Personalized Exposure Control Using Adaptive Metering and Reinforcement Learning

Huan Yang, Baoyuan Wang, Noranart Vesdapunt, Minyi Guo, *Fellow, IEEE*, and Sing Bing Kang, *Fellow, IEEE*

Abstract—We propose a reinforcement learning approach for real-time exposure control of a mobile camera that is personalizable. Our approach is based on Markov Decision Process (MDP). In the camera viewfinder or live preview mode, given the current frame, our system predicts the change in exposure so as to optimize the trade-off among image quality, fast convergence, and minimal temporal oscillation. We model the exposure prediction function as a fully convolutional neural network that can be trained through Gaussian policy gradient in an end-to-end fashion. As a result, our system can associate scene semantics with exposure values; it can also be extended to personalize the exposure adjustments for a user and device. We improve the learning performance by incorporating an adaptive metering module that links semantics with exposure. This adaptive metering module generalizes the conventional spot or matrix metering techniques. We validate our system using the MIT FiveK [1] and our own datasets captured using iPhone 7 and Google Pixel. Experimental results show that our system exhibits stable real-time behavior while improving visual quality compared to what is achieved through native camera control.

Index Terms—Auto exposure, reinforcement learning, personalization

1 Introduction

Real-time auto-exposure is a camera operation that enables high-quality photo capture. In smartphone cameras, this fundamental operation is typically based on simple metering over a predefined area or areas, and the analysis is independent of the scene. The newer smartphone cameras are capable of real-time detection of faces, and thus capable of using facial information to influence the exposure setting. This is especially useful for capturing backlit scenes. There are new high-end smart phones that claim to be able to detect scene categories beyond faces (e.g., Huawei P20 Pro and Xiaomi Mi 8). Details of how scene information is utilized for improving exposure control are not available, and it is not clear if the user personalization feature is available as well.

Currently, if the user is not satisfied with the viewfinder (live preview) exposure and wishes to modify it, he/she would need to tap on the region of interest and then manually tweak the exposure by the sliding bar before the shutter press, as shown in Fig. 1. However, during this "tap-and-tweak" process, it would be easy to miss the best moment to capture. Our goal is develop a system that automatically produces the exposure that is acceptable to the user and thus obviates the need for such manual adjustment.

We are not aware of published research work done on using generic scene information for real-time exposure control, not to mention additionally catering to specific user preferences. It is evident from the MIT FiveK [1] dataset that how an image is exposed through tonal adjustment is person-dependent. The notion of personalization of image enhancement has also been studied elsewhere (e.g., [2], [3], [4]), which points to its importance.

1.1 Challenges

How would one design a real-time personalized exposure control system? Let us first look at how the control of exposure EV works mathematically. (The definition of exposure is given in the Appendix, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TVCG.2018.2865555.) Since we bypass the hardware metering function, the exposure value EV_{i+1} for next frame I_{i+1} is fully determined by the current frame I_i (at time i). Let us denote $\mathrm{F}(I_i)$ as the function of predicting the exposure adjustment Δ^i_{EV} given input frame I_i, $\mathrm{F}(I_i) = \Delta^i_{EV}$. The camera will then apply $EV_{i+1} = EV_i + \mathrm{F}(I_i)$ to capture the next frame I_{i+1}; this process is iterated in the viewfinder mode.[1]

An ideal way of implementing this idea is to collect paired training data (I_i, Δ^i_{EV}) and then perform supervised learning to learn function F. Hypothetically, we can do the

- *H. Yang and M. Guo are with the Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai 200000, China. E-mail: yanghuanflc@sjtu.edu.cn, guo-my@cs.sjtu.edu.cn.*
- *B. Wang, N. Vesdapunt, and S.B. Kang are with Microsoft Research, Redmond, WA 98052-5321. E-mail: {baoyuanw, noves}@microsoft.com, sbkang@ieee.org.*

Manuscript received 5 Mar. 2018; revised 4 Aug. 2018; accepted 8 Aug. 2018. Date of publication 15 Aug. 2018; date of current version 30 Aug. 2019. (Corresponding author: Baoyuan Wang). Recommended for acceptance by K. Myszkowski. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below. Digital Object Identifier no. 10.1109/TVCG.2018.2865555

1. In practice, the updated EV_{i+1} will not be immediately applied to frame I_{i+1}. This is because third-parties typically do not have direct access to the image signal processing (ISP) firmware. Instead, the latency through API calls lasts some number of frames (from our experience, 3 to 5 for iPhone 7 and Google Pixel).

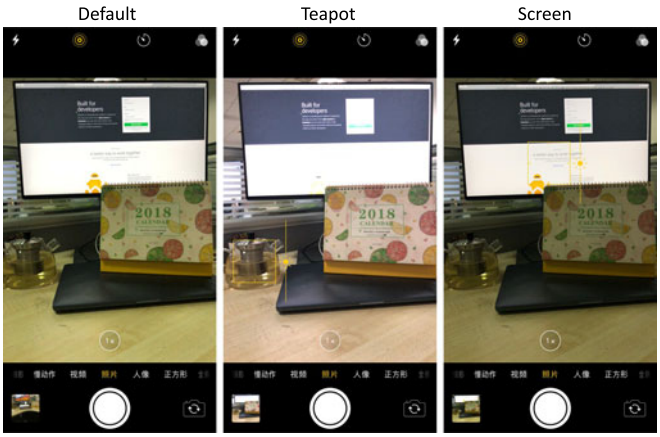


Fig. 1. The "tap-and-tweak" strategy for manually optimizing exposure. The user selects the region of importance by tapping on it, and the camera responds by prioritizing exposure over that region.

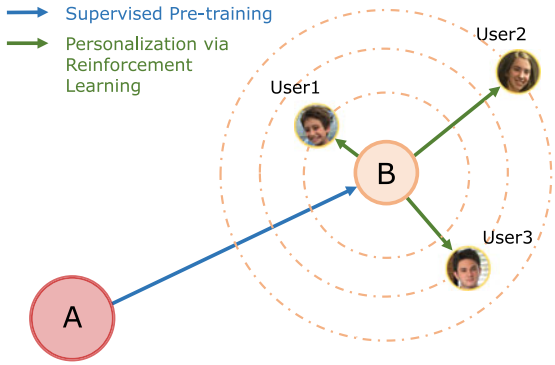


Fig. 2. Our coarse-to-fine learning strategy. At the coarse level, we perform supervised pre-training first to mimic the native camera, which enables us to change from A to B. The fine adjustment is achieved by training the reinforcement learning module on B and learning personalization for different users.

following: use exposure bracketing to capture various images with different exposure values to create a dataset, then conduct a labeling exercise selecting the best exposure. Subsequently, apply any supervised learning to learn a regression function F that can map any incorrect exposure to its correct one. However, it is not practical to ask each user to capture and annotate a large scale bracketing dataset in order to train a personalized model. An alternative to acquire training data would be through the "tap-and-tweak" mechanism, but again, this approach would not be practical. This is because finding the label Δ_{EV} associated with the actual corresponding frame in the viewfinder would be non-trivial. Without direct access to the camera hardware (especially for the iPhone), there is a lag when invoking the camera API, which would easily result in mismatched training pairs. All those challenges motivate us to develop an automatic system with a more practical means for acquiring training data.

1.2 Our Approach

Any well-designed non-native (third party) exposure control system should first perform as well as the native camera for the "easy" cases (e.g., scenes with good uniform lighting) while improving on more challenging scenes that require object prioritization (e.g., back-lit scenes). This inspires us to adopt a "coarse-to-fine" learning strategy, with "coarse" being pre-training using native camera data and "fine" being fine-tuning using on-line learning. This strategy is illustrated in Fig. 2.

To enable semantic understanding, we use a fully convolutional neural (FCN) network to represent the exposure prediction function F. The FCN network is pre-trained through supervised learning on a synthetic dataset. After the pre-training, our model mimics the behavior of the native camera and can be deployed to each end-user. We call this model as the basic or average model. Once deployed, the basic model is then fine-tuned locally through the on-line learning module.

During the on-line stage, at time t, the hardware can only choose one specific EV and capture a corresponding image I, which is unlike full exposure bracketing. Note that it is impractical to ask users to directly provide the annotation of Δ_{EV} for image I without the corresponding exposure bracketing set. However, the user could instead provide feedback on the exposure after capture, namely, if the captured image is "under-exposed," "correctly exposed," or "over-exposed."

Such feedback serves as a reward signal to indirectly supervise how the system intelligently selects Δ_{EV} for a given image. This is where reinforcement learning comes in; it allows both data collection and personalization to be feasible and scalable. After we accumulate a new batch of images with their corresponding reward signals, the local model is then fine-tuned through back-propagation on this new batch based on a Gaussian policy gradient method. This process is iterated until all the feedback signals are positive.

Most native cameras have default metering options for auto-exposure. Each option works by assigning a relative weighting to different spatial region. However, such weighting schemes are all heuristically pre-defined, such as spot or center-weighted metering. (Newer cameras also use face information for exposure prioritization.) To generalize metering and improve the learning performance, we introduce an adaptive metering module into the FCN network. For each image frame, the adaptive metering module outputs a weighting map which is element-wise multiplied by another learned exposure map. The whole system is learned end-to-end. In this paper, we show the effectiveness of the adaptive metering module both visually and quantitatively. Once the model is trained, during run-time, we directly feed-forward the current frame into the network to get the output Δ_{EV}, which then used to capture the next frame.

Our major contributions are as follow:

- To the best of our knowledge, our work is the first to address personalized real-time exposure control based on learned scene semantics.
- We propose a practical "coarse-to-fine" learning strategy which first uses supervised learning to achieve a good anchor point, followed by refinement through reinforcement learning. We believe that such a training strategy could inspire similar approaches to other problems in computer vision.
- We introduce an adaptive metering module that can automatically infer user prioritization in a scene. We show that it improves the learning performance.

TABLE 1
Feature Comparison between Post-Processing Approaches, Native Approach, and Our Approach

	Post-processing	Native	Ours
Scene-Aware	Partially[(a)]	Partially[(b)]	Yes
Personalized	Partially[(a)]	No	Yes
Real-Time[(c)]	No	Yes	Yes

Notes: (a) *Many techniques are not content-aware and/or personalized.* (b) *Exposure prioritization is based on faces and possibly very simple scenes, such as sky [15].* (c) *Real-time on mobile devices.*

- We develop an end-to-end system and implement it on iPhone 7 and Google Pixel and demonstrate good performance in terms of both speed and visual quality. Our system learns the proper exposure for a variety of scenes and lighting conditions, and outperforms its native camera counterparts for challenging cases that require general semantic understanding.

2 Related Work

In this section, we briefly review prior work on auto-exposure and post-processing techniques for tonal enhancement.

