# Recurrent Neural Networks with Intra-Frame Iterations for Video Deblurring

Seungjun Nah Sanghyun Son Kyoung Mu Lee

Department of ECE, ASRI, Seoul National University, Seoul, Korea
seungjun.nah@gmail.com, {thstkdgus35, kyoungmu}@snu.ac.kr

## Abstract

*Recurrent neural networks (RNNs) are widely used for sequential data processing. Recent state-of-the-art video deblurring methods bank on convolutional recurrent neural network architectures to exploit the temporal relationship between neighboring frames. In this work, we aim to improve the accuracy of recurrent models by adapting the hidden states transferred from past frames to the frame being processed so that the relations between video frames could be better used. We iteratively update the hidden state via reusing RNN cell parameters before predicting an output deblurred frame. Since we use existing parameters to update the hidden state, our method improves accuracy without additional modules. As the architecture remains the same regardless of iteration number, fewer iteration models can be considered as a partial computational path of the models with more iterations. To take advantage of this property, we employ a stochastic method to optimize our iterative models better. At training time, we randomly choose the iteration number on the fly and apply a regularization loss that favors less computation unless there are considerable reconstruction gains. We show that our method exhibits state-of-the-art video deblurring performance while operating in real-time speed.*

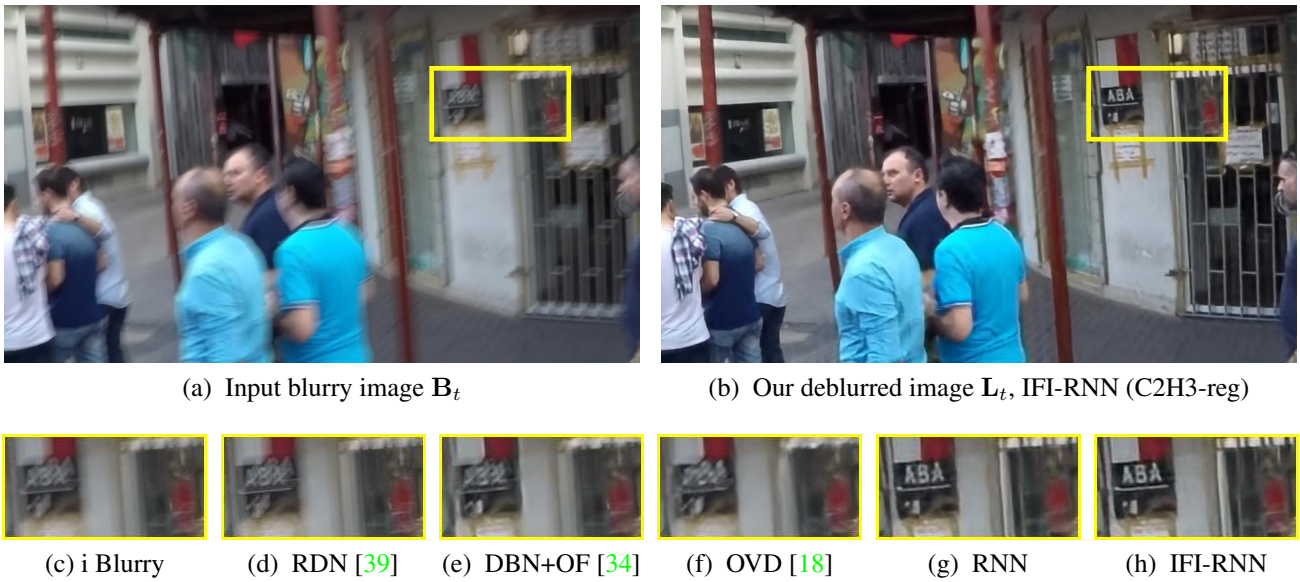
## 1. Introduction

Videos captured in dynamic environments typically contain blurs where the relative motions occur. Hand-held cameras are more likely to be shaken during shooting, and fast-moving objects can exist at any time in the scene. Especially, a long exposure time is required in the low-light environment or for the widely used mobile cameras. Since the motions during the exposure time directly cause the blurs in captured frames, blurs are among the most common degradation artifacts in videos. Those motions of various objects or a camera give rise to spatially non-uniform blurs that make the deblurring problem challenging. In real-world scenarios, the problem becomes more challenging since the sharp video frames should be recovered without knowing the information of the spatially varying motions or the local blur kernels. Furthermore, abrupt motions often cause severe blurs with diverse strengths and types.

In video deblurring, it is crucial to analyze the relevant information between consecutive frames as well as the information in the target frame. In recent deep neural network based approaches, several designs of CNNs and RNNs are adopted to incorporate temporal information. Su et al. [34] introduced a 2-stage approach to handle misplacement from large motions between frames and the fuse information between the frames. A sequence of frames is spatially aligned to the middle frame by homography or optical flow. Those frames are then fed into a CNN to get a deblurred middle frame. On the other hand, Wieschollek et al. [39] and Kim et al. [18] proposed recurrent network architectures that can operate on arbitrary length videos. While [39] used information from past frames by simply copying features, [18] presented a Dynamic Temporal Blending module on a fast RNN. The module blends the hidden state from past frames and feature from the current frame to transfer the temporal information through hidden states.

These neural network-based approaches mainly focus on how to adopt the related information from the neighboring frames to restore the target frame and show significant improvements. However, these methods try to handle temporal relation in a single-step operation, which may not be optimal. Traditionally, the difficulty of estimating motion information or blur kernel from multiple frames was mitigated by iterative estimation steps [44, 41, 17, 1, 31]. Furthermore, handling the neighbor frames with the alignment from optical flow [34] or heavy neural network [39] is expensive in computation. Thus, to resolve these issues, temporal information transfer method both fast and more optimal is required.

