LiTMNet: A Deep CNN for Efficient HDR Image Reconstruction from a Single LDR Image

Guotao Wu[a,*], Ran Song[a,b,*], Mingxin Zhang[a], Xiaolei Li[a,**], Paul L. Rosin[c]

[a] *School of Control Science and Engineering, Shandong University, Jinan, China*
[b] *Institute of Brain and Brain-Inspired Science, Shandong University, Jinan, China*
[c] *School of Computer Science and Informatics, Cardiff University, Cardiff, UK*

Abstract

Existing methods can generate a high dynamic range (HDR) image from a single low dynamic range (LDR) image using convolutional neural networks (CNNs). However, they are too cumbersome to run on mobile devices with limited computational resources. In this work, we design a lightweight CNN, namely LiTMNet which takes a single LDR image as input and recovers the lost information in its saturated regions to reconstruct an HDR image. To avoid trading off the reconstruction quality for efficiency, LiTMNet does not only adapt a lightweight encoder for efficient feature extraction, but also contains newly designed upsampling blocks in the decoder to alleviate artifacts and further accelerate the reconstruction. The final HDR image is produced by nonlinearly blending the network prediction and the original LDR image. Qualitative and quantitative comparisons demonstrate that LiTMNet produces HDR images of high quality comparable with the current state of the art and is 38× faster as tested on a mobile device. Please refer to the supplementary video for additional visual results.

Keywords: HDR image reconstruction, lightweight CNN, inverse tone mapping

1. Introduction

Standard digital cameras quintessentially fail to store all dynamic ranges of a scene in challenging lighting conditions such as glare, improper contrast and poorly distributed light. Consequently, the captured images often contain unfavorably saturated regions. Thus, the demand for capturing more dynamic ranges has driven the ongoing development of high dynamic range (HDR)

*These authors contributed equally to this work.
**Corresponding author
Email address: qylxl@sdu.edu.cn@sdu.edu.cn (Xiaolei Li)

imaging techniques. A traditional approach to HDR image reconstruction is to take a sequence of low dynamic range (LDR) images at different exposure levels (known as bracketed exposures) and fuse them into a single HDR image [1], [2] or a single LDR image with improved appearance [3], [4]. An alternative [5], [6], [7], commonly referred to as inverse tone-mapping operators (iTMOs), is to reconstruct an HDR image from a single exposed LDR image by expanding the dynamic range of the LDR image.

In general, traditional approaches can produce more visually appealing images than iTMOs when the sequence of LDR images are perfectly aligned. However, most images are captured at different exposures, and ghosting artifacts may also arise due to moving contents or camera shake [8]. This is a serious limitation as cameras are often hand-held and real scenes often contain moving objects. On the contrary, iTMOs based on a single LDR image do not have the problems of alignment and ghosting artifacts.

In recent years, a number of CNN-based inverse tone-mapping methods [9], [5], [6] have produced convincing HDR results from a single LDR image. However, they incur a heavy computational cost as the CNNs employed in these methods contain computationally expensive components such as the 3D convolutional layers in [9], the VGG16 encoder in [5], and the full resolution branch in [6]. Hence, using such methods is commonly not a serious issue on high-end devices, but could be arduous when deployed on a mobile device, which nowadays is the favorite hand-held camera equipment. However, designing a good iTMO that enables HDR reconstruction on mobile devices is challenging, which should 1) produce **results of high quality**, as today mobile phone users become more and more picky about image quality, 2) obtain **high computational efficiency** so that the method can be easily deployed on mobile devices and provide a pleasurable user experience when running it, and 3) have **good generalization ability** to various images, so that it can be widely used by individuals with diverse backgrounds.

Inspired by the success of lightweight CNNs [10], [11], [12], it is intuitive to substitute some inefficient component in a state-of-the-art HDR reconstruction network for a lightweight one with similar functionality for acceleration purpose. However, such a replacement is not straightforward and often error-prone. For instance, we found that in the state-of-the-art method HDRCNN [5], directly replacing the VGG16 encoder with MobileNetV2 [10] which both extract deep features from the input LDR image caused a significant degradation of the reconstruction quality of the HDR image. Technically, it is difficult to avoid trading off performance for efficiency in HDR image reconstruction.

To address such a dilemma, this paper proposes a novel lightweight CNN, named as LiTMNet, short for Lightweight Inverse Tone Mapping Network. The LiTMNet is 38x faster than the HDRCNN on a mobile phone while it can reconstruct HDR images of comparably high quality. Specifically, the LiTMNet takes a single exposed LDR image as input, and aims to estimate and recover the lost information in the highlighted regions. Such a strategy is motivated by two reasons. First, perceptual experiments [13] have shown that the perceived quality of an image degrades substantially with over-exposure, which means

that restoring the information of highlight is of vital importance for improving visual quality. Second, to train a CNN to learn an end-to-end mapping from an 8-bit LDR image to a 32-bit HDR image directly is quite challenging and may lead to the difficulty of convergence. By contrast, only predicting the data of over-exposure regions enables the CNN to take shorter time for a certain level of convergence.

The LiTMNet is composed of three components: encoder, skip-connections and decoder. It is trained with a hybrid loss, measuring the differences between images at both pixel and feature levels. The encoder first extracts a multi-level feature representation of the input image. Then, the decoder makes full use of these features to reconstruct the details in the saturated regions. We propose a new upsampling block for the decoder to improve the quality of reconstruction by mitigating some artifacts. To avoid blurring effects, extracted features from the contracting path are combined with the upsampled features by skip-connections.

In summary, the main contributions of this work are:

- A novel lightweight CNN that is 38x faster than the state-of-the-art method [5] on mobile devices while producing comparable results.
- A new computationally efficient upsampling block that largely fixes the reconstruction artifacts.
- A publicly available android application based on LiTMNet, offering end-to-end and efficient HDR image reconstruction.

2. Related Work

Many methods for reconstructing HDR images from one or multiple LDR images have been presented. We categorize them into two groups based on their underlying rationales.

2.1. Exposure Bracketing

Exposure bracketing is a widely known technique for reconstructing HDR images. It takes multiple LDR images with different exposures and then combines them into a single HDR image [1] (typically followed by tone mapping for display) or an enhanced LDR image by exposure fusion [3]. Assuming that cameras are mounted on tripods and thus scenes are static, such methods generally produce high quality results indeed. However, such an assumption is weak in real-world applications and the ghosting problem would degrade the final result if it no longer holds.

Over the past few years, researchers have made a great effort in handling dynamic scenes while reducing ghosting artifacts [14], [15]. However, these methods usually rely on the robust alignment of LDR images, which remains an arduous problem. Thus, instead of designing a complicate but possibly brittle de-ghosting algorithm, we use a single LDR image to infer the HDR image and remove the ghosting artifacts thoroughly. This strategy works owing to the fact that the ghosting problem is essentially caused by the misalignment between multiple LDR images.

2.2. Inverse Tone Mapping

Inverse tone mapping is an ill-posed problem as information is lost due to the saturation of camera sensor. Since HDR pixels have much wider variations than LDR pixels, inferring HDR pixels from LDR pixels through inverse tone mapping is a challenging task. In the early stage, Landis [16] expanded all pixels in an LDR image based on power functions. Rempel et al. [17] first processed the input with an inverse gamma curve and noise filtering, then conducted brightness enhancement of saturated regions to generate the HDR image. Akyüz et al. [18] boosted the dynamic range of an LDR image linearly for HDR capable displays. Masia et al. [13] found that a simple iTMO based on gamma expansion is less likely to introduce artifacts, and proposed a method to automatically set a gamma value for enhancing visible details. Although these methods were relatively robust to various LDR images, they cannot retrieve the lost information from the LDR images.

Recently, many CNNs were proposed to tackle domain-specific, ill-posed problems in image processing and have achieved promising results, including denoising [19], super-resolution [20], colorization [21] and depth estimation [22], etc. Several iTMOs have been constructed based on CNNs due to their powerful learning capacity. Unlike the previous iTMOs, these CNN-based methods are designed to reproduce the missing information. Endo et al. [9] made an attempt to infer a sequence of exposure-bracketed LDR images from a single LDR input, and then these synthetic images were fused into an HDR image. To learn the relative changes of pixel values, they used 3D deconvolutional layers, which are computationally expensive even on a high-end desktop computer. By assuming that the solar azimuth is the same in all images, Zhang and Lalonde [23] proposed an HDR reconstruction method specifically designed to recover the extremely high dynamic range close to the sunlight while the resolutions of HDR outputs were limited to 128×64 pixels.

2.3. Difference to Closely Related Work

As the proposed LiTMNet relies only on a single LDR image to reconstruct an HDR image, it fundamentally differs from the methods that require multiple LDR images for reconstruction, such as [24], [25] and [26].

Although LiTMNet is closely related to HDRCNN [5] based on a hybrid dynamic range auto-encoder as well, both the encoder and the decoder of LiTMNet are significantly different from those of HDRCNN: 1) The encoder layers in HDRCNN based on VGG16 are mostly replaced with MobileNetV2 [10] in LiTMNet, leading to a considerable improvement of efficiency; 2) However, since simply replacing the VGG16 encoder with MobileNetV2 leads to a poor HDR reconstruction, LiTMNet uses a different decoder with a specifically designed skip-connection scheme and a new upsampling block to mitigate the reconstruction artifacts.

Compared with some latest methods such as FHDR [27] and SingleHDR [28], the network architecture of LiTMNet is much lighter as the former repeatedly uses feedback blocks which are dense and the latter consists of four sub-networks trained jointly by combining six loss functions.

