions of them as $(\lceil k/c \rceil \times \lceil ck \rceil) \rightarrow (\lceil ck \rceil \times \lceil k/c \rceil)$, where $0 < c \leq k$ is a parameter to control the aspect ratio. For instance, when we set $c = 5$ for the case of $k = 5$, we have $(1 \times 25) \times (25 \times 1)$. We consider only the case where both sides of each kernel are odd, in order to keep the size of the input and output feature maps exactly the same, and thus the possible combinations we have for $k = 5$ are the following four patterns: $(9 \times 3) \rightarrow (3 \times 9)$, $(3 \times 9) \rightarrow (9 \times 3)$, $(25 \times 1) \rightarrow (1 \times 25)$ and $(1 \times 25) \rightarrow (25 \times 1)$. The example of the resulting CTLC block with $(1 \times 25) \rightarrow (25 \times 1)$ kernels can be illustrated in Fig. 3b.

To examine the impact of the kernel shapes on the final depth estimation accuracy, we evaluate the accuracy when we change only the kernel shapes while keeping the rest (further details of the protocol will be given later in Sec. 4). We tested two feature extraction backbone networks, DenseNet-161 and ResNet-50. Table 1 shows the results for the five different combinations of the kernel shapes. The results show that the shape of the convolution kernels significantly affects the final depth prediction performance. Interestingly, the performance of the square kernel is the worst for both feature extraction backbone networks, and the longer the kernel shapes, the better the performance. We also evaluated cases that use kernels having the same lengths as the input feature map as adopted in WSM

[1] Although [3] uses $(5 \times 5) \rightarrow (3 \times 3)$, we in this work restrict it to be $(5 \times 5) \rightarrow (5 \times 5)$ for systematic discussion.

Table 1: **Impact of kernel shapes in upsampling block on depth estimation performance.** The scores are evaluated on NYU Depth V2 dataset. All the kernels have almost the same number of parameters (5×5). The best score for each metric and each feature extraction backbone (DenseNet-161 or ResNet-50) is shown in bold.

Backbone	Kernel Shape	higher is better			lower is better			
		δ_1	δ_2	δ_3	AbsRel	SqRel	RMSE	$\text{RMSE}_{\log}$
DenseNet-161	$(5 \times 5) \rightarrow (5 \times 5)$	0.850	0.975	0.995	0.126	0.075	0.431	0.156
	$(9 \times 3) \rightarrow (3 \times 9)$	0.857	0.977	0.995	0.122	0.072	0.426	0.154
	$(3 \times 9) \rightarrow (9 \times 3)$	0.857	0.978	**0.996**	0.121	0.071	0.425	0.153
	$(25 \times 1) \rightarrow (1 \times 25)$	0.867	0.978	0.995	0.119	0.070	0.412	0.148
	$(1 \times 25) \rightarrow (25 \times 1)$	**0.871**	**0.979**	0.995	**0.116**	**0.068**	**0.409**	**0.147**
	$(H \times 1) \rightarrow (1 \times W)$	0.873	0.978	0.996	0.118	0.070	0.407	0.147
	$(1 \times W) \rightarrow (H \times 1)$	0.871	0.979	0.996	0.115	0.068	0.413	0.147
ResNet-50	$(5 \times 5) \rightarrow (5 \times 5)$	0.812	0.965	0.992	0.143	0.094	0.478	0.177
	$(9 \times 3) \rightarrow (3 \times 9)$	0.829	0.968	0.993	0.137	0.087	0.459	0.169
	$(3 \times 9) \rightarrow (9 \times 3)$	0.828	0.968	0.993	0.137	0.087	0.461	0.170
	$(25 \times 1) \rightarrow (1 \times 25)$	0.863	0.977	0.994	0.120	0.073	0.418	0.150
	$(1 \times 25) \rightarrow (25 \times 1)$	**0.867**	**0.980**	**0.995**	**0.117**	**0.069**	**0.413**	**0.148**
	$(H \times 1) \rightarrow (1 \times W)$	0.864	0.976	0.994	0.120	0.073	0.413	0.150
	$(1 \times W) \rightarrow (H \times 1)$	0.866	0.979	0.996	0.117	0.070	0.413	0.149

[10], e.g., $(1 \times W) \rightarrow (H \times 1)$, but they did not give a consistent performance improvement over our CTLC counterparts such as $(1 \times 25) \rightarrow (25 \times 1)$, despite their non-negligible increases in the number of parameters[2]. These results suggest the effectiveness of the core concept of our CTLC.