We set up a baseline model in a light and fast convolutional RNN architecture that exploits the inter-frame information. Like [18], we deliver information from past frames to the current frame in the form of hidden state. To let the propagated hidden state fit to the target frame, we employ



(a) Input blurry image $\mathbf{B}_t$ (b) Our deblurred image $\mathbf{L}_t$, IFI-RNN (C2H3-reg)

(c) i Blurry (d) RDN [39] (e) DBN+OF [34] (f) OVD [18] (g) RNN (h) IFI-RNN

Figure 1: Deblurring results comparison with the state-of-the-art methods. (g) result of our model with dual RNN cells without iteration. (h) result of our 3-iteration model with stochastic regularization (IFI-RNN (C2H3-reg)).

an iterative hidden state update scheme within a single inter-frame time-step. We refer to this operation as intra-frame iterations. As the intra-frame iteration is in the same form as inter-frame operation, no modification of the architecture or additional parameters are required. Additionally, we investigate and analyze the schemes of intra-frame recurrence by varying the composition of the RNN cells. (i.e., single-cell and dual-cell methods) We experimentally show that the proposed intra-frame recurrence scheme results in substantial improvements in the restoration accuracy.

We train each model with a predefined intra-frame iteration number. On average, more iterations bring performance improvements. However, not all frames are best restored from the maximum number of iterations. As this is the case when more computation induces degradation, we cast this as an imperfect optimization issue. We adopt a stochastic strategy [36] to employ regularization effect to improve iterative models. As the models with different iteration numbers share an architecture, we regard less iteration models as parts of larger iteration models. During training, the number of internal iterations is chosen randomly. However, our regularization loss term favors fewer calculations. Several works have been reported that training a model with partial computation paths at random improves accuracy [33, 38, 13, 9, 36]. We implement the training by using a gating unit that decides iteration numbers. Note that our primary goal is to improve performance by regularizing RNN cells. Therefore, we drop the gating function at inference and prevent the model from showing a stochastic or adaptive behavior. The result of our regularized dual-cell method is displayed in Fig. 1.

Our contributions in this paper is summarized as follows:

- We present a simple yet effective RNN-based video deblurring method that exploits both the intra-frame (internal) and inter-frame (external) recurrent schemes. By updating the hidden state multiple times internally during a single time-step, our model produces better results without modifying the architecture.

- We study various types of intra-frame iteration strategies. For recurrent networks with different internal cell parameters, we investigate the effect of partial recurrence to investigate more optimal hidden state update strategy.

- Finally, we develop a single model that can be trained to handle various internal recurrence paths (iterations). Our loss function is composed of a data term that aims to minimize restoration error and a prior term that favors shorter computation path. We train our multi-path network in a stochastic way. Owing to the regularization effect of the stochastic training that prevents the co-adaptation of layers, the flexible intra-frame iterative model provides more improved deblurring results.

- Through extensive empirical tests and evaluations, we demonstrate the superiority of the proposed model over the current state-of-the-art methods in both deblurring accuracy and computational efficiency.

## 2. Related Works

In this section, we describe previous works related to our research.

### Video Deblurring

In the early studies of video deblurring, the concept of lucky imaging was adopted where sharp contents replaced the blurry ones in pixel [29] and patch [5] level. Later, deconvolution based methods were widely studied where kernels are estimated from inter-frame relation. Temporal information was exploited to predict the global motion and to generate a sharp panorama scene from a blurry video [25]. To handle differently blurred regions, Wulff and Black [41] studied layered blur model that segments an image into layers and deconvolved each layer separately to improve the estimation of both the blur kernels and the latent image. Kim et al. [17, 19] proposed a segmentation-free dynamic video deblurring method where locally varying blur kernels were approximated from bidirectional optical flows. These methods formulate the problem as a non-convex energy minimization framework of which variables include the local blur kernels and the latent images. Thus, many deconvolution algorithms for deblurring [15, 16, 44, 17, 35] resolve this issue by iteratively optimizing the energy function.

Recently, [30, 34] introduced video datasets that contain realistic blurry frames and corresponding sharp ground-truth frames. As the frames in a video recorded by a high-speed camera are sharp and slowly changing, the average of several subsequent frames can mimic a blurry frame captured at a longer exposure. With the advent of realistic blur datasets, there have been proposed a few deep learning-based methods for single image [30] and video [34] deblurring. Similarly, Wieschollek et al. [39] synthesized the training data by downsampling and interpolating 4k-8k resolution videos.

Su et al. [34] proposed a CNN-based algorithm called DBN. It takes a stack of 5 successive frames as an input and deblurs the middle frame among them. To handle severely blurred frames, they also aligned their input frames with the optical flow as a pre-processing. On the other hand, RDN [39] uses an encoder-decoder architecture model that can process arbitrary length videos. RDN utilizes temporal skip connections so that features extracted in the previous frames can directly propagate to the next frame. In advance, OVD [18] proposed a recurrent network whose hidden state carries the temporal information from the past time-steps. In the recurrent architecture, they added a dynamic temporal blending module so that the hidden state from the previous time-step is adapted to the current frame. Furthermore, Spatio-temporal Transformer Network [20] was applied to improve DBN and OVD by making use of long-range pixel correspondences.

In this paper, we aim to improve the deblurring quality using recurrent neural networks by updating the hidden states to be more optimal for predicting the output. In the viewpoint of making better use of hidden states, our work is closely related to [18, 20]. However, we reuse existing parameters without introducing any extra module.