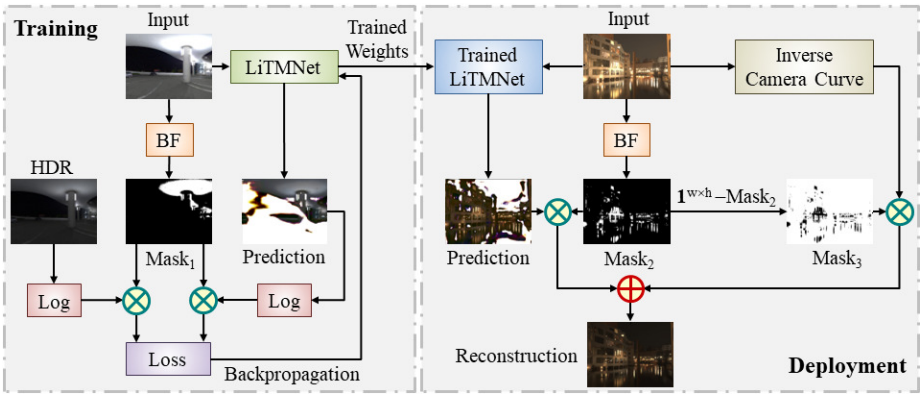


Figure 1: Workflow of the proposed method. Log denotes the logarithmic operation. The inverse camera curve is implemented using $f(x) = x^2$. BF denotes the blending function defined in Eq. (1). The images annotated as HDR, Prediction and Reconstruction respectively are shown with low exposure for display purpose.

3. Method

In this section, to make this paper self-contained, we first provide the formulation of the problem, followed by an overview of the proposed pipeline. Then, we present the core component of our method, the LiTMNet, where the newly designed upsampling block vital for boosting the reconstruction quality is elaborated.

3.1. Problem Formulation

Given an LDR image of single exposure, our objective is to reconstruct the lost information in the saturated regions. To produce HDR images, HDRCNN [5] combines the reconstructed pixels with linearized LDR inputs via a blending function with a mask image which highlights the saturated regions and keeps the unsaturated regions unmodified. It is a triangle weighting scheme in which the weight α_i is defined as

$$\alpha_i = \frac{max(0, max_c(L_{i,c}) - \tau)}{1 - \tau} \tag{1}$$

where τ is set to 0.95 in [5] and $L_{i,c}$ denotes the LDR input pixel with spatial index i and channel c.

Following such an objective, an intuitive way to attain a more efficient network is to replace some layers of the HDRCNN with lightweight versions while technically, such a replacement might not be trivial. We thus propose a method based on such a blending strategy where the core component is the efficient LiTMNet for predicting the pixels in the saturated regions.

3.2. Workflow of the Proposed Method

Fig. 1 illustrates the overview of our method in both training and deployment modes. In the training mode, Eq. (1) is applied to the LDR input to create $Mask_1$. Guided by such a mask, the LiTMNet desirably focuses on the reconstruction of the lost information in the saturated regions. The loss function compares the output of the LiTMNet to the ground truth HDR image in the log domain. This is important as it weakens the influence of extremely large pixel values of the HDR image and thus stabilizes the gradients in the training process. It is worth mentioning that large luminance values will not significantly affect the training of the network. This is because as shown in Fig. 1, the loss is calculated after the log operation which effectively suppresses large luminance values. The linearization applied onto the input image makes the network focus on the reconstruction of the over-exposed image region so that it does not need to carry out domain transform with extra cost. Both linearization and log transformation are simple but widely used schemes to optimize the distribution of luminance values in HDR image reconstruction [5], [29]. In the deployment mode, the log transformation is no longer needed as it is only used for calculating the loss. Since the LiTMNet focuses on over-exposure regions, artifacts may appear in other areas of the image predicted by the LiTMNet. Therefore, the final result is the combination of the unsaturated areas in the LDR image and the saturated areas in the prediction. Apart from $Mask_1$, we further define two masks for the combination. $Mask_2$ in Fig. 1 is identical with $Mask_1$, and $Mask_3$ is defined as

$$Mask_3 = \mathbb{1}^{w \times h} - Mask_2 \tag{2}$$

where w and h are the width and the height of the input LDR image respectively. It can be seen that $Mask_3$ extracts the unsaturated pixels from the original LDR image that are finally combined with LiTMNet output.

3.3. Proposed LiTMNet

As shown in Fig. 2, the proposed LiTMNet is an encoder-decoder architecture with skip-connections which pass the feature output at each level of the encoder to the corresponding level of the decoder. Different from HDRCNN, these skip-connections have no domain transformations which contain heavy element-wise operators. Also, an inverse camera curve and a log transformation are applied to the input and the output of LiTMNet at the training stage respectively. While the inverse camera curve is generally unknown, we implement a gamma function $f(x) = x^2$ as its rough approximation to linearize the camera curve. We selected this approximate function based on the observation of the five representative camera response functions generated by k-means clustering from a diverse dataset of real-world camera response functions [30] as detailed in [9]. We observed that these representative curves are in the form of gamma function $x^{-\alpha}$. Thus, the inverse camera function is x^{α}. To determine the value of α, we make it a learnable parameter with the initial value of 2.2. Then, through training the LiTMNet where α is optimized as well, it turns out

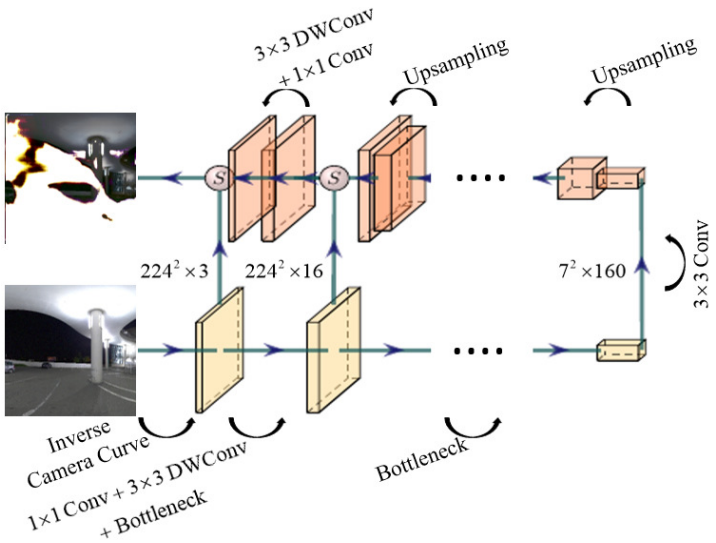


Figure 2: Architecture of LiTMNet. *S* denotes the skip layer.

that the learned optimal value is around 1.9. We thus simply set it to the integer 2 for efficiency purpose.

3.3.1. Encoder

The encoder is based on MobileNetV2 [10] with modifications for better performance. By applying a depthwise convolutional layer with a stride of 1 at the beginning of the encoder, our network produces feature representation at full resolution. By contrast, MobileNetV2 downsamples the input immediately as a convolution with a stride of 2 is used at its entrance. Then, we remove the batch normalization layers in the original bottleneck as it could cause degradation of the reconstruction quality as observed in the super resolution literature [20]. Moreover, our task does not need mid-level feature representation as rich as it is provided in MobileNetV2, and we thus balance the depth of bottleneck sequence across spatial resolutions.

The linear bottleneck is important for MobileNetV2 as it helps to maintain the non-linearity of the network while not causing the lost of much information. However, the linear bottleneck conflicts with the domain transformation skip-connection (DTSC), a crucial component of the HDRCNN. This is because the linear bottleneck produces feature maps consisting of values in the range of $\mathbb{R}$. Note that the inverse camera curve in the DTSC contains a square operation. Thus using the linear bottleneck and DTSC simultaneously would inappropriately change the feature maps as the square operation maps them from $\mathbb{R}$ to $\mathbb{R}_{\geq 0}$.

To tackle this problem, we decompose the domain transform of DTSC into an inverse camera curve and a logarithmic operation, and only applies them to the input and the output of LiTMNet respectively during training. We just ap-

Table 1: The details of the encoder

Input	Operator	Output
$h_i \times w_i \times c_i$	1×1 Conv2d, ReLU	$h_i \times w_i \times (t \times c_i)$
$h_i \times w_i \times (t \times c_i)$	3×3 DWConv s=s, ReLU	$\frac{h_i}{s} \times \frac{w_i}{s} \times (t \times c_i)$
$\frac{h_i}{s} \times \frac{w_i}{s} \times (t \times c_i)$	Linear 1×1 Conv2d	$\frac{h_i}{s} \times \frac{w_i}{s} \times d_i$

(a) The bottleneck without batch normalization. Instead, it uses ReLU and transforms from $h_i \times w_i \times c_i$ to $\frac{h_i}{s} \times \frac{w_i}{s} \times d_i$, with stride s, and expansion factor t. DWConv is depthwise convolution.

