STAR: A Structure-aware Lightweight Transformer for Real-time Image Enhancement

Zhaoyang Zhang[1,2] Yitong Jiang[2] Jun Jiang [2] Xiaogang Wang[1] Ping Luo[3] Jinwei Gu[2,4]
[1]The Chinese University of Hong Kong [2]SenseBrain Research [3]The Univesity of Hong Kong [4]Shanghai AI Laboratory
{zhaoyangzhang@link,xgwang@ee}.cuhk.edu.hk
{jiangyitong,jiangjun,gujinwei}@sensebrain.ai pluo@cs.hku.hk

Abstract

Image and video enhancement such as color constancy, low light enhancement, and tone mapping on smartphones is challenging, because high-quality images should be achieved efficiently with a limited resource budget. Unlike prior works that either used very deep CNNs or large Transformer models, we propose a structure-aware lightweight Transformer, termed STAR, for real-time image enhancement. STAR is formulated to capture long-range dependencies between image patches, which naturally and implicitly captures the structural relationships of different regions in an image. STAR is a general architecture that can be easily adapted to different image enhancement tasks. Extensive experiments show that STAR can effectively boost the quality and efficiency of many tasks such as illumination enhancement, auto white balance, and photo retouching, which are indispensable components for image processing on smartphones. For example, STAR reduces model complexity and improves image quality compared to the recent state-of-the-art [19] on the MIT-Adobe FiveK dataset [7] (i.e., $1.8dB$ *PSNR improvements with* 25% *parameters and* 13% *float operations.)*

1. Introduction

Recent years have witnessed significant progress on a variety of image and video enhancement tasks with learning-based methods, such as super-resolution, denoising, demosaicking, low light enhancement, color constancy (i.e., white balance), tone mapping. Yet, there are still two key challenges for deploying these methods on edge devices. First, these methods must process high-resolution images efficiently within a very limited computation budget, making trade-offs between model flexibility and computation efficiency. Second, they need to incorporate structural and global information of input images in order to achieve high-quality stable results, especially for tasks such as color constancy, low light enhancement, and tone mapping. Even for tasks with local support, such as denoising and demosaicking, structure-aware region-based processing can often produce better results [33].

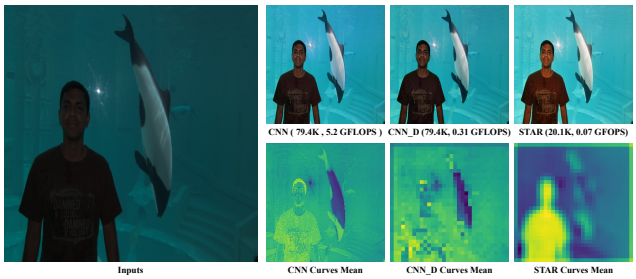

Figure 1. Comparison of curve estimation results between CNN and STAR for low light enhancement. "CNN" denotes the original backbone model used by [19] that predicts per-pixel curves. For fair comparison, we also train the same CNN model with downsampling (denoted as CNN-D), so that both STAR and CNN-D predict per-token curves. Both the enhanced results and model complexity (number of parameters, FLOPS) are shown. We also visualize the predicted curves by plotting the mean of α (See more details in Section 4.1).

To address these two challenges, prior work can be summarized into three categories. The first approach is to use stacked, very deep CNNs [49, 9, 17, 45, 19]. In order to maintain high-frequency details, the spatial resolution is often kept unchanged, and thus these methods have a large computational cost and memory footprint. The second approach is to estimate one set of global adjustment function [52, 35, 24, 31], but they lack the flexibility to handle the complexity of real-world scenarios (e.g., mixed illumination for white balance, HDR scenes for tone mapping). Finally, the third approach is to explicitly use a segmentation network to divide images into semantically-meaningful regions and process each region separately [33, 51]. However, this approach also has limitations such as requiring per-pixel annotated datasets.

In this paper, we proposed STAR (Structure-aware Transformer), a general lightweight backbone for various

real-time image post-processing tasks. STAR is formulated to capture long-range dependencies among image patches, which naturally and implicitly captures the structure relationships within an image. STAR is a general architecture that can be easily adapted for a variety of learning-based image enhancement tasks. Rather than stacked convolution layers, STAR is based on the Transformer module, which mainly consists of multi-head self-attention and full-connected layers only. Transformer [44] is widely used in natural language processing for its high training efficiency towards long-distance dependencies and tremendous model capacity. Models building upon Transformer achieved surprising performance that even surpasses human recognition ability in specific language tasks [11].

Specialized in image enhancement tasks, we design the STAR network, which can be free of stacked convolution and thus more efficient in extracting structural information. In STAR, patches of the image are tokenized into token embeddings, much like word embeddings in NLP. Rather than computing pixel-wise dependencies directly, STAR explicitly learns token-wise dependencies for image patches. Fig.1 shows an example. As shown in Fig.1, STAR delivers high efficiency to enhance an image. In addition to high efficiency (0.07 GFLOPS), as shown, STAR can also implicitly learn the semantic structure and thus deliver more semantic-meaningful results than CNNs [19]. Instead of having one module for general information, as suggested in [27, 48], we employ a specialized two-branch design naming long-short Range Transformer to ensure STAR can focus on capturing global contexts thus reducing computations.

We further validate STAR on several recent image enhancement methods, including illumination enhancement [19], white balance [2] and photo retouching with 3D lookup table [52]. Experimental results show that STAR can often effectively boost the performance of these tasks with significantly reduced model complexity, which offers great advantages for real-time processing on edge devices. For instance, on low light enhancement using DCE-Net [19] , compared to CNN backbone, STAR-based methods can achieve $1.8dB$ PSNR improvement, while only requiring 25% parameters and 13% float operations (FLOPS).