2.1 Auto-Exposure

Except a few early papers (e.g., [5], [6]), there appears to be little published research work on exposure control for mobile cameras. For competitive reasons, camera manufacturers do not usually publicize full details of their technique for real-time auto-exposure, though information for a few camera models exists (see, for example, [7]). In native mode, cameras rely on built-in metering to measure the amount of light for determining the proper exposure values. This works well in many cases. However, assuming different options for meter mode are available, users may need to modify the meter mode (e.g., spot, matrix) and focus area for handling more challenging conditions such as back-lit scenes. Techniques for more specialized treatment of front- and back-lit scenes have been described (e.g., [8]), but they fundamentally rely on histogram analysis that does not jointly consider object importance and lighting condition. Some native cameras on popular phones prioritize exposure on detected face regions [9], [10], [11]; many allow user interaction to select a region of interest for appropriate exposure adjustment. In contrast, our auto-exposure approach supports more general scenes beyond faces, and it does not require user interaction.

2.2 Post-Processing Tonal Adjustment

There are many post-processing techniques for enhancing image quality (e.g., [1], [4], [12], [13], [14], [15], [16], [17], [18], [19], [20], [21]) or achieve certain artistic effects, (e.g., [2], [22], [23], [24]). We believe that optimizing exposure during capture is valuable as it captures details of important objects; once they are lost due to inappropriate exposure, no amount of post-processing would help to recover the original information. More recently, [25] introduced a new framework for photo retouching that tries to model and learn the photo editing operations using both adversarial and reinforcement learning. Although they also model the exposure adjustment in addition to color and contrast, the problem setting is significant different from ours and their goal is not for real-time personalized exposure control. Table 1 compares three features (ability to be scene-aware, personalized, and real-time for mobile devices) for post-processing techniques, the native camera, and our system.

2.3 High Dynamic Range (HDR) Imagery

The system reported in [26] optimizes image quality through raw burst capture and efficient image processing. While it has an example-based auto-exposure component along the pipeline, the goal is to deliberately underexpose the image to better tailor the subsequent HDR+ fusion. In addition, Hasinoff et al. use low-level features and nearest neighbor search to account for semantics and weighted blending of exposure values from matched examples, whereas we use an end-to-end deep learning system for semantic understanding related to optimal exposure values. A more recent paper [27] proposes a lightweight CNN approach to regress the affine transforms in the bilateral space in order to create edge-preserving filters for real-time image enhancement. While their results look impressive, their system requires pair-wise training data to perform fully supervised learning. By comparison, we apply reinforcement learning due to the lack of directly label data.

Compared with the above approaches, another major differentiator is our work can be treated as a solution to a control problem; we not only require the steady-state exposure value to be optimal, but we also want fast convergence without oscillatory behavior for different scenes and lighting conditions in the viewfinder.

3 System Overview

Our system is depicted in Fig. 3. It is based on a fully convolutional neural network.

3.1 Network Structure

Apart from the input and output, our system consists of three sequentially connected components: (1) backbone network (within the green box), (2) adaptive metering module (within the blue box), and (3) reinforcement module (within the yellow box).

3.1.1 *Backbone Module*

A backbone network can be any network structure as long as it extracts semantically meaningful representations. In the context of real-time exposure control, the backbone network should be designed for both accuracy and run-time performance. Recent studies on image classification (e.g., [28], [29], [30]) show that the backbone could be significantly reduced in terms of the number of layers and model size without losing accuracy. In addition, the state-of-the-art system [31] for object detection shows that when the head is carefully designed, the backbone does not have to be very deep.

3.1.2 *Adaptive Metering Module*

To predict the important regions for exposure prioritization, we introduce an adaptive metering module as a means for semantic-based metering. (The adaptive metering module

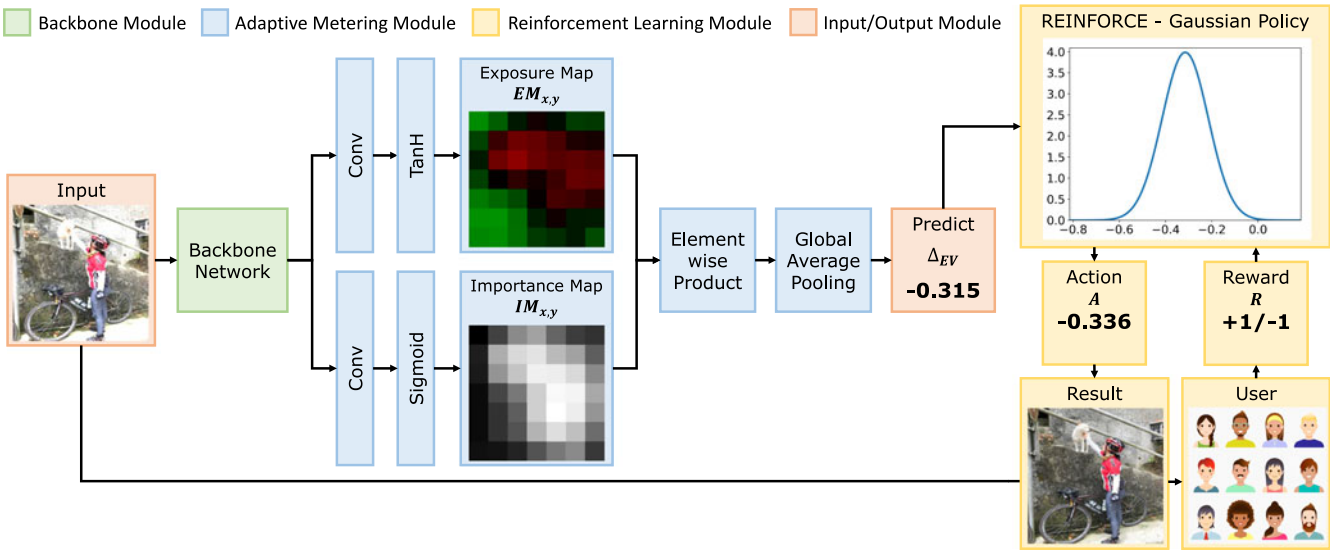


Fig. 3. Overview of our personalized auto-exposure framework. The backbone network is designed as a light-weight deep convolutional neural network which trades-off between accuracy and performance on mobile devices. The adaptive metering module (which contains the importance map IM and exposure map EM) learns prioritization of exposure setting based on scene content. The reinforcement module uses policy gradient with Gaussian policy for online learning; the reward is based on user feedback. Note that in exposure map, we use green, red and black to represent positive, negative and zero Δ_{EV} respectively.

sits on top of the backbone net.) The goal of this module is to learn both an exposure map (denoted as EM_{xy}) and an importance map (denoted as IM_{xy}) in parallel. The exposure and importance maps are element-wise multiplied and then processed through a global average pooling layer to output the final Δ_{EV} (Fig. 3).

It is possible to train a general neural network to learn Δ_{EV} implicitly. However, using a general version may result in convergence and performance issues. Our network design with split EM and IM branches has good convergence behavior. This design is also intuitive in that it mimics how exposure is typically computed in current camera systems (namely, prioritizing image areas and computing overall exposure based on the resulting weight distribution). Our network is trained to prioritize based on scene.

3.1.3 Reinforcement Learning Module

As described in Section 1, it is not practical to ask each user to capture and annotate a large scale bracketing dataset in order to train our system. Instead, it is easier to ask user to label the exposure quality in one of the three categories: "over-exposed","under-exposed" and "well-exposed". So, if an image is labeled as "well-exposed", the output delta EV should be zero; when an image is labeled as "under-exposed" ("over-exposed"), the output delta EV should be positive (negative). Such label information is incomplete and weak, since it does not provide the exact exposure value for a given photo. To overcome these challenges, we propose a third module that is based on reinforcement learning. During run-time, the reinforcement module would no longer be required. We can directly feed the current frame into the network and output the corresponding Δ_{EV}, which is then used to capture the next frame.

3.2 Training Strategy

To apply reinforcement learning to our personalized exposure control system, we have to define the state, action, and reward. Intuitively, the state represents any frame in the viewfinder, regardless their exposure quality, while the action is defined as any Δ_{EV} that lies in the range of possible exposure adjustments. We use a 1D Gaussian parameterized by mean μ and a constant standard deviation σ to sample an action during the training stage. It is reasonable to ask users to provide feedback on exposure quality through a simple selection from over-, under- or well-exposed options. Once acquired, such data are directly used as the reward signal to train the whole system. We believe that this design choice facilitates on-line learning to personalize the exposure model for each user or even each device.

Unfortunately, directly training from scratch would require significant amounts of data and thus result in slow convergence; this is especially cumbersome to generate a personalized model. Instead, we first train the backbone network with an adaptive metering module via supervised learning to replicate a basic exposure control behavior. Since such a pre-trained network can already give us a good initial estimate Δ_{EV}, we only need to fine-tune the reinforcement module by locally refining Δ_{EV} based on the reward signals (as shown in Fig. 2). This significantly simplifies learning while being more likely to end up with a better local minimum. In addition, since fine-tuning is done in an end-to-end training fashion, the reinforcement learning module can more easily customize the learned semantic features embedded in all the layers to handle exposure control. More training is expected to improve robustness and accuracy of the control policy.

4 Exposure Control

Unlike post-processing techniques that are applied to photos *after* they have been captured, our goal is real-time exposure control in the viewfinder mode. In addition to generating correct steady-state exposure values, our system also needs to take into account time performance, convergence speed, and avoidance of oscillatory behavior.

4.1 Exposure-Control as Markov Decision Process

To simplify the model and reduce the computational overhead, we assume that the exposure control process in the viewfinder mode can be treated as a Markov Decision Process (MDP): the new state s^t (next captured frame I_{i+1}) is determined by the current state s (current frame I_i) and action a (exposure adjustment Δ_{EV}). In other words, given s and a, the new state is assumed to be independent of previous states and actions. Therefore, in order to achieve fast convergence and reduce the observation latency in the viewfinder, we want the next captured frame to be as close as possible to the optimal one given the current frame, which requires the exposure adjustment Δ_{EV} to be directly close to optimal one as well. If Δ_{EV} is too large, it may result in overshooting and therefore causing oscillation, while a value that is too small would result in slow convergence and large latency.

We want to learn an exposure prediction function $F(I_i)$ to generate the optimal exposure adjustment Δ_{EV}. As mentioned earlier, there is no scalable way to collect training pairs (I, Δ_{EV}) for supervised learning. To overcome this problem, we use reinforcement learning with a Gaussian policy to learn $F(I_i)$. One advantage of our approach is the ease with which the exposure control model can be personalized for each user. More details are given in Section 5.

4.2 Adaptive Metering

Smart phone cameras generally rely on firmware metering modes (spot, matrix, center weighted, and global) to determine the exposure value. Newer high-end cameras are capable of detecting faces and possibly categorizing simple scenes, and such information is used to optimize exposure. However, to our best knowledge, there are no technical details on how these cameras make use of scene semantics for automatic real-time auto-exposure.

To adjust the exposure, users are given the option to manually tap on the screen to select the region of interest. There is also the option to additionally adjust the luminance (Fig. 1). Given the manual nature of the exposure change, it would be easy to miss good moments to capture (e.g., when capturing a fast moving object of interest such as a bird in flight). It is much more desirable to have the camera automatically produce the optimal exposure at all times—this is the goal of our work.

The hardware metering modes are heuristic approaches for exposure control. For better generalization (beyond faces), we need a systematic approach to adaptively predict a metering importance map based on the scene. Our importance map is a normalized weighting map that is used to display the importance of each spatial region (e.g., car, building, pet, and flower) as learned through examples. Therefore, we propose to develop an adaptive metering module which consists of the exposure adjustment map (how exposure should change locally) and the weighting map (spatial distribution of importance).