There are several possible reasons. First, the sequence of a long-range convolution and its transposition allows the upsampling block to have a far larger field of view than a sequence of two square kernels, as shown in Fig. 1. The importance of the size of the field of view in deep monocular depth estimation has been pointed out in several studies, e.g., [3]. However, to the best of our knowledge, there has been no attempt to increase its size by changing only the shape of the convolution kernels in monocular depth estimation. Another reason may be that information obtained from pixels in the horizontal and vertical lines is often vital in depth estimation. An RGB image and a depth map captured in a natural setting are often taken almost parallel to the ground. In this case, in which case, if no object is present in the scene, pixels in the same horizontal line are considered to be almost equidistant from the camera. Furthermore, a recent study on deep monocular depth estimation [35] suggests that the network pays attention to the vertical position of the object to estimate the depth. From these observations, the pixels on the same horizontal and vertical lines provide important clues, which can be effectively captured by our CTLC.

[2] For example, $(1 \times W) \rightarrow (H \times 1)$ has 1.46 times the number of parameters than $(1 \times 25) \rightarrow (25 \times 1)$ for the DenseNet-161 backbone.

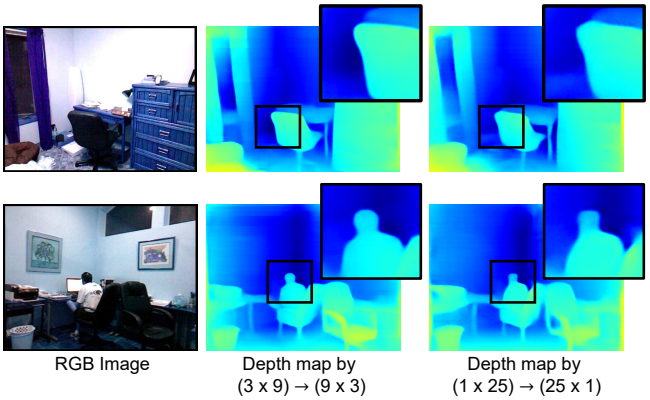

Fig. 4: **Examples of estimated depth map**. Each row shows an RGB image (left) and corresponding depth maps estimated by $(3 \times 9) \rightarrow (9 \times 3)$ (center) and $(1 \times 25) \rightarrow (25 \times 1)$ (right), respectively. The area inside the black frame is enlarged and displayed at the top right of each image.

3.3 Final CTLC Block

We propose our final CTLC block by introducing two modifications to the architecture shown in Fig. 3b. One is the parallelization of CTLCs to improve the accuracy, and the other is leveraging the idea of sub-pixel convolution [36] to reduce the number of parameters and FLOPs.

Parallel CTLC. Fig. 4 shows some examples of the depth maps estimated by the upsampling block with $(3 \times 9) \rightarrow (9 \times 3)$ and $(1 \times 25) \rightarrow (25 \times 1)$. The results shown in Table 1 show that $(1 \times 25) \rightarrow (25 \times 1)$ tends to perform better than $(3 \times 9) \rightarrow (9 \times 3)$ in average. However, looking at the examples in the figure, we can see that $(3 \times 9) \rightarrow (9 \times 3)$ better recovers corners and roundish objects (such as a human) than $(1 \times 25) \rightarrow (25 \times 1)$. This may be because $(3 \times 9) \rightarrow (9 \times 3)$ is closer to square than $(1 \times 25) \rightarrow (25 \times 1)$, and thus is useful for restoring local shapes that would otherwise be difficult to recover without looking vertical and horizontal directions at the same time. Furthermore, since the second convolution sees only the features aggregated by the first convolution, the order of the kernels to be applied determines the priority of the directions. According to the suggestion above by [35], this should be determined based on the presence or absence of objects. These discussions suggest that a single sequence of CTLC may not be sufficient to capture the full spectrum of depth information.