2. Related Work

2.1. Learning-based Image Enhancement in Camera Imaging Pipeline

Recently, with the success of deep convolutional neural networks (CNNs), many appealing learning-based image enhancement methods achieve promising results such as Camera ISP [34, 32, 38], demosaicking [41, 14], denoising [14, 18], white balancing [2, 1], HDR reconstruction [17, 39, 28, 29], exposure correction [19, 45] and color enhancement [45, 52]. Nevertheless, many of these methods rely on heavy computation and memory footprint, which may hinder their deployment on hardware-constrained devices, such as smartphones or other embedded systems. Some recent methods are proposed for efficient image enhancement on mobile platforms. The first category is to estimate a set of global adjustment function like [52, 35, 24, 31]. These methods mainly process the downsampled images and predict variables of a set of global manipulating functions. However, many works like [35, 24] show that applying global functions only is not able to provide adequate and flexible enhancement capability. Another category of image enhancement methods is formulated to be a semantic-aware prediction, which trains the CNN model to estimate mapping or transformation functions with semantic masks [33, 43]. Models in these methods are usually versatile because the predicted results are conditioned to semantic contents like other pixel-wise methods like [49, 9, 17, 45, 19]. But such methods have limitations such as requiring an extra segmentation subnetwork and per-pixel annotated datasets to predict masks.

2.2. Transformer Module

Transformer is proposed by [44] for machine translation, where the multi-head self-attention and feed-forward MLP are stacked to capture the long-range correlation between words. Transformer-based networks show high model capacity and have since become the state-of-the-art method in many NLP tasks [11, 6, 36]. The core of Transformer is to characterize the dependencies of two distant tokens by a multi-head self-attention mechanism. This property endows Transformer with the potential to capture latent correlation among large and complex data sources.

Transferring the pre-trained Transformer from NLP to vision tasks also becomes a hot issue. [46, 40] attempts to combine visual and linguistic representations to build comprehensive pre-trained Transformers for both vision and language data. Besides, relying on large-scale external data [42], recent work ViT [12] has implemented high competitive image ImageNet [10] classification results using networks mainly constructed by Transformer layers. Inspired by that, [8] further demonstrate that pre-training such a pure Transformer model on ImageNet could improve network performance on low-level vision tasks (super-resolution, denoising, deraining). However, a large-scale pre-trained model also requiring massive parameters (e.g. IPT [8] model has over 114 M parameters and 33G FLOPS, which is impractical to deploy on mobile platforms for image and video enhancement. In contrast, STAR is designed to be lightweight and can achieve real-time performance, which to our knowledge is the first lightweight Transformer for image enhancement.

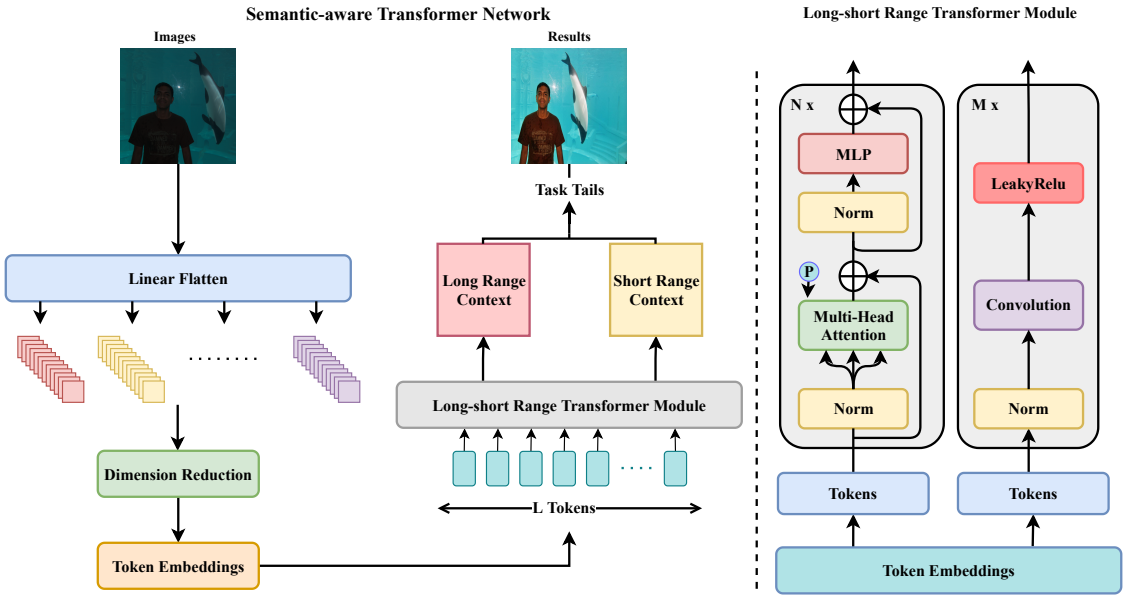

Figure 2. Network overview. Left: We flatten an image feature map into patches, then linearly embed them into tokens after dimension reduction. The generated tokens will be fed to a long-short range Transformer module. The resulting sequence of Transformer will be reshaped further used by downstream tasks by adding task tail. Right: Illustration of long-short range Transformer. We adopt a two branch design to reduce model complexity by explicitly separating local and global context extraction. Long-range dependencies are extracted by **N** cascaded Transformer encoder blocks. We adopt the standard Transformer design as suggested by [12, 44] with position embeddings (denoted as **P**)

3. Structure-Aware Transformer Network

An overview of the STAR model is depicted in Fig. 2. We first tokenize an image $\mathbf{I} \in \mathbb{R}^{H\times W\times C_I}$ into a sequence of flattened tokens $\mathbf{I}_T \in \mathbb{R}^{L\times C_T}$, where C_I and C_T are the number of channels. The generated tokens will be received by the long-short range Transformer module as inputs, which will then output two structural maps $\mathbf{S}_l, \mathbf{S}_s \in \mathbb{R}^{L\times L\times C_S}$ of long and short-range respectively. The predicted two structural maps could then be used to further estimate curves or transformation for image enhancement tasks.