Input	Operator	t	c_i	n	s
$224^2 \times 3$	Linear 1×1 Conv2d	-	16	1	1
$224^2 \times 16$	3×3 DWConv, ReLU	-	16	1	1
$224^2 \times 16$	Bottleneck	1	16	1	1
$224^2 \times 16$	Bottleneck	2	32	2	2
$112^2 \times 32$	Bottleneck	6	32	2	2
$56^2 \times 32$	Bottleneck	6	48	3	2
$28^2 \times 48$	Bottleneck	6	96	3	2
$14^2 \times 96$	Bottleneck	6	160	2	2
$7^2 \times 160$	-	-	-	-	-

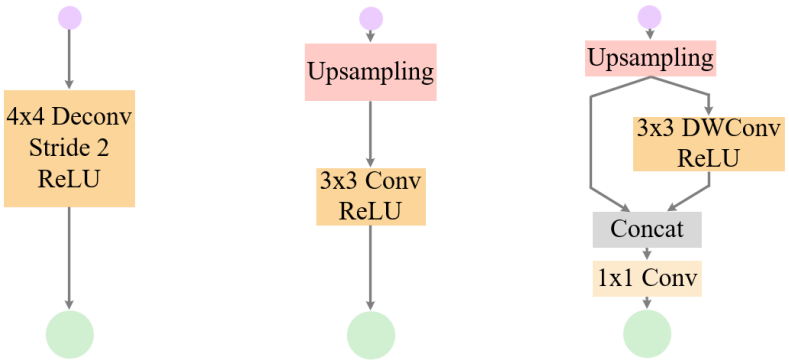
(b) Each row except the final one describes a sequence of one or more identical layers, repeated n times. c_i is the number of the output channels of all layers in the same sequence. In each sequence, only the first layer uses a stride s and all the other layers have a stride of 1. The expansion factor t is only used in the bottleneck as described in (a).

plied skip-connections to the bottlenecks in the LiTMNet. And in the deployment mode, the logarithmic operation is removed as explained in Section 3.2. Experimental evidence in Section 4.3 suggests that such a tactic improves both the performance and the efficiency. The details of the encoder are summarized in Table 1.

3.3.2. Decoder

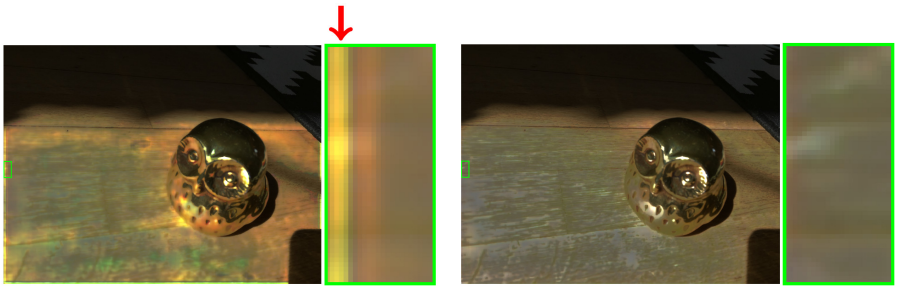
The decoder mainly consists of a sequence of upsampling blocks and skip layers. A skip layer makes a linear combination of its inputs. I.e., given two $h_i \times w_i \times c_i$ features, the skip layer concatenates them along the channel axis and applies a 1×1 convolution to produce an $h_i \times w_i \times c_i$ output.

Unlike HDRCNN [5] using deconvolution in the decoder, we propose a new upsampling block as illustrated in Fig. 3. Our upsampling block is also significantly different from the widely used vanilla resize-convolution block [31]. We replace the standard convolution with a lightweight depthwise separable convolution (DSC) to save computational cost. A standard convolution takes as input an $h_i \times w_i \times c_i$ feature map and applies a convolutional kernel $K \in \mathbb{R}^{k \times k \times c_i \times d_i}$ to output an $h_i \times w_i \times d_i$ feature. Such a process has the compu-



(a) (b) (c)

Figure 3: (a) The deconvolution block used in HDRCNN [5]; (b) The vanilla resize-convolution block; (c) The proposed upsampling block.



(a) Using convolution (b) Using DSC

Figure 4: Closeups of the reconstructions using the standard convolution (a) and the DSC (b). The red arrow points to the position of the black boundary effect.

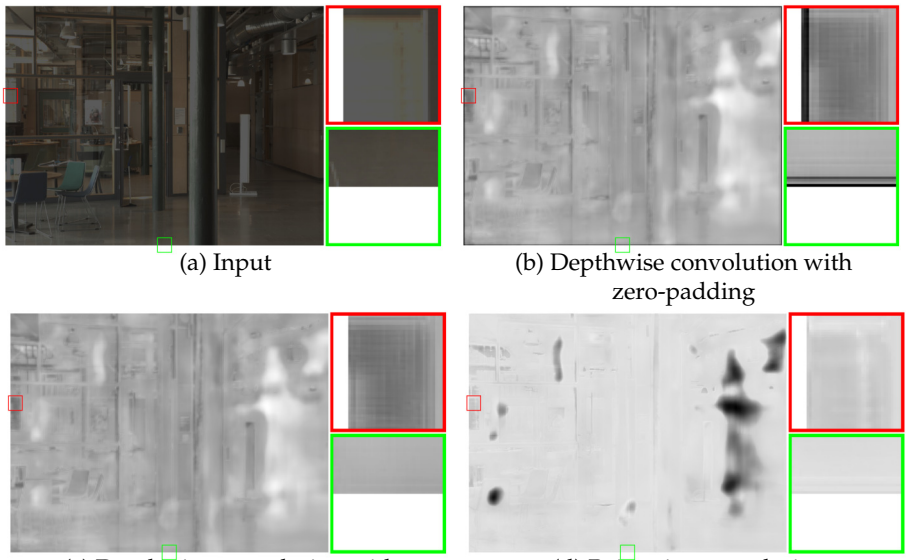
tational cost of $h_i \cdot w_i \cdot c_i \cdot d_i \cdot k^2$. In comparison, the lightweight DSC used in the proposed upsampling block has a computational cost of:

$$h_i \cdot w_i \cdot c_i \cdot (k^2 + 2d_i) \tag{3}$$

which is the sum of the depthwise and pointwise convolution. In the implementation, we use $k = 3$ and thus the computational cost of our upsampling block is about 4 times less than that of the vanilla resize-convolution block, albeit subject to the output channel d_i.

Model quantization and pruning can also be employed to further accelerate the LiTMNet. This is because they are essentially post-processing methods for model acceleration and usually take effect after the main training stage although fine-tuning might be required. Thus, they do not conflict with DSC which achieves the acceleration by changing the network architecture before the main training stage.

Moreover, we experimentally found that using depthwise separable convo-



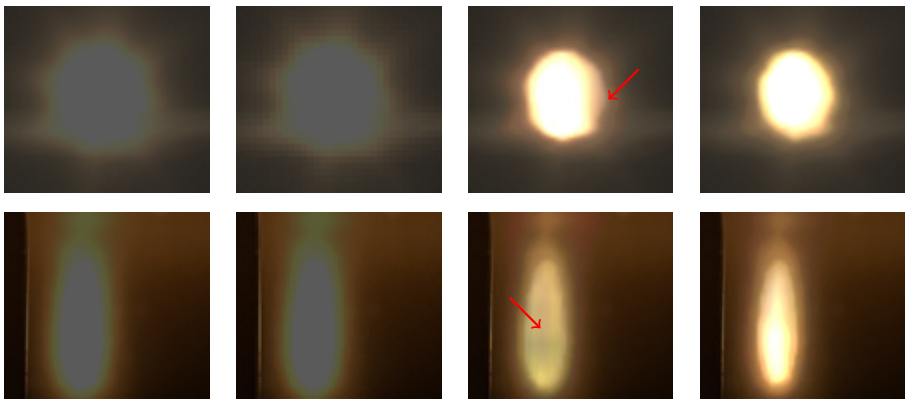
(a) Input (b) Depthwise convolution with zero-padding

(c) Depthwise convolution with mirror-padding (d) Pointwise convolution

Figure 5: Intermediate feature maps. The feature map generated by the depthwise convolution with zero-padding (b) contains a black boundary, while using mirror-padding (c) avoids it. The pointwise convolution (d) takes (b) as input and avoids the black boundary as well.

lution can help to alleviate boundary artifacts as shown in Fig. 4. This is probably because the included pointwise convolution learns to combine its input features into artifact-free ones. To further investigate the issue, we visualize intermediate output features of depthwise and pointwise convolution in the last upsampling block respectively, and select a typical one as shown in Fig. 5. Note that padding zeros to a feature map actually adds irrelevant information to it. It is not surprised that applying a well trained convolutional filter to these areas will lead to unwanted results, such as the feature maps with a black boundary as shown Fig. 5(b). By contrast, although the pointwise convolution takes these features as input as well, it learns combinations to produce some features without the black boundary as shown in Fig. 5(d). Such features are similar to those computed by the depthwise convolution with mirror-padding as shown in Fig. 5(c). This implies that the pointwise convolution effectively mitigates the black boundary effect by producing well structured features. It can be consequently seen that due to the pointwise convolutions of preceded blocks, the black boundary shown in Fig. 5(b) is relatively small and does not lead to serious artifacts as shown in Fig. 4.

Only replacing the standard convolution with the depthwise separable convolution is not sufficient according to Fig. 6(c) which exhibits unnatural reconstructions. To address this problem, we concatenate the upsampled features and the features computed by the depthwise convolution along the channel axis before passed to pointwise convolution as illustrated in Fig. 3 (c). Differ-



(a) Input (b) Upsampled input (c) Without SC (d) With SC

Figure 6: Closeups of the reconstructions without and with the shortcut connection (SC). Upsampled inputs are created by using downsampling followed by upsampling.

ent from a residual connection which typically adds the output of a layer to its input, we take advantage of the pointwise convolution to learn a proper combination of these features rather than simply performing addition. Moreover, another benefit of doing so is that the upsampling process is less likely to cause boundary artifacts as shown in Fig. 6(b). The shortcut connection provides upsampled features for pointwise convolution, and thus the network is able to learn how to make use of these features. This strategy improves the reconstruction quality as shown in Fig. 6(d). The details of the whole decoder are summarized in Table 2.

