Fast Particle Analysis Using Machine Learning

John Dang[1], **David Wang**[2], **Qianwen Zhong**[3], Yi Luo[2], Hyouarm Joung[2,3], and Aydogan Ozcan[2,3,4,5]

[1]Computer Science Department , [2]Bioengineering Department, [3]Electrical and Computer Engineering Department, [4]California NanoSystems Institute (CNSI), [4]Department of Surgery, David Geffen School of Medicine, University of California, Los Angeles, CA 90025 USA

Visit *innovate.ee.ucla.edu* and *org.ee.ucla.edu/hhmi* for more info

Abstract and Introduction

Monitoring coagulation of pathogens and antibody-labeled microbeads can give a simple but reliable detection solution to some diseases such like meningitis. Time lapse images can be captured to dynamically resolve the morphological change using a portable lens-free microscope. Although an in-line hologram enjoys the benefit of volumetric imaging and high portability, an additional auto-focusing process is required before any morphological analyzation. Also, while object detection and measurement is a solved question with many solutions, the computational complexity increases as a more precise analysis is performed. The large computational cost, as a result, puts restriction on the democratization of the detection method. Here we propose a potential solution using neural networks, efficiently resolving morphological properties while using much less computational time. A Matlab script was first applied to 840 pre-processed holographic images to analyze the number of particles and average diameter, which was later used as ground truth information. A convolutional neural network (CNN) and a recurrent neural network (RNN) were trained, using the same dataset, to perform morphological analysis. A tenfold processing time reduction is observed using the CNN instead of Matlab script, while performance is maintained at the same level.

Materials and Methods

A Matlab script is used for analysis of particle numbers. To process each image, a bilateral filter is first applied to both reduce noise and preserve edges. The bilateral filter is defined as:

$$I^{\text{filtered}}(x) = \frac{1}{W_p}\sum_{x_i\in\Omega} I(x_i) f_r(\|I(x_i) - I(x)\|) g_s(\|x_i - x\|),$$

and normalization term, W_p is defined as:

$$W_p = \sum_{x_i\in\Omega} f_r(\|I(x_i) - I(x)\|) g_s(\|x_i - x\|).$$

A Canny edge detection method is then applied to create a binary image with only the edge of each particle. In order to ensure detection of significant particles that do not have complete boundaries, edge linking was implementd. The edgelink function finds endings and junctions and creates a sparse matrix to mark them. Starting from these endpoints, the function tracks any unlabeled edges and looks for unlabeled pixels while simultaneously labeling image pixels. After identifying available endpoints, line segments are drawn between endpoints if possible, given the specified tolerance . If the maximum deviation is larger than the allowed tolerance, the line is shortened to the maximum deviation and the procedure is repeated. As a result, edges are essentially broken down into line segments. After linking the edges to form whole cell aggregates, any leftover gaps were further filled .

A modified version of MobileNetV2 is used here to measure particle diameter. A 512×512 image is first passed through a single depth-wise convolution and max-pooling to extract high level features, which is then passed to a MobileNetV2 module that outputs a feature vector. This feature vector is passed to a single neuron dense layer that outputs the predicted average diameter of particles in image. The loss function is calculated as loss = $(prediction - target)^2$. The network structure is shown in figure 1(a).

Materials and Methods (cont'd)

