Lensless Image Classification for Machine-based Decision Making

Ganghun Kim
Department of Electrical and Computer Engineering
University of Utah

Brian Rodriguez
Department of Electrical and Computer Engineering
University of Utah

Rajesh Menon
Department of Electrical and Computer Engineering
University of Utah

***Abstract*— Deep Learning (DL) has accelerated advancements in Image Classification via convolutional neural networks (CNNs). However, these image classification tasks have been widely trained on images taken with typical cameras; human-centric images. Here, we present a CNN trained using data taken by a single CMOS image sensor with no lens. We created a dataset of lensless images comprised of handwritten digits taken from the MNIST dataset. Then, we trained a CNN on this dataset and were able to show that for 10 digits, the CNN is able to classify lensless images with 97% accuracy.**

I. Introduction

Wide-scale deep learning algorithms have pushed image classification tasks to its limits. State of the art architectures have been able to classify human-centric images with astonishing accuracy, but human-centric images require expensive cameras [1], [2]. Recently, lensless imaging has been gaining traction [3]–[5]. These lensless images are taken using a low cost and light-weight image sensor without the use of a camera lens. Previously, we demonstrated that machine learning algorithms can accurately classify lensless images for 2 digits with 99% accuracy using support vector machines (SVMs) [6]. Here, we extend this idea and use DL to accurately classify lensless images for 10 digits. Moreover, we trained a CNN on a dataset of lensless images that we captured using a lensless camera. Our lensless camera consists of a simple

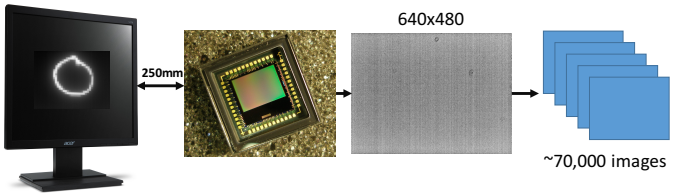

Fig. 1: Lensless imaging process from left to right. The process begins with an LCD screen that displays an image of a digit then the image sensor captures an image of the screen. These images are collected into a dataset and used as input for the neural network.

CMOS image sensor. We used a liquid-crystal display (LCD) that displayed images of hand-written digits taken from the open-sourced MNIST dataset [7]. The LCD rests 250mm away from the CMOS sensor with an exposure time of 150ms and averaged over 100 frames to reduce noise. Using this procedure, we captured 70,000 lensless images and created a dataset with appropriate labels. These lensless images are 640x480 (307,200 pixels per image) which for scalability purposes are resized to 240x240. Our results demonstrate that CNNs can accurately classify multiple classes of data taken directly from a lensless camera.

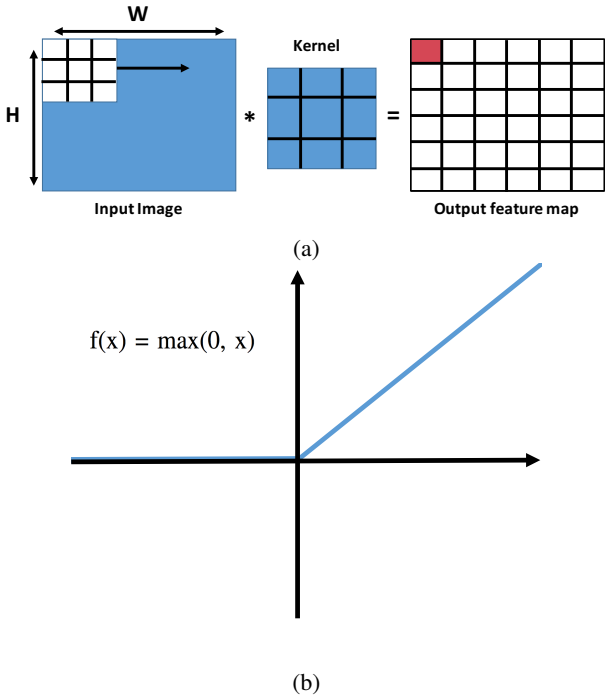

Fig. 2: (a) Example of the convolution operation with a 3x3 receptive field where one stride to the right corresponds to one scalar output on the feature map (filter). (b) Shows the graph of the ReLU activation function. The function takes the max of either 0 or x. This is important because this means that the neuron is either on or off.