3.3.3. Loss Function

To train the network, we define a hybrid loss combining the ℓ_1-norm loss $\mathcal{L}_1$ and the perceptual loss $\mathcal{L}_p$. The hybrid loss can be written as

$$\mathcal{L}(\hat{\boldsymbol{y}}, \boldsymbol{H}) = \mathcal{L}_1(\hat{\boldsymbol{y}}, \boldsymbol{H}) + \beta \mathcal{L}_p(\hat{\boldsymbol{y}}, \boldsymbol{H}) \tag{4}$$

where $\hat{\boldsymbol{y}}$ and $\boldsymbol{H}$ are the network prediction output by the LiTMNet and the ground truth HDR image, respectively. β is used to balance the losses and set to 1 in our work. Specifically, the ℓ_1-norm loss can be formulated as

$$\mathcal{L}_1(\hat{\boldsymbol{y}}, \boldsymbol{H}) = \frac{1}{N} \sum_{i,c} \left| \hat{y}^{\alpha}_{i,c} - H^{\alpha}_{i,c} \right| \tag{5}$$

where $\hat{y}^{\alpha}_{i,c} = \alpha_i(\log(\hat{y}_{i,c} + \epsilon))$, $H^{\alpha}_{i,c} = \alpha_i(\log(H_{i,c} + \epsilon))$ and N is the number of pixels. Note that the log operation effectively avoids the large luminance dominance for the computation of $\mathcal{L}_1$.

Perceptual loss makes use of a *loss network* ϕ to measure high-level perceptual and semantic differences between images [32]. In our implementation,

Table 2: Details of the decoder

Input	Operator	Output
$h_i \times w_i \times c_i$	Upsampling	$2h_i \times 2w_i \times c_i$
$2h_i \times 2w_i \times c_i$	3×3 DWConv s=1, ReLU	$2h_i \times 2w_i \times c_i$
$2 \times 2h_i \times 2w_i \times c_i$	Concatenate	$2h_i \times 2w_i \times 2c_i$
$2h_i \times 2w_i \times 2c_i$	Linear 1×1 Conv2d	$2h_i \times 2w_i \times d_i$

(a) The upsampling block. DWConv denotes depthwise convolution.

Input	Operator	Output
$2 \times h_i \times w_i \times c_i$	Concatenate	$h_i \times w_i \times 2c_i$
$h_i \times w_i \times 2c_i$	1×1 Conv2d, ReLU	$h_i \times w_i \times c_i$

(b) The skip layer.

Input	Operator	c_i
$7^2 \times 160$	3×3 Conv2d s=1, ReLU	160
$7^2 \times 160$	Upsampling block	96
$2 \times 14^2 \times 96$	Skip layer	96
$14^2 \times 96$	Upsampling block	48
$2 \times 28^2 \times 48$	Skip layer	48
$28^2 \times 48$	Upsampling block	32
$2 \times 56^2 \times 32$	Skip layer	32
$56^2 \times 32$	Upsampling block	32
$2 \times 112^2 \times 32$	Skip layer	32
$112^2 \times 32$	Upsampling block	16
$2 \times 224^2 \times 16$	Skip layer	16
$224^2 \times 16$	3×3 DWConv s=1, ReLU	16
$224^2 \times 16$	Linear 1×1 Conv2d	3
$2 \times 224^2 \times 3$	Skip layer	3
$224^2 \times 3$	-	-

(c) c_i denotes the number of the output channels. The details of the upsampling block and the skip layer are described in (a) and (b), respectively.

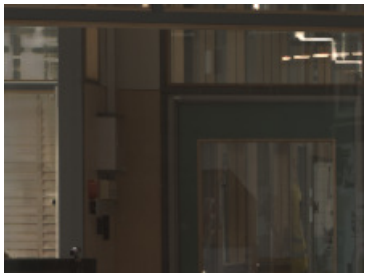
ϕ is the VGG19 network [33] pretrained on the ImageNet dataset [34]. The perceptual loss can be formulated as

$$\mathcal{L}_p(\hat{\boldsymbol{y}}, \boldsymbol{H}) = \frac{1}{N_l} \sum_l \sum_{i,c} |\phi_l(\hat{y}^\alpha)_{i,c} - \phi_l(H^\alpha)_{i,c}|^2 \tag{6}$$

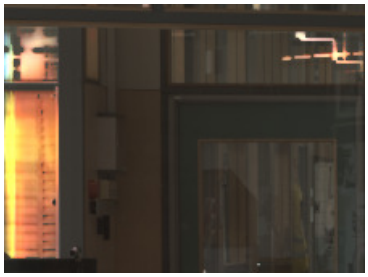
where ϕ_l indicates the feature maps produced by the first convolution (after activation) before the l-th maxpooling layer within the VGG19 network. N_l is the total number of pixels of ϕ_l. We use $l = \{1, 2, 3, 4, 5\}$ in the experiments. $\hat{y}^\alpha$ represents an image and $\hat{y}^\alpha_{i,c}$ is a pixel with spatial index i and channel c of that image.

(a) LDR input image

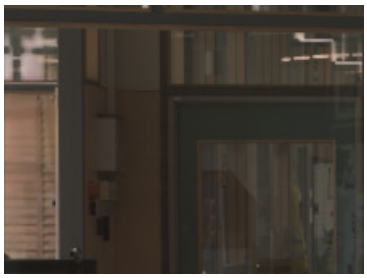
(b) Ground truth



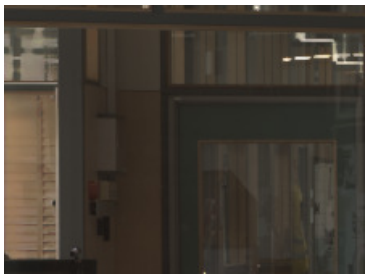
(c) Deconvolution
Total reconstruction time 5.0s



(d) VRC with zero padding
Total reconstruction time 4.0s



(e) VRC with mirror padding
Total reconstruction time 4.7s



(f) Proposed upsampling block
Total reconstruction time 2.7s

Figure 7: Comparisons made on an android device between the deconvolution, the vanilla resize-convolution block (VRC) and the proposed upsampling block.

3.3.4. Network Efficiency

Without the proposed upsampling block, it is still possible to build a network using deconvolution or vanilla resize-convolution block. After training such a network, a measurement of inference time was conducted on an android device equipped with a Snapdragon 835 processor. The resolution of the tested images is 1024×768. As shown in Fig. 7, the network with deconvolutions is the slowest. Using vanilla resize-convolution blocks with zero padding takes about 4 seconds to reconstruct an image, but we experimentally found that padding zeros to the blocks would lead to not only boundary artifacts but also color shift. By replacing zero padding with mirror padding, the artifacts and

color shift disappear while using mirror padding costs more inference time. One possible explanation is that mirror padding need to read the feature map first, and then pad the corresponding values back. As a result, the memory consumption in this process could be higher than zero padding, which just pads zeros to the feature map directly. With the proposed upsampling block, not only the issue of the boundary artifacts is addressed, but also the efficiency is significantly improved.

Overall, the proposed LiTMNet is more efficient than the state-of-the-art method HDRCNN [5] for two reasons. First, we replace the VGG encoder of the HDRCNN with a modified MobileNetV2 encoder. According to the MobileNetV2 paper, it performs particularly efficiently on CPUs (e.g. the mobile devices) due mainly to the specific design of the depthwise separable convolution (DSC). To generate an $h_i \times w_i \times d_j$ tensor from an input $h_i \times w_i \times d_i$ tensor with a kernel $K \in R^{k \times k \times d_i \times d_j}$, the standard convolution requires the computational cost of $h_i \times w_i \times d_i \times d_j \times k \times k$. By contrast, the DSC just costs $h_i \times w_i \times d_i \times (d_j + k \times k)$. However, replacing the original encoder of HDRCNN with MobileNetV2 is not straightforward as we discussed in the introduction and our work addresses the issues caused by this replacement. Second, we propose a new upsampling block and demonstrate that it is efficient on an android device.