3.1. Tokenization

To transform an image to tokens, a naive way is to flatten image into raw patches as mentioned in [8]. In that case, the features $\mathbf{I} \in \mathbb{R}^{H\times W\times C_I}$ are reshaped into a sequence of patches and treat them as tokens, i.e., $\mathbf{T} \in \mathbb{R}^{(\frac{H}{P}\times\frac{W}{P})(P^2\times C_I)}$, where P is the patch size and $L = (\frac{H}{P}\times\frac{W}{P})$ is the number of patches. However, this strategy will result in large memory consumption. Specifically, the input token vectors is formulated to have large dimension $t_i \in \mathbb{R}^{P^2\times C_I}, i = 1, 2, ..., N$, requiring massive parameters for training (e.g., 33 M parameters in [8]). An alternative strategy is to derive input token sequence from feature maps of a CNN [12, 50]. As a special case, the patches size here can be regarded as 1x1 after spatially downsampling, and tokens are extracted by a stacked convolution operation.

To support efficient real-time image enhancement, we implement the tokenization pipeline with the following steps. As shown in Fig. 2, we first flatten the full resolution feature into a sequence of patches. A dimension reduction operation for each patch is cascaded behind that. After that, we further extract tokens of each patch by learning linear embedding. The key to alleviating memory usage is proper dimension reduction.

In this paper, we have compared three tokenization strategies as depicted in Fig. 3. The most typical tokenization strategy is Linear Head [12, 8, 25], where the inputs are divided into patches and then linearly embedded. As mentioned before, for image enhancement such a strategy is too heavy. To reduce computation, we first try the Conv Head strategy as adopted in [15, 12]. Rather than computing the large $P^2C \times C_T$ (C means input channel dimensions and C_T denotes Transformer dimension) projection, we use a preprocessing CNN with gradually reducing spatial size. The token sequence is obtained by simply flatten the spatial dimension of the feature map. We further implement the Mean Head strategy, where the spatial size is reduced by Adaptive Average Pooling directly. This is inspired by squeeze-and-excitation block [23]. With Mean Head, we can maximumly reduce the tokenization complexity, but note that here we follow the assumption that the texture information (i.e. corners) is not crucial for learning structural context for specific image enhancement tasks (i.e. white balance). Similar to [50], we apply a 7×7 convolution

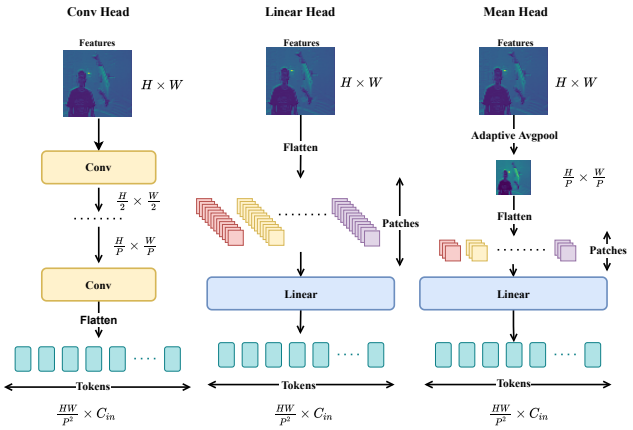

Figure 3. Illustration of tokenization methods used in STAR. Here $H \times W$ denotes the spatial resolution of inputs and C_T denotes Transformer dimension. **Conv head strategy**: The inputs will be fed to stacked convolution layers with downsampling. As described in [12], the token sequence is obtained by simply flatten the spatial dimension of the feature map and then projecting them to Transformer dimension. **Linear head strategy**: Splitting feature maps into patches directly and then projection to embedding. **Mean head strategy**: Reducing the spatial size using adaptive average pooling, then following linear head strategy.

with stride 4 and output channels 16 on the image and then feed output features to the above tokenization modules for more informative representations. We further quantitatively compare the three strategies. See Section 5.1.

3.2. Long-short Range Transformer Module

Unlike existing vision transformer methods [12, 25, 8] that use standard Transformer [44] architecture only, we adopt a two-branch Transformer design to process token sequences. As shown in Fig. 2, instead of feeding the entire token embeddings ($\mathbf{T} \in \mathbb{R}^{\frac{HW}{P^2} \times C_T}$) into both branches , we split them into two parts along the channel dimension. The divided embeddings ($\{\mathbf{T}_{long}, \mathbf{T}_{short}\} \in \mathbb{R}^{\frac{HW}{P^2} \times \frac{C_T}{2}}$) will be fed to the each branch respectively. Such practice effectively reduces the overall computation.

As shown in Fig. 2, for the left branch we use a normal transformer module as in [44]. Each Transformer contains multi-head self-attention module and a MLP with skip connection. Following [12], we use LayerNorm [4] as normalization and GELU [22] non-linearity function. We also add a 1D learnable position embedding $p \in \mathbb{R}^{\frac{C_T}{2}}$ to Transformer inputs to retain position information like [8, 12]. For the right branch of local relationships, we apply convolution to cover adjacent tokens. Unlike processing sentences in NLP where 1-D convolution [48, 27] are used to capture 1-D dependency, we use 2D convolutions to process rearranged image tokens. Instead of embedding the convolution inner Transformer module , we place the convolution branch in parallel with the whole Transformer module. Specifically, for long-range context ($\mathbf{y_{long}}$) and short-range context ($\mathbf{y_{short}}$), we have

$$\mathbf{T}_0 = \mathbf{T}_{long} + p \tag{1}$$

$$\tilde{\mathbf{T}}_n = MSA(LN(\mathbf{T}_{n-1})) + \mathbf{T}_{n-1}, n = 1...\mathbf{N} \tag{2}$$

$$\mathbf{T}_n = MLP(LN(\tilde{\mathbf{T}}_n)) + \tilde{\mathbf{T}}_n, n = 1...\mathbf{N} \tag{3}$$

$$\mathbf{y_{long}} = LN(\mathbf{T}_N) \tag{4}$$

$$\mathbf{y_{short}} = CNN(LN(\mathbf{T}_{short})) \tag{5}$$

MSA, MLP,and LN denote multi-head self-attention, multi-layer perceptron and layer normalization respectively. $\mathbf{N}$ is the depth of Transformer (number of basic transformer blocks). The resulting $\mathbf{y_{long}}$ and $\mathbf{y_{short}}$ can then used for various image enhancement tasks by adding a task tail. (e.g., adding linear layers to predict tone curves.)