We, therefore, propose a CTLC block that uses multiple sequences of different kernel shapes in parallel. The architecture of the proposed parallel CTLC block is indicated by the red dotted box in Fig. 5. The input feature map passes through parallel branches of four CTLCs with different shape patterns and then concatenated into a single feature map. Compared to the original CTLC block

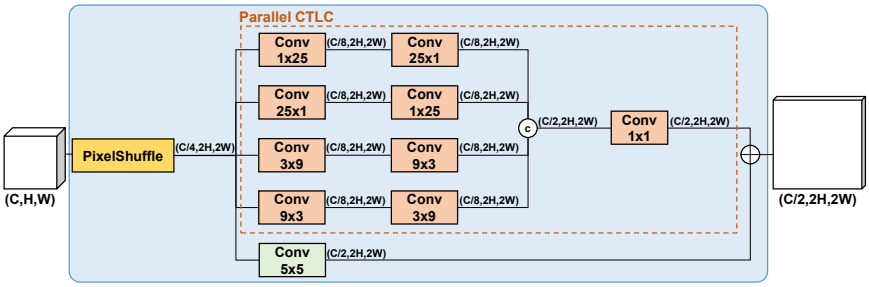

Fig. 5: **Full CTLC block**. (C, H, W) means the number of channels, height, and width of the input feature map. A batch normalization layer and ReLU follow each convolution layer. ⓒ and ⊕ mean concatenation and element-wise addition, respectively.

having only a single branch, the number of channels in each branch is reduced to 1/4. The concatenated feature map is linearly projected by (1×1) convolution. This parallel architecture allows the upsampling block to effectively capture different context information, which leads to better estimation accuracy.

Sub-pixel convolution. UnPool is not efficient because it doubles the spatial resolution of the input feature map without reducing the number of channels. Only a quarter of the pixels in the resulting feature map have non-zero values. This increases the number of parameters of the subsequent convolution layers and requires more FLOPs. We resolve this issue based on the idea of sub-pixel convolution [36] and replace UnPool with the pixel shuffle operation (PixelShuffle) that relocates the pixels of the tensor of shape $C \times H \times W$ to that of $C/4 \times 2H \times 2W$. The number of channels can be reduced to 1/4 while retaining the same information. This makes it possible to suppress both the number of parameters and FLOPs to 1/3 or less without losing depth estimation accuracy.

Fig. 5 shows the architecture of our final CTLC block. Unless otherwise specified, the proposed method uses this architecture for the upsampling block.

4 Experiments

We evaluate the performance of our CTLC upsampling block for monocular depth estimation.

4.1 Protocol

We follow the common protocol adopted in most of the previous papers (e.g., [2–4]) as detailed below.

Dataset. We use two datasets that are commonly used to evaluate monocular depth estimation. The first one is **NYU Depth V2** [11] which is the most

widely-used benchmark dataset for indoor depth estimation tasks. It consists of RGB and depth images of 464 indoor scenes with a resolution of 480×640. Following the official split, we sampled 48k RGB-depth pairs from the raw RGB/depth sequences of the official training set (249 scenes) for training and used all the 654 images from the official test set (215 scenes) for evaluation. Each depth map is spatiotemporally aligned with the corresponding RGB image and hole-filled by applying a cross-bilateral filter. The second one is **KITTI** [12] which is frequently used for outdoor monocular depth estimation evaluation. KITTI consists of image frames and depth maps of 61 outdoor scenes, including "city", "residential", "road", and "campus" categories. All the images are captured by a camera and LiDAR sensor mounted on a car with a resolution of 375×1241. Following the common protocol called "Eigen split" introduced by [1], we use $23,488$ images from 32 categories for training and 697 images extracted from the remaining 29 scenes for evaluation. Our method is evaluated with the ground truth depth within $80m$ or $50m$.

Feature extraction backbone. We use three types of feature extraction backbone networks including DenseNet-161 [37], ResNet-50 [27], and MobileNetV2 [38]. The original architecture of DenseNet-161/ResNet-50 consists of four dense/residual blocks for feature extraction and one classification block (a fully-connected layer after a global average pooling layer) at the top to output the final classification results. Following [3, 4], we adapted these architectures to the monocular depth estimation task by replacing their classification block with the upsampling network shown in Fig. 2. For MobileNetV2, we also replaced the top convolution and average pooling layers for classification with the upsampling network. The parameters are initialized with those pre-trained on ImageNet and fine-tuned with the training set of the corresponding dataset.

Training details. We use PyTorch for implementing all the models and running the experiments. The training is performed using Adam optimizer for 50 epochs. The learning rate is initially set to 10^{-4} and polynomially decayed during training to 10^{-5}. Other parameters we use are $\beta_1 = 0.9$, $\beta_2 = 0.999$, a weight decay of 10^{-4}, and a mini-batch size of 6. For all the experiments, we use the scale-invariant loss [1] to train our networks.

$$\ell(e) = \frac{1}{n}\sum_{i}^{n} e_i^2 - \frac{1}{n^2}\left(\sum_{i}^{n} e_i\right)^2 \tag{1}$$

where e_i and n are the pixel-level difference at i-th pixel between the estimated and ground truth depth maps and the number of pixels, respectively. We also use several standard data augmentation techniques commonly used in monocular depth estimation [1, 4, 7]. The RGB and depth images are randomly rotated in $[-5, +5]$ (deg), horizontally flipped with a 50% chance, and each of the RGB values is scaled by $[0.5, 1.5]$. We train our model on a random crop of size 416×544 for NYU Depth V2 and 385×513 for KITTI.