4. Experimental Results

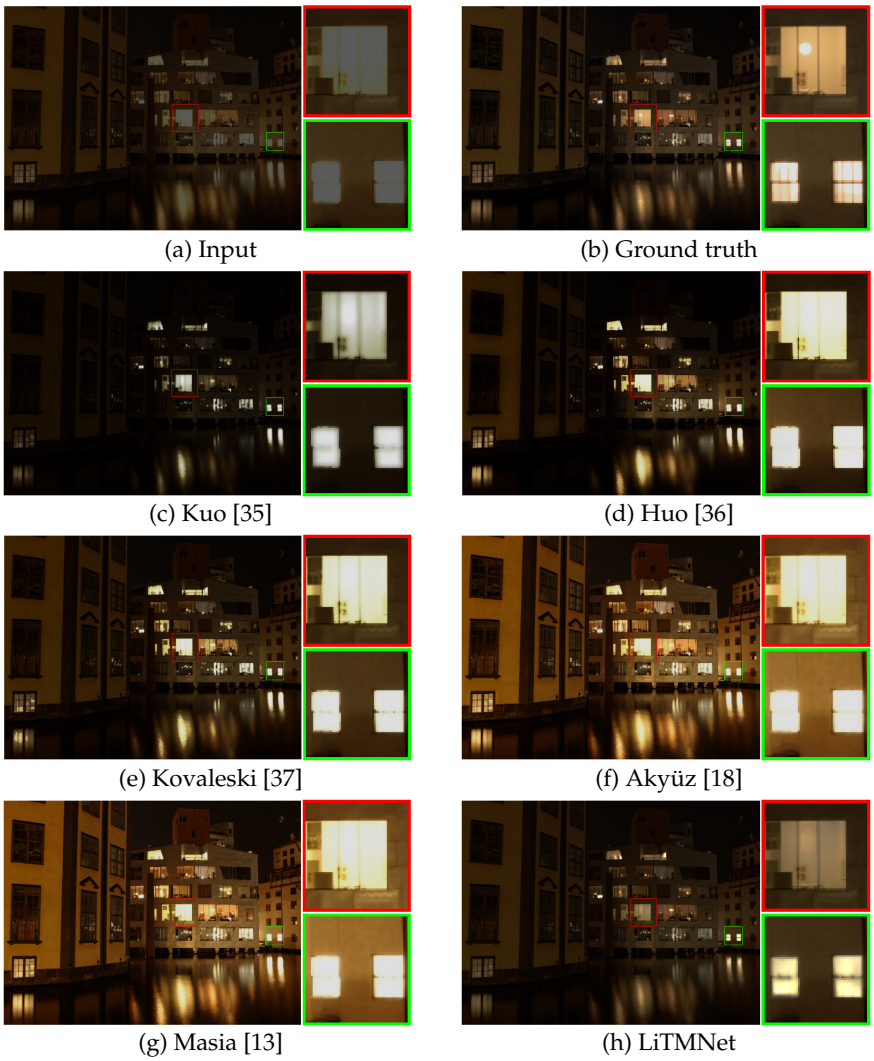
In this section, we first elaborate the experiment settings, and then present both qualitative and quantitative comparisons to evaluate the reconstruction quality as well as the efficiency of the proposed method. Finally, we perform an ablation study to investigate the effectiveness of LiTMNet and discuss a failure case. Additional visual examples including implementations through an android application on a mobile device can be found in the supplementary video.

4.1. Experimental Settings

4.1.1. HDR Dataset

Since LiTMNet learns a mapping function from some saturated pixels in the input LDR image to its corresponding pixels in the HDR image, the training data are HDR–LDR image pairs.

The HDR images are required to cover the dynamic range of various real-world scenarios and be taken by different imaging devices. We collect a total of 3500 HDR images in the public domain from the Internet, which cover representative scenarios including indoors, outdoors, night-time and daytime, etc. We then use a virtual camera provided in [5] to capture the LDR patches. The patches are created by cropping an HDR image with random sizes at random positions, followed by a camera curve. The patches are then resized to 320×320 pixels and the training dataset is further augmented via random flipping. Finally, we produced a set of 169K augmented LDR patches with their corresponding HDR references used as the ground truth data for training.



(a) Input (b) Ground truth

(c) Kuo [35] (d) Huo [36]

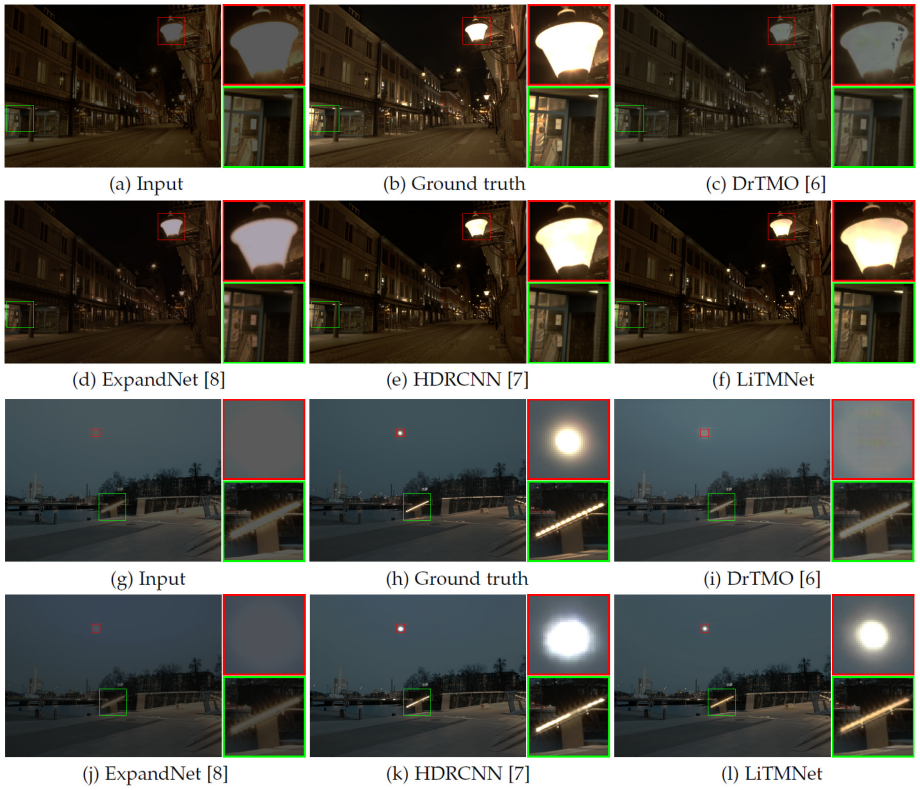
(e) Kovaleski [37] (f) Akyüz [18]

(g) Masia [13] (h) LiTMNet

Figure 8: Comparisons with five traditional iTMOs.

4.1.2. Training

We implement the network with TensorLayer [38] and initialize different components of the network using different strategies. For the skip layers, we initialize their weights by following the strategy proposed in [5]. For all the other components of the network, since there is no available pre-trained model, we perform the truncated normal initialization. After the initialization, we first train the network on a simulated dataset generated using the simulation method detailed in [5]. In this simulated dataset, the ground truth data are created by increasing the exposure of an LDR image with no saturated regions. After the training based on the simulated dataset, we further train the



(a) Input (b) Ground truth (c) DrTMO [6]

(d) ExpandNet [8] (e) HDRCNN [7] (f) LiTMNet

(g) Input (h) Ground truth (i) DrTMO [6]

(j) ExpandNet [8] (k) HDRCNN [7] (l) LiTMNet

Figure 9: Comparisons with CNN-based methods including DrTMO [9], ExpandNet [6] and HDR-CNN [5]. The inputs are created from the testing set of [5].

network using the aforementioned HDR dataset. We use ADAM optimizer with learning rate 2×10^{-5} and train the network for 1 million steps on each dataset, respectively.

4.2. Comparisons with iTMOs

We perform visual and quantitative comparisons between LiTMNet and ten representative iTMOs, including Akyüz et al. [18], Huo et al. [36], Kuo et al. [35], Masia et al. [13], Kovaleski and Oliveira [37], DrTMO [9], ExpandNet [6], HDRCNN [5], FHDR [27] and SingleHDR [28] where the last five are based on CNNs.

4.2.1. Visual Comparison

Fig. 8 compares the LiTMNet with five traditional iTMOs which do not use CNNs. These methods can expand the dynamic range of an LDR image to create an HDR sense when viewed on an HDR display. However, they fail to recover the lost information. By contrast, the LiTMNet learns a complex nonlinear function to reconstruct the lost details caused by sensor saturation,

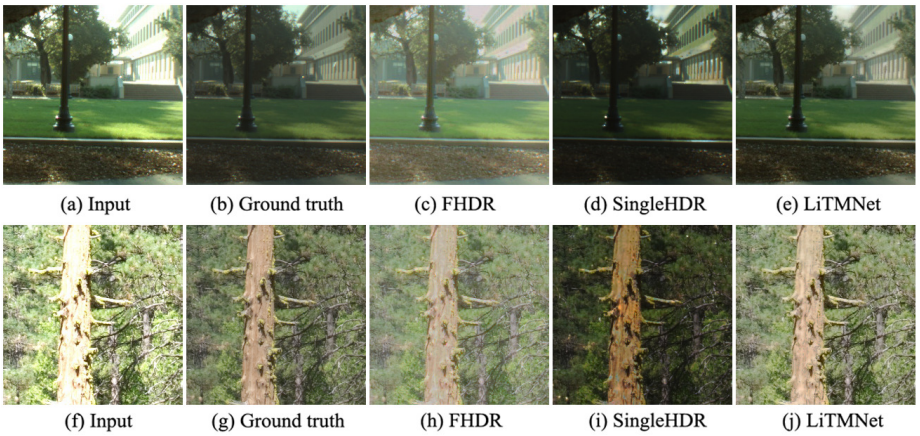


Figure 10: Comparisons with the latest CNN-based methods including FHDR [27] and SingleHDR [28]. The input images are from the FHDR testing set.

based on the information around the saturated regions and the context of the image.

Fig. 9 shows comparisons between LiTMNet and three CNN-based iTMOs. The LDR inputs are created from the testing set provided by [5]. It can be seen that in both scenes, the LiTMNet reconstructed the missing details in the saturated regions (shown in red and green boxes) that looks highly consistent with the ground truth. The HDRCNN produced generally satisfying results while the moon in Fig. 9(k) did not look natural. ExpandNet and DrTMO are less reliable when handling over-exposed pixels. For example, ExpandNet suffers from color shift and DrTMO produced undesirable artifacts in the regions highlighted with red and green boxes. Fig. 10 shows visual comparisons between the LiTMNet and two latest CNN-based iTMOs, FHDR [27] and SingleHDR [28]. It can be seen that our LiTMNet produced images globally more consistent with the ground truth while exhibiting good local details.