4. Image Enhancement with STAR

To explore model versatility, we evaluate STAR on three applications using the most recent CNN methods: curve estimation for illumination enhancement [19], auto white balance [2] and photo retouching [52]. For a fair comparison, all the following experiments are based on public codes released by authors. We keep all experimental setups other than the backbone model (i.e., datasets, data augmentation, training strategy, ...) unchanged.

4.1. Illumination Enhancement

Many photos are often captured under inadequate and unbalanced lights. Such low-light photos not only suffer from compromised aesthetic quality but also challenge many fundamental downstream vision tasks like classification and object detection. To tackle this issue, we take one of the most efficient recent low-light enhancement methods DCE-Net (which is claimed to be 3× faster on GPU than the previous methods [45, 26, 47]) as our baseline to evaluate STAR.

In DCE-Net [19], a Deep Curve Estimation Network (DCE-Net) is devised to estimate a set of best-fitting Light-Enhancement curves (LE-curves) given an input image. The framework maps all pixels of the inputs' RGB channels by applying curves iteratively to obtain the final toned image. Quadratic curves $LE(I(\mathbf{x}); \alpha) = I(\mathbf{x}) + \alpha I(\mathbf{x})(1 - I(\mathbf{x}))$ are used in DCE-Net, where $I(\mathbf{x})$ and $LE(I(\mathbf{x}); \alpha)$ denote the adjusted image and original inputs respectively. The curve parameters $\alpha_i \in [-1, 1]^{H \times W}$ for i-th iteration are per-pixel predicted by a CNN with feature cascade. To evaluate our method, we replace this CNN backbone by the proposed STAR and estimate α_i using the resulting in structural contexts (y_{short}, y_{long}). With derived α_i, the calculation of LE-curve for i-th iteration can be formulated as:

$$\tilde{\alpha}_i = FC([y_{short}, y_{long}]) \tag{6}$$

Figure 4. Qualitative comparison of DCE-Net with different backbone on low light enhancement on MIT-Adobe FiveK [7].

$$\alpha_i = \psi(tanh(\tilde{\alpha}_i)) \tag{7}$$

$$LE_i(I(\mathbf{x}); \alpha_i) = LE_{i-1}(\mathbf{x}) + \alpha_i LE_{i-1}(\mathbf{x})(1 - LE_{i-1}(\mathbf{x})) \tag{8}$$

where FC denotes a learned projection to predict curves, $\psi(.)$ is an interpolation function to map per-token curves to per-pixel curves. We refer above pipeline as "STAR-DCE" in the following sections in contrast to CNN-based DCE-Net.

4.2. Auto White Balance

White balance (WB) is a fundamental low-level computer vision task applied to all camera images. Specifically, WB is formulated to normalize the effect of the captured scene's illumination so that all the captured objects appear to be under ideal "white light". We follow recent work [2] and implements STAR for WB with an encoder/decoder scheme. In this section, we will further demonstrate how STAR can be employed in an encoder/decoder network (i.e., U-Net [37]).

According to [2], a CNN is applied to produce the edited images $\mathbf{I}_{WB^{(i)}\downarrow}$ with target WB setting $i \in A, T, S$[1]. The network is built upon 4-level encoder/decoders with 2×2 max-pooling and transposed convolution. [2] use a multi-decoder architecture consisting of two units: (1) A 4-level encoder for extracting multi-scale latent representation of images. (2) Three 4-level decoders corresponding to AWB, Incandescent WB, and Shade WB settings. To employ Transformer, we use the proposed STAR the replace the original encoder. We also adopt the multi-decoder design which is shown to yield better performance than vanilla U-Net [37] by [2]. Specifically, the input images are first divided into 16×16 patches (corresponding to the original 4-level encoder) and then projected to tokens. After that, we feed the tokens to our long-short range Transformer and the generated structural context $[y_{long}, y_{short}]$ with spatial size $\frac{H}{16} \times \frac{W}{16}$ will then be used by subsequent three decoders. Following the prior practice in illumination enhancement, we adopt the Mean Head tokenization and two-branch transformer design. We keep the first convolution ($3 \times 24 \times 3 \times 3$) layer and use its producing feature maps to guide the image generation of decoders. Like [2], the whole STAR module is shared by three decoders and jointly trained with them.

[1] A: AWB, T: Incandescent WB, S: Shade WB

4.3. Photo Retouching with 3D-LUTs

We choose 3D-LUTs (lookup tables) learning as an instance to show how STAR improves the learning of global adjustments. 3D-LUTs are usually applied to adjust the hue, saturation, exposure, color and tone of photos in camera imaging pipelines. As one of the most classical yet widely used photo adjustment techniques, 3D LUT can achieve stable photo enhancement performance at very high efficiency. We evaluate STAR on 3D-LUTs estimation following the framework of [52], which learns several (e.g., 3) basis 3D LUTs to interpolate images and fuses the results by using CNN to learn a linear combination of them.

Conventional 3D LUT based image enhancement methods have a main limitation that the 3D LUTs are mostly manually and only provide a fixed transformation. To adapt to different scenes, [52] learns several LUTs and uses a small CNN predictor to fuse them. Although efficient and stable, as discussed in [52], such 3D-LUT model have limitations inherited from global adjustments. Once the 3D LUT is determined for an input image, it is the same for dif-

ferent local areas within the image. Naturally, such strategies produce unsatisfactory results for images requiring local enhancements (e.g., target images with high dynamic range). [52] intends to tackle this problem by applying a local tone mappings [13] as preprocessing. However, such a local tone mapping method is still time-consuming for high-resolution images and this issue is left to be future work by [52].

To apply our model, we replace the CNN directly with STAR and predict combination weights on $[y_{short}, y_{long}]$. Rather than predicting global adjustments weight for LUT directly, STAR predicts 32×32 token-wise weights, which are then used to fuse interpolated images of each LUT.

5. Experiments

5.1. Illumination Enhancement Results

This section compares our STAR with the original CNN model on curve estimation with the following setups.