These two maps are computed using two different activation functions, namely, *TanH* and *Sigmoid*. The feature map computed using *TanH* is interpreted as the exposure map (EM). Each value (within $[-1.0, 1.0]$) represents the amount of exposure, with positive values indicating over-exposure and negative values indicating under-exposure. On the other hand, the feature map computed using *Sigmoid* is interpreted as the importance map (IM). Each value (within $[0.0, 1.0]$) represents the importance for determining exposure. The exposure and importance maps are element-wise multiplied and then processed through a global average pooling layer to output the final Δ_{EV}:

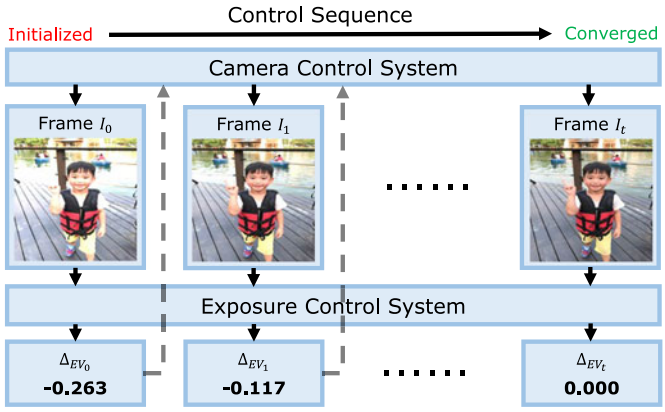


Fig. 4. Flow of our exposure control system during run-time.

$$\Delta_{EV} = \frac{1}{HW} \sum_{x=1}^{H} \sum_{y=1}^{W} EM_{x,y} \times IM_{x,y}. \tag{1}$$

Suppose we have the ground-truth exposure adjustment Y_i for each image I_i. Training the exposure prediction function $F(I_i)$ is straightforward, by minimizing the following loss:

$$\mathcal{L} = \frac{1}{N} \sum_{i=1}^{N} (F(I_i) - Y_i)^2. \tag{2}$$

Because we use a deep convolution neural net to represent function F, minimizing loss $\mathcal{L}$ is done simply through back propagation. However, in practice, as we discussed before, there is no ground-truth for each image. In this paper, we propose to use reinforcement learning with a Gaussian policy to learn the function F (Section 5). In Section 8, we show examples to illustrate that our system learns semantic information for adaptive metering.

As shown in Fig. 4, once the model is deployed, during run-time, our exposure control system will be used for capturing a new frame during the viewfinder mode. The moment the viewfinder is turned on, the hardware captures the first frame denoted as I_0, which is then fed into our exposure prediction network to obtain Δ_{EV_0}. This is added to EV_0 to generate the new exposure value Δ_{EV_1}, which is passed to the hardware control system to capture the new frame. If the predicted Δ_{EV} drops to zero, it means the exposure has converged to a steady state (assuming static camera and scene).

Ideally, the steady state is achieved in one step. In practice, this is not possible due to firmware latency, especially when there is no direct control over the firmware. Empirically, even though we train our system to directly predict the optimal exposure adjustment, it takes 3 to 5 frames for convergence.

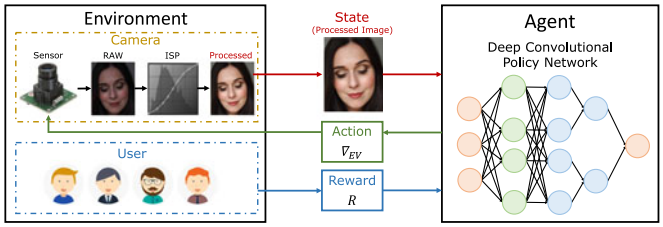


Fig. 5. The MDP components in our control system.

5 Reinforcement Learning

As discussed earlier, there is no scalable way to easily obtain pair-wise training data (consisting of (I, Δ_{EV}) pairs) for supervised learning. As can be seen in the MIT FiveK [1] dataset, exposure and tone adjustment are highly person-dependent. In addition, given differences in the camera firmware and ISP (image signal processing), the temporal response and steady-state appearance vary from one device to another. In practice, it is rare for third parties to have direct control of the camera hardware, and the detailed operations of the ISP (which likely involve nonlinear mappings) are mostly unknown.

Despite these difficulties, we believe it is better to train using processed RGB images rather than raw (Bayer filtered) images. This is because we wish *the final processed images to achieve the correct exposure*, i.e., images generated right after passing through the ISP pipeline. Training using raw images would be problematic because the camera firmware and ISP are typically black boxes; what may work for one camera would likely not work as well for another. We cast exposure control as an MDP (Markov Decision Process) problem and use reinforcement learning to train the system.

5.1 Formulation

In the context of reinforcement learning, an image is a state, with $\mathcal{S}$ representing the set of *states*. $\mathcal{A}$ is the set of possible *actions*, where each action is one specific exposure adjustment value Δ_{EV}. The *environment* consists of two components, namely, camera and user. The user responses are used to provide the *reward* to the image captured by camera while the camera continually receives new exposure adjustment commands. The *agent* is the fully convolutional neural network shown in Fig. 3.

Fig. 5 shows the components in our MDP framework. The goal of the agent is to refine its control policy to adapt to different users and camera firmware based on the rewards. Therefore, even though the camera firmware and ISP are collectively treated as a single black box, we can generate good camera exposure control behavior indirectly through the learned policy network.

At each time step t, the camera acquires an image s_t, which is then judged by a user as being over-exposed, under-exposed, or well-exposed. (Section 5.3 provides more details on how such feedback is converted to reward used to train the whole system.) The database of image-reward pairs is used to then update the policy network through a policy-gradient algorithm with Gaussian policy. We discuss the policy-gradient algorithm in Section 5.2. While not currently demonstrated in this work, our formulation is general enough to customize device behavior.

5.2 Policy Gradient with Gaussian Policy

To train the system, we choose policy gradient rather than Q-learning for two reasons. One reason is policy gradient generally performs better than Q-learning on a large action space. The other reason is policy gradient supports continuous action space while Q-learning usually requires discrete actions.

As a comparison, we partition the exposure adjustment values into small discrete values and then use classification to train the system. However, we encounter the problem of temporal oscillatory behavior. As a result, we choose to directly regress the exposure adjustment value Δ_{EV} as a continuous variable, with values closer to the optimal being penalized less than those farther away. Based on this observation, we use a Gaussian to serve as the policy function to sample each action during training.

Let π_θ (parametrized by θ) be the policy function and $J(\theta)$ the expected total reward; the goal is to maximize $J(\theta)$:

$$\begin{aligned} J(\theta) &= \mathbb{E}_{\pi_\theta}[r] \\ &= \sum_{s\in\mathcal{S}} d(s) \sum_{a\in\mathcal{A}} \pi_\theta(a|s) r(s,a), \end{aligned} \tag{3}$$

where $d(s)$ is the probability distribution for each state, $\pi_\theta(a|s)$ is the conditional probability distribution of each possible action a given the state s, and $r(s,a)$ is the reward after applying action a given state s.

Following REINFORCE [32], we compute the derivative of the objective function $J(\theta)$ with respect to the parameters θ as

$$\begin{aligned} \nabla_\theta J(\theta) &= \nabla_\theta \sum_{s\in\mathcal{S}} d(s) \sum_{a\in\mathcal{A}} \pi_\theta(a|s) r(s,a) \\ &= \sum_{s\in\mathcal{S}} d(s) \sum_{a\in\mathcal{A}} \nabla_\theta \pi_\theta(a|s) r(s,a). \end{aligned} \tag{4}$$

With

$$\nabla_\theta \pi_\theta(a|s) = \pi_\theta(a|s) \nabla_\theta \log \pi_\theta(a|s), \tag{5}$$

Eq. (4) can rewritten as

$$\begin{aligned} \nabla_\theta J(\theta) &= \sum_{s\in\mathcal{S}} d(s) \sum_{a\in\mathcal{A}} \pi_\theta(a|s) \nabla_\theta \log \pi_\theta(a|s) r(s,a) \\ &= \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(a|s) r(s,a)]. \end{aligned} \tag{6}$$

As mentioned before, we use a Gaussian function to model the policy. As such, π_θ can be formulated as

$$\pi_\theta = \mathcal{N}(\mu(s); \sigma^2) = \mathcal{N}(F_\theta(s); \Sigma), \tag{7}$$

where F_θ is the parameterized fully convolutional network that outputs Δ_{EV} for any input image, as discussed in Section 4. For simplicity, we directly take the network output Δ_{EV} as the mean value $\mu(s)$ of our Gaussian policy function, and take a constant value as its variance Σ. We empirically set Σ to 0.1 in our current experiments. During training, we sample the action based on this Gaussian policy function, while at run-time, we directly use the output as the final action to control the exposure adjustments.

Based on Gaussian policy, $\log \pi_\theta(a|s)$ in Eq. (6) can be rewritten as

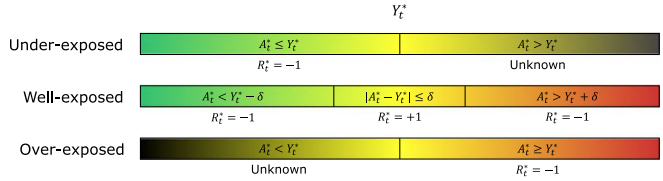


Fig. 6. Strategy to compute reward, with $\delta = 0.1$.

$$\log \pi_\theta(a|s) = -\frac{1}{2}||F_\theta(s) - a||^2_{\Sigma^{-1}} + \text{const}, \tag{8}$$

where $||F_\theta(s) - a||^2_{\Sigma^{-1}} = (F_\theta(s) - a)^T \Sigma^{-1} (F_\theta(s) - a)$.

Then the derivative of $\log \pi_\theta(a|s)$ in Eq. (6) can be further rewritten as

$$\nabla_\theta \log \pi_\theta(a|s) = -\Sigma^{-1}(F_\theta(s) - a)\nabla_\theta F_\theta(s). \tag{9}$$

As a result, the objective function in Eq. (6) is rewritten as

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}[-r(s,a)\Sigma^{-1}(F_\theta(s) - a)\nabla_\theta F_\theta(s)]. \tag{10}$$

To estimate $\nabla_\theta J(\theta)$ in Eq. (10), we sample over the action space using the probability distribution $\pi_\theta(a|s)$:

$$\nabla_\theta J(\theta) \approx -\frac{1}{N}\sum_{i=1}^{N} r(s_i, a_i)\Sigma^{-1}(F_\theta(s_i) - a_i)\nabla_\theta F_\theta(s_i). \tag{11}$$

We describe the reward function $r(s_i, a_i)$ in the next section.

5.3 Reward Setting and Supervised Pre-Training

Earlier, we discussed that it is easier to collect a database of images and their corresponding user feedback on exposure quality (over-exposed, under-exposed, or well-exposed). However, we still need to address two challenges before applying reinforcement learning.

