https://doi.org/10.2352/ISSN.2470-1173.2020.14.COIMG-344

Background subtraction in diffraction X-ray images using deep CNN

Rodrigo Aranguren Carmona, Gady Agam, and Thomas Irving;
Illinois Institute of Technology, Chicago, IL 60616

Abstract

Diffraction X-ray images provide molecular level information in tissue under hydrated, physiological conditions at the physiologically relevant millisecond time scale. When processing diffraction x-ray images there is a need to subtract background produced during the capture process prior to making measurements. This is a non-uniform background that is strongest at the diffraction center and decays with increased distance from the center. Existing methods require careful parameter selection or assume a specific background model. In this paper we propose a novel approach for background subtraction in which we learn to subtract background based on labeled examples. The labeled examples are image pairs where in each pair one of the images has diffraction background and the second has the background removed. Using a deep convolutional neural network (CNN) we learn to map an image with background to an image without it. Experimental results demonstrate that the proposed approach is capable of learning background removal with results close to ground truth data (PSNR $>$ 68, SSIM $>$ 0.99) and without having to manually select background parameters.

Introduction

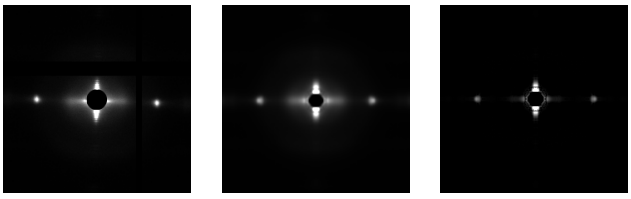
X-ray diffraction is the only technique that can provide molecular level information in tissue under hydrated, physiological conditions at the physiologically relevant millisecond time scale. Thus, the results of X-ray diffraction experiments have told us much about what we know about the molecular events involved in muscle contraction.

To facilitate the analysis of diffraction X-ray images, there is a need to remove diffraction background. This is a non-uniform background that is strongest at the diffraction center and goes down when the distance from the center increases. Existing algorithms for background subtraction use morphological filters (e.g. white-top-hat) or try to model the background (e.g. using a truncated Gaussian). Such methods require selecting parameters (e.g. radius in white-top-hat) or assume a specific background model (e.g. truncated Gaussian) and in this way introduce bias into the background subtraction process. Adjusting parameters in such methods is often manual. Incorrect background adjustments may impact measured quantities in the images.

In this paper we propose a novel approach for background subtraction in which we learn to subtract background based on labeled examples. The labeled examples are image pairs where in each pair one of the images has diffraction background and the second has the background removed. Using a deep convolutional neural network (CNN) we learn to map an image with background to an image without it. Experimental results demonstrate that the proposed approach is capable of learning background removal with results close to ground truth data (PSNR $>$ 68, SSIM $>$ 0.99) and without having to manually select background parameters.

The algorithms used to prepare the training data are part of the MuscleX [1] application suite we developed. Currently available methods in this program include: white-top-hat filtering, calculation of a circularly-symmetric background subtraction, 2D convex hull, Paul Langan's roving window method, and smoothed subtraction based on [2], which has two alternatives: smoothed Gaussian and smoothed boxcar.

In order to achieve the best performance, the algorithms can be used in combination or alone (an example is shown in Figure 1), but they require to be tuned on a per-image basis. While default parameters provided by [1] usually operate adequately, there are cases where the algorithms do not perform equally well under all circumstances.



(a) Before quadrant folding

(b) After quadrant folding. No background subtraction

(c) After quadrant folding. With background subtraction

Figure 1: Fiber diffraction patterns at different stages of processing

Proposed approach

Our approach is based on a modified U-Net [3]. We replace the cross entropy loss and the final soft-max activation, usually suited for classification tasks in the original U-Net with the mean absolute error and LeakyReLU [4, 5] nonlinearity respectively. The combination of cross entropy loss and soft-max is typical of classification tasks, whereas the problem of background subtraction is in essence a problem of reconstruction. The output in our model is the final image without background, and so we need our network to produce continuous pixel values instead of discrete label predictions for every pixel location. Subsequently, we build our network so the final output has the same number of channels as the input. In contrast to the original U-Net, we also want our output image to preserve the resolution of the input. In order to achieve this, we add zero padding, preventing the image size to decrease after every convolution.