Table 2: **Comparison of upsampling blocks on NYU Depth V2** with three backbone networks. #Params and GFLOPs are the number of trainable parameters and gigaFLOPs of each upsampling network, respectively. "Ours (UP)" and "Ours (PS)" use UnPool and PixelShuffle for upsampling, respectively. The best and the second-best are in bold and underlined, respectively.

Backbone	Upsampling Block	higher is better			lower is better				#Params	GFLOPs
		δ_1	δ_2	δ_3	AbsRel	SqRel	RMSE	$\text{RMSE}_{\log}$	(million)	
ResNet-50	UpConv [3]	0.773	0.955	0.990	0.157	0.111	0.520	0.194	25.2	117.2
	UpProj [3]	0.812	0.965	0.992	0.143	0.094	0.478	0.177	58.2	293.0
	Sub-pixel Conv [36]	0.802	0.962	0.992	0.147	0.099	0.485	0.181	18.1	117.2
	Atrous Conv [19]	0.855	0.975	0.994	0.125	0.075	0.427	0.156	58.2	293.0
	Self-attention [33]	0.830	0.970	0.994	0.136	0.086	0.452	0.167	25.7	120.6
	DUpsampling [30]	0.840	0.972	**0.995**	0.129	0.079	0.442	0.162	123.9	53.3
	DASPP [21]	0.857	**0.978**	**0.995**	0.121	0.074	0.433	0.154	97.5	332.6
	WSM [10]	0.812	0.969	0.993	0.158	0.102	0.474	0.176	**17.4**	**38.0**
	Ours (UP)	**0.866**	0.976	0.994	0.119	0.073	**0.412**	**0.149**	41.4	256.3
	Ours (PS)	0.865	**0.978**	**0.995**	**0.117**	**0.071**	0.415	0.150	17.7	77.4
DenseNet-161	UpConv [3]	0.843	0.975	0.995	0.128	0.077	0.436	0.160	24.4	136.2
	UpProj [3]	0.850	0.975	0.995	0.126	0.075	0.431	0.156	57.5	340.6
	Sub-pixel Conv [36]	0.846	0.975	**0.996**	0.127	0.075	0.430	0.158	17.4	136.3
	Atrous Conv [19]	0.866	0.977	0.995	0.118	0.070	0.419	0.150	57.5	340.6
	Self-attention [33]	0.861	0.978	0.995	0.121	0.071	0.417	0.151	24.9	140.0
	DUpsampling [30]	0.856	0.977	**0.996**	0.122	0.072	0.426	0.153	123.2	66.7
	DASPP [21]	0.874	**0.979**	0.995	0.115	0.068	0.410	0.146	96.8	385.3
	WSM [10]	0.830	0.975	0.995	0.150	0.091	0.456	0.167	21.8	**44.2**
	Ours (UP)	**0.877**	**0.979**	0.995	**0.114**	**0.067**	**0.404**	**0.145**	50.6	297.1
	Ours (PS)	0.874	**0.979**	0.995	0.115	0.068	0.406	0.146	**16.9**	89.4
MobileNetV2	UpConv [3]	0.773	0.955	0.990	0.157	0.111	0.520	0.194	7.6	45.8
	UpProj [3]	0.778	0.952	0.989	0.157	0.115	0.526	0.196	17.8	114.5
	Sub-pixel Conv [36]	0.762	0.949	0.988	0.162	0.120	0.541	0.201	5.4	45.8
	Atrous Conv [19]	0.820	**0.967**	0.992	0.138	0.092	0.478	0.174	17.8	114.5
	Self-attention [33]	0.801	0.960	0.990	0.150	0.103	0.498	0.183	7.7	47.5
	DUpsampling [30]	0.805	0.964	**0.993**	0.143	0.095	0.489	0.178	106.8	20.8
	DASPP [21]	0.827	**0.967**	0.992	**0.133**	**0.089**	0.474	0.170	29.9	133.5
	WSM [10]	0.788	0.959	0.991	0.163	0.115	0.515	0.188	6.8	**14.8**
	Ours (UP)	**0.833**	**0.967**	0.991	0.134	0.090	0.466	**0.169**	15.7	100.3
	Ours (PS)	0.829	**0.967**	0.992	0.135	**0.089**	**0.465**	**0.169**	**5.3**	30.2

Evaluation metrics. We employ the following metrics for evaluation. The predicted and ground truth depth values at i-th pixel ($1 \leq i \leq n$) are denoted by $\hat{z}_i$ and z_i^*.