### Burst Deblurring

Under low-light conditions, a burst of photographs is likely to be blurred due to hand tremor. In [43, 3] the sparse prior of blur kernels and spatial gradient of latent images are investigated to obtain sharp images. On the other hand, some alignment-free methods were suggested by posing a joint problem of multiple image registration and deblurring [42, 4, 45].

Then, Delbracio and Sapiro [6, 7] presented a simple yet efficient burst deblurring method without relying on kernel estimation and deconvolution. They utilized spectral information in the Fourier domain where information from less blurred images is more weighted. Wieschollek et al. [40] further extended [6] by learning a hybrid network that decides the weights for Fourier burst accumulation and the deconvolution filter. Furthermore, a recently proposed permutation-invariant model by Aittala and Durand [2] improved the restoration quality significantly in the presence of noise, blur, and saturation. We also augment noise in the training process like [2, 30].

### Stochastic Neural Network Training

Most of the neural networks are designed to process equally for every input. However, training the networks as it is not always known to be optimal. Therefore, several randomized training strategies have been proposed to regularize the optimization process. The most classical types of stochastic regularization techniques are Dropout [12, 33] and DropConnect [38]. While Dropout randomly deactivates the outputs of fully-connected layers, DropConnect disconnects the weights of the layer at training time. They are known to prevent co-adaptation of features and regularize the network to avoid overfitting.

In ResNets [10, 11], residual blocks contain shortcut connections where their inputs are directly headed to the output in parallel with convolution features. Veit et al. [37] observed that this could be interpreted as an ensemble of exponentially many shallower networks. In ResNets, surprisingly, removing or permuting several layers do not cause catastrophic degradation. Furthermore, ResNets trained with random skips of residual blocks showed ameliorated classification accuracy [13]. Similarly, FractalNet [24] showed that drop-path training could also exhibit regularization effect.

Recently, more advanced stochastic training techniques were proposed, letting the stochastic path to be chosen by the model itself. Graves [9] proposed an adaptive computa-

tion time (ACT) algorithm where the number of recurrence steps between the inputs is decided by the network with an estimated halting score, instead of using a predefined fixed number of iterations. Figurnov et al. [8] extended the ACT to spatial locations of ResNets [11] so that every pixel would have different network depth.

The most relevant study to ours is the work by Veit and Belonge [36]. They added a gating unit in each block of the ResNet that could switch-off rather irrelevant layers. To computationally benefit from the switching, the output from the gates should be hard binary rather than being soft. The training of the hard gate is enabled by using the back-propagation with Gumbel-SoftMax relaxation [14, 28]. In contrast to the previous methods focusing on acceleration with a moderate increase in error, they exhibit improved accuracy compared to the original ResNet for image classification.

In our experiments, we find that several different numbers of intra-frame iterations are beneficial in general. Hence, we conjecture that training a single generic model that could operate in the variable number of intra-frame iterations is possible, regarding the shared architecture between our models. We aim to benefit from regularization effect through training our model in stochastic paths. To let our model decide the iteration number itself, we implement a stochastic gate function that determines if additional iteration is to be used or not. To jointly train the gate as well as the main network, we design a regularization loss term that favors less computation together with the content (L2) loss. We adopt the Gumbel-Softmax trick [14, 28] that has been used in [36] to route the model in a single prediction path that is discretely decided from the iteration number. Our regularized models exceed their original models in deblurring performance, both quantitatively and qualitatively.

# 3. Proposed Method

In this section, we describe how we develop our model. In section 3.1, we describe our baseline RNN model and the formulation terminologies. In section 3.2, we explain the concept of our intra-frame iteration model and analyze possible iteration strategies. Lastly, we describe more advanced training methods for our intra-frame iteration RNN in section 3.3.

## 3.1. Recurrent Video Deblurring Networks

Let us denote the blurry video, ground-truth sharp video, and the predicted latent video as $\mathbf{B} = \{\mathbf{B}_t\}$, $\mathbf{S} = \{\mathbf{S}_t\}$, $\mathbf{L} = \{\mathbf{L}_t\}$ with the frame index $t \in \{1 \dots T\}$, respectively.

We construct our baseline architecture as a recurrent neural network so that temporal information can propagate over video frames like [18]. Then, our network operates on the blurry input video by following recurrence operation.

$$(\mathbf{L}_t, \mathbf{h}_t) = \mathcal{F}\left(\mathbf{B}_t, \mathbf{h}_{t-1}\right),$$

where $\mathcal{F}$ refers to our RNN cell. The cell consists of several components, $\mathcal{F}_\mathbf{B}$, $\mathcal{F}_\mathbf{R}$, $\mathcal{F}_\mathbf{L}$, $\mathcal{F}_\mathbf{h}$ as shown in Fig. 2. First, $\mathcal{F}_\mathbf{B}$ extracts the feature $f_{\mathbf{B}_t}$ from a blurry frame. Then, $\mathcal{F}_\mathbf{R}$ produces the intermediate feature $f_{\mathbf{B}_t}$ that is used for $\mathcal{F}_\mathbf{L}$ and $\mathcal{F}_\mathbf{h}$ to estimate the latent frame $\mathbf{L}_t$ and hidden state $\mathbf{h}_t$, respectively. $\mathbf{h}_t$ is the hidden state that is produced at $t$-th time-step and will be propagated to $t+1$-th time-step. We initialize $\mathbf{h}_0$ with zero.

The RNN cell consists of strided convolutions ($\mathcal{F}_\mathbf{B}$) followed by ResBlocks ($\mathcal{F}_\mathbf{R}$, $\mathcal{F}_\mathbf{h}$) and up-convolutions ($\mathcal{F}_\mathbf{L}$). Note that we use ResBlocks without batch normalization [30, 26]. Please refer to the supplementary materials for layer specifications.