The encoding and decoding of information in the U-Net, may

cause block artifacts in some regions of the output images. These artifacts are not addressed in the original U-Net due to the fact that it focuses on producing a segmentation map instead of a reconstructed image. In a segmentation map, details and gradients are not preserved as the only relevant information is a label for each pixel.

Such artifacts were partially addressed by increasing the receptive field of the convolutions. While increasing the input patch size will increase the receptive field, this comes at a cost of adding more parameters without actually dealing with the underlying issue. To address this we apply atrous convolutions [6] with rates greater than 1 on all blocks. Dilation rates 2 and 4 were tested. This is effectively a way of increasing the receptive field without adding more parameters to the network.

Block artifacts have been shown to be associated with learnable transposed convolutions in decoder stages when performing upsampling operations [7]. The proposed U-Net has several such layers in the decoding path. Replacing the transposed convolutions with simpler upsampling operations followed by interpolation largely solves the issue of block patterns. Using 3×3 convolutions after the upsampling produced the best results. There was no difference between nearest neighbor and bilinear interpolation.

To further deal with block artifacts, we modify the loss term in addition to the previous steps described. Let I be the input image, and $\hat{I}$ be the image with its background subtracted. We propose a new loss term that can better preserve changes of intensity and prevent block artifacts. By encouraging the intensity to be smooth on small gradients and sharp on large gradients, contrast is improved. This technique is used in recent reconstruction and enhancement applications of deep neural networks [8]. The loss term we use in our work is as follows:

$$\mathscr{L}_s = \sum_p \omega^p_{x,c}(\partial_x \hat{I}_p)^2 + \omega^p_{y,c}(\partial_y \hat{I}_p)^2 \tag{1}$$

This term is computed over all pixels p. In this expression ∂_x and ∂_x are the partial derivatives in both spatial directions; $\hat{I}$ is the predicted image without background and $\omega^p_{y,c}$ and $\omega^p_{y,c}$ are smoothness weights defined as:

$$\begin{aligned} \omega^p_{x,c} &= (|\partial_x L^p|^\theta_c + \varepsilon)^{-1} \\ \omega^p_{y,c} &= (|\partial_y L^p|^\theta_c + \varepsilon)^{-1} \end{aligned} \tag{2}$$

where L is the logarithm of input image I; θ is a hyperparameter that controls the weight of the image gradient and ε is a small term introduced to prevent division by zero. With this term, our final loss is constructed as follows:

$$\mathscr{L} = (1-\alpha)\mathscr{L}_r + \alpha\mathscr{L}_s \tag{3}$$

where α is a hyperparameter in the range $[0-1]$ that weights both loss terms.

We tested introducing a final refinement to our network. Similar to [9], we add a skip connection between the input image and the final output. This addresses the vanishing gradient problem [10], which is common on deep networks. Intuitively this approach makes sense when applied to our task, as the network will learn the difference between the input and the ground truth. While this was thought to be beneficial, this refinement introduced different issues with our network, which yielded pixel values that sometimes were outside of the intended range. Thus, this refinement is not included in the final network.

The final network architecture is shown in Figure 2. Our model uses 4 convolutional blocks on the encoding path. Each one of them has two convolutional layers with LeakyReLU activation (inspired by [11], we use $\alpha = 0.2$). We preserve the spatial dimensions using zero padding on all convolutions. This way, downsampling is only done with max pooling layers. There is a final bottleneck convolutional block without pooling. The decoding path uses previously mentioned upsampling operations followed by regular convolutions. The output is concatenated to the matching layer from the encoding path and fed into two consecutive convolutional layers with LeakyReLU activation and zero padding. When the original resolution is reached, we subtract the result from the input image, yielding the output.

The model is implemented in Keras [12] with TensorFlow backend [13]. Inputs are cropped or resized to have the same length on both spatial dimensions, depending on the experiment. Input and ground truth patches are resolution and normalized to be in the range 0-1. We minimize the loss using the Adam optimizer with an initial learning rate of 1e-5. The rest of parameters for the optimizer are the recommended by [14]. We train for up to 1000 epochs when using the synthetic datasets and 100 when using real data. Batch size is either 4, 8 or 16 depending on the size of the input patch. PSNR and SSIM are used as additional metrics. Training is done on NVidia Tesla K80 and P100 GPUs.