To further demonstrate the robustness of the LiTMNet, we evaluate it on the images taken by mobile phone cameras. Fig. 11 shows the reconstructions from some JPEG photos taken by the XiaoMI 6 and the Motorola Nexus 6 cameras. The photos taken by the Nexus 6 camera are publicly available in the hdrplus dataset [40]. The LiTMNet and the HDRCNN both generalize well to such photos. Although their reconstructions sometimes vary (e.g. the street light in the leftmost column), the results are generally reasonable and in line with image context. For ExpandNet and DrTMO, the shortcomings still exist in the images captured by mobile devices and very few details were successfully reconstructed.

We further compare LiTMNet and HDRCNN in Fig. 12. It can be seen that LiTMNet produced smooth and natural reconstructions at different exposure levels while HDRCNN suffers from blocking artifacts. Presumably, this is because HDRCNN uses deconvolutions in the decoder. The deconvolution has

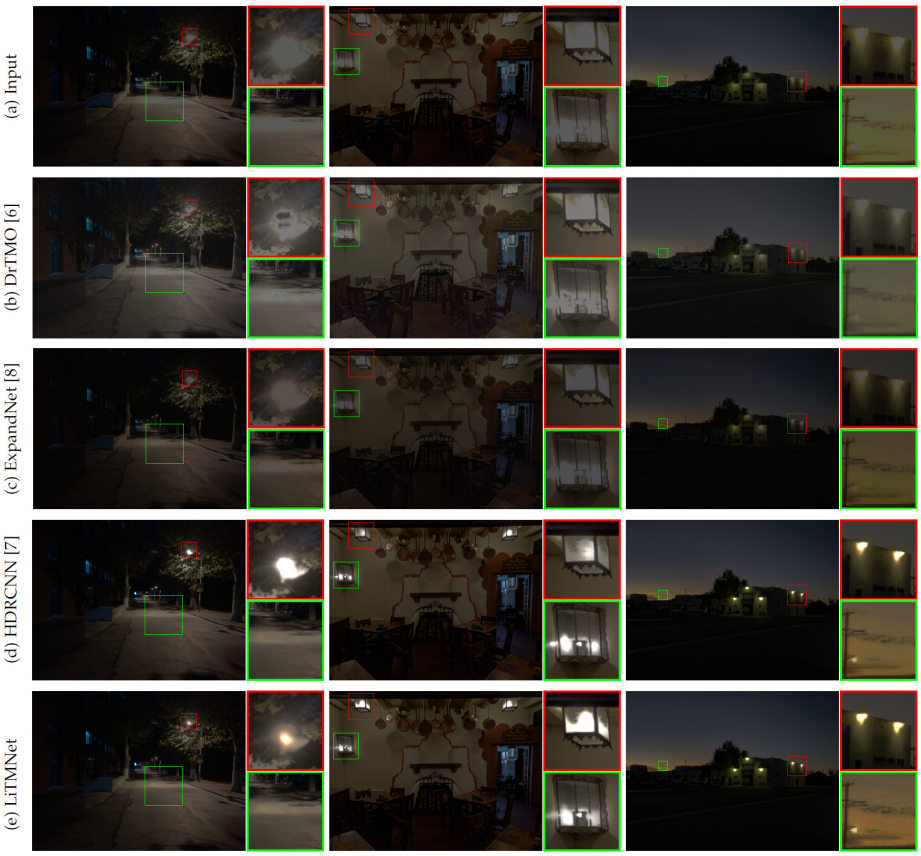


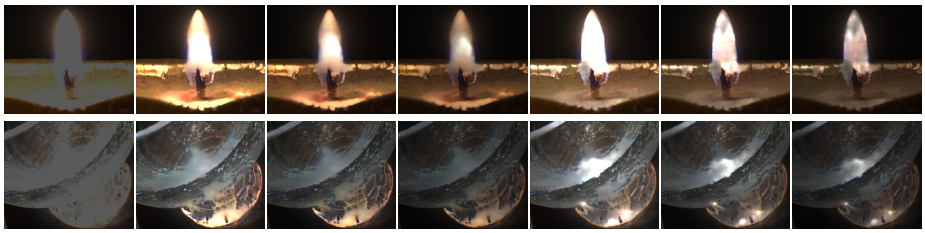
Figure 11: Comparisons of the HDR reconstructions produced by different CNN-based methods. The input images were taken by mobile phone cameras.

overlaps when estimating its output, and thus fails to learn a smooth output.

It is noteworthy that whether the input images are compressed or not could have an effect on this task. The HDRCNN thus contains two sets of learned weights respectively. Here we select the set of learned weights corresponding to the compressed images for comparison as this set is more robust for a variety of images. And the LiTMNet is also trained on the compressed data.

4.2.2. Quantitative Comparisons

We comparatively evaluate our method using three testing datasets including HDRCNN [5], FHDR [27] and HDR-Real [28] in terms of Peak Signal to Noise Ratio (PSNR), Multi-Scale Structural Similarity (MS-SSIM), HDR-VDP-2.2 [41], and run time for images with 1024×768 and 512×512 (required by [9]) pixel resolutions on a mobile device and a desktop computer respectively. For fair comparisons, we always fine-tune a competing network with the particular



Input Image | L=-3 | L=-4 | L=-5 | L=-3 | L=-4 | L=-5

Figure 12: Comparisons between LiTMNet and HDRCNN [5] at three different exposure levels. The images are cropped for better showing details. From left to right, the first column show the input images provided in HDRCNN. The second to the fourth columns show the reconstructions using LiTMNet at different exposure levels. The fifth to the seventh columns show the reconstructions produced by HDRCNN.

Table 3: Average PSNR, MS-SSIM and HDR-VDP-2.2 scores on HDRCNN [5], FHDR [27] and SingleHDR [28] datasets where the top three scores are colored in red, green and blue respectively. * and † correspond to the run time on a desktop computer and a mobile phone respectively. A perceptual uniformity encoding [39] is applied to the prediction and the reference for the PSNR and MS-SSIM metrics.

	HDRCNN Dataset [5]			FHDR Dataset [27]			HDR-Real Dataset [28]				
Method	PSNR	MS-SSIM	HDR-VDP-2.2	PSNR	MS-SSIM	HDR-VDP-2.2	PSNR	MS-SSIM	HDR-VDP-2.2	Run Time*(s)	Run Time†(s)
Akyüz [18]	16.3095	0.657	50.1871	17.8851	0.7330	52.7138	15.9309	0.4935	46.3400	-	-
Huo [36]	17.1657	0.6081	48.1339	13.4138	0.3452	46.5124	18.0752	0.4993	44.9631	-	-
Kovaleski [37]	21.4111	0.7096	51.2502	18.4549	0.6480	49.8993	17.3074	0.4949	46.8521	-	-
Kuo [35]	18.5208	0.4961	50.0052	16.2404	0.5400	51.1630	18.2212	0.4730	47.2662	-	-
Masia [13]	16.3955	0.6674	50.2791	16.3955	0.6674	50.2791	15.6464	0.5162	46.6087	-	-
ExpandNet [6]	21.3691	0.7413	53.5859	19.5439	0.7450	54.9577	15.8136	0.4022	45.7593	8.38	29.84
DrTMO [9]	19.3449	0.6892	51.8473	17.8003	0.5502	47.2141	16.2973	0.5272	47.5734	20.45	-
HDRCNN [5]	23.1966	0.8006	54.9102	23.6058	0.8430	57.1519	19.4555	0.5559	48.3908	4.64	104.36
FHDR [27]	19.2606	0.7119	52.3717	23.0218	0.8293	56.7724	16.5605	0.4988	47.2921	38.32	-
SingleHDR[28]	23.3986	0.7541	54.8830	23.4792	0.8343	57.7513	18.3068	0.5511	48.1131	11.32	-
LiTMNet	22.9296	0.7867	54.4645	23.1422	0.8252	56.9444	19.5510	0.5598	48.2623	0.43	2.72

training dataset when available.

For the testing dataset of HDRCNN [5], the input data are single exposed LDR images generated from the 96 HDR images contained in the HDRCNN testing set. For a fair comparison, instead of creating LDR images using the virtual camera provided in HDRCNN, we use five representative camera response functions, selected using k-means clustering, from a diverse database of real-world camera response functions [30] as detailed in [9]. In order to use PSNR and MS-SSIM for comparing the reconstructed HDR images, a perceptual uniformity encoding [39] is applied to the prediction and the reference. Experiments are carried out on both a desktop computer with a quad-core Intel i7-4700MQ CPU and an android phone equipped with a Snapdragon 835 CPU. We thus recorded the run time of varying methods on different devices. In order to run the trained network on a mobile phone, we convert it to TensorFlow Lite format (TF-Lite) so that the well-developed TF-Lite can be used.

Table 4: Configurations of different networks. DT skip-connection denotes that the connection includes a domain transformation.

	A	B	C	D	LiTMNet
Original MobileNetV2	✓	✓	✓		
Modified encoder				✓	✓
Vanilla upsampling	✓	✓			
Proposed upsampling			✓	✓	✓
Zero padding	✓		✓	✓	✓
Mirror padding		✓			
DT skip-connection	✓	✓	✓	✓	
Skip-connection					✓

Since the reconstruction is of HDR content, it needs tone mapping before displayed on mobile phones. In our implementation, we employ the Reinhard tone mapping algorithm provided by OpenCV.