- δ_α: ratio of pixels whose relative error is within 1.25^α. We use $\alpha \in \{1, 2, 3\}$. Higher is better.
- AbsRel: mean absolute relative error, i.e., $\frac{1}{n}\sum_i^n |z_i^* - \hat{z}_i|/z_i^*$. Lower is better.
- SqRel: squared version of absolute relative error, i.e., $\frac{1}{n}\sum_i^n (z_i^* - \hat{z}_i)^2/z_i^*$. Lower is better.
- RMSE: root mean square error, i.e., $\sqrt{\frac{1}{n}\sum_i^n (z_i^* - \hat{z}_i)^2}$. Lower is better.
- $\text{RMSE}_{\log}$: logarithmic root mean square error, i.e., $\sqrt{\frac{1}{n}\sum_i^n (\log z_i^* - \log \hat{z}_i)^2}$. Lower is better.

Table 3: **Comparison with state-of-the-art methods on NYU Depth V2.** The best and the second-best scores for each metric and backbone network are shown in bold and are underlined, respectively.

Method	Backbone	higher is better			lower is better	
		δ_1	δ_2	δ_3	AbsRel	RMSE
Saxena et al. [14]	-	0.447	0.745	0.897	0.349	1.214
Eigen et al. [2]	VGG	0.769	0.950	0.988	0.158	0.641
Liu et al. [17]	AlexNet (Custom)	0.650	0.906	0.976	0.213	0.759
Laina et al. [3]	ResNet-50	0.811	0.953	0.988	0.127	0.573
Kendall et al. [26]	DenseNet (Custom [39])	0.817	0.959	0.989	**0.110**	0.506
Xu et al. [6]	ResNet-50	0.811	0.954	0.987	0.121	0.586
Ma et al. [4]	ResNet-50	0.810	0.959	0.989	0.143	0.514
Lee et al. [18]	ResNet-152	0.815	0.963	0.991	0.139	0.572
Fu et al. [7]	ResNet-101	0.828	0.965	0.992	0.115	0.509
Qi et al. [25]	ResNet-50	0.834	0.960	0.990	0.128	0.569
Heo et al. [10]	ResNet-50	0.816	0.964	0.992	0.135	0.571
Lee et al. [40]	DenseNet-161	0.837	0.971	0.994	0.131	0.538
Ours	ResNet-50	0.865	0.978	**0.995**	0.117	0.415
Ours	DenseNet-161	**0.874**	**0.979**	**0.995**	0.115	**0.406**

4.2 Comparison with Existing Upsampling Blocks

We first compare our CTLC with the following eight existing upsampling blocks. For all the upsampling blocks, we use the same base network architecture given in Fig. 2 and the training protocol.

- **UpConv** [3] uses only the bottom branch of UpProj. Other parts of the entire network is the same as that shown in Fig. 2.
- **UpProj** [3] which is exactly Fig. 3a.
- **Sub-pixel Conv** [36] is the same as UpProj except that it uses PixelShuffle instead of UnPool.
- **Atrous Conv** [19] changes the dilation rate of all convolution layers in the upper branch of UpProj to 2.
- **Self-attention** [33] uses UpConv as the base architecture, but a self-attention block (see Figure 2 in [33]) is placed on top of the first two upsampling blocks.
- **DUpsampling** [30] uses three (5×5) convolutions and one DUpsampling layers, following [30].
- **DASPP** [21] is the same as the UpProj but has one DASPP block before the first upsampling block right after the feature extraction backbone network.
- **WSM** [10] uses the WSM block (see Fig. 5 in [10]) for each upsampling block in Fig. 2.

We use NYU Depth V2 for this experiment.

The results are presented in Table 2. Overall, we observe that our CTLC outperforms all the other upsampling blocks for all the feature extraction backbone networks. This clearly shows the effectiveness of our CTLC block. The results provide several interesting observations. First, when compared to UpProj

Table 4: **Comparison with state-of-the-art methods on KITTI.** "cap" gives the maximum depth used for evaluation. "Ours (raw)" and "Ours (GT)" are trained with raw depth maps and post-processed ground truth depth maps, both available on the official page. The best scores for each metric are shown in bold.