Experimental results

To evaluate the proposed approach we employ two commonly used metrics (PSNR and SSIM). High values for both metrics indicate good results. During training these metrics are computed on the entire image, whereas during evaluation of the final images, only the center region is used (region of interest).

The training dataset containing actual data consists of 3000 diffraction images. In order to produce the training pair, each sample is processed using MuscleX's Quadrant Folding [1]. An image is folded and has its background removed using a specific method. Therefore, given a diffraction image, we are able to produce a training pair for every method. The input to the network is the folded image with background, and the ground truth is the folded image without background, as produced by one of the available methods using the provided parameters. Using this automated procedure, we are able to produce up to 18000 training pairs. We split the dataset into training, validation and testing subsets (2000/500/500). However, some cases exist where the folding fails, either because the correct center is not found or the equator orientation is not properly calculated (see Figure 3). In order to remove the majority of these failed cases, we compute the average folded image and discard the 10% that deviate the most from it.

We focus our training on a subset of the available background subtraction methods. We drop the algorithms that produce the most similar results. In order to make the decision on which ones to discard, we subtract the background from 500 images taken from our training dataset using all methods. We compute the average mean absolute error and the PSNR for all possible pairs (including the input image) for all images. We then average those values.

We identify algorithms that produce similar results by observing low MAE and high PSNR values. First, "smoothed Gaus-