The results are summarized in Table 3. The run time of DrTMO, FHDR and SingleHDR on the mobile phone is unavailable since they are too heavy to run on such a device. Also, we only recorded the run time for the CNN-based methods as the traditional methods were implemented in MATLAB instead of Python. Although according to the scores, the performance of the LiTMNet is comparable with SingleHDR and slightly lower than the HDRCNN, it is about 38$\times$ faster than HDRCNN with the best performance when running on a mobile phone. This is because the LiTMNet has a computational cost of 19.4 GFLOPs (giga floating-point operations). In comparison, ExpandNet [6] and HDRCNN [5] have the computational costs of 322.5 and 1290.7 GFLOPs, respectively. And the LiTMNet surpasses the other three CNN-based methods, i.e. FHDR, ExpandNet and DrTMO, in respect of both reconstruction quality and speed. It is noteworthy that HDRCNN [5] performs well on the HDRCNN dataset as its hyperparameters are well adapted to the HDRCNN dataset. The FHDR dataset [27] contains a number of sample images from the HDRCNN dataset and thus the data distributions of the two datasets are relatively similar. Therefore, HDRCNN also performs well on the FHDR datasets. The HDR-Real dataset [28] is significantly different from the HDRCNN and the FHDR datasets and thus we observed the performance variations over the three datasets.

4.3. Ablation Study

In this section, we show how the components described in Section 3 affect the performance of our network via ablation study. First, we conduct a quantitative analysis for the variant of LiTMNet where the shortcut connection in the proposed upsampling block illustrated in Fig. 3 (c) is removed. Second, we produce another variant where the perceptual loss $\mathcal{L}_p$ defined in Eq. (6) is removed during the training. We also construct variant networks of the LiMNet as summarized in Table 4, which actually contains four ablation studies:

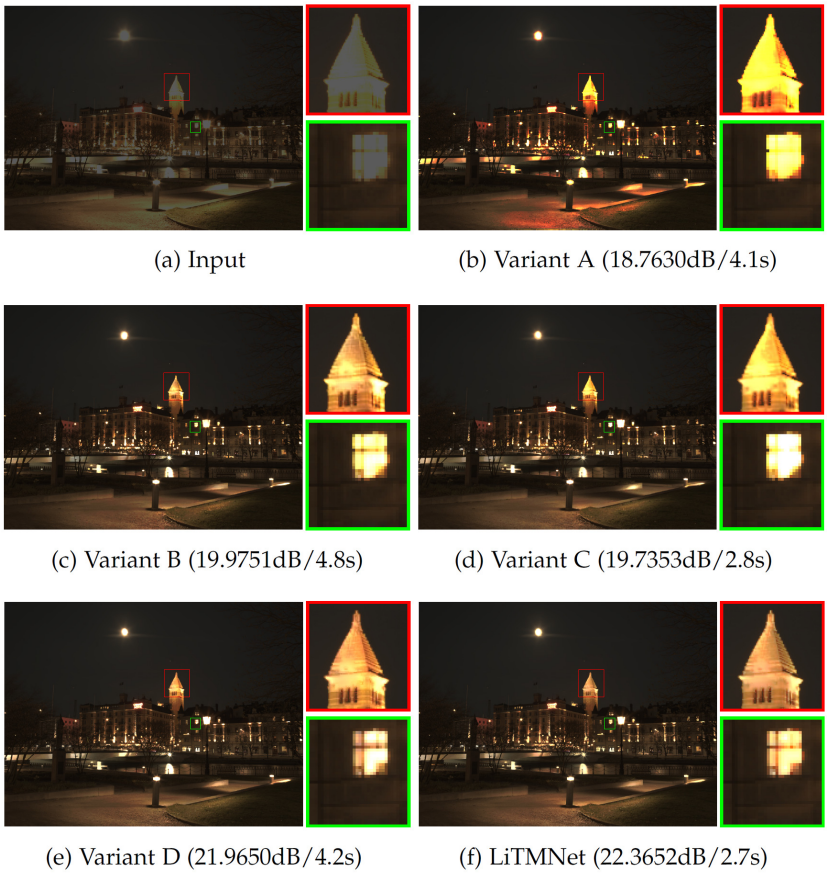
Table 5: Quantitative results of the variants of LiTMNet on the HDRCNN dataset [5] in terms of PSNR, MS-SSIM, HDR-VDP-2.2 and the run time on a mobile phone.

LiTMNet Variants	PSNR	MS-SSIM	HDR-VDP-2.2	Run Time (s)
w/o SC	21.8755	0.7715	53.8905	2.63
w/o $\mathcal{L}_p$	22.4726	0.7841	54.3353	2.72
Variant A	21.0147	0.7546	52.8891	4.13
Variant B	20.7112	0.7602	53.6376	4.81
Variant C	21.4289	0.7705	53.6857	2.83
Variant D	21.0851	0.7580	52.6475	4.22
LiTMNet	22.9296	0.7867	54.4645	2.72

1) Rows 1 and 2 deliver the ablation study between Variants C and D, which attempts to demonstrate that our modified encoder performs better than MobileNetV2; 2) Rows 3 and 4 deliver the ablation study between Variants A and C, which attempts to demonstrate the superiority of the proposed upsampling block over the vanilla upsampling block; 3) Rows 5 and 6 deliver the ablation study between Variants A and B, which attempts to demonstrate the benefit of using zero padding instead of mirror padding; 4) Rows 7 and 8 deliver the ablation study between Variants D and the full version of LiTMNet, which attempts to demonstrate that skip-connection outperforms DT skip-connection in our work. We provide both quantitative and qualitative results of the variants for comparisons.

The quantitative results of the ablation study are reported in Table 5 where "w/o SC" denotes the variant without the shortcut connection in the upsampling block illustrated in Fig. 3 (c) and "w/o $\mathcal{L}_p$" represents the variant without the perceptual loss $\mathcal{L}_p$ defined in Eq. (6). It can be seen the LiTMNet outperforms all variants on the HDRCNN dataset [5] in terms of PSNR, MS-SSIM and HDR-VDP-2.2, which demonstrates the superiority of the proposed LiTMNet and experimentally justifies the design choices of its architecture when alternative network components are available. The qualitative results are consistent with the quantitative ones. As shown in Fig. 13, compared to Variants A-D, the LiTMNet does not only produce the better results, but also is more efficient. By comparing Fig. 13 (c) with (d), we can see that the proposed upsampling block produces results of high quality and is very efficient as well. We also found that a significant improvement is achieved by replacing the MobileNetV2 with our redesigned encoder in Variant C. The highlighted regions (shown in the red boxes) and the lost details (shown in the green boxes) are better recovered in Variant D, although it is less efficient than Variant C. Therefore, Variant D can be regarded as a trade-off between performance and efficiency.

Unlike Variant D using a DTSC at each level of feature representation, LiTMNet applies an inverse camera curve to the input image. Such an alteration improves not only the efficiency, but also the reconstruction quality. It is probably because our redesigned encoder utilizes linear bottlenecks to extract multi-level features while the square operation included in the DTSC maps the learned features inappropriately from $\mathbb{R}$ to $\mathbb{R}_{\geq 0}$, leading to the loss of information con-



(a) Input (b) Variant A (18.7630dB/4.1s)

(c) Variant B (19.9751dB/4.8s) (d) Variant C (19.7353dB/2.8s)

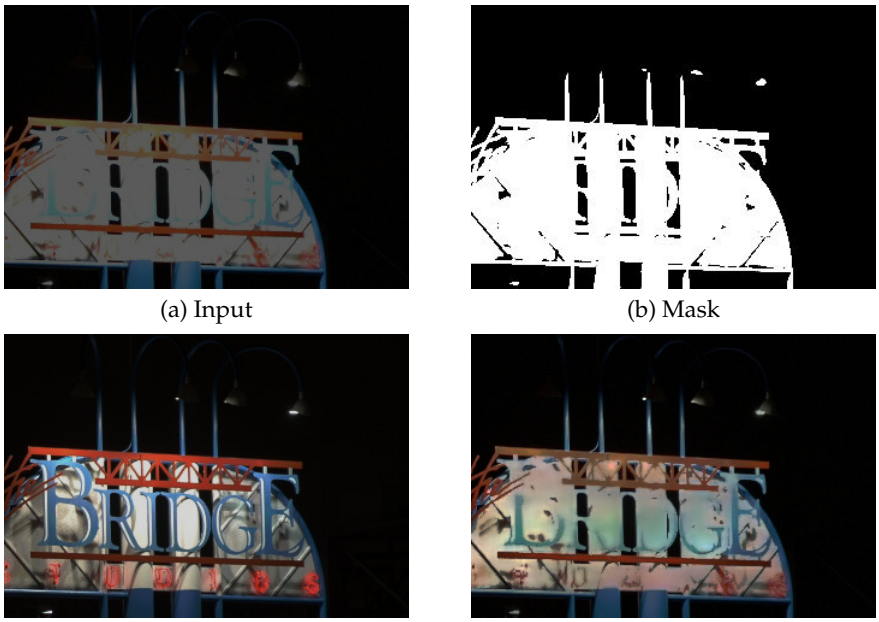
(e) Variant D (21.9650dB/4.2s) (f) LiTMNet (22.3652dB/2.7s)

Figure 13: Comparisons of different configurations with regard to PSNR and run time.

veyed by the features.

4.4. Failure Case

Guided by the HDR image, the LiTMNet learns to infer the missing information from an LDR input. If few details are available in a saturated region, the LiTMNet may fail to recover the structures properly. Fig. 14 shows a failure case where a large fraction of the pixels are saturated. Although the LiTMNet makes use of little information in the neighborhood to synthesize the missing details, the shape of the letters and the color of the pillars are not estimated correctly. This is a common limitation even shared by the state-of-the-art iTMOs, such as HDRCNN and SingleHDR.