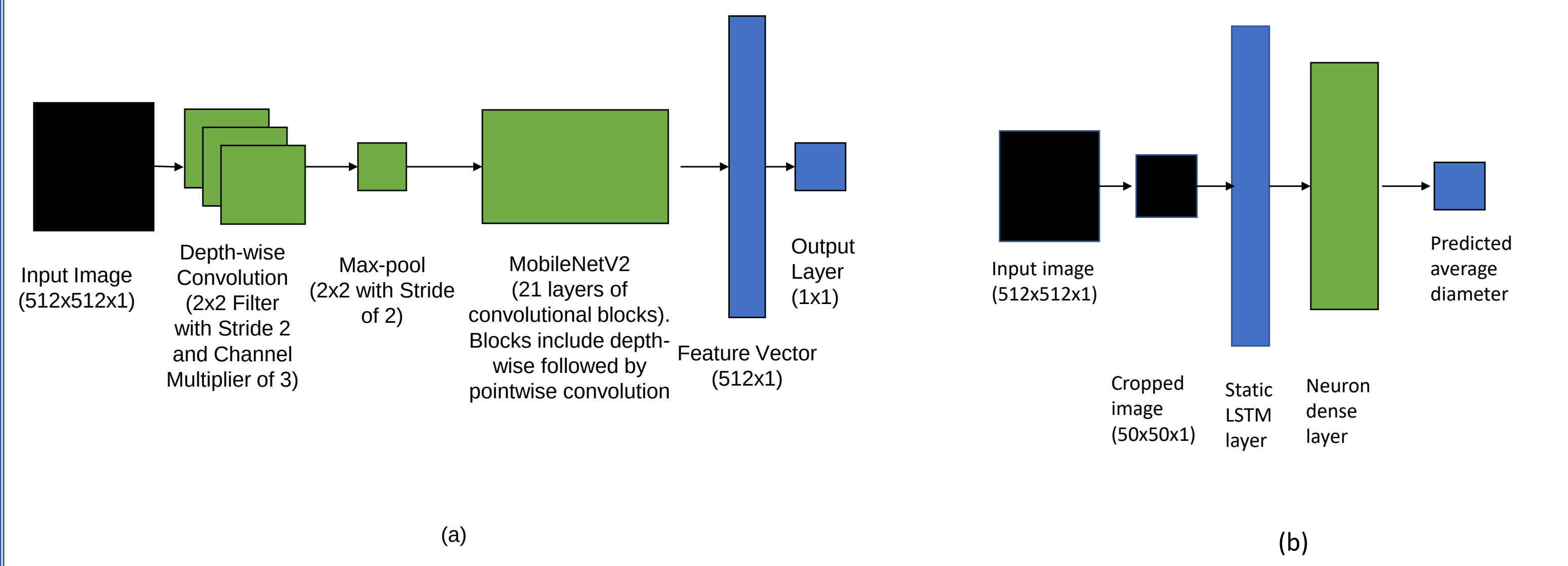
Input Image (512x512x1)
Depth-wise Convolution (2x2 Filter with Stride 2 and Channel Multiplier of 3)
Max-pool (2x2 with Stride of 2)
MobileNetV2 (21 layers of convolutional blocks). Blocks include depth-wise followed by pointwise convolution
Feature Vector (512x1)
Output Layer (1x1)
(a)
Input image (512x512x1)
Cropped image (50x50x1)
Static LSTM layer
Neuron dense layer
Predicted average diameter
(b)

Figure 1 Neural network architectures (a) CNN architecture based on pre-trained MobileNetV2 and (b) LSTM-RNN neural network.

After the CNN was trained, the Tensorflow checkpoint files were converted to a Tensorflow Lite model (flatbuffer .tflite file) for inference on a mobile device. The conversion was completed using the tensorflow.Lite.Converter module in Tensorflow 1.13 (Python 3.6) Tensorflow Lite converts regular Tensorflow operations to Tensorflow Lite Operations that are lower latency and smaller in size due to the smaller flatbuffer binary file format. Additionally, post-training quantization was applied to the model during the conversion process, which converts model weights to 8 bits of precision, reducing the model file size. These optimizations allow the model to be used for inference on devices with tighter computational and memory constraints, without sacrificing significant performance when compared to its desktop counterpart.

A Long-short-term-memory (LSTM) RNN was also applied to the same task. A 50×50 image is cropped from a 512×512 image and passed as an 2D-matrix input into a LSTM. The output is passed into a single neuron dense layer to obtain predicted diameter of all particles. The loss function is also defined as loss $= \sum(prediction - ground\ truth)^2$. The LSTM neural network is shown in figure 1(b).

Results

Matlab code is slow but accurate. By using a bilateral filter and an edge linking algorithm, incomplete boundaries present after reading in the image were successfully linked so that all major cell aggregates were detected. The average