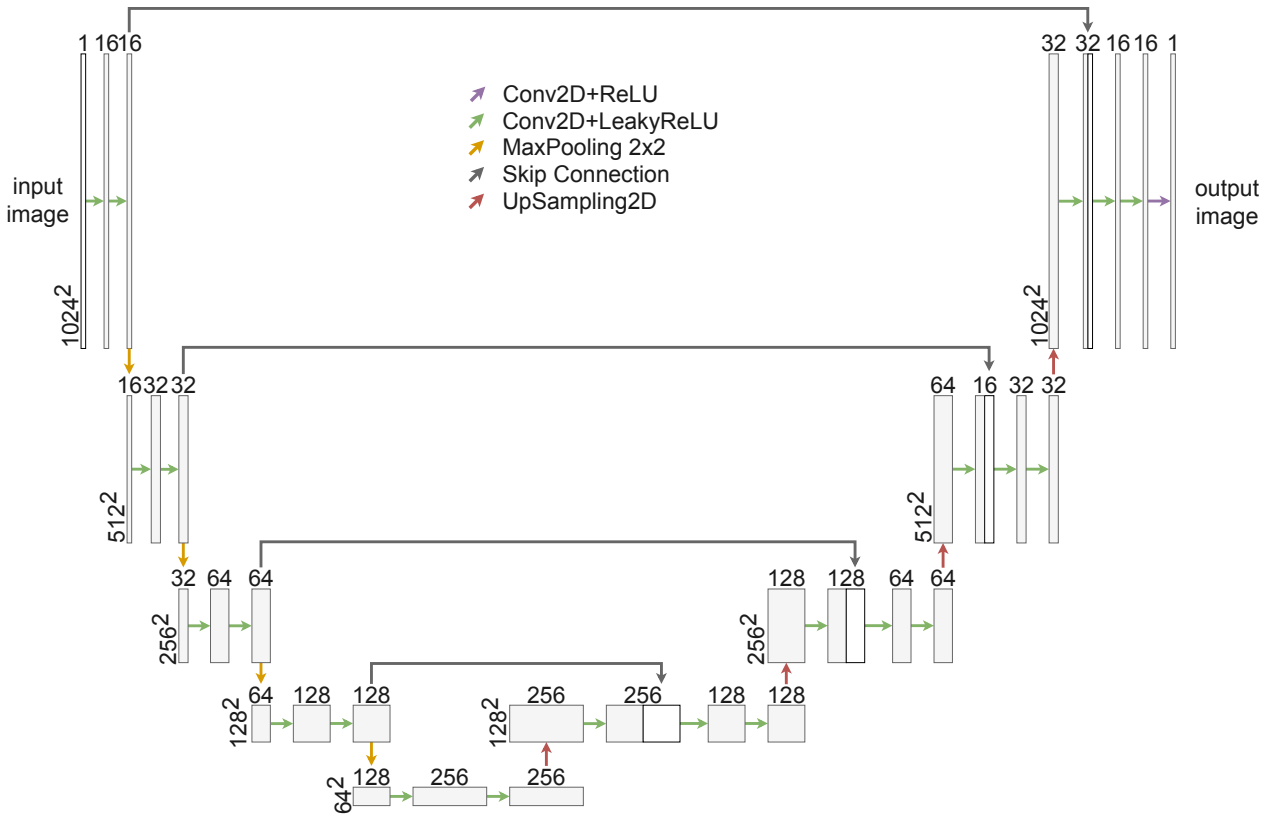


Figure 2: Model architecture

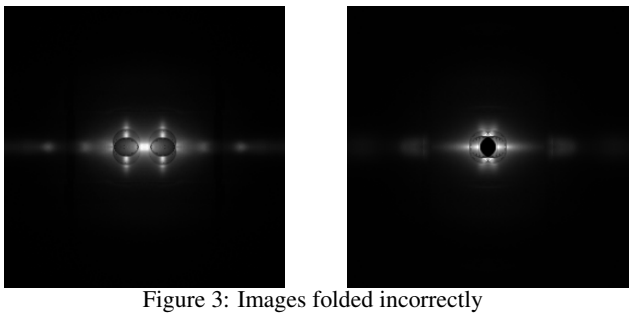


Figure 3: Images folded incorrectly

Table 1: Best validation metrics for trained models using unified normalization

Method	MAE	PSNR	SSIM
White-top-hats	0.9e-4	68.54	0.9998
Circularly symmetric	0.7e-4	61.01	0.9997
Smooth Gaussian	0.7e-4	64.77	0.9998
Mixed average	2.6e-4	51.63	0.9973

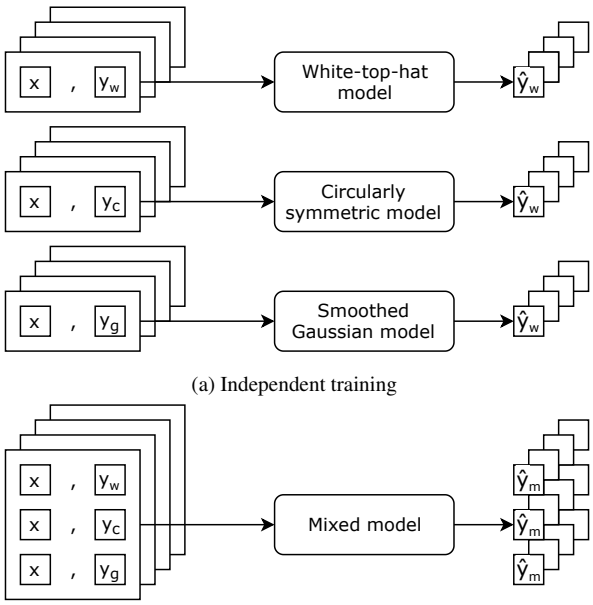
sian" and "smoothed boxcar" are the most similar, which leads to dropping "smoothed boxcar". We then observe the high similarity between "circularly symmetric" and "roving window". We discard the latter. Finally, there is some resemblance between the results from "white-top-hat" and "2D convex hull". With this, our final three algorithms that we use for training are: "white-top-hat", "smoothed Gaussian" and "circularly symmetric".

Having selected the methods that perform background subtraction in the most unique way, a model is trained for each of them independently (Figure 4a). Once the performance obtained by these models is sufficient, we additionally train a model using a dataset that contains all images produced by the three different background subtraction methods at the same time (Figure 4b). Effectively, every image is fed into the network three times, each one with a different ground truth. This mixed model is trained for three times the amount of epochs to compensate the increased dataset size.

As can be observed in the metrics present in Table 1, the proposed background subtraction network achieves results with very high similarity to the ground truth for all background subtracted data (subtracted using different methods). This is also true for the mixed model which is trained using data produced by a collection of algorithms.

From the table we observe that white top hat performed significantly better than the rest. This could be attributed to the way normalization is performed. In these models, we normalized both input and ground truth to the range [0-1] by dividing them by the maximum value present at the input. Intuitively using this scaling makes sense for an architecture designed to learn differences between input and output. Furthermore, we would be able to scale the output back to the range of the input, which would be useful during inference. This strategy benefited the "white-top-hats" model, as the ground truth images have pixel intensities in similar ranges to the input. Both "circularly symmetric" and "smoothed Gaussian" output much smaller values.