Datasets. We choose the MIT-Adobe FiveK [7] (FiveK) to compare our methods with CNN. The FiveK dataset contains 5,000 image pairs with retouched ground-truth by experts (A - E). We follow previous methods [45, 17, 24, 35] to use only the Output from Expert C and randomly split images into two parts: 500 images for validation and testing, remaining 4,500 images for training. For a fair comparison, we use datasets split by [45]. Both the training and evaluation images are resized to 1200×900 based on their longest side like [19]. Note that here we choose MIT-Adobe FiveK rather than datasets released by [19] or other datasets corresponding to specific methods because we intend to show that STAR can work as a general backbone, which should be evaluated on common and widely-used datasets.

Model Complexity. DCE-Net [19] choose a CNN with 7 layers, 32 maximum output channels as its network backbone (79.42k parameters). Following that, we first set the Transformer dimension (C_{in}) to be the same 32. The inner MLP dimensions are also kept the same. Both the CNN and STAR model predicts 24 curves (totally 8 iterations for 3 channels) like practice in [19]. Each image will be divided to 32×32 tokens in STAR. In this task, we observe strong overfitting when applying STAR with large Transformer depth. Consequently, we use STAR of Transformer depth 1 as defaults. Additional ablation studies towards model complexity are performed including changing model width/depth, using Mean Head tokenization and employing a two-branch design.

Training and Evaluation. Similar to [52], images are processed in full resolution (1200×900) when evaluation and resized to 256×256 during training. Following [19], Adam [30] optimizer is applied with fixed learning rate $1e^{-4}$. We use a simple L1-norm as loss function for both the CNN and STAR based models to fairly compare the model performance. We evaluate model efficiency using the following metrics: memory consumption (number of parameters), model complexity (float operations, FLOPS)[2], enhancement performance (PSNR/SSIM) and inference latency.

	Tokenization	Branches	Parameters (K)	FLOPS (G)	PSNR (dB)	SSIM
DCE-Net	-	-	79.4	5.20	22.7	0.870
DCE-Net$_D$	-	-	79.4	0.51	22.2	0.866
STAR-DCE	Linear	1	79.6	0.17	24.0	0.882
STAR-DCE	Conv	1	32.6	0.10	24.5	0.894
STAR-DCE	Mean	1	23.3	0.07	24.5	0.892
STAR-DCE	Linear	2	77.8	0.15	24.1	0.885
STAR-DCE	Conv	2	30.8	0.08	24.4	0.894
STAR-DCE	Mean	2	**20.1**	**0.05**	24.5	0.893

Table 1. Comparisons of Tokenization and Two-branch strategy. DCE-Net and DCE-Net$_D$ denote the 7-layer CNN model with/without downsampling. We report model size and total float operations (FLOPS) and corresponding average PSNR on FiveK evaluation set. The top results are bold.

To identify the most efficient STAR setups, we perform several ablation studies:

Ablation I: Tokenization and Long-short Range Capturing. We first compare the mentioned Conv / Linear / Mean Head tokenization. As shown in the table. 1, the three tokenization methods bring approaching model performance despite the varying model size. As a comparison, we also train DCE-Net with downsampling to token size 32×32 , denoted by DCE-Net$_D$. We observe that STAR model could hardly benefit from complex tokenization methods especially the most wide-used Linear Head [8, 12, 25]. In contrast, simply averaging the feature maps can reduce more than 50% computations and memory usage without performance loss, and the model can be further compressed by applying a two-branch (long-short range) design. The results support that texture information like corners are not essential for such toning tasks. Subsequent experiments perform the Mean Head tokenization and two-branch design as defaults for better efficiency.

Ablation II: Slimmer Model. This section explores the model scalability of CNNs and STAR by using fewer channels/layers. We reduce network width to 16 (for both STAR and CNN) and depth to 3 (for CNN) to observe the sensitivity of model performance towards model size. Table. 2 reports the quantitative results. As shown, reducing CNN depth to 3 will severely harm model performance (over 10 dB PSNR drops). Besides, reducing CNN width to 16 will also result in obvious degradation on outputs quality (i.e., 0.9 dB PSNR and 0.04 SSIM drop for DCE-Net$_D$). Qualitative comparison can be seen in Fig. 4 The above results illustrate the limitation of the pure CNN model to capture structural representations. To derive a more efficient model, simply reducing the depth/width of CNN may also lead to poor performance.

Runtime Latency. We have shown the effectiveness

[2]FLOPS are calculated with inputs of size $3 \times 256 \times 256$ in this application

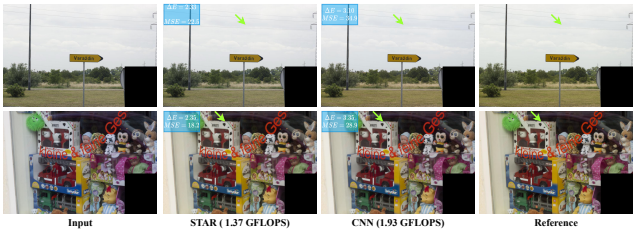

Figure 5. Qualitative comparison of AWB correction on Cube+ [5]. We show both model complexity (FLOPS) and performance (ΔE 2000, MSE). Green arrows indicate the obvious color difference.

	Width	Layers	Parameters (K)	FLOPS (G)	PSNR (dB)	SSIM
DCE-Net	32	7	79.4	5.20	22.7	0.870
DCE-Net$_D$	32	7	79.4	0.31	22.2	0.866
DCE-Net	16	7	23.6	1.55	21.6	0.820
DCE-Net$_D$	16	7	23.6	0.10	21.3	0.824
DCE-Net	32	3	24.0	1.57	12.3	0.605
STAR-DCE	32	-	20.1	0.04	**24.5**	**0.893**
STAR-DCE	16	-	7.2	0.01	23.0	0.845

Table 2. Comparisons of STAR-DCE and DCE-Net with reduced depth/width. DCE-Net$_D$ denotes DCE-Net with downsampling.