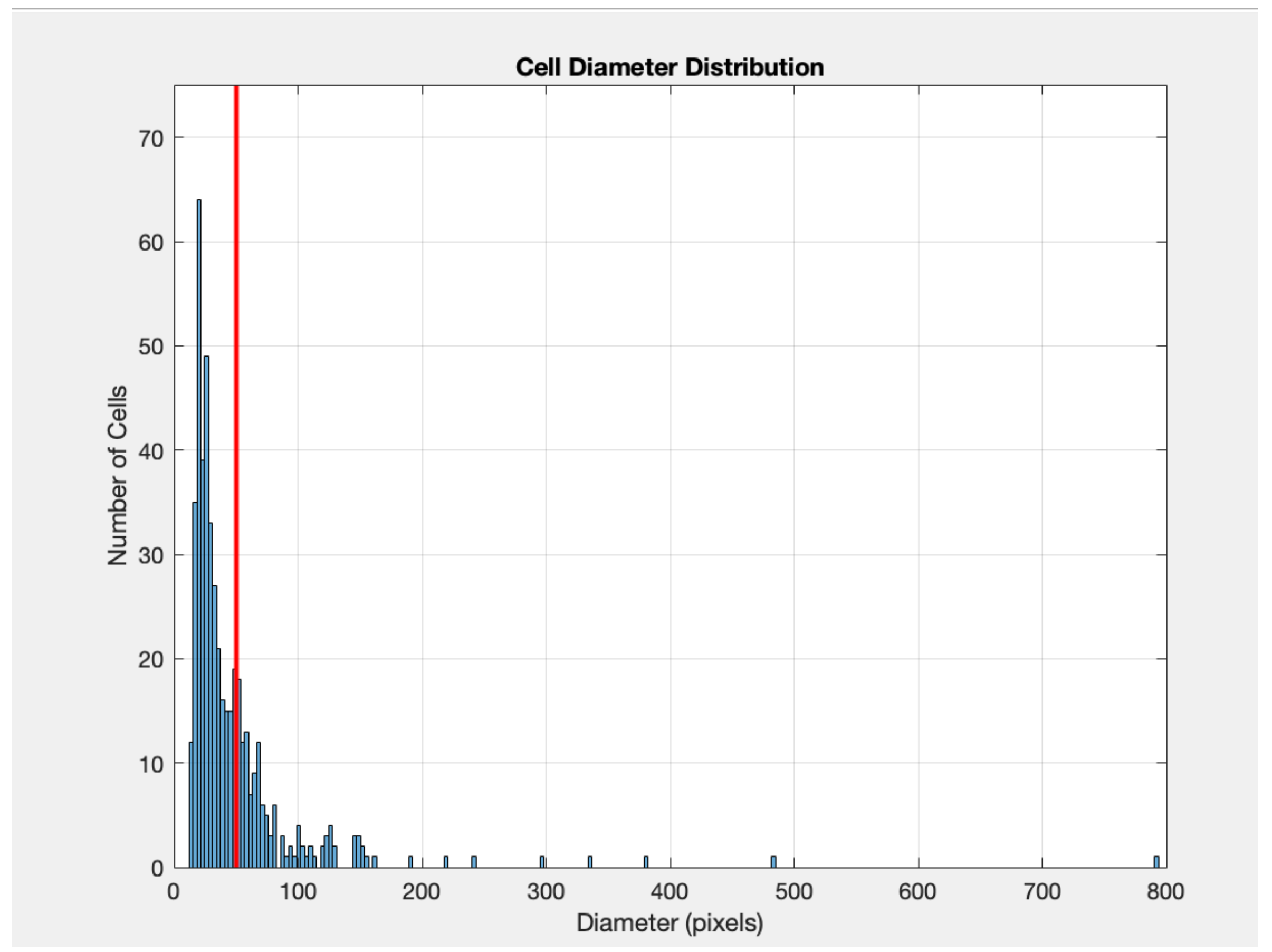

Figure 2 Matlab analyze results. The average diameter is shown as the red line.

Results (cont')