In Table 3 we show the metrics using independent normalization, specifically focusing on the center region of interest (^{c}), where differences between algorithms are more noticeable. This confirms that this area is what our models trained on "circularly



(a) Independent training

(b) Simultaneous training

Figure 4: Training strategies

Table 2: Best validation metrics for trained models using independent normalization

Method	MAE	PSNR	SSIM
White-top-hats	3.7e-4	62.96	0.9979
Circularly symmetric	2.8e-4	56.97	0.9985
Smooth Gaussian	2.7e-4	57.17	0.9989

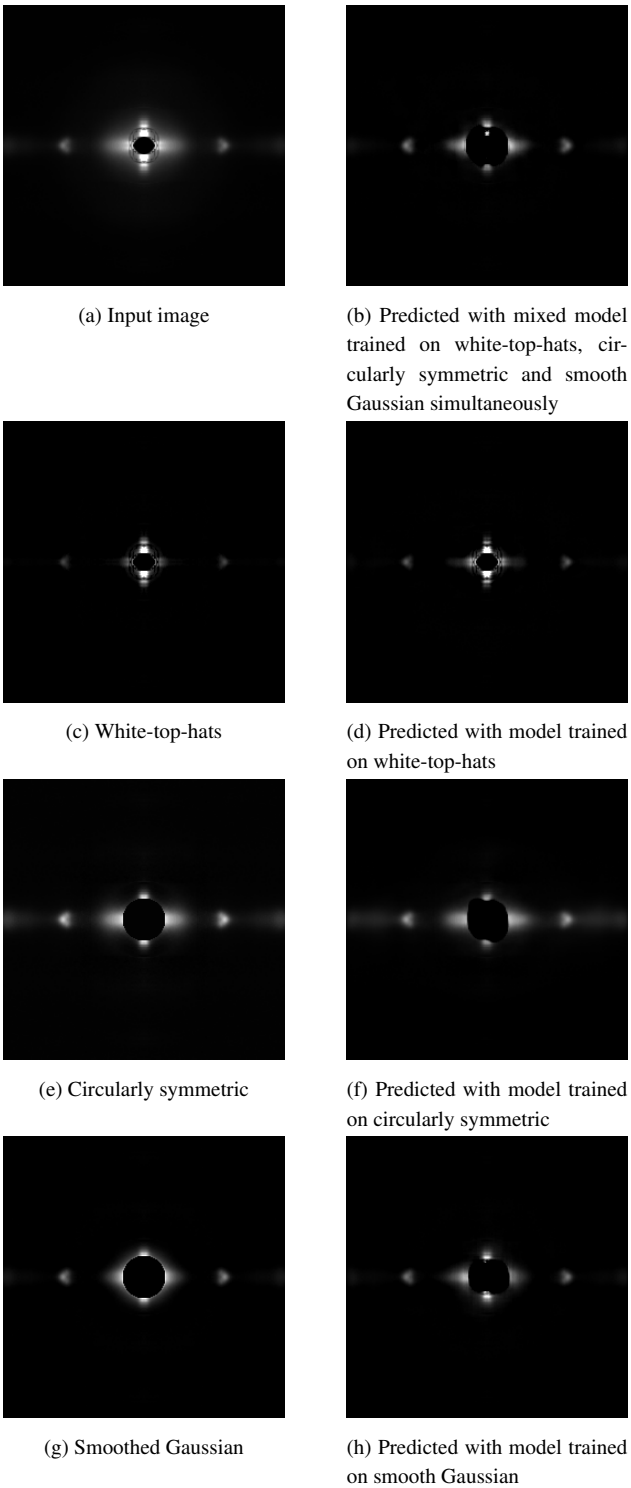
symmetric" and "smoothed Gaussian" are struggling to replicate.

In Figure 5 we show a visual comparison between output from all models given an input folded image with background taken from the test set.