The first challenge is that once our basic system is deployed, before any personalization learning is conducted, we can only collect the feedback signals for those images directly captured by our system. More specifically, in the context of reinforcement learning, for each state image I, we can only obtain the feedback signal for action A, where $A = \Delta_{EV} = 0$. This is because, our exposure control system is generally considered to have converged (i.e., $\Delta_{EV} = 0$) before shutter is pressed.

So far, what we can use to train are only the state-action-reward tuples $(I, A = \Delta_{EV} = 0, R)$. During the personalization training stage, our Gaussian policy requires sampling of different actions $A \neq 0$ for state image I. The second challenge is how to get the corresponding rewards, given that it is not practical to require the user to provide such feedback.

To augment our training set, we synthesize a set of new images from each original image I corresponding to different exposures. Prior to synthesis, we linearize the image intensities by applying a power curve (with $\gamma = 2.2$).[2] We use Adobe Lightroom to synthesize new images I_t from linearized images by synthetically changing Δ_{EV} from $-2.0EV$ to $2.0EV$ with a step of $0.2EV$; note that we reintegrate the nonlinearity to these synthetic images using the power curve with $\gamma = 1.0/2.2$. Each new synthesized image I_t corresponds to one action $Y_t^* = -2.0EV + t * 0.2EV$, i.e., represents the consequence of applying that action to the original image I. For any image I_t (synthetic or real), we need to define its reward R_t^* based on both Y_t^* and the original feedback signals for I (where $I = I_{10}$). Fig. 6 illustrates how to compute the reward R_t^* for any sampled action A_t^*.

Our reward system is based on the intuition that for any "well-exposed" image, a small perturbation of the exposure (i.e., small Δ_{EV}) is acceptable. Eventually, as the magnitude of Δ_{EV} increases, the image will become either "under-exposed" or "over-exposed". For the case that the image is "under-exposed", if we further decrease the exposure will obviously result a more unacceptably "under-exposed" image. The logic can be similarly extended to the "over-exposure" case.

In our system, if the original image is annotated as "well-exposed", then for any other synthetic image I_t and any sampled action A_t^*, as long as A_t^* is very close to Y_t^*, the reward R_t^* should be positive which is shown as the middle region in the second row in Fig. 6. Conversely, if the sampled action is far away from Y_t^*, we should penalize that action through a negative reward (corresponding to the left and right regions in the second row in Fig. 6).

If the original image is annotated as "under-exposed", any sampled action A_t^* decreasing its exposure should make it even more "under-exposed". In this case, the reward R_t^* should be negative, which corresponds to the left region in the first row in Fig. 6. However, for the same "under-exposure" case, it is uncertain as to what the reward should be if the sampled action A_t^* is unexpectedly larger than Y_t^* without further supervision. We ignore such cases in the training which is shown as unknown region in the first row in Fig. 6. (The same situation applies when we deal with images annotated as "over-exposed" which is shown in the third row in Fig. 6.)

Reinforcement learning generally needs lots of data and long convergence time for it to be effective. As shown in Fig. 2, we propose to pre-train the network using supervised learning. On one hand, it is reasonable to assume that any personalized exposure control system should at least perform as well as a native camera; on the other hand, from a data-collection standpoint, supervised pre-training for mimicking native camera can be conducted easily on a relative large scale set, which does not require human annotation.

6 Datasets

In Section 8, we show experimental results to demonstrate the effectiveness of supervised pre-training and our adaptive metering module as well as the performance of reinforcement learning for personalized exposure control. The results are based on four different datasets, namely, Flickr, MIT FiveK, iPhone 7, and Google Pixel datasets. Table 2 shows the number of images in each dataset. We now describe how each dataset is used.

Each dataset is used for a different purpose: Flickr dataset for supervised pre-training, MIT FiveK dataset for personalization, and iPhone 7 and Google Pixel datasets for enhancement of native camera performance of the respective smartphones.

2. Making the image linear is necessary to simulate changes in exposure.

TABLE 2
Datasets Used in Our Experiments

	Flickr	MIT FiveK [1]	iPhone 7	Google Pixel
#Images	11,000	5,000	26,679	7,056
Format	JPEG	RAW	JPEG	JPEG

6.1 Flickr Data for Supervised Pre-Training

Our system requires a dataset that can be used for supervised pre-training in order to mimic the exposure control behavior of native cameras. To this end, we downloaded 100,000 images that were captured by mobile devices from Flickr. To improve image diversity, we remove duplicates and then randomly sample 11,000 of them to serve as the original dataset for supervised pre-training. As discussed in Section 5, to further augment the training set, for each image, we use Lightroom to synthesize 20 images corresponding to ΔEV from $-2.0EV$ to $+2.0EV$ at a step of $0.2EV$. As a result, we end up with a large pair-wise training set, each of which contains a synthetic image and its corresponding Δ_{EV}.

The supervised pre-training is straightforward, where the loss function is simply the euclidean loss. We simplify pre-training by assuming the original image to be well-exposed. Please note that the upper bound performance of the pre-training model matches that of the native camera. In this work, the pre-training model is just a starting point that is to be enhanced, in order to exceed the performance of the native camera app and adapt to personalized preferences.

6.2 MIT FiveK Dataset for Personalization

One approach to test the performance of reinforcement learning for personalization would be to deploy our basic model (after supervised pre-training) to the user's device first and then capture and annotate many images as being under-exposed, well-exposed, or over-exposed. Instead, we choose to leverage the MIT FiveK dataset [1].

The MIT FiveK dataset contains 5000 RAW images, each of which was retouched by five experts using Lightroom. In our work, we consider exposure adjustments only. Fig. 7 shows a histogram that illustrates the variation of expert labels for each image. We count only images with variation below 1.5, which cover 95 percent of the whole dataset. We also sample images for five experts within the largest bin for the variation of 0.255 in our settings; they exhibit significant differences in exposure adjustment. In the dataset, about 75 percent of the images have variation larger than 0.255. This analysis lends credence to our claim that exposure personalization is important under the MIT FiveK dataset.

Given the RAW format, it is straightforward to synthesize 20 new images by varying Δ_{EV} for each original image in Lightroom. However, we use the final processed images as the training data, because ultimately we want the final processed (rather than RAW) image to be personalized.

We randomly sample 500 of original images for testing; for each image, since we know the ground-truth exposure adjustment for each of the five experts, validation is fairly straightforward. We use the other 4,500 images for training. However, unlike supervised learning which directly uses the ground-truth as the signal to train the network, we instead use *indirect* feedback signals in the form of exposure change Δ_{EV}.

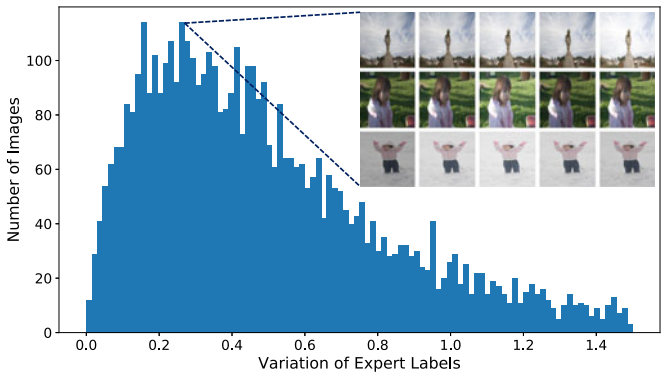


Fig. 7. Histogram for the variation of expert labels and some image samples of five experts within the largest bin.

Δ_{EV} is computed as the difference in exposure between the current (synthetic) exposure and that of ground-truth. The synthetic image will be annotated as well-exposed if its Δ_{EV} is very close to the ground-truth (when the difference is less than 0.1). Otherwise, it will be labeled as either over-exposed or under-exposed based on the deviation. The mapping of reward values is given in Fig. 6. We train a separate model for each expert, and then report their respective performance on the test set.

6.3 iPhone 7 and Google Pixel Datasets for Native Camera Enhancement

To further evaluate the effectiveness of our approach, we conducted offline simulations on two popular smartphones, namely iPhone 7 and Google Pixel. The goal of this experiment is to demonstrate that our system can be used to enhance the exposure quality for native cameras beyond personalization. The motivation behind this is that for most native cameras, the exposure metering and adjustment are not semantic-aware beyond faces. It is highly desirable to be able to meter the exposure based on generic semantic content.

We captured 26,679 images using an iPhone 7 and 7,056 images using a Google Pixel for a variety of scene content. We randomly sample 90 percent of images from each set for training and use the rest for validation to avoid over-fitting. As for training, we hired 5 judges to annotate the exposure quality as being under-exposed, well-exposed, or over-exposed; we then consolidated the annotation results through simple majority voting. We instructed the judges to pay more attention to foreground objects they deem important (such as building, pets, face, humans, cars, and plants) when judging the exposure quality.[3]

To augment the training set, we again use Lightroom to synthesize 9 more images with Δ_{EV} from -1.0 to 1.0 with a step size 0.25 for each original image. We then fine-tune the basic model using all these images as well as the feedback signals through the reinforcement learning module. The

3. Although the final annotation is consolidated by voting and not personalized, it is conceptually possible to train based on individual judges as a means for personalizing the exposure. In our work, this was not done because this experiment is to verify if our approach can be used to enhance the native camera for general users.

fine-tuning is done separately on the iPhone 7 and Google Pixel datasets to generate two different models. We then deployed them to the respective smartphones and performed comparisons with native camera performance through a user study.

7 Implementation Details

Before we report our results, we provide more details on system implementation, namely on network topology, training, and evaluation.

7.1 Network Topology

The network topology of our system consists of the backbone network and adaptive metering module. We use a trimmed SqueezeNet [30] as the backbone network to achieve a good trade-off between accuracy and run-time speed. For SqueezeNet, we keep the layers from the bottom up to "fire7" and discard the other layers. We then add a dropout layer with dropout ratio as 0.5 to reduce the risk of over-fitting.

Our adaptive metering module is designed to predict soft metering regions so as to generalize the default hardware metering mode. Sitting on the top of backbone net, our adaptive metering module branches out into two small subnets; one is used to predict the importance map while the other predicts the local exposure adjustment map. We use a 1x1 convolution layer followed by *TanH* and *Sigmoid* activation functions, respectively (Fig. 3). These two maps are then element-wise multiplied before being applied to the global average pooling layer for the final result.

7.2 Training, Run-Time Details and Evaluation Metric

We use PyTorch to train our system. The input images are all resized to 128×128, with a batch size of 128. The labels are normalized within $[-1.0, 1.0]$. All the networks were trained for 35 epochs with a stepped learning rate (denoted as r) policy; r is reduced by half every 15 epochs. For supervised pre-training, we use the SGD solver with the momentum of 0.9, weight decay of 0.0002, and learning rate of 0.003. For reinforcement learning, we use the Adam solver with an initial learning rate of 0.0001.

During run-time, once we get the new EV to apply, we decompose EV into ISO(s) and Exposure duration time(t) according to the equation $EV = \log_2 t * s$, as defined in Appendix, available in the online supplemental material. To avoid noises, we always prefer a small ISO whenever possible. However, when the duration time exceeds the hardware limit, we will keep increasing ISO to the next level until we find a duration time that supported by the hardware. Note that, the duration time is equivalent to another commonly used concept "Shutter Speed" in here.