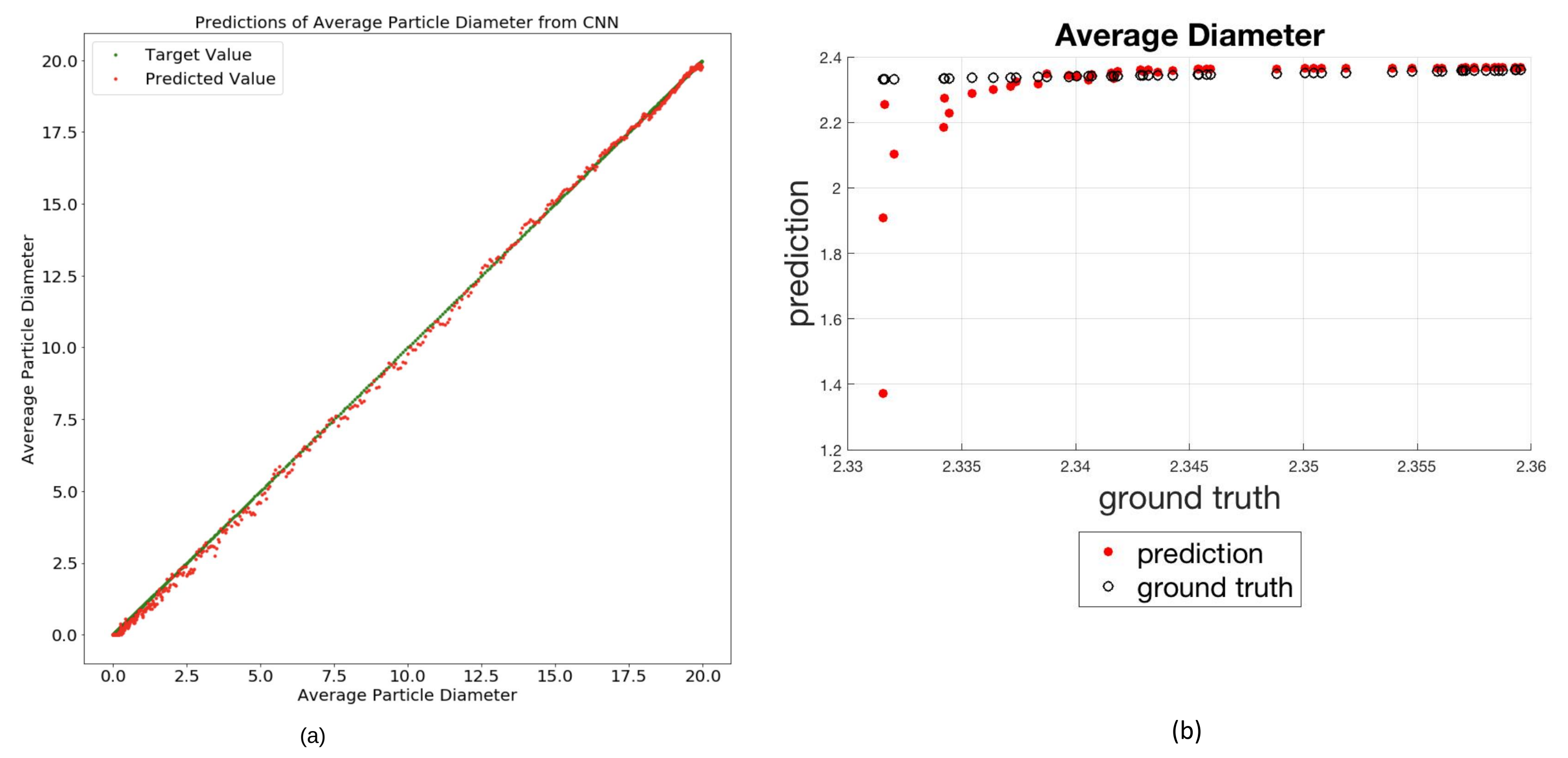

Figure 3 Neural network prediction vs. ground truth for (a) CNN and (b) LSTM-RNN.

diameter was found to be ~7.89 pixel, which is 8.11μm. The processing time for a 512×512 image is 277.19s.

But this may be affected by the presence of noise that was not properly filtered. With the current parameters, some larger cell aggregates appear to have been sectioned off into smaller aggregates, which will adversely affect the statistics generated.

Neural networks returned a faster and comparable result. On average, our CNN can make a prediction on an image in 0.01 seconds on a laptop and in 0.14 seconds on a Raspberry Pi 3. The model achieved 0.0125 mean-squared error loss on the test set, meaning on average diameter prediction is within 0.111. The mean of the average diameter data is 8.576 and standard deviation of average diameter data is 8.338. The CNN's prediction error is well below the standard deviation of the data, indicating that the network is expressive enough to accurately perform the particle sizing task. The quantized TF-lite module achieves predictions on average within 0.118 of the target. The original model had a total size of 8.5 megabytes on disk. After conversion to Tensorflow Lite and post-training quantization, the model size was reduced to 463 kilobytes. These results indicate that it is possible to achieve both high accuracy and efficiency for machine learning on mobile devices-- even on devices that cost only $40 like a Raspberry Pi.

Comparingly, an RNN performed slightly worse than the CNN, which is mainly resulting from lack of training data and insufficient training time. The network output is good when particle diameter is in the range of 2 to 5 micron. However, the performance degraded when particle size getting larger showing a sign of overfitting.

Acknowledgements

We thank the Howard Hughes Undergraduate Research Program, NSF PATHS-UP, and Vehbi Koç Foundation for their funding and support.