Table 3: Best validation metrics for trained models using independent normalization (center region)

Method	MAE	PSNR	SSIM
White-top-hats^c	10.1e-4	51.31	0.9951
Circularly symm.^{c}	8.7e-4	48.60	0.9956
Smooth Gaussian^c	7.3e-4	47.94	0.9973
Mixed^c	38.3e-4	36.85	0.9504

With the visual results shown, we can confirm the models train the subtraction method they were trained on. We can bring attention to the mixed model, which shares characteristics with the others. The center region is mostly learned from the "circularly symmetric" and "smooth Gaussian". However, the equator line has the background removed more aggressively due to the influence of the "white-top-hats" images in the dataset. On the other hand, "white-top-hats" (both the algorithm and the trained model) tend to remove more from the diffraction points as well. The mixed model is more forgiving in that sense, somewhat resembling "circularly symmetric".



(a) Input image

(b) Predicted with mixed model trained on white-top-hats, circularly symmetric and smooth Gaussian simultaneously

(c) White-top-hats

(d) Predicted with model trained on white-top-hats

(e) Circularly symmetric

(f) Predicted with model trained on circularly symmetric

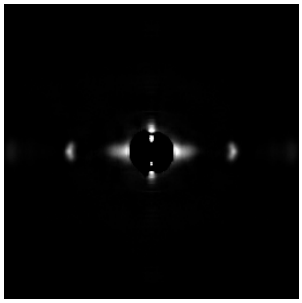
(g) Smoothed Gaussian

(h) Predicted with model trained on smooth Gaussian

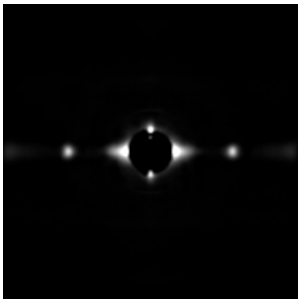
Figure 5: Comparison between algorithms and output images produced by different models. Pixel intensities adjusted for improved visualization.

Limitations

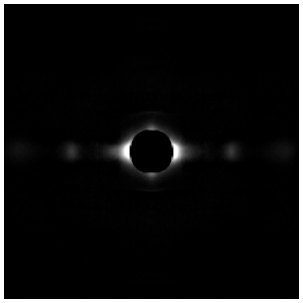
Due to the specific problem this model is aiming to solve, the data that is available for training is very limited. We have generated our training pairs using the parameters provided by MuscleX. Seeing that we are able to replicate the methods of background subtraction, ideally we would want to have a curated dataset with samples where every sample is the best possible image without background. Ideally, we would want this process to be fully auto-



(a) Predicted image closer to white-top-hats
PSNR: 30.96 (3.73 above average)



(b) Predicted image closer to circularly symmetric
PSNR: 31.92 (2.45 above average)



(c) Predicted image closer to smoothed Gaussian
PSNR: 31.12 (2.03 above average)

Figure 6: Cases where the mixed model performs the closest to every subtraction method

matic, but due to the lack of a quantitative measure of how good a method is removing background, this is not possible at the moment. The alternative would be to collect these images from experts that use the program. The next step would be to train on those images, so our model would be able to learn from the best possible image without background.

Another limitation of our used data is the cases where the folding fails. While we have taken measures to prevent the majority of them from getting into the datasets, it is still possible that inaccurate folds are present, contaminating the data.

Conclusion

In this paper we propose several models based on the U-Net architecture for background subtraction in diffraction X-ray images. We propose several modifications to the U-Net and propose several strategies to handle block patterns in the produced results. Experimental results on actual diffraction X-ray images show that we are able to replicate the performance of traditional background subtraction algorithms without having to manually adjust parameters. We are also able to learn from several traditional background subtraction algorithms simultaneously.

Acknowledgments

This project was supported by grant 9 P41 GM103622 (T.C.I.) from the National Institute of General Medical Sciences of the National Institutes of Health.

References

[1] J. Jiratrakanvong, X. Li, G. Nikseresht, J. Shao, M. Menendez, J. Li, W. Ma, G. Agam, and T. Irving, "biocatiit/musclex: Muscle x 1.14.12." `https://doi.org/10.5281/zenodo.3360909`, Aug. 2019.

[2] M. I. Ivanova and L. Makowski, "Iterative Low-Pass Filtering for Estimation of the Background in Fiber Diffraction Patterns," *Acta Crystallogr A Found Crystallogr*, vol. 54, pp. 626–631, Sept. 1998.

[3] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," *LNCS*, vol. 9351, pp. 234–241, 10 2015.

[4] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *in ICML Workshop on Deep Learning for Audio, Speech and Language Processing*, 2013.