II. Background

Convolutional neural networks are widely used for object recognition, computer vision, and image classification tasks. Recent advancements such as YOLOv3, a real-time object detection network, and Inception-v4, a large-scale image classification network, have set high expectations for proceeding networks [2], [8]. While these networks occupy different subfields in DL, most convolutional neural networks

use the same operations to perform classification. Ideas such as convolution, pooling, and activation functions are the core of a convolutional neural network.

Convolution (Conv) is an operation that takes in an input image or filter and performs a dot product within the scope of its receptive field which corresponds to a scalar point on an output feature map (filter) as shown in Figure 2a. A receptive field is the area in which the convolution operation performs its dot product on. In Figure 2a the receptive field is a 3x3 area as denoted by the white 3x3 box. After it does a dot product on the first section the receptive field slides to the right and performs another dot product until it reaches the end of the image.

Since convolution is a linear operation, using an activation function allows a neural network to learn nonlinear relations in data. In this paper we utilize the widely used Rectified Linear Unit activation function (ReLU) refer to Figure 2b [9]. Maxpooling is a type of pooling operation which is more notably used for nonlinearly down-sampling its input. Depending on the receptive field size of the pooling operation, it takes a max filter over the receptive field and takes the highest value in the receptive field as shown in Figure 3a.

After a series of convolutions and maxpooling layers it is typical to see either one or two fully-connected layers (FC) at the end of a network. An FC layer is what a CNN utilizes to make high-level predictions based on the data. Every neuron in an FC layer is connected to all of the previous layer's activations allowing the network to "see" what high-level features the convolution layers extracted from the data.

Other important concepts regarding neural networks are back-propagation, mini-batch size and learning rate. Back-propagation is the backbone of how a neural network is able to learn [10]. Back-propagation is a type of gradient descent optimization algorithm that propagates an error value calculated using a loss function. It takes this error value and computes the partial derivative with respect to a parameter value to calculate the gradients of all the parameters in the network. This allows the network to learn values for all of its parameters to achieve a global minimum that will make the error value the smallest.

A mini-batch size is the amount of data inputs a neural network will "see" before it computes an error value. This allows the network to learn in small incremental steps, rather than learning over an entire dataset. Learning rates impact parameter updates. The bigger the learning rate the more a parameter will change but the lower the learning rate the less a parameter will be impacted by an update from back-propagation.

III. Network Architecture

We propose a CNN architecture that is able to classify lensless images at a 97% accuracy. Our architecture is comprised of two main sections; feature section and a classifying section. The feature section has 4 subsections containing convolutional layers and max pooling layers. Each convolutional layer is followed by batch normalization and a ReLU activation function [9], [11]. The 1x1 convolutional

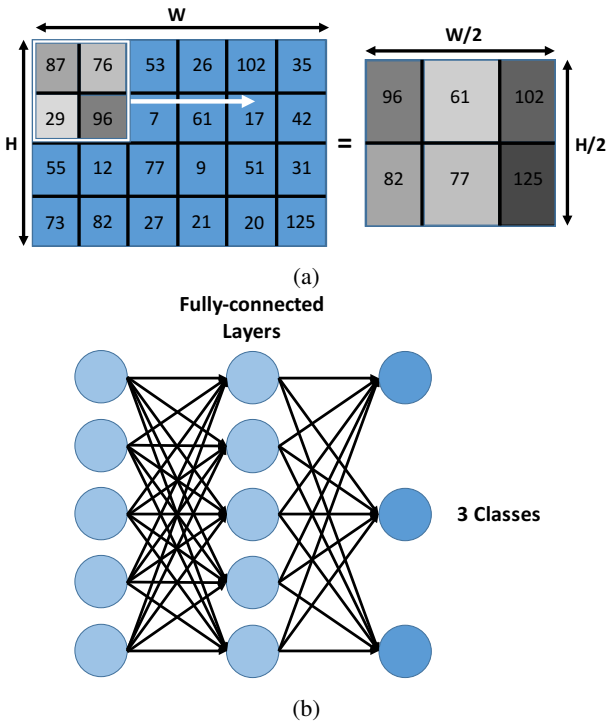

Fig. 3: (a) An example of a maxpooling operation. In this example we use a 2x2 receptive field and a stride of 2 to down-sample the image in half. (b) Example of a fully-connected layer. All of the connections in the previous layer are connected to the next layer's neurons, in this case, there are two fully-connected layers where the last fully-connected layer is connected to the three classes of an arbitrary dataset.

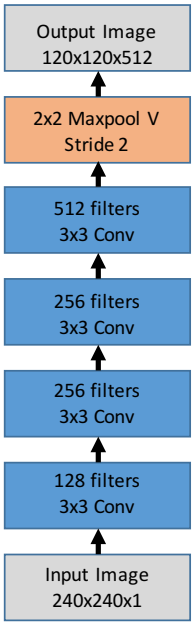

Fig. 4: Section 1 of the Convolutional Neural Network. Input images are sized to 240x240 and gray-scaled.

layers are used for dimensionality reduction, these layers follow every pooling layer except for the last [12]. The 1x1

convolutions allowed for our network to converge faster.

Following the feature section is a classifying section. This section contains two fully-connected layers, where the last fully-connected layer classifies between the 10 different classes. After the fully-connected layers is a softmax function that compresses the output scores of the CNN into values ranging from 0 to 1 where all of the values sum up to 1. This is important in multi-class classification problems since the softmax function outputs action probabilities relating to each class. [13], [14].

Overfitting is a problem in neural networks that impacts the validation accuracy, it's a problem where the neural network memorizes the input data instead of learning the required features to make accurate predictions. To combat this problem we employed dropout on our final fully-connected layer with a probability of .5 [15]. Dropout is an idea that "turns off" or more accurately zeroes out neurons which forces the network to learn even more connections between a layer and in our case, the layer before a prediction is made.

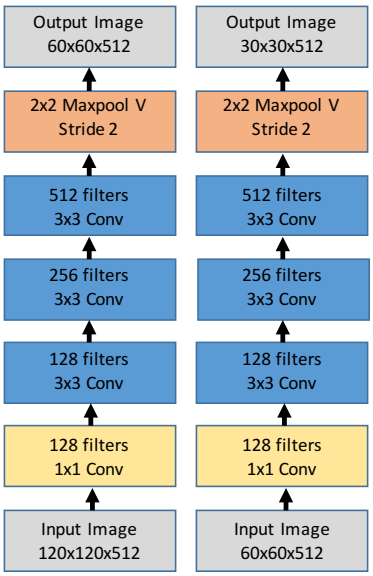

Fig. 5: Section 2 (Left) and Section 3 (Right) of the network. These two sections are identical except for their output dimensions of the images.

IV. Training Methodology

We created our network using Pytorch, a highly extensible deep learning framework [16]. Our network was trained using stochastic gradient descent with a momentum of .9 on a single NVidia Tesla V100 GPU [17], [18]. We used a learning rate of .001 and a mini-batch size of 16. The learning rate was decayed when the loss plateaued; decaying one time over 40 epochs. The input data are gray-scaled (by default) and the images are resized to 240x240. 5,400 images were taken from each class and split into two groups, training and testing groups; 3,200 training images and 2,200 testing images. Testing images are used to benchmark our model and test for overfitting in the network.