We train our baseline model with the $L2$ loss between the estimated latent video and the ground-truth sharp video such as

$$\mathcal{L}_{\text{content}} = \frac{1}{TCHW} \sum_{t=1}^{T} \|\mathbf{L}_t - \mathbf{S}_t\|_2^2,$$

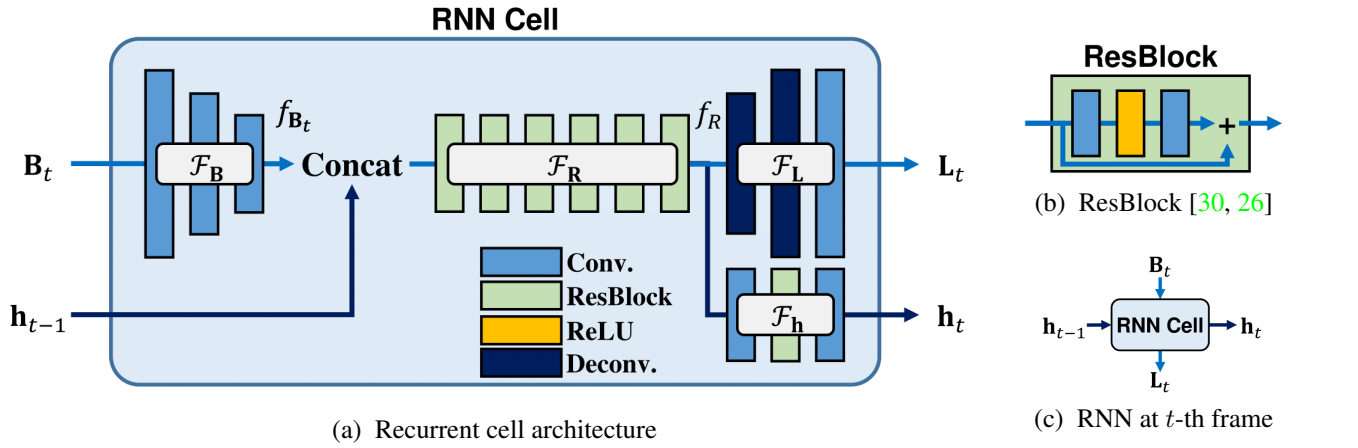
where $C$, $H$, $W$ denote the number of channels (3 for RGB color videos), height, and the width of the training samples, respectively.

## 3.2. Intra-frame iteration Models

The most crucial part of RNNs against CNNs is the hidden state that brings the performance gain as CNNs have no temporal connections. Therefore, it is essential to have *good* hidden states so that they could better help to predict more accurate outputs at the current frame as well as at the next frame. In this regard, we attempt to make *better* use of hidden states by intra-frame iteration before passing it to the next RNN cell.

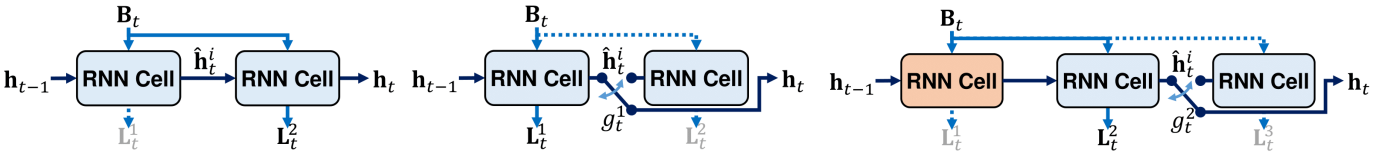
We implement this idea by utilizing our baseline RNN cell architecture. First, we compute the initial hidden state $\hat{\mathbf{h}}_t^0$ at a certain time step $t$ from the blurry input $\mathbf{B}_t$ and the previous hidden state $\mathbf{h}_{t-1}$ using our RNN cell. Then, we feedback $\hat{\mathbf{h}}_t^0$ to the cell again without changing $\mathbf{B}_t$ to update the hidden state. After updating the hidden state for $N$ iterations, we finally generate a latent output frame $\mathbf{L}_t$ at that time step with the updated hidden state $\hat{\mathbf{h}}_t^N$. Note that the blur feature extractor $\mathcal{F}_\mathbf{B}$ and the latent frame estimator $\mathcal{F}_\mathbf{L}$ are used only once despite the number of iterations.

We provide two different types of iteration: the single cell and the dual cell method. In the single cell method, we use the same parameters to estimate both the initial hidden state and the updated hidden state. On the other hand, in a dual cell method, we use two RNN cells and use each of them for a different purpose. Only the second cell is used to update the hidden states and predict latent frames. Although the dual cell method requires more parameters




(a) Recurrent cell architecture (b) ResBlock [30, 26] (c) RNN at $t$-th frame

Figure 2: The baseline architecture of IFI-RNN (Ours)




(a) Training with fixed iterations (b) Stochastic training (single cell) (c) Stochastic training (dual cell)

Figure 3: Training methods of IFI-RNN using different hidden state update schemes.

than the single cell approach, it can bring significant performance gain as different sets of parameters can dedicate to different roles. From now on, we denote the single and dual cell models with prefix C1 and C2, respectively. Also, we put a suffix H with the hidden state iterations. For example, C2H2 denotes the dual cell model which updates its hidden state two times.

We describe the two intra-frame hidden state updating methods in Algorithm 1. In an architectural viewpoint, our methods virtually increase the depth of RNN cell, enlarging the receptive field and its capacity. In other words, our hidden states can be better optimized by a virtually deeper model.