Since we are dealing with a regression task, we choose the Mean Absolutely Error (*MAE*) as the evaluation metric:

$$MAE = \sum_{i=1}^{N} |\Delta_{EV_i} - Y_i|, \tag{12}$$

where N is the number of testing images, and EV_i and Y_i are the prediction and ground-truth labels of image i, respectively.

TABLE 3
Testing MAE Comparison between Reinforcement Learning from ImageNet (RL) and Reinforcement Learning with Our Pre-Training (RL+PT)

	Mean	ExpA	ExpB	ExpC	ExpD	ExpE
RL	0.2457	0.3160	0.2917	0.3526	0.3069	0.3144
RL+PT	**0.2019**	**0.2936**	**0.2349**	**0.2725**	**0.2491**	**0.2518**

8 Experimental Results

In this section, we first show how our system performed on exposure personalization using the MIT FiveK dataset. Results on this dataset demonstrate that our system is able to learn exposure personalization from five experts individually. For real-time deployment, it is ideal to conduct an extensive deployment of on-line learning for personalization on devices. However, it requires a significant amount of engineering effort to develop such a system as well as a sufficiently large number of user feedback signals for empirical validation. This type of evaluation is hard to be conclusive due to the lack of ground truth. By comparison, the MIT FiveK dataset allows for a more controlled validation process via known ground truth. In Section 8.2, we evaluate the ability of our system to enhance native camera exposure on two popular smartphones through a user study.

8.1 Results on MIT FiveK

As mentioned earlier, the MIT FiveK dataset is used as proof-of-concept for personalization-based reinforcement learning. It is also used for evaluating different parts of our system.

8.1.1 Evaluation of Supervised Pre-Training

Without supervised pre-training, one can choose to fine-tune the whole network by pre-training based on the classification task using ImageNet. However, since the adaptive metering module is not used in any previous classification network, we can only borrow the pre-trained weights in the backbone network from those pre-trained on ImageNet (such as the pre-trained SqueezeNet used in our experiments). All the other new layers are initialized with standard zero mean and 0.01 standard deviation Gaussian function.

Table 3 shows the effect of fine-tuning from ImageNet and fine-tuning from our supervised pre-trained weights for different personalized models. The MAE numbers support our design decision to use supervised pre-training, since they are reduced for all five personalized models. This is likely because the pre-training step allows the model to learn more relevant representations tailored for exposure control, and not generic features used for image classification.

8.1.2 Evaluation of Adaptive Metering Module

The goal of our proposed adaptive metering module is to generalize the heuristic hardware metering options by adaptively predicting the importance map for each image. The baseline approach is simply outputting the exposure map only, treating each local region with same weight for aggregating the final exposure adjustment. As comparison, we remove the importance branch and the subsequent

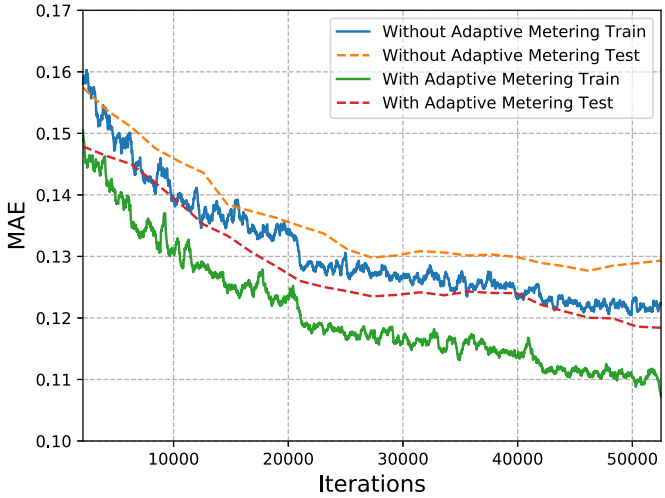


Fig. 8. Training and testing performance graphs that show the effect of using the adaptive metering module.

element-wise multiple layer and trained a baseline model. Fig. 8 shows that during the pre-training stage, adding the adaptive metering module significantly reduces the regression error for both the training and testing. This appears to indicate that the importance branch learns some meaningful maps for prioritization of certain local areas at the expense of others.

To further illustrate the effectiveness of the adaptive metering module, we show both exposure map and importance map for the testing images under different scene content and lighting. For visualization, the exposure map is shown in pseudo-color while the intensities of the importance map is linearly converted to $[0, 255]$. The red regions in the exposure map represent over-exposed areas (where exposure needs to be reduced) while the green regions represent under-exposed areas (where exposure needs to be increased). As for the importance map, the brighter the region is, the higher the priority for exposure adjustment.

Fig. 9 shows the exposure and importance maps for representative images. Our method appears to be able to predict semantically reasonable exposures as shown in the second row. Notice that for image (a), where there is no discernible local foreground of interest, the maps are close to being uniform. For the other images with local foregrounds of interest, the maps are substantially more unevenly distributed. Visually, they correlate with objects of interest. In particular, for image (f), our system deems the building to be over-exposed; there are more details on the facade after exposure adjustment.

Our network is based on FCN (fully convolutional networks), which can be easily modified to accommodate larger input and output feature maps that are capable of storing more details. Fig. 10 shows the exposure and importance maps associated with input sizes of 128×128, 256×256, and 512×512. Notice the increasingly better detail with higher input size; this is evidence that our system is able to learn scene semantics.

Interestingly, even without direct supervision, our end-to-end system is capable of learning the latent importance maps. This may be attributed to our local-to-global aggregation design through both the element-wise multiple layer and global average pooling layer. This suggests that an adaptive weighted importance map is effective in generalizing the heuristic hardware metering modes.

8.1.3 *Evaluation of Reinforcement Learning*

Table 4 compares the performance of fully supervised learning and reinforcement learning using the MIT FiveK database for training and testing. Fully supervised learning is

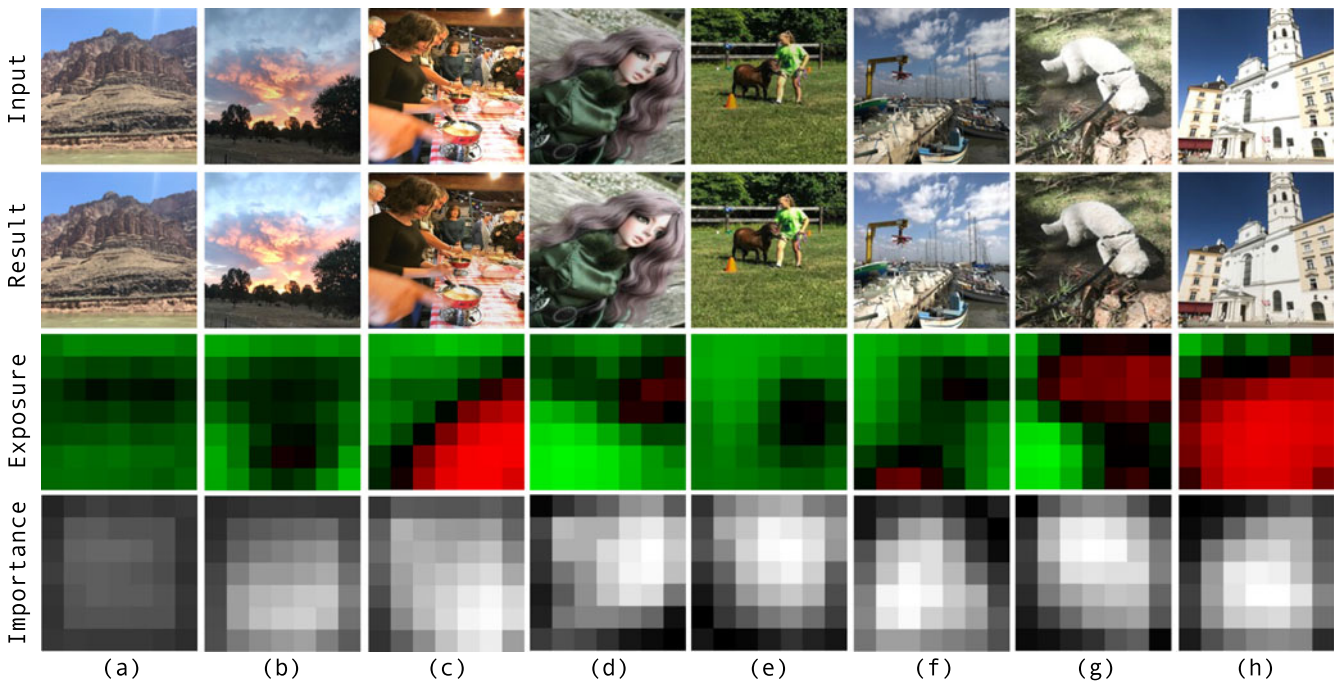


Fig. 9. Exposure and importance maps on representative images. For image (a), the foreground occupies most of the scene; here, the importance map is close to being uniform. For image (b-c), there are no objects with obvious semantics, for such cases, the importance maps are focusing on the region exposed incorrectly. For image(d-h), objects with obvious semantics can be found in the scene which the importance map should focus on such regions accordingly.

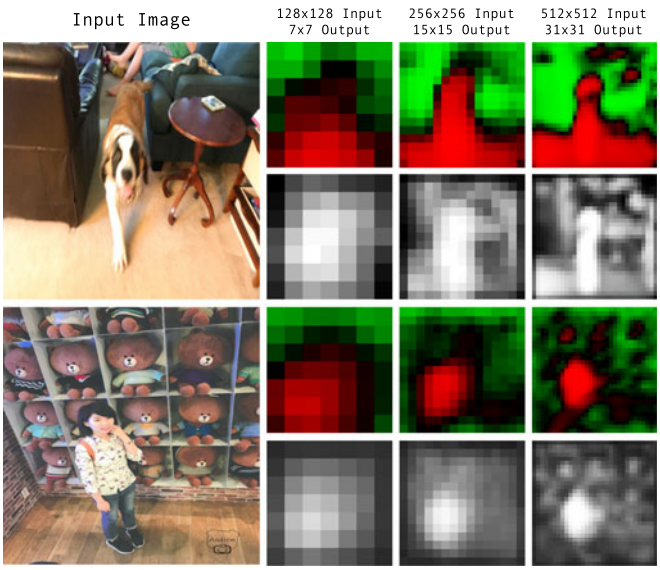


Fig. 10. Exposure and importance maps for three different input sizes (128×128, 256×256, and 512×512). The corresponding output feature map sizes are 7×7, 15×15, and 31×31.