Method	Backbone	cap	higher is better			lower is better			
			δ_1	δ_2	δ_3	AbsRel	SqRel	RMSE	$\text{RMSE}_{\log}$
Saxena et al. [14]	-	80m	0.601	0.820	0.926	0.280	3.012	8.734	0.361
Eigen et al. [1]	AlexNet	80m	0.692	0.899	0.967	0.190	1.515	7.156	0.270
Liu et al. [17]	AlexNet (Custom)	80m	0.647	0.882	0.961	0.217	1.841	6.986	0.289
Godard et al. [5]	ResNet-50	80m	0.861	0.949	0.976	0.114	0.898	4.935	0.206
Kuznietsov et al. [41]	ResNet-50	80m	0.862	0.960	0.986	0.113	0.741	4.621	0.189
Gan et al. [28]	ResNet-50	80m	0.890	0.964	0.985	0.098	0.666	3.933	0.173
Fu et al. [7]	ResNet-101	80m	0.932	0.984	0.994	0.072	0.307	**2.727**	0.120
Ours (raw)	DenseNet-161	80m	0.896	0.972	0.990	0.093	0.519	3.856	0.155
Ours (GT)	DenseNet-161	80m	**0.951**	**0.992**	**0.998**	**0.064**	**0.271**	2.945	**0.101**
Garg et al. [42]	Alexnet (Custom)	50m	0.740	0.904	0.962	0.169	1.080	5.104	0.273
Godard et al. [5]	ResNet-50	50m	0.873	0.954	0.979	0.108	0.657	3.729	0.194
Kuznietsov et al. [41]	ResNet-50	50m	0.875	0.964	0.988	0.108	0.595	3.518	0.179
Gan et al. [28]	ResNet-50	50m	0.898	0.967	0.986	0.094	0.552	3.133	0.165
Fu et al. [7]	ResNet-101	50m	0.936	0.985	0.995	0.071	0.268	2.271	0.116
Ours (raw)	DenseNet-161	50m	0.911	0.975	0.991	0.086	0.399	2.933	0.145
Ours (GT)	DenseNet-161	50m	**0.956**	**0.994**	**0.999**	**0.065**	**0.199**	**2.141**	**0.098**

which is used as the basic model of CTLC, we can see that CTLC consistently improves performance and significantly reduces the number of parameters and FLOPs. This proves the validity of the idea of changing the shape of two consecutive convolutions into spatially transposed long-range convolutions. Second, replacing UnPool (Ours (UP)) with PixelShuffle (Ours (PS)) greatly reduces the number of CTLC parameters and FLOPs while maintaining accuracy. Note that depending on the architecture of the upsampling block, accuracy may not always be maintained. More specifically, there is no significant difference in accuracy between UpProj and Sub-pixel Conv when DenseNet-161 is selected as the feature extraction backbone. However, using ResNet-50 or MobileNetV2 gives a performance gap between them. This may be because our CTLC has a much longer field-of-view than the range of the pixels shuffled by PixelShuffle, which also illustrates its advantage. Third, our CTLC is significantly more accurate than WSM that also uses long-range convolutions. Unlike CTLC, WSM compresses the input feature map vertically or horizontally using a convolutional layer of the same length as the input feature map. This reduces FLOPs, but can result in significant loss of spatial information. In contrast, our CTLC mitigates this by using the cascades of two long-range convolutions to capture spatial information in both directions efficiently. This may allow CTLC to achieve high accuracy with fewer parameters than other methods, including WSM.

Comparison with state-of-the-art methods. We compare the performance of our CTLC with several state-of-the-art monocular depth estimation methods.

Table 5: **Ablation study.** Performance on NYU Depth V2 with DenseNet-161 feature extraction backbone is reported. #Params and GFLOPs are the number of trainable parameters and gigaFLOPs of each upsampling network, respectively. The best scores for each metric are shown in bold.