### 3.3. Regularization by Stochastic Training

The performance gains from iteration, however, become marginal for higher iteration models. For example, the C1H4 model (single cell four iterations) does not perform better than C1H3 model in Fig. 5. We also observe that for each image, the best performing model is not always the one with more iterations. Fig. 4 shows the number of images that are best restored by the single-cell method with different iterations. Although many images prefer more iterations for better restoration, a nontrivial amount of images favor lesser iterations. Since we use the same RNN cell for each iteration, it is natural to conjecture that we can train a model that can deblur each input frame with different it-

**Algorithm 1** Deblurring with intra-frame hidden state update

**procedure** SINGLE CELL METHOD($\mathbf{B}_t, \mathbf{h}_{t-1}$)
  $f_{\mathbf{B}_t} = \mathcal{F}_\mathbf{B}(\mathbf{B}_t)$
  $\hat{\mathbf{h}}_t^0 \leftarrow \mathbf{h}_{t-1}$
  **for** $i = 1 \dots N$ **do**
    $f_R^i = \mathcal{F}_R\left(f_{\mathbf{B}_t}, \hat{\mathbf{h}}_t^{i-1}\right)$
    $\hat{\mathbf{h}}_t^i = \mathcal{F}_\mathbf{h}(f_R)$
  $\mathbf{h}_t \leftarrow \hat{\mathbf{h}}_t^N$
  $\mathbf{L}_t = \mathcal{F}_L\left(f_R^N\right)$
  **return** $\mathbf{L}_t, \mathbf{h}_t$

**procedure** DUAL CELL METHOD($\mathbf{B}_t, \mathbf{h}_{t-1}$)
  $f_{\mathbf{B}_t,1} = \mathcal{F}_{\mathbf{B},1}(\mathbf{B}_t)$
  $f_{\mathbf{B}_t,2} = \mathcal{F}_{\mathbf{B},2}(\mathbf{B}_t)$
  $\hat{\mathbf{h}}_t^0 = \mathcal{F}_{\mathbf{h},1}(\mathcal{F}_{\mathbf{R},1}(f_{\mathbf{B}_t}, \mathbf{h}_{t-1}))$
  **for** $i = 1 \dots N$ **do**
    $f_{R,2}^i = \mathcal{F}_{\mathbf{R},2}\left(f_{\mathbf{B}_t,2}, \hat{\mathbf{h}}_t^{i-1}\right)$
    $\hat{\mathbf{h}}_t^i = \mathcal{F}_{\mathbf{h},2}\left(f_{R,2}^i\right)$
  $\mathbf{h}_t \leftarrow \hat{\mathbf{h}}_t^N$
  $\mathbf{L}_t = \mathcal{F}_\mathbf{L}\left(f_{R,2}^N\right)$
  **return** $\mathbf{L}_t, \mathbf{h}_t$

erations in a stochastic way. Therefore, we attempt to take advantage of the regularization effect from using stochastic computational path for training.

First, we add a gating unit $g(\cdot) \in \{0, 1\}$ that looks into the hidden state and decides if the model will compute one more iteration or not. We calculate the score for iteration by global average pooling [27] followed by two fully connected layers activated by ReLU [23]. Then discrete binary sampling is done with the Gumbel-SoftMax trick [14, 28]. At the training time, when the gate is on, we update the hidden state once more. Otherwise, we stop the iteration and return the deblurred frame. Second, we employ a regularization term, that favors fewer iterations when the loss is already small enough. We set a target average iteration ratio, $\tau = 0.75$. Compared to the models with a fixed iteration number, this loss prefers stopping the iteration with the probability of $1 - \tau$. We define the term as L2 loss between the average gate activation over a mini-batch with iterations and $\tau$.

$$\mathcal{L}_{\text{reg}} = \frac{1}{T} \sum_{t=1}^{T} \sum_{i=1}^{N} \left( \mathbb{E}\left[ g_t^i \right] - \tau \right)^2,$$

where $\mathbb{E}[\cdot]$ is an average operation, $g_t^i = g\left(\hat{\mathbf{h}}_t^i\right)$ at iteration $i$ at time-step $t$, and $N$ is the maximum iteration threshold we set during training. Thus, our final loss term becomes

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{content}} + \lambda \mathcal{L}_{\text{reg}},$$

with $\lambda$ as the weight for the regularization term. Note that our primary purpose of the stochastic training is to improve the results by regularizing the co-adaptation of parameters, rather than making our model to show stochastic behavior. So, we remove the gating unit after training so that the system provides the results of a specified number of iterations.

The performances of regularized models are shown as dotted lines in Fig. 5. We add '-reg' suffix to our IFI-RNN models to refer models trained with regularization like C2H3-reg.

## 4. Experimental Results

### 4.1. Datasets

We have tested our algorithm (denoted as IFI-RNN) on the GOPRO dataset [30]. The GOPRO dataset contains 2103 training samples from 22 sequences and 1111 evaluation samples from 11 sequences. We generated blur and sharp image pairs from 240 fps videos. Those high-speed video frames are averaged in a gamma-transformed domain to mimic images taken in longer exposure time with nonlinear camera response function (CRF). To suppress the noise

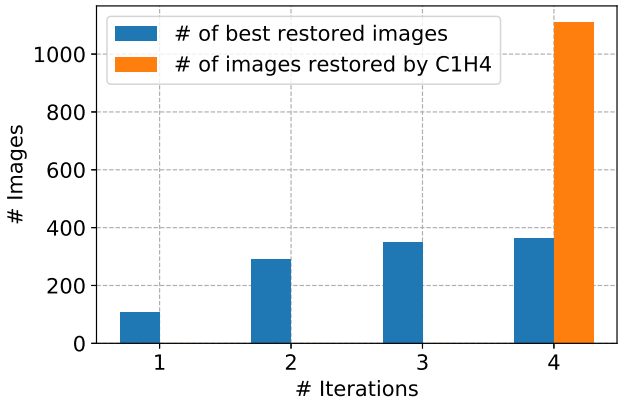



Figure 4: Blue bars show the number of images that are best restored by the single-cell method according to the iterations. Orange bar represents the total number of images restored by C1H4 model. We used downsampled GOPRO test images [30]. Refer to section 4.1 for details.