	Platform	Latency (s)
*SRIE [16]	Matlab (CPU)	12.1865
*LIME [20]	Matlab (CPU)	0.4914
*RetinexNet [47]	Tensorflow (GPU)	0.1200
*DeepUPE [45]	Tensorflow (GPU)	0.0210
*EnlightenGAN [26]	PyTorch (GPU)	0.0078
*DCE-Net [19]	PyTorch (GPU)	0.0025
DCE-Net [19]	PyTorch (GPU)	0.0037
DCE-Net$_D$ [19]	PyTorch (GPU)	0.0027
STAR-DCE	PyTorch (GPU)	**0.0019**

Table 3. Runtime latency comparisons of STAR-DCE / DCE-Net and existing methods. "*" denotes results reported by [19] (tested on Nvidia 2080Ti and Inter i7 6700). The remaining results are from our re-implementation (tested on Nvidia 1080 Ti and Intel Xeon 6126)

of STAR-DCE compared the CNN baselines on both low-light enhancement quality (i.e., PSNR: 24.5, SSIM: 0.893 *vs*. PSNR: 22.2, SSIM: 0.866) and theoretical complexity (i.e., 20.1K parameters, 0.04G FLOPS *vs*. 79.4K parameters, 0.51G FLOPS). To further measure STAR'S real-time performance, we evaluate its inference speeds in contrast to [19] and other existing methods. The images are resized to a resolution of 1200×900 and tested for 32 times on Nvidia 1080 Ti GPU with 11GB memory. We also include results tested by [19] as a comparison. Table. 3 reports the averaged results. As can be seen, DCE-Net is of high efficiency and over $3\times$ faster than existing methods. By further using the STAR backbone, DCE-Net can even have an extra $2\times$ speed-up.

5.2. White Balance Results

Datasets. We adopt Rendered WB dataset [3] and the Cube+ dataset [5] to compare STAR with CNNs. The Rendered WB dataset [3] consists of two subsets: Set 1 (62,535 images captured by seven different DSLR cameras) and Set 2 (2,881 images captured by a DSLR camera and four mobile phone cameras). Set 1 is split into three folds by [3]. Following [2], We train our networks on 12,000 images randomly selected from the first and second folds of Set 1. The remaining third fold (referred to as Set 1-$Test$) together with the Set 2 (2,881 images) and the Cube+ dataset (10,242 images) are used for testing.

	Datasets	Parameters (M)	FLOPS (G)	MAE	MSE	ΔE 2000
WB editting [2]	Set 1–Test	4.37	1.93	3.12	82.55	3.77
WB editting+STAR	Set 1–Test	3.32	**1.37**	3.24	79.95	3.62
WB editting [2]	Set 2	4.37	1.93	3.75	124.97	4.90
WB editting+STAR	Set 2	3.32	**1.37**	3.67	118.27	4.79
WB editting [2]	Cube+	4.37	1.93	3.45	80.96	4.59
WB editting+STAR	Cube+	3.32	**1.37**	3.31	75.8	4.32

Table 4. AWB comparisons of CNN encoder and STAR encoder on white balance datasets. "WB editting" denotes the original CNN encoder in [2] and "WB editting+STAR" denotes the transformer encoder. We report the mean of the image MAE (mean angular error), MSE (mean square error), and ΔE 2000 for the each evaluation set.

Training Strategy. For a fair comparison, we keep most of the training strategy in [2] unchanged. For each image, four 128×128 patches are randomly sampled for training together with their corresponding ground truth. Geometric augmentations including random rotation and flipping are added for data augmentation. The networks are trained using MAE (Mean Absolute Error) loss to minimize L1-norm between the generated and ground truth patches. Model parameters are optimized using Adam [30] with initial learning rates $1e^{-4}$, which are reduced by 0.5 every 25 epochs.

Quantitative Results. In Table. 4 and Fig. 5, we provide a comparison between CNN encoder and STAR encoder on three evaluation sets. Following [2], we use the mean MAE (mean angular error), MSE (mean square error), and ΔE 2000 to evaluate the produced results. For model size and FLOPS[3] in the table, we only count the encoder and AWB decoder as only AWB results are evaluated. From Table. 4 we can see that with STAR encoder, the methods can yield approaching (even slightly better) results with 25% fewer parameters and 30% fewer computations. Note that such compression ratios are achieved by only replacing the CNN encoder. We also have tried to replace the entire CNN decoder with Transformers (to have a smaller model) but this idea yields poor performance when reconstructing high-resolution images. We leave this topic to be future works to further reducing the model computations.

5.3. Photo Retouching Results

Datasets. We conduct experiments on HDR+ [21] to demonstrate the performance improvement of STAR. HDR+ is a burst photography dataset for research of high dynamic range and low-light imaging on mobile cameras. The same to [52], we use the TIF images transformed from

[3]Calculated with $3 \times 128 \times 128$ inputs [2]

Figure 6. Qualitative results of photo retouching by learning 3D LUTs with CNN and STAR on HDR+ dataset [21].

	Parameters (K)	FLOPS (G)	PSNR	SSIM	ΔE
CNN$_{global}$	270.1	0.07	23.5	0.885	7.93
CNN$_{semantic}$	270.1	0.29	25.6	0.884	5.81
CNN-half$_{global}$	74.0	0.02	22.8	0.881	8.54
CNN-half$_{semantic}$	74.0	0.08	24.4	0.881	6.24
STAR-64$_{global}$	109.4	0.08	23.7	0.885	7.24
STAR-64$_{semantic}$	109.4	0.08	**26.8**	0.887	**5.42**
STAR-32$_{global}$	43.4	0.03	23.5	0.884	7.75
STAR-32$_{semantic}$	43.4	0.03	**26.5**	0.883	**5.77**

Table 5. Comparisons of photo retouching by CNNs and STAR on HDR+ dataset. Here $_{global}$ and $_{semantic}$ denotes learning per-image weight / per-token weight respectively. STAR-32 and STAR-64 represents STAR model with Transformer dimension 32 / 64.

an intermediate result of the aligned and merged frames (in DNG format) as inputs, while the JPG images generated by manually fine-tuned HDR imaging pipeline as ground truth. Following the dataset settings of the photo retouching application in [52], both the inputs and targets are compressed in JPG format and have an 8-bit dynamic range. Since most scenes in the HDR+ dataset are not aligned between the intermediate frame and the ground-truth, we use the well-aligned Nexus 6p subset (consists of 675 scenes for training and 250 scenes for testing.) according to [52].