Method	higher is better			lower is better		#Params	GFLOPs
	δ_1	δ_2	δ_3	AbsRel	RMSE	(million)	
UpProj	0.850	0.975	0.995	0.126	0.431	57.5	340.6
+ Long-range	0.871	0.979	0.995	0.119	0.412	57.5	340.6
+ Parallel CTLC	**0.877**	**0.979**	**0.995**	**0.114**	**0.404**	50.6	297.1
+ PixelShuffle	0.874	**0.979**	**0.995**	0.115	0.406	**16.9**	**89.4**

The results with NYU Depth V2 and KITTI are given in Table 3 and Table 4, respectively. Overall, our method outperforms all the other methods in most cases, which proves the remarkable superiority of our CTLC. The diversity of some detailed configurations over the different methods (e.g., differences of feature extraction backbone networks) makes it not easy to perform entirely fair comparisons for all the methods. However, ours achieves better performance even with ResNet-50 backbone than several methods that adopt much stronger backbone networks such as DenseNet with 100 layers [26] or ResNet-101 [7].

Ablation study. We conduct an ablation study to analyze the effectiveness of each idea of our method. The results on NYU Depth V2 are shown in Table 5. From the normal UpProj block, the depth estimation accuracy is improved by changing the two convolution kernels in the upper branch to a pair of transposed long-range convolutions $(1 \times 25) \rightarrow (25 \times 1)$ (+ Long-range) without changing the number of parameters and FLOPs. Introducing a parallel CTLC block (+ Parallel CTLC) further improves accuracy. Finally, by replacing UnPool with PixelShuffle, both the number of parameters and FLOPs are significantly reduced while preserving accuracy. These results confirm the validity of each idea.

5 Conclusions

We introduced an upsampling block called Cascaded Transposed Long-range Convolutions (CTLC) for monocular depth estimation. Despite its simplicity of changing the kernel shape of two successive convolutions to spatially transposed long-range convolutions, it yields significantly better performance at reasonably smaller computational costs compared with existing upsampling blocks. We also demonstrated that the depth estimation network involving our final CTLC block outperforms the state-of-the-art depth estimation methods. One interesting suggestion is that changing the shape of the convolution kernels can boost the depth estimation performance, which will bring a new possibility for improving the performance of depth estimation in a way that can be easily applied to many types of network architectures. Applying the idea to other structured prediction tasks such as image segmentation would be an interesting future direction.

References

1. Eigen, D., Puhrsch, C., Fergus, R.: Depth map prediction from a single image using a multi-scale deep network. In: Proc. NeurIPS. (2014)
2. Eigen, D., Fergus, R.: Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In: Proc. ICCV. (2015)
3. Laina, I., Rupprecht, C., Belagiannis, V.: Deeper depth prediction with fully convolutional residual networks. In: Proc. 3DV. (2016)
4. Ma, F., Karaman, S.: Sparse-to-Dense: Depth prediction from sparse depth samples and a single image. In: Proc. ICRA. (2018)
5. Godard, C., Aodha, O.M., Brostow, G.J.: Unsupervised monocular depth estimation with left-right consistency. In: Proc. CVPR. (2017)
6. Xu, D., Ricci, E., Ouyang, W., Wang, X., Sebe, N.: Multi-scale continuous CRFs as sequential deep networks for monocular depth estimation. In: Proc. CVPR. (2017)
7. Fu, H., Gong, M., Wang, C., Batmanghelich, K., Tao, D.: Deep ordinal regression network for monocular depth estimation. In: Proc. CVPR. (2018)
8. Chen, Z., Badrinarayanan, V., Drozdov, G., Rabinovich, A.: Estimating depth from RGB and sparse sensing. In: Proc. ECCV. (2018)
9. Ron, D., Duan, K., Ma, C., Xu, N., Wang, S., Hanumante, S., Sagar, D.: Monocular depth estimation via deep structured models with ordinal constraints. In: Proc. 3DV. (2018)
10. Heo, M., Lee, J., Kim, K.R., Kim, H.U., Kim, C.S.: Monocular depth estimation using whole strip masking and reliability-based refinement. In: Proc. ECCV. (2018)
11. Silberman, N., Hoiem, D., Kohli, P., Fergus, R.: Indoor segmentation and support inference from RGBD images. In: Proc. ECCV. (2012)
12. Geiger, A., Lenz, P., Stiller, C., Urtasun, R.: Vision meets robotics: The KITTI dataset. IJRR **32** (2013) 1231–1237
13. Saxena, A., Chung, S.H., Ng, A.Y.: Learning depth from single monocular images. In: Proc. NeurIPS. (2006)
14. Saxena, A., Sun, M., Ng, A.Y.: Make3D: Learning 3D scene structure from a single still image. IEEE TPAMI **31** (2009) 824–840
15. Liu, B., Gould, S., Koller, D.: Single image depth estimation from predicted semantic labels. In: Proc. CVPR. (2010)
16. Karsch, K., Liu, C., Kang, S.B.: Depth transfer: Depth extraction from video using non-parametric sampling. IEEE TPAMI **36** (2014) 2144–2158
17. Liu, F., Shen, C., Lin, G., Reid, I.: Learning depth from single monocular images using deep convolutional neural fields. IEEE TPAMI **38** (2016) 2024–2039
18. Lee, J.H., Heo, M., Kim, K.R., Kim, C.S.: Single-image depth estimation based on Fourier domain analysis. In: Proc. CVPR. (2018)
19. Chen, L.C., Papandreou, G., Kokkinos, I., Murphy, K., Yuille, A.L.: DeepLab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs. IEEE TPAMI **40** (2017) 834–848
20. Lee, J.H., Han, M.K., Ko, D.W., Suh, I.H.: From big to small: Multi-scale local planar guidance for monocular depth estimation. In: arXiv preprint arXiv:1907.10326. (2019)
21. Yang, M., Yu, K., Zhang, C., Li, Z., Yang, K.: DenseASPP for semantic segmentation in street scenes. In: Proc. CVPR. (2018)
22. Irie, G., Kawanishi, T., Kashino, K.: Robust learning for deep monocular depth estimation. In: Proc. ICIP. (2019)