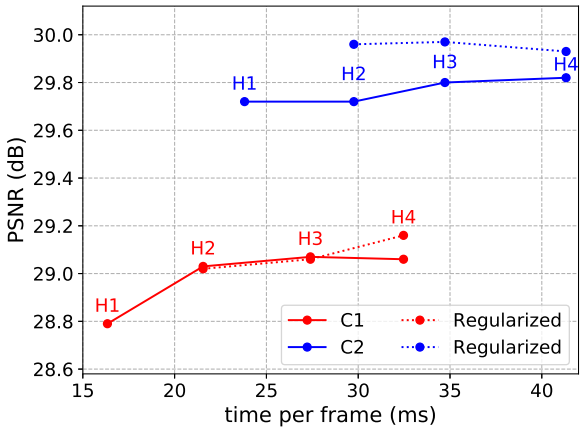



Figure 5: PSNR and the running time of our methods, evaluated on downsampled GOPRO test set at resolution $960 \times 540$. Refer to section 4.1 for details.

and video compression artifacts, we downsampled the original video resolution from $1280 \times 720$ to $960 \times 540$ before averaging.

We also use a similar dataset from Su et al. [34]. This dataset also consists of paired samples synthesized from 240 fps videos. It provides 61 sequences containing 5708 training pairs and 10 sequences including 1000 evaluation pairs. However, we do not evaluate with the method proposed by Köhler et al. [22] as [34]. Instead, we evaluate PSNR and SSIM as is without post-processing such as alignment. In addition to the original captured frames, they interpolate intermediate sharp frames from optical flow estimation to generate smooth blur frames. The original and

interpolated frames are averaged altogether to synthesize blurs under linear CRF assumption.

To compare our method with previous methods, we use the test video sequences from the above two datasets except for the first four frames and the last frame in each video, as [18] does not provide the results for them. Also, we show the deblurring results of real videos to demonstrate the generalization capability of our method.

### 4.2. Implementation details

We train our models on the GOPRO dataset [30] with ADAM optimizer [21] where $\beta_1 = 0.9$ and $\beta_2 = 0.999$. We train each model for 500 epochs in total. Beginning from the initial learning rate of $10^{-4}$, we anneal the learning rate by half after every 200 epochs. We set the regularization loss weight $\lambda = 10$. During training, we sample 12-frame $256 \times 256$ RGB patch sequences from the dataset to construct a mini-batch of size 4. Random augmentations are applied to those samples with geometric transforms including vertical and horizontal flips as well as $90^o$ rotation. Also, we add zero-mean Gaussian noise to blurry inputs, where its standard deviation is sampled from another Gaussian distribution $\mathcal{N}\left(0, 2^2\right)$ to blurry inputs NVIDIA GTX 1080 Ti GPUs were used for all of our experiments. We implemented our models with PyTorch 0.4.1 [32] built with CUDA 9.2 and cuDNN 7.1. Our source code will be released publicly.

### 4.3. Comparisons on GOPRO [30] dataset

We evaluate our method and other methods on the downsampled GORPO dataset. We report the evaluation results of all the comparing methods in terms of PSNR, SSIM and the running time in Table 1. From these results, it is clear that the proposed intra-frame iteration scheme and the stochastic training method improve the performance of our model significantly compared with the other state-of-the-art methods. Furthermore, surprisingly, our method is much faster than the others, despite having internal iterative operations. For visual comparison, please refer to Fig. 1.

| Method | PSNR / SSIM | Speed (fps) |
|---|---|---|
| DBN+OF [34] | 27.08 / 0.8429 | 1.72† |
| RDN [39] | 25.19 / 0.7794 | 7.37 |
| OVD [18] | 26.82 / 0.8245 | 9.24 |
| IFI-RNN (C1H1) | 28.79 / 0.8647 | 61.2 |
| IFI-RNN (C1H2) | 29.03 / 0.8712 | 46.4 |
| IFI-RNN (C1H3) | 29.07 / 0.8730 | 36.5 |
| IFI-RNN (C1H4) | 29.06 / 0.8730 | 30.8 |
| IFI-RNN (C1H4-reg) | **29.16 / 0.8760** | 30.8 |
| IFI-RNN (C2H1) | 29.72 / 0.8884 | 42.0 |
| IFI-RNN (C2H2) | 29.72 / 0.8885 | 33.6 |
| IFI-RNN (C2H3) | 29.80 / 0.8900 | 28.8 |
| IFI-RNN (C2H4) | 29.82 / 0.8913 | 24.2 |
| IFI-RNN (C2H3-reg) | **29.97 / 0.8947** | 28.8 |
| IFI-RNN (C2H4-reg) | 29.93 / 0.8943 | 24.2 |

Table 1: Deblurring accuracy comparison on the downsampled GOPRO dataset [30]. For our method IFI-RNN, C1 and C2 refer to single-cell and dual-cell method, respectively. †Note that the above speed does not include the optical flow estimation time for [34]. All the running times are averaged from 10 runs on the test set.

| Method | PSNR / SSIM |
|---|---|
| DBN+OF [34] | 30.14 / 0.8913 |
| RDN [39] | 26.98 / 0.8076 |
| OVD [18] | 29.97 / 0.8696 |
| IFI-RNN (C1H1) | 30.07 / 0.8823 |
| IFI-RNN (C1H4-reg) | 30.10 / 0.8849 |
| IFI-RNN (C2H1) | 30.74 / 0.8974 |
| IFI-RNN (C2H3-reg) | **30.80 / 0.8991** |
| IFI-RNN (C2H4-reg) | 30.73 / 0.8976 |

Table 2: Deblurring accuracy comparison on the dataset from [34].