Training Strategy. We train and evaluate both CNNs and STAR models on 480p resolutions. During training, each image will be randomly cropped into 256×256 patches then fed to the model as inputs. As mentioned before, we kept all the training settings other than the backbone model unchanged. To be specific, we optimize both the CNN and STAR model with Adam optimizer and fixed learning rate $1e^{-4}$. Input patches are randomly flipped for data augmentation.

Quantitative Results. We employ three metrics to evaluate model performance: PSNR, SSIM, and deltaE (ΔE). Smaller ΔE and higher PSNR / SSIM represent better performance. To fairly compare with STAR, we not only train CNN to predict global LUT weights for the entire image like [52], but also use the same architecture to predict 32×32 per-token weights by changing strides of the last two layers from 2 to 1. We denote such two strategies as $_{global}$ and $_{semantic}$ respectively. As shown in Table. 5, predicting semantic weights significantly improve the methods' performance on processing HDR+ images. However, such improvements are at the cost of heavy computation (i.e., for [52], over $3\times$ FLOPS[4]). Experiments show that the obtained results using our STAR achieve 26.8 dB PSNR with 40% parameters and approaching FLOPS compared to CNNs. Moreover, when we reduce the width of the STAR by half, the model can still maintain high performance with only a 0.3 dB PSNR drop. This shows the STAR architecture has high efficiency to capture semantic context. Fig. 6 visualizes examples of the enhanced results.

6. Conclusion and Discussion

In this work, we have presented Semantic-aware Transfomer (STAR), a new lightweight deep learning backbone for image enhancements. The proposed methods allow fast inference with small memory usage. We conduct experiments to evaluate STAR on three common applications including illumination enhancements, white balance and photo retouching with 3D LUTs. Quantitative and qualitative results demonstrate the efficiencies and effectiveness of the proposed STAR with the state-of-the-art prior work. In our future work, we plan to extend STAR for more image and video enhancement tasks such as denoising and super-resolution.

7. Acknowledgement

This work is supported in part by Centre for Perceptual and Interactive Intelligence Limited, in part by the General Research Fund through the Research Grants Council of Hong Kong under Grants (Nos. 14202217, 14203118, 14208619,27208720), in part by Research Impact Fund Grant No. R5001-18.

[4]Calculated with $3 \times 256 \times 256$ inputs

References

[1] Mahmoud Afifi. Semantic white balance: Semantic color constancy using convolutional neural network. *arXiv preprint arXiv:1802.00153*, 2018. 2

[2] Mahmoud Afifi and Michael S Brown. Deep white-balance editing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1397–1406, 2020. 2, 4, 5, 7

[3] Mahmoud Afifi, Brian Price, Scott Cohen, and Michael S Brown. When color constancy goes wrong: Correcting improperly white-balanced images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1535–1544, 2019. 7

[4] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016. 4

[5] Nikola Banić, Karlo Koščević, and Sven Lončarić. Unsupervised learning for color constancy. *arXiv preprint arXiv:1712.00436*, 2017. 7

[6] Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020. 2

[7] Vladimir Bychkovsky, Sylvain Paris, Eric Chan, and Frédo Durand. Learning photographic global tonal adjustment with a database of input/output image pairs. In *CVPR 2011*, pages 97–104. IEEE, 2011. 1, 5, 6

[8] Hanting Chen, Yunhe Wang, Tianyu Guo, Chang Xu, Yiping Deng, Zhenhua Liu, Siwei Ma, Chunjing Xu, Chao Xu, and Wen Gao. Pre-trained image processing transformer. *arXiv preprint arXiv:2012.00364*, 2020. 2, 3, 4, 6

[9] Yu-Sheng Chen, Yu-Ching Wang, Man-Hsin Kao, and Yung-Yu Chuang. Deep photo enhancer: Unpaired learning for image enhancement from photographs with gans. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6306–6314, 2018. 1, 2

[10] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 2

[11] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018. 2

[12] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. 2, 3, 4, 6

[13] Frédo Durand and Julie Dorsey. Fast bilateral filtering for the display of high-dynamic-range images. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 257–266, 2002. 6

[14] Thibaud Ehret, Axel Davy, Pablo Arias, and Gabriele Facciolo. Joint demosaicking and denoising by fine-tuning of bursts of raw images. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8868–8877, 2019. 2

[15] Patrick Esser, Robin Rombach, and Björn Ommer. Taming transformers for high-resolution image synthesis. *arXiv preprint arXiv:2012.09841*, 2020. 3

[16] Xueyang Fu, Delu Zeng, Yue Huang, Xiao-Ping Zhang, and Xinghao Ding. A weighted variational model for simultaneous reflectance and illumination estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2782–2790, 2016. 7

[17] Michaël Gharbi, Jiawen Chen, Jonathan T Barron, Samuel W Hasinoff, and Frédo Durand. Deep bilateral learning for real-time image enhancement. *ACM Transactions on Graphics (TOG)*, 36(4):1–12, 2017. 1, 2, 6

[18] Shuhang Gu, Yawei Li, Luc Van Gool, and Radu Timofte. Self-guided network for fast image denoising. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2511–2520, 2019. 2

[19] Chunle Guo, Chongyi Li, Jichang Guo, Chen Change Loy, Junhui Hou, Sam Kwong, and Runmin Cong. Zero-reference deep curve estimation for low-light image enhancement. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1780–1789, 2020. 1, 2, 4, 6, 7

[20] Xiaojie Guo, Yu Li, and Haibin Ling. Lime: Low-light image enhancement via illumination map estimation. *IEEE Transactions on image processing*, 26(2):982–993, 2016. 7

[21] Samuel W Hasinoff, Dillon Sharlet, Ryan Geiss, Andrew Adams, Jonathan T Barron, Florian Kainz, Jiawen Chen, and Marc Levoy. Burst photography for high dynamic range and low-light imaging on mobile cameras. *ACM Transactions on Graphics (TOG)*, 35(6):1–12, 2016. 7, 8

[22] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016. 4

[23] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7132–7141, 2018. 3

[24] Yuanming Hu, Hao He, Chenxi Xu, Baoyuan Wang, and Stephen Lin. Exposure: A white-box photo post-processing framework. *ACM Transactions on Graphics (TOG)*, 37(2):1–17, 2018. 1, 2, 6