TABLE 4
Testing MAE for Fully Supervised Learning (FSL) and Reinforcement Learning (RL)

	Mean	ExpA	ExpB	ExpC	ExpD	ExpE
FSL	0.1926	0.2770	0.2272	0.2688	0.2412	0.2450
RL	0.2019	0.2936	0.2349	0.2725	0.2491	0.2518

TABLE 5
Testing MAE Matrix Showing the Effect of Applying a Trained Model on Testing Data with Different Ground-Truths (Depending on the Expert)

		Test				
		ExpA	ExpB	ExpC	ExpD	ExpE
Train	Mean	0.3862	0.2483	0.3355	0.2702	0.2919
	ExpA	**0.2936**	0.3659	0.4897	0.3952	0.4433
	ExpB	0.4143	**0.2349**	0.3037	0.2588	0.2731
	ExpC	0.5057	0.2816	**0.2725**	0.2703	0.2706
	ExpD	0.4008	0.2506	0.3151	**0.2491**	0.2791
	ExpE	0.4679	0.2588	0.2881	0.2645	**0.2518**

done by directly regressing Δ_{EV} using ground-truth data, and its numbers can be considered as upper-bound performance. The numbers for reinforcement learning are very competitive even though it makes use of coarse-grained feedback signals ("over-exposed","under-exposed," or "well-exposed").

In addition to the five personalized models available from the MIT FiveK database, we also compute the average Δ_{EV} of all the five ground-truth labels for each image. We train a "mean" model that to represent the preference of the average person. We then use these six trained models to run the test set for each expert (same testing images but different Δ_{EV} labels). Unsurprisingly, results shown in Table 5 indicate that the personalized model trained for each expert generates the smallest error for that same expert. This demonstrates the necessity for personalization of exposure adjustment.

TABLE 6
Accuracy of Prediction from One Expert Given the Ground Truth (Trained) from Another Expert

		Percentage of Images				
		ExpA	ExpB	ExpC	ExpD	ExpE
Train	ExpA	**67%**	13%	2%	13%	4%
	ExpB	10%	**42%**	13%	19%	16%
	ExpC	3%	16%	**52%**	13%	16%
	ExpD	14%	18%	14%	**38%**	15%
	ExpE	3%	19%	15%	14%	**48%**

Each number in the table is the percentage of images which the predictions of one model is nearest to the ground-truth Δ_{EV}. For example, only 2 percent of the images predicted by ExpC are closest to the ground truth from ExpA.

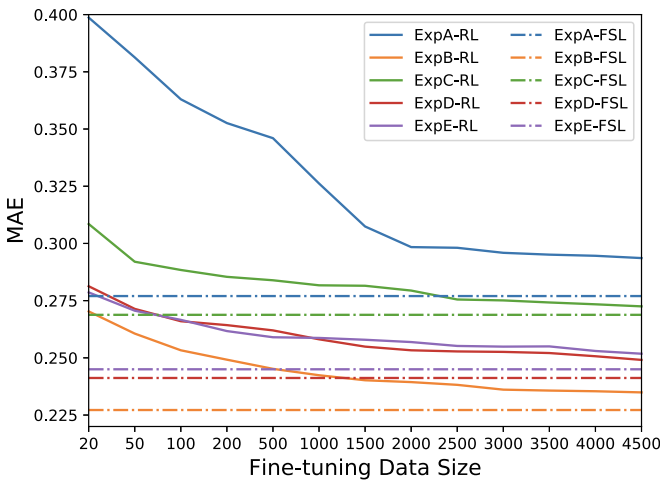


Fig. 11. Testing MAE as a function of training data size. The solid lines represent performance of our reinforcement learning based method while the dotted lines are the reference (lower-bound) performance based on fully supervised learning.

To further show that our model is able to learn personalized exposure preferences, we run our model for each expert on the test set from the MIT FiveK dataset. We compute the percentage of images which Δ_{EV} predicted by our model is nearest to the ground-truth Δ_{EV} (compared to the ground truth for the other four experts); the results are shown in Table 6. There are two interesting observations. First, for any row or column, the highest percentage is where the prediction and ground truth are from the same expert. Second, there is significant variation in percentages across the table. These observations support the existence of exposure preference in the database and show that our method is able to reasonably capture these preferences.

Fig. 12 shows results for two experts (A and C) who have significant differences in exposure preferences. For each example, we show close-ups of two regions to highlight the difference in exposure preference in the third and sixth columns. The different exposure preferences are reflected in the different exposure and importance maps shown in the fourth and seventh columns.

We also evaluate the performance of reinforcement learning with respect to the size of training data. Fig. 11 shows that the regression error monotonically decreases with increase of training data size, which is expected. Each solid line represents the testing performance of each personalized model trained by reinforcement learning, while each dotted line

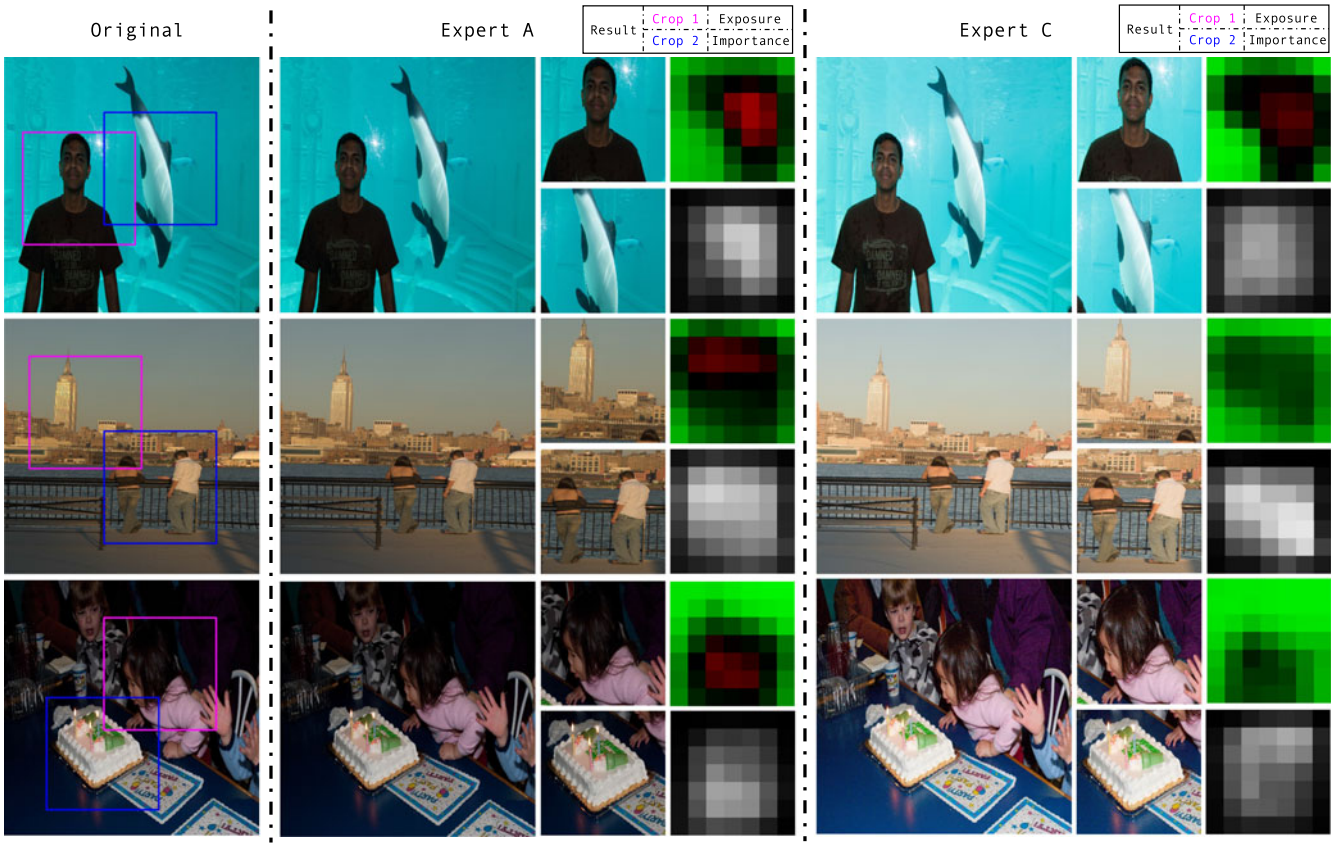


Fig. 12. Exposure and importance map visualization of testing examples for experts A and C. Note that for each example, the exposure and importance maps are for the entire image.

denotes the corresponding testing performance of the reference model that was trained by fully supervised learning with all 4,500 training images. Each dotted horizontal line represents upper bound performance. The errors seem to drop more quickly when the images are fewer than 200 for experts A, B, C, D, and E. Also, after increasing the number of images to a few hundred, the performance of our RL-based method approximates that of fully supervised learning.

8.1.4 Comparison with other Methods

Table 7 compares our performance in terms of MSE (mean squared error) against those for the systems of Hwang et al. 2012 [33] and Yan et al. 2016 [2] (both of which are post-processing techniques with spatially-varying operations, and both report MSE as quantitative objective evaluation). In particular, Yan et al. [2] use a set of hand-crafted and deep learning features to learn spatially-variant local mapping functions and predict the final L channel value of each pixel in CIE*LAB space. In contrast, our method is more light-weight and can be considered as a global approach. It is noted that MSE is not a good metric for perceptual quality [25], [33]. Regardless, our results are still competitive, with the advantages of our system being geared for personalization and allowing real-time capture. In addition, our system is trained without the need of full supervision, which is more scalable for practical deployment.

TABLE 7
Comparison of MSE Errors Obtained with Our Method (RL) and Previous Methods (Fully Supervised Learning) on the MIT-Adobe FiveK Database

Method	Ran.250 (L,a,b)	H.50 (L,a,b)
Hwang et al. 2012 [33]	15.01	12.03
Ours (RL)	13.51	11.21
Yan et al. 2016 [2]	9.85	8.36

8.2 Results for Exposure Control Enhancement

As proof-of-concept for improving exposure control though semantic-awareness (beyond faces), we trained two different models, one for iPhone 7 and the other for Google Pixel. The training was done using their own respective datasets.

We then developed a test app that is deployed to both iPhone 7 and Google Pixel. We can control the exposure only through API calls and not directly through the firmware. Not surprisingly, we noticed there is latency in the API calls; the amount of latency is 3-5 frames. Our results show that even though the models were trained using synthetic datasets, our exposure control system is still able to achieve good trade-off among the steadily state, temporal oscillation, and run-time speed. On iPhone 7, our model takes 17 ms to predict the exposure; on Google Pixel, it takes around 27 ms.

For performance comparison, we mounted two phones side-by-side on rig and captured pictures simultaneously. To address the view difference issue, we followed the approach proposed by [22], [34]) to first align two images by local features and then crop for best comparison.

Fig. 13 shows a few comparisons with iPhone 7 and Google Pixel. When the face is not frontal, iPhone 7 typically fails to detect it, leaving it under-exposed in a backlit scene.