### 4.4. Comparisons on [34] Dataset and Real Videos

We also compared the performances on the dataset in [34]. In this case, we fine-tuned our GOPRO models with the training subset of [34]. In Table 2, our model also improves performance with iterations and regularization for both C1 and C2 models. Furthermore, IFI-RNN C2 models show state-of-the-art performance. In Fig. 6, our IFI-RNN recovers the text and legs more clearly. Also, our results on real videos also clarify blurred textures in Fig. 7.

## 5. Conclusion

In this paper, we proposed a method to ameliorate the recurrent network for video deblurring. By iteratively updating the hidden state to the target frame, our method removes blurs in the video frames more effectively. Furthermore, we train our model with a regularization term that could enhance prediction accuracy through stochastic computation paths. Our method does not require additional parameters while being fast and accurate compared to other state-of-the-art methods.

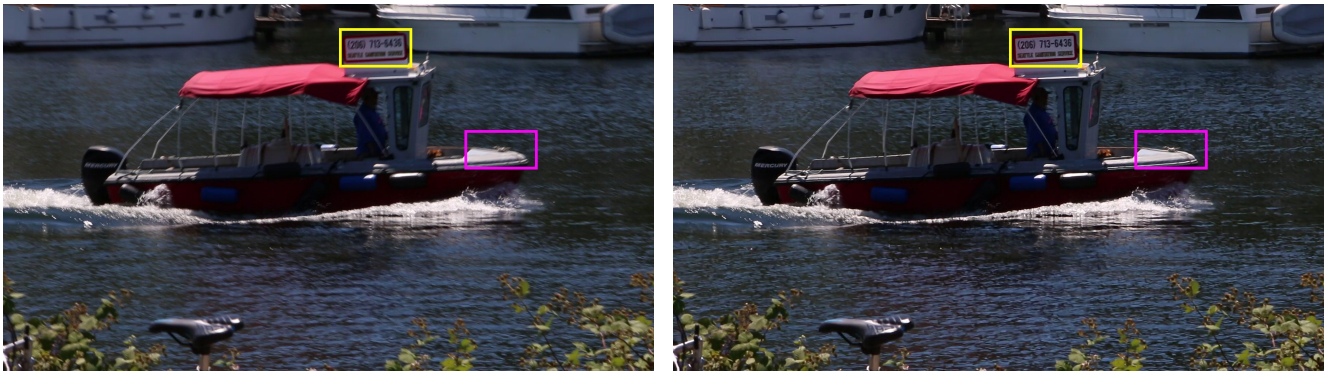
## Acknowledgement

This work was partially supported by LG Electronics and the National Research Foundation of Korea (NRF) grant funded by the Korea Government(MSIT) (No. NRF-2017R1A2B2011862)

(a) Blur (b) Deblurred (Ours)

(c) Blur (d) RDN [39] (e) OVD [18] (f) DBN+OF [34] (g) IFI-RNN(C2H3-reg)

Figure 6: Deblurring results on [34] dataset.



(a) Blur (b) Deblurred (Ours, IFI-RNN(C2H4-reg))

(c) Blur (d) RDN [39] (e) OVD [18] (f) DBN+OF [34] (g) IFI-RNN(C2H3-reg)

Figure 7: Deblurring results of real video.

## References


[1] Byeongjoo Ahn, Tae Hyun Kim, Wonsik Kim, and Kyoung Mu Lee. Occlusion-aware video deblurring with a new layered blur model. *arXiv preprint arXiv:1611.09572*, 2016. 1

[2] Miika Aittala and Fredo Durand. Burst image deblurring using permutation invariant convolutional neural networks. In *ECCV*, 2018. 3

[3] Jian-Feng Cai, Hui Ji, Chaoqiang Liu, and Zuowei Shen. Blind motion deblurring using multiple images. *Journal of Computational Physics*, 228(14):5057–5071, 2009. 3

[4] Sunghyun Cho, Hojin Cho, Yu-Wing Tai, and Seungyong Lee. Registration based non-uniform motion deblurring. In *Computer Graphics Forum*, volume 31, pages 2183–2192. Wiley Online Library, 2012. 3

[5] Sunghyun Cho, Jue Wang, and Seungyong Lee. Video deblurring for hand-held cameras using patch-based synthesis. *ACM Transactions on Graphics (TOG)*, 31(4):64, 2012. 3

[6] Mauricio Delbracio and Guillermo Sapiro. Burst deblurring: Removing camera shake through fourier burst accumulation. In *CVPR*, 2015. 3

[7] Mauricio Delbracio and Guillermo Sapiro. Removing camera shake via weighted fourier burst accumulation. *IEEE Transactions on Image Processing (TIP)*, 24(11):3293–3307, 2015. 3

[8] Michael Figurnov, Maxwell D Collins, Yukun Zhu, Li Zhang, Jonathan Huang, Dmitry P Vetrov, and Ruslan Salakhutdinov. Spatially adaptive computation time for residual networks. In *CVPR*, 2017. 4