[25] Yifan Jiang, Shiyu Chang, and Zhangyang Wang. Transgan: Two transformers can make one strong gan. *arXiv preprint arXiv:2102.07074*, 2021. 3, 4, 6

[26] Yifan Jiang, Xinyu Gong, Ding Liu, Yu Cheng, Chen Fang, Xiaohui Shen, Jianchao Yang, Pan Zhou, and Zhangyang Wang. Enlightengan: Deep light enhancement without paired supervision. *IEEE Transactions on Image Processing*, 30:2340–2349, 2021. 4, 7

[27] Zihang Jiang, Weihao Yu, Daquan Zhou, Yunpeng Chen, Jiashi Feng, and Shuicheng Yan. Convbert: Improving bert with span-based dynamic convolution. *arXiv preprint arXiv:2008.02496*, 2020. 2, 4

[28] Nima Khademi Kalantari and Ravi Ramamoorthi. Deep high dynamic range imaging of dynamic scenes. *ACM Trans. Graph.*, 36(4):144–1, 2017. 2

[29] Nima Khademi Kalantari, Eli Shechtman, Connelly Barnes, Soheil Darabi, Dan B Goldman, and Pradeep Sen. Patch-based high dynamic range video. *ACM Trans. Graph.*, 32(6):202–1, 2013. 2

[30] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 6, 7

[31] Satoshi Kosugi and Toshihiko Yamasaki. Unpaired image enhancement featuring reinforcement-learning-controlled image editing software. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 11296–11303, 2020. 1, 2

[32] Zhetong Liang, Jianrui Cai, Zisheng Cao, and Lei Zhang. Cameranet: A two-stage framework for effective camera isp learning. *IEEE Transactions on Image Processing*, 30:2248–2262, 2021. 2

[33] Orly Liba, Longqi Cai, Yun-Ta Tsai, Elad Eban, Yair Movshovitz-Attias, Yael Pritch, Huizhong Chen, and Jonathan T Barron. Sky optimization: Semantically aware image processing of skies in low-light photography. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 526–527, 2020. 1, 2

[34] Seonghyeon Nam and Seon Joo Kim. Modelling the scene dependent imaging in cameras with a deep neural network. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1717–1725, 2017. 2

[35] Jongchan Park, Joon-Young Lee, Donggeun Yoo, and In So Kweon. Distort-and-recover: Color enhancement using deep reinforcement learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5928–5936, 2018. 1, 2, 6

[36] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019. 2

[37] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015. 5

[38] Eli Schwartz, Raja Giryes, and Alex M Bronstein. Deepisp: Toward learning an end-to-end image processing pipeline. *IEEE Transactions on Image Processing*, 28(2):912–923, 2018. 2

[39] Pradeep Sen, Nima Khademi Kalantari, Maziar Yaesoubi, Soheil Darabi, Dan B Goldman, and Eli Shechtman. Robust patch-based hdr reconstruction of dynamic scenes. *ACM Trans. Graph.*, 31(6):203–1, 2012. 2

[40] Weijie Su, Xizhou Zhu, Yue Cao, Bin Li, Lewei Lu, Furu Wei, and Jifeng Dai. Vl-bert: Pre-training of generic visual-linguistic representations. *arXiv preprint arXiv:1908.08530*, 2019. 2

[41] Runjie Tan, Kai Zhang, Wangmeng Zuo, and Lei Zhang. Color image demosaicking via deep residual learning. In *IEEE Int. Conf. Multimedia and Expo (ICME)*, volume 2, page 6, 2017. 2

[42] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. *arXiv preprint arXiv:2012.12877*, 2020. 2

[43] Yi-Hsuan Tsai, Xiaohui Shen, Zhe Lin, Kalyan Sunkavalli, and Ming-Hsuan Yang. Sky is not the limit: semantic-aware sky replacement. *ACM Trans. Graph.*, 35(4):149–1, 2016. 2

[44] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *arXiv preprint arXiv:1706.03762*, 2017. 2, 3, 4

[45] Ruixing Wang, Qing Zhang, Chi-Wing Fu, Xiaoyong Shen, Wei-Shi Zheng, and Jiaya Jia. Underexposed photo enhancement using deep illumination estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6849–6857, 2019. 1, 2, 4, 6, 7

[46] Yue Wang, Shafiq Joty, Michael R Lyu, Irwin King, Caiming Xiong, and Steven CH Hoi. Vd-bert: A unified vision and dialog transformer with bert. *arXiv preprint arXiv:2004.13278*, 2020. 2

[47] Chen Wei, Wenjing Wang, Wenhan Yang, and Jiaying Liu. Deep retinex decomposition for low-light enhancement. *arXiv preprint arXiv:1808.04560*, 2018. 4, 7

[48] Zhanghao Wu, Zhijian Liu, Ji Lin, Yujun Lin, and Song Han. Lite transformer with long-short range attention. *arXiv preprint arXiv:2004.11886*, 2020. 2, 4

[49] Zhicheng Yan, Hao Zhang, Baoyuan Wang, Sylvain Paris, and Yizhou Yu. Automatic photo adjustment using deep neural networks. *ACM Transactions on Graphics (TOG)*, 35(2):1–15, 2016. 1, 2

[50] Li Yuan, Yunpeng Chen, Tao Wang, Weihao Yu, Yujun Shi, Francis EH Tay, Jiashi Feng, and Shuicheng Yan. Tokens-to-token vit: Training vision transformers from scratch on imagenet. *arXiv preprint arXiv:2101.11986*, 2021. 3

[51] Xiaoyu Yue, Zhanghui Kuang, Zhaoyang Zhang, Zhenfang Chen, Pan He, Yu Qiao, and Wei Zhang. Boosting up scene text detectors with guided cnn. *arXiv preprint arXiv:1805.04132*, 2018. 1

[52] Hui Zeng, Jianrui Cai, Lida Li, Zisheng Cao, and Lei Zhang. Learning image-adaptive 3d lookup tables for high performance photo enhancement in real-time. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020. 1, 2, 4, 5, 6, 7, 8