(a) Input (b) Mask

(c) Ground truth (d) LiTMNet

Figure 14: Failure case. The mask is created using Eq. (1), which indicates over-exposure pixels. Images are cropped for showing better details.

5. Conclusions

Reconstructing an HDR image from a single exposed LDR image is a challenging task particularly when sufficient computing power is inaccessible. This paper presents a method for recovering saturated image regions to reconstruct HDR images via lightweight CNNs. The optimized encoder and the proposed upsampling block provide a dedicated solution for this task as they are computationally efficient. Moreover, extensive visual and quantitative comparisons show that the LiTMNet is both effective and efficient. To the best of our knowledge, the LiTMNet is the first deep iTMOs that can be applied to mobile devices for efficient end-to-end HDR reconstruction.

Although the LiTMNet is much faster than existing deep iTMOs, it is still not fast enough for 12 megapixel inputs. Thus future work will be to make the network even lighter by using a combination of guided upsamplings. However, such a scheme might cause the risk of degraded reconstructions and some schemes for mitigating this issue have to be investigated.

[1] P. E. Debevec, J. Malik, Recovering high dynamic range radiance maps from photographs, in: ACM SIGGRAPH 2008 classes, ACM, 2008, p. 31.

[2] T.-H. Oh, J.-Y. Lee, Y.-W. Tai, I. S. Kweon, Robust high dynamic range imaging by rank minimization, IEEE transactions on pattern analysis and machine intelligence 37 (2014) 1219–1232.

[3] T. Mertens, J. Kautz, F. Van Reeth, Exposure fusion: A simple and practical alternative to high dynamic range photography, in: Computer graphics forum, volume 28, Wiley Online Library, 2009, pp. 161–171.

[4] Z. Li, J. Zheng, Z. Zhu, S. Wu, Selectively detail-enhanced fusion of differently exposed images with moving objects, IEEE Transactions on Image Processing 23 (2014) 4372–4382.

[5] G. Eilertsen, J. Kronander, G. Denes, R. K. Mantiuk, J. Unger, Hdr image reconstruction from a single exposure using deep cnns, ACM Transactions on Graphics 36 (2017) 178.

[6] D. Marnerides, T. Bashford-Rogers, J. Hatchett, K. Debattista, Expandnet: A deep convolutional neural network for high dynamic range expansion from low dynamic range content, in: Computer Graphics Forum, volume 37, 2018, pp. 37–49.

[7] P. Chen, L. Li, X. Zhang, S. Wang, A. Tan, Blind quality index for tone-mapped images based on luminance partition, Pattern Recognition 89 (2019) 108–118.

[8] O. T. Tursun, A. O. Akyüz, A. Erdem, E. Erdem, The state of the art in hdr deghosting: A survey and evaluation, in: Computer Graphics Forum, volume 34, 2015, pp. 683–707.

[9] Y. Endo, Y. Kanamori, J. Mitani, Deep reverse tone mapping., ACM Trans. Graph. 36 (2017) 177–1.

[10] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, L.-C. Chen, Mobilenetv2: Inverted residuals and linear bottlenecks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 4510–4520.

[11] D. Zhang, J. Zhang, Q. Zhang, J. Han, S. Zhang, J. Han, Automatic pancreas segmentation based on lightweight dcnn modules and spatial prior propagation, Pattern Recognition 114 (2021).

[12] G. Li, M. Zhang, J. Li, F. Lv, G. Tong, Efficient densely connected convolutional neural networks, Pattern Recognition 109 (2021).

[13] B. Masia, S. Agustin, R. W. Fleming, O. Sorkine, D. Gutierrez, Evaluation of reverse tone mapping through varying exposure conditions, in: ACM transactions on graphics (TOG), volume 28, ACM, 2009, p. 160.

[14] W. Zhang, W.-K. Cham, Gradient-directed multiexposure composition, IEEE Transactions on Image Processing 21 (2011) 2318–2323.

[15] Z. Li, Z. Wei, C. Wen, J. Zheng, Detail-enhanced multi-scale exposure fusion, IEEE Transactions on Image Processing 26 (2017) 1243–1252.

[16] H. Landis, Production-ready global illumination, Siggraph course notes 16 (2002) 11.

[17] A. G. Rempel, M. Trentacoste, H. Seetzen, H. D. Young, W. Heidrich, L. Whitehead, G. Ward, Ldr2hdr: on-the-fly reverse tone mapping of legacy video and photographs, in: ACM transactions on graphics (TOG), volume 26, ACM, 2007, p. 39.

[18] A. O. Akyüz, R. Fleming, B. E. Riecke, E. Reinhard, H. H. Bülthoff, Do hdr displays support ldr content?: a psychophysical evaluation, in: ACM Transactions on Graphics (TOG), volume 26, ACM, 2007, p. 38.

[19] Y. Quan, Y. Chen, Y. Shao, H. Teng, Y. Xu, H. Ji, Image denoising using complex-valued deep cnn, Pattern Recognition 111 (2021).

[20] P. V. Arun, I. Herrmann, K. M. Budhiraju, A. Karnieli, Convolutional network architectures for super-resolution/sub-pixel mapping of drone-derived images, Pattern recognition 88 (2019) 431–446.

[21] M. He, D. Chen, J. Liao, P. V. Sander, L. Yuan, Deep exemplar-based colorization, ACM Transactions on Graphics (TOG) 37 (2018) 1–16.

[22] S. Pillai, R. Ambruş, A. Gaidon, Superdepth: Self-supervised, super-resolved monocular depth estimation, in: 2019 International Conference on Robotics and Automation, 2019, pp. 9250–9256.

[23] J. Zhang, J.-F. Lalonde, Learning high dynamic range from outdoor panoramas, in: Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 4519–4528.

[24] Q. Yan, D. Gong, Q. Shi, A. v. d. Hengel, C. Shen, I. Reid, Y. Zhang, Attention-guided network for ghost-free high dynamic range imaging, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 1751–1760.

[25] Y. Chen, G. Jiang, M. Yu, Y. Yang, Y.-S. Ho, Learning stereo high dynamic range imaging from a pair of cameras with different exposure parameters, IEEE Transactions on Computational Imaging (2020).

[26] Q. Yan, L. Zhang, Y. Liu, Y. Zhu, J. Sun, Q. Shi, Y. Zhang, Deep hdr imaging via a non-local network, IEEE Transactions on Image Processing 29 (2020) 4308–4322.

[27] Z. Khan, M. Khanna, S. Raman, Fhdr: Hdr image reconstruction from a single ldr image using feedback network, arXiv preprint arXiv:1912.11463 (2019).

[28] Y.-L. Liu, W.-S. Lai, Y.-S. Chen, Y.-L. Kao, M.-H. Yang, Y.-Y. Chuang, J.-B. Huang, Single-image hdr reconstruction by learning to reverse the camera pipeline, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 1651–1660.

[29] S. Wu, J. Xu, Y.-W. Tai, C.-K. Tang, Deep high dynamic range imaging with large foreground motions, in: Proceedings of the European Conference on Computer Vision (ECCV), 2018, pp. 117–132.

[30] M. D. Grossberg, S. K. Nayar, What is the space of camera response functions?, in: IEEE Conference on Computer Vision and Pattern Recognition, volume 2, 2003, pp. II–602.

[31] X. Wang, G. Oxholm, D. Zhang, Y.-F. Wang, Multimodal transfer: A hierarchical deep convolutional neural network for fast artistic style transfer, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 5239–5247.

[32] J. Johnson, A. Alahi, L. Fei-Fei, Perceptual losses for real-time style transfer and super-resolution, in: European conference on computer vision, Springer, 2016, pp. 694–711.

[33] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, arXiv preprint arXiv:1409.1556 (2014).

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

[35] P.-H. Kuo, C.-S. Tang, S.-Y. Chien, Content-adaptive inverse tone mapping, in: 2012 Visual Communications and Image Processing, IEEE, 2012, pp. 1–6.

[36] Y. Huo, F. Yang, L. Dong, V. Brost, Physiological inverse tone mapping based on retina response, The Visual Computer 30 (2014) 507–517.

[37] R. P. Kovaleski, M. M. Oliveira, High-quality reverse tone mapping for a wide range of exposures, in: 2014 27th SIBGRAPI Conference on Graphics, Patterns and Images, IEEE, 2014, pp. 49–56.

[38] H. Dong, A. Supratak, L. Mai, F. Liu, A. Oehmichen, S. Yu, Y. Guo, TensorLayer: A Versatile Library for Efficient Deep Learning Development, ACM Multimedia (2017). URL: `http://tensorlayer.org`.

[39] T. O. Aydın, R. Mantiuk, H.-P. Seidel, Extending quality metrics to full luminance range images, in: Human Vision and Electronic Imaging XIII, volume 6806, International Society for Optics and Photonics, 2008, p. 68060B.

[40] S. W. Hasinoff, D. Sharlet, R. Geiss, A. Adams, J. T. Barron, F. Kainz, J. Chen, M. Levoy, Burst photography for high dynamic range and low-light imaging on mobile cameras, ACM Transactions on Graphics 35 (2016).

[41] M. Narwaria, R. Mantiuk, M. P. Da Silva, P. Le Callet, Hdr-vdp-2.2: a calibrated method for objective quality prediction of high-dynamic range and standard images, Journal of Electronic Imaging 24 (2015) 010501.