[9] Alex Graves. Adaptive computation time for recurrent neural networks. *arXiv preprint arXiv:1603.08983*, 2016. 2, 3

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 3

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *ECCV*, 2016. 3, 4

[12] Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012. 3

[13] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q Weinberger. Deep networks with stochastic depth. In *ECCV*, 2016. 2, 3

[14] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016. 4, 6

[15] Tae Hyun Kim, Byeongjoo Ahn, and Kyoung Mu Lee. Dynamic scene deblurring. In *CVPR*, 2013. 3

[16] Tae Hyun Kim and Kyoung Mu Lee. Segmentation-free dynamic scene deblurring. In *CVPR*, 2014. 3

[17] Tae Hyun Kim and Kyoung Mu Lee. Generalized video deblurring for dynamic scenes. In *CVPR*, 2015. 1, 3

[18] Tae Hyun Kim, Kyoung Mu Lee, Bernhard Scholkopf, and Michael Hirsch. Online video deblurring via dynamic temporal blending network. In *ICCV*, 2017. 1, 2, 3, 4, 7, 8

[19] Tae Hyun Kim, Seungjun Nah, and Kyoung Mu Lee. Dynamic video deblurring using a locally adaptive blur model. *IEEE transactions on pattern analysis and machine intelligence*, 40(10):2374–2387, 2018. 3

[20] Tae Hyun Kim, Mehdi S. M. Sajjadi, Michael Hirsch, and Bernhard Scholkopf. Spatio-temporal transformer network for video restoration. In *ECCV*, 2018. 3

[21] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. 7

[22] Rolf Köhler, Michael Hirsch, Betty Mohler, Bernhard Schölkopf, and Stefan Harmeling. Recording and playback of camera shake: Benchmarking blind deconvolution with a real-world database. In *ECCV*, 2012. 6

[23] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, pages 1097–1105, 2012. 6

[24] Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. Fractalnet: Ultra-deep neural networks without residuals. *arXiv preprint arXiv:1605.07648*, 2016. 3

[25] Yunpeng Li, Sing Bing Kang, Neel Joshi, Steve M Seitz, and Daniel P Huttenlocher. Generating sharp panoramas from motion-blurred videos. In *CVPR*, 2010. 3

[26] Bee Lim, Sanghyun Son, Heewon Kim, Seungjun Nah, and Kyoung Mu Lee. Enhanced deep residual networks for single image super-resolution. In *CVPR Workshops*, 2017. 4, 5

[27] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. *arXiv preprint arXiv:1312.4400*, 2013. 6

[28] Chris J Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. *arXiv preprint arXiv:1611.00712*, 2016. 4, 6

[29] Yasuyuki Matsushita, Eyal Ofek, Weina Ge, Xiaoou Tang, and Heung-Yeung Shum. Full-frame video stabilization with motion inpainting. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 28(7):1150–1163, 2006. 3

[30] Seungjun Nah, Tae Hyun Kim, and Kyoung Mu Lee. Deep multi-scale convolutional neural network for dynamic scene deblurring. In *CVPR*, 2017. 3, 4, 5, 6, 7

[31] Liyuan Pan, Yuchao Dai, Miaomiao Liu, and Fatih Porikli. Simultaneous stereo video deblurring and scene flow estimation. In *CVPR*, 2017. 1

[32] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017. 7

[33] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014. 2, 3

[34] Shuochen Su, Mauricio Delbracio, Jue Wang, Guillermo Sapiro, Wolfgang Heidrich, and Oliver Wang. Deep video deblurring for hand-held cameras. In *CVPR*, 2017. 1, 2, 3, 6, 7, 8

[35] Jian Sun, Wenfei Cao, Zongben Xu, and Jean Ponce. Learning a convolutional neural network for non-uniform motion blur removal. In *CVPR*, 2015. 3

[36] Andreas Veit and Serge Belongie. Convolutional networks with adaptive inference graphs. In *ECCV*, 2018. 2, 4

[37] Andreas Veit, Michael J Wilber, and Serge Belongie. Residual networks behave like ensembles of relatively shallow networks. In *NIPS*, pages 550–558, 2016. 3

[38] Li Wan, Matthew Zeiler, Sixin Zhang, Yann Le Cun, and Rob Fergus. Regularization of neural networks using dropconnect. In *ICML*, 2013. 2, 3

[39] Patrick Wieschollek, Michael Hirsch, Bernhard Scholkopf, and Hendrik P. A. Lensch. Learning blind motion deblurring. In *ICCV*, 2017. 1, 2, 3, 7, 8

[40] Patrick Wieschollek, Bernhard Schölkopf, Hendrik PA Lensch, and Michael Hirsch. End-to-end learning for image burst deblurring. In *ACCV*, 2016. 3

[41] Jonas Wulff and Michael Julian Black. Modeling blurred video with layers. In *ECCV*, 2014. 1, 3

[42] Haichao Zhang and Lawrence Carin. Multi-shot imaging: Joint alignment, deblurring and resolution-enhancement. In *CVPR*, 2014. 3

[43] Haichao Zhang, David Wipf, and Yanning Zhang. Multi-image blind deblurring using a coupled adaptive sparse prior. In *CVPR*, 2013. 3

[44] Haichao Zhang, David Wipf, and Yanning Zhang. Multi-observation blind deconvolution with an adaptive sparse prior. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 36(8):1628–1643, 2014. 1, 3

[45] Haichao Zhang and Jianchao Yang. Intra-frame deblurring by leveraging inter-frame camera motion. In *CVPR*, 2015. 3