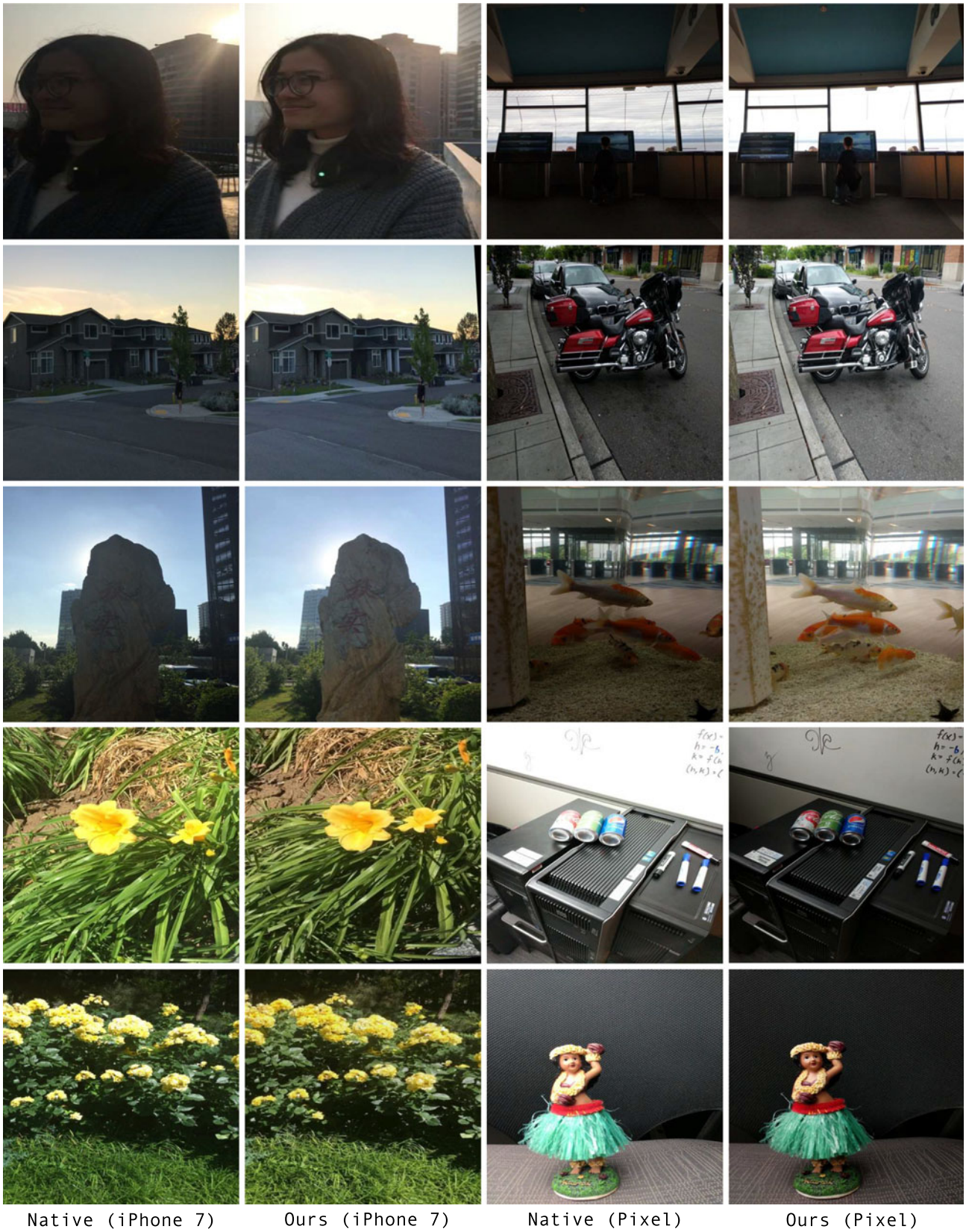


Fig. 13. Side-by-side capture of various scenes using the native camera apps of iPhone 7 and Google Pixel and ours. Our system tend to better expose foreground objects.

Our model is able to better expose foreground subjects at the expense of slightly over-exposing the background. A similar assessment can be made for Google Pixel (e.g., the scene with the person with the back facing the camera). The native camera apps for both iPhone 7 and Google Pixel tend to optimize over the whole image, and as a result, under-expose the foreground for scenes with bright backgrounds. Our system is able to prioritize exposure for non-person

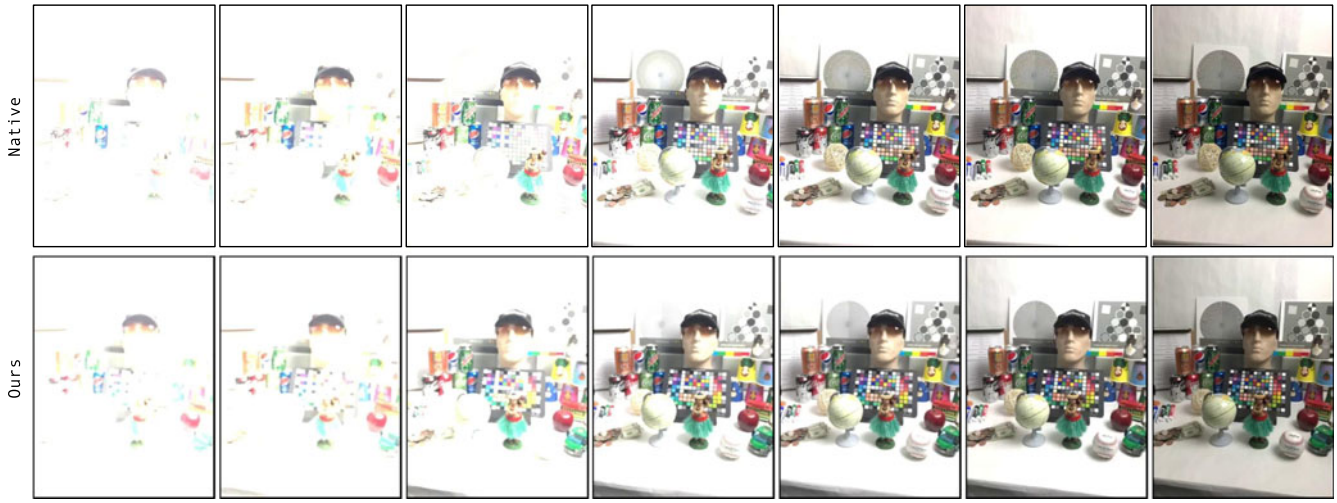


Fig. 14. Comparison of exposure dynamics between our model and the iPhone 7 native camera (both in response to a sudden increase in lighting).

objects as well, e.g., the building and motorcycle in the second and third rows. For the examples in the last two rows, our system instead reduces the exposure for better detail. This is in comparison to the over-exposed versions from iPhone 7 and Google Pixel's native camera app. These examples illustrate the ability of our model to learn scene semantic representations for better exposure control.

Fig. 14 shows snapshots that represent the response of our system to abrupt lighting changes. The dynamics of exposure varying mimics the native camera app well. Please refer to supplemental videos, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TVCG.2018.2865555 for more examples of how our deployed real-time exposure control system responds in real-time to camera motion in different scenes and changing lighting conditions.

8.2.1 User Study

We performed a user study to compare performance of our system with that of the native camera app. To do this, we first captured a 100 pair of images using two iPhone 7s placed side-by-side on many different scenes; one is running the native camera while the other is running our app. Similarly, we captured another 100 pairs of images for side-by-side comparison on two Google Pixel phones. In the user study, a subject is shown a pair of images, one being our result and the other from the native camera app. The images are randomly arranged. In this A-B test, the subject is asked to select amongst the choices "Left is better", "Right is better," or "Equally good."

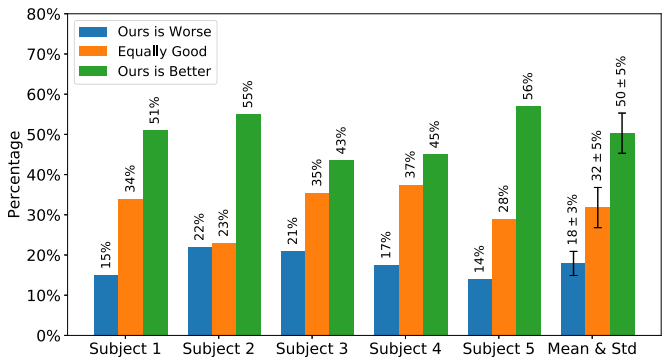


Fig. 15. Histogram showing preferences for each of the five subjects as well as the mean result with standard deviation.

There are five subjects in the study, with each looking at the same 200 image pairs. The results, shown in Fig. 15, validate our design decisions for a reinforcement learning based exposure control system. These results are also significant in that the subjects in the user study were not involved in annotating the training data.

To further analyze the variation of five subjects for each image pair, we evaluate the consistency of five subjects for each image pair in Table 8. In this table, $N_{consistency}$ is used to denote the maximum number of subjects (up to five) having the same label for each image pair. For example, $N_{consistency} = 4$ means four subjects having the same opinion for an image pair. 81% of the image pairs have at least four subjects agreeing with each other.

We also evaluate the variation of five subjects through Kendall's coefficient of concordance[35] which denotes as W here. In order to show the result of all 200 image pairs, we take an average of W_i that computes from each image pair respectively. The final averaged $W = 0.67$ which shows strongly concordancy among five subjects.

9 Discussion

In this section, we discuss a number of issues on system design and feature extensions.

9.1 Training on HDR or DSLR Images

In principle, our proposed system can be trained on HDR images, in which case, f has to take a RAW image as input,

TABLE 8
The Percentage of Image Pairs with Different $N_{consistency}$

$N_{consistency}$	Percentage
5	55% (110/200)
4	26% (52/200)
3	18% (36/200)
≤ 2	1% (2/200)

with the corresponding exposure quality label judged based its the processed image. However, in practice, accessing HDR and RAW images is challenging on most mobile devices, so collecting the RAW (HDR) images is not as easy as collecting JPEG images for training data. At runtime, the same type of image would need to be accessed in real-time; even if the RAW image is accessible, it would be done at the expense of performing the ISP (image signal processing) pipeline in software, which would be unacceptably slow.

For the DSLR images, since each device's ISP pipeline is different, for the same RAW image and equivalent exposure settings, the output is very likely to be different. As a result, learning from RAW data from one device and applying it to another (in our case mobile cameras) would be problematic.

9.2 Dealing with HDR Content

The typical scene is inherently of high dynamic range, and it is better to capture HDR if possible. This is especially true for backlit or frontlit scenes. It is certainly possible to incorporate some of our ideas to select the bracketing exposures for optimal semantic-aware HDR capture. Given that sequential capture is required for HDR, the problem is significantly more challenging due to possible camera and/or scene motion. HDR also requires post-processing, which is outside the scope of our work.

9.3 Camera 3A Integration

In this work, we wish to demonstrate the most obvious feature of the phone in live viewfinder mode, namely exposure. In practice, auto-exposure works in conjunction with auto-focus and auto-white balance, collectively termed the *camera 3A*. The amount of training data that are required to account for 3A would be substantially larger, but the principle would be similar. An end-to-end deep learning based 3A system (given the recent advances in color constancy [36]) is future work.

9.4 Incorporate with User Input

Our system is designed to predict the optimal exposure to obviate the need for user manual interaction, such as "Tap-and-Tweak". It is possible to add manual overrides through manual area specification through lassoing or tapping on an image area. In principle, our system should still work by replacing the learned importance map with a heuristically defined importance map based on user selection.