23. Jiao, J., Cao, Y., Song, Y., Lau, R.: Look deeper into depth: Monocular depth estimation with semantic booster and attention-driven loss. In: Proc. ECCV. (2018)
24. Cheng, X., Wang, P., Yang, R.: Depth estimation via affinity learned with convolutional spatial propagation network. In: Proc. ECCV. (2018)
25. Qi, X., Liao, R., Liu, Z., Urtasun, R., Jia, J.: GeoNet: Geometric neural network for joint depth and surface normal estimation. In: Proc. CVPR. (2018)
26. Kendall, A., Gal, Y.: What uncertainties do we need in Bayesian deep learning for computer vision? In: Proc. NeurIPS. (2017)
27. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proc. CVPR. (2015)
28. Gan, Y., Xu, X., Sun, W., Lin, L.: Monocular depth estimation with affinity, vertical pooling, and label enhancement. In: Proc. ECCV. (2018)
29. Rengarajan, V., Balaji, Y., Rajagopalan, A.N.: Unrolling the shutter: CNN to correct motion distortions. In: Proc. CVPR. (2017)
30. Tian, Z., He, T., Shen, C., Yan, Y.: Decoders matter for semantic segmentation: Data-dependent decoding enables flexible feature aggregation. In: Proc. CVPR. (2019)
31. Wang, X., Girshick, R., Gupta, A., He, K.: Non-local neural networks. In: Proc. CVPR. (2018)
32. Huang, Z., Wang, X., Huang, L., Huang, C., Wei, Y., Liu, W.: CCNet: Criss-cross attention for semantic segmentation. In: Proc. ICCV. (2019)
33. Zhang, H., Goodfellow, I., Metaxas, D., Odena, A.: Self-attention generative adversarial networks. In: Proc. ICML. (2019)
34. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z.: Rethinking the inception architecture for computer vision. In: Proc. CVPR. (2016)
35. van Dijk, T., de Croon, G.: How do neural networks see depth in single images? In: Proc. ICCV. (2019)
36. Shi, W., Caballero, J., Huszár, F., Totz, J., Aitken, A.P., Bishop, R., Rueckert, D., Wang, Z.: Real-time single image and video super-resolution using an efficient subpixel convolutional neural network. In: arXiv preprint arXiv:1609.05158. (2016)
37. Huang, G., Liu, Z., van der Maaten, L., Weinberger, K.Q.: Densely connected convolutional networks. In: Proc. CVPR. (2017)
38. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.C.: MobileNetV2: Inverted residuals and linear bottlenecks. In: Proc. CVPR. (2018)
39. Jégou, S., Drozdzal, M., Vázquez, D., Romero, A., Bengio, Y.: The one hundred layers tiramisu: Fully convolutional densenets for semantic segmentation. In: Proc. CVPR Workshops. (2017)
40. Lee, J.H., Kim, C.S.: Monocular depth estimation using relative depth maps. In: Proc. CVPR. (2019)
41. Kuznietsov, Y., Stückler, J., Leibe, B.: Semi-supervised deep learning for monocular depth map prediction. In: Proc. CVPR. (2017)
42. Garg, R., Kumar, V.B.G., Carneiro, G., Reid, I.: Unsupervised CNN for single view depth estimation: Geometry to the rescue. In: Proc. ECCV. (2016)