[5] B. Xu, N. Wang, T. Chen, and M. Li, "Empirical evaluation of rectified activations in convolutional network," *CoRR*, vol. abs/1505.00853, 2015.

[6] L. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, pp. 834–848, Apr. 2018.

[7] A. Odena, V. Dumoulin, and C. Olah, "Deconvolution and Checkerboard Artifacts," *Distill*, vol. 1, p. 10.23915/distill.00003, Oct. 2016.

[8] R. Wang, Q. Zhang, C.-W. Fu, X. Shen, W.-S. Zheng, and J. Jia, "Underexposed Photo Enhancement Using Deep Illumination Estimation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6849–6857, 2019.

[9] K. H. Jin, M. T. McCann, E. Froustey, and M. Unser, "Deep Convolutional Neural Network for Inverse Problems in Imaging," *IEEE Trans. on Image Process.*, vol. 26, pp. 4509–4522, Sept. 2017.

[10] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. CVPR*, pp. 770–778, jun 2016.

[11] C. Chen, Q. Chen, J. Xu, and V. Koltun, "Learning to see in the dark," in *Proc. CVPR*, pp. 3291–3300, jun 2018.

[12] F. Chollet, "keras." `https://github.com/fchollet/keras`, 2015.

[13] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. J. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Józefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. G. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. A. Tucker, V. Vanhoucke, V. Vasudevan, F. B. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," *CoRR*, vol. abs/1603.04467, 2016.

[14] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *International Conference on Learning Representations*, 12 2014.

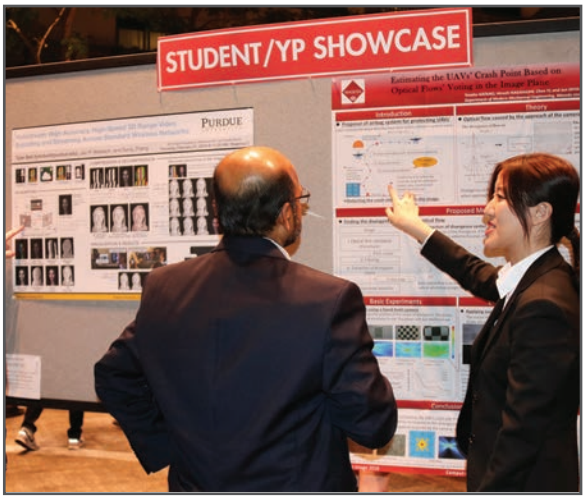
JOIN US AT THE NEXT EI!

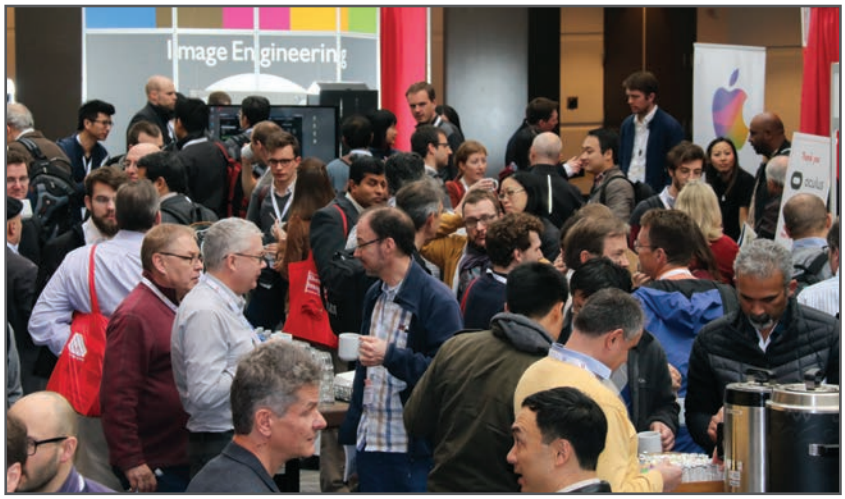
IS&T International Symposium on

Electronic Imaging

SCIENCE AND TECHNOLOGY

Imaging across applications . . . Where industry and academia meet!





• SHORT COURSES • EXHIBITS • DEMONSTRATION SESSION • PLENARY TALKS •
• INTERACTIVE PAPER SESSION • SPECIAL EVENTS • TECHNICAL SESSIONS •

www.electronicimaging.org

IS&T
imaging.org