10 Conclusions

In this paper, we propose a new semantic-aware exposure control system based on reinforcement learning. Our system consists of three major components: (1) supervised pre-training to mimic the native camera's control behavior, (2) an adaptive metering model, and (3) a reinforcement learning model that can be trained for both personalization and enhancing the native camera capabilities. We conducted extensive experiments on MIT FiveK and our own captured datasets. The encouraging results validate our system design. It would be useful to be able to decide on the need to change exposure time or gain (ISO). In particular, accounting for scene motion is important. As an example, in sports photography, it is common practice to increase shutter speed (i.e., reduce exposure time) while increasing the ISO. Another issue is that of speeding up performance; while this can be accomplished through model compression (e.g., by pruning [37] and binarization [38]), more work is required to ensure that accuracy is not too severely compromised.

Acknowledgments

This work was partially done when the first author was an intern at Microsoft AI&R.

References

[1] V. Bychkovsky, S. Paris, E. Chan, and F. Durand, "Learning photographic global tonal adjustment with a database of input/output image pairs," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2011, pp. 97–104.

[2] Z. Yan, H. Zhang, B. Wang, S. Paris, and Y. Yu, "Automatic photo adjustment using deep neural networks," *ACM Trans. Graph.*, vol. 35, no. 2, pp. 11:1–11:15, Feb. 2016.

[3] J. C. Caicedo, A. Kapoor, and S. B. Kang, "Collaborative personalization of image enhancement," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2011, pp. 249–256.

[4] S. B. Kang, A. Kapoor, and D. Lischinski, "Personalization of image enhancement," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2010, pp. 1799–1806.

[5] R. Brunner, "Automatic exposure control based on multiple regions," Dec. 6, 2012, US Patent App. 13/151,165. [Online]. Available: http://www.google.la/patents/US20120307107

[6] N. Kehtarnavaz, H.-J. Oh, I. Shidate, Y. F. Yoo, and R. Taluri, "New approach to auto-white-balancing and auto-exposure for digital still cameras," *Sensors Camera Syst. Sci. Ind. Digit. Photography Appl. III*, vol. 4669, pp. 268–277, 2002.

[7] N. Sampat, S. Venkataraman, T. Yeh, and R. L. Kremens, "System implications of implementing auto-exposure on consumer digital cameras," *Sensors Cameras Appl. Digit. Photography*, vol. 3650, pp. 100–108, 1999.

[8] Z. Guo, Y. Gu, and H. Yao, "Auto-exposure algorithm based on luminance histogram and region segmentation," *Appl. Mech. Mater.*, vol. 543-547, pp. 2278–2282, 2014.

[9] E. W. Jin, S. Lin, and D. Dharumalingam, "Face detection assisted auto exposure: supporting evidence from a psychophysical study," *Digit. Photography VI*, vol. 7537, 2010, Art. no. 75370K.

[10] Y. Jeon, H. Park, Y. Yoon, and Y. Baek, "Design and implementation of multi exposure smart vehicular camera applying auto exposure control algorithm based on region of interest," *J. Korean Inst. Commun. Inf. Sci.*, vol. 42, pp. 181–192, Jan. 2017.

[11] M. Yang, Y. Wu, J. Crenshaw, B. Augustine, and R. Mareachen, "Face detection for automatic exposure control in handheld camera," in *Proc. IEEE Int. Conf. Comput. Vis. Syst.*, Jan. 2006, pp. 17–17.

[12] N. Joshi, W. Matusik, E. H. Adelson, and D. J. Kriegman, "Personal photo enhancement using example images," *ACM Trans. Graph.*, vol. 29, no. 2, pp. 12:1–12:15, Apr. 2010.

[13] M. Belen and G. Diego, "Content-aware reverse tone mapping," in *Proc. Int. Conf. Artif. Intell.: Technol. Appl.*, 2016, pp. 235–238.

[14] F. Berthouzoz, W. Li, M. Dontcheva, and M. Agrawala, "A framework for content-adaptive photo manipulation macros: Application to face, landscape, and global manipulations," *ACM Trans. Graph.*, vol. 30, no. 5, pp. 120:1–120:14, Oct.

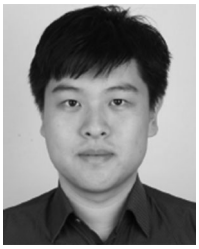
[15] L. Kaufman, D. Lischinski, and M. Werman, "Content-aware automatic photo enhancement," in *Computer Graphics Forum*, vol. 31, no. 8. Hoboken, NJ, USA: Wiley Online Library, 2012, pp. 2528–2540.

[16] D. Lischinski, Z. Farbman, M. Uyttendaele, and R. Szeliski, "Interactive local adjustment of tonal values," *ACM Trans. Graph.*, vol. 25, no. 3, 2006, pp. 646–653.

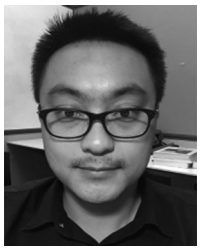
[17] Z. Lou, T. Gevers, N. Hu, and M. P. Lucassen, "Color constancy by deep learning," in *Proc. British Mach. Vis. Conf.*, 2015, pp. 76.1–76.12.

[18] D. Guo, Y. Cheng, S. Zhuo, and T. Sim, "Correcting over-exposure in photographs," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2010, pp. 515–521.

[19] D. Xu, C. Doutre, and P. Nasiopoulos, "Correction of clipped pixels in color images," *IEEE Trans. Vis. Comput. Graph.*, vol. 17, no. 3, pp. 333–344, Mar. 2011.
[20] G. Eilertsen, J. Kronander, G. Denes, R. K. Mantiuk, and J. Unger, "HDR image reconstruction from a single exposure using deep cnns," *ACM Trans. Graph.*, vol. 36, no. 6, 2017, Art. no. 178.
[21] Y. Endo, Y. Kanamori, and J. Mitani, "Deep reverse tone mapping," *ACM Trans. Graph.*, vol. 36, no. 6, 2017, Art. no. 177.
[22] B. Wang, Y. Yu, and Y.-Q. Xu, "Example-based image color and tone style enhancement," *ACM Trans. Graph.*, vol. 30, no. 4, 2011, Art. no. 64.
[23] Y. HaCohen, E. Shechtman, D. B. Goldman, and D. Lischinski, "Non-rigid dense correspondence with applications for image enhancement," *ACM Trans. Graph.*, vol. 30, no. 4, 2011, Art. no. 70.
[24] L. A. Gatys, A. S. Ecker, and M. Bethge, "Image style transfer using convolutional neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 2414–2423.
[25] Y. Hu, H. He, C. Xu, B. Wang, and S. Lin, "Exposure: A white-box photo post-processing framework," *ACM Trans. Graph.*, vol. 37, no. 2, 2018, Art. no. 26.
[26] S. W. Hasinoff, D. Sharlet, R. Geiss, A. Adams, J. T. Barron, F. Kainz, J. Chen, and M. Levoy, "Burst photography for high dynamic range and low-light imaging on mobile cameras," *ACM Trans. Graph.*, vol. 35, no. 6, 2016, Art. no. 192.
[27] M. Gharbi, J. Chen, J. T. Barron, S. W. Hasinoff, and F. Durand, "Deep bilateral learning for real-time image enhancement," *ACM Trans. Graph.*, vol. 36, no. 6, 2017, Art. no. 118.
[28] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv: 1704.04861*, 2017.
[29] J. Redmon, "Darknet: Open source neural networks in C," 2013–2016. [Online]. Available: http://pjreddie.com/darknet/
[30] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and < 0.5MB model size," *arXiv:1602.07360*, 2016.
[31] Z. Li, C. Peng, G. Yu, X. Zhang, Y. Deng, and J. Sun, "Light-head r-cnn: In defense of two-stage object detector," *arXiv: 1711.07264*, 2017.
[32] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Mach. Learn.*, vol. 8, no. 3/4, pp. 229–256, 1992.
[33] S. J. Hwang, A. Kapoor, and S. B. Kang, "Context-based automatic local image enhancement," in *Proc. Eur. Conf. Comput. Vis.*, 2012, pp. 569–582.
[34] A. Ignatov, N. Kobyshev, R. Timofte, K. Vanhoey, and L. Van Gool, "Dslr-quality photos on mobile devices with deep convolutional networks," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 3277–3285.
[35] Kendall's coefficient of concordance. [Online]. Available: https://en.wikipedia.org/wiki/Kendall
[36] Y. Hu, B. Wang, and S. Lin, "FC4: Fully convolution color constancy with weighted average pooling," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jul. 2017, pp. 4085–4094.
[37] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," *Int. Conf. Learn. Representations*, 2016.
[38] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "Xnor-net: Imagenet classification using binary convolutional neural networks," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 525–542.



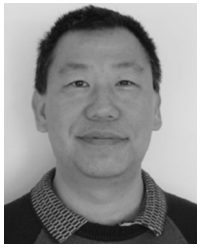
Huan Yang received the BS degree in computer science from Shanghai Jiao Tong University, China, in 2014 . He is currently working toward the PhD degree in computer science at Shanghai Jiao Tong University. His current research interests include deep learning, reinforcement learning, computer vision, image processing, real-time video processing, and image photography.



Baoyuan Wang received the BS and PhD degrees in computer science and engineer from Zhejiang University, in 2007 and 2012, respectively. He is currently a principal researcher with Microsoft AI team, and his research interests include deep learning based computational photography for intelligent capture, intelligent action as well as novel content generation. He has shipped several important technologies to Microsoft Pix, Xbox One, Swift-Key and other Microsoft products. From 2006 to 2007, he was an engineer intern with Infosys, Bangalore, India. From 2009 to 2012, he was a research intern with Microsoft Research Asia, working on data-driven methods for computational photography. He was a lead researcher with Microsoft Research Asia from 2012 to 2015.

Noranart Vesdapunt received the BEng degree in computer engineering from Chulalongkorn University and the MS degree in computer vision from Carnegie Mellon University. He is a software engineer with Microsoft Research. His research interests include computational photography, visual recognition, video analytics, and deep learning on mobile devices.

Minyi Guo received the BS and ME degrees in computer science from Nanjing University, China, in 1982 and 1986, respectively, and the PhD degree in information science from the University of Tsukuba, Japan, in 1998. From 1998 to 2000, he had been a research associate with NEC Soft, Ltd. Japan. He was a visiting professor with the Department of Computer Science, Georgia Institute of Technology. He was a full professor with the University of Aizu, Japan, and is the head of the Department of Computer Science and Engineering, Shanghai Jiao Tong University, China. His research interests include automatic parallelization and data-parallel languages, bioinformatics, compiler optimization, high-performance computing, deep learning, and pervasive computing. He is a fellow of the IEEE.



Sing Bing Kang received the PhD degree in robotics from Carnegie Mellon University, Pittsburgh, in 1994. He is a principal researcher with Microsoft Corporation, and his research interests include image and video enhancement, and image-based modeling. He has coedited two books (Panoramic Vision and Emerging Topics in Computer Vision) and coauthored two books (Image-Based Rendering and Image-Based Modeling of Plants and Trees). On the community service front, he has served as an area chair for the major computer vision conferences and as papers committee member for SIGGRAPH and SIGGRAPH Asia. He was program chair for ACCV 2007 and CVPR 2009, and was an associate editor-in-chief for the *IEEE Transactions on Pattern Analysis and Machine Intelligence* from 2010-2014. He is a fellow of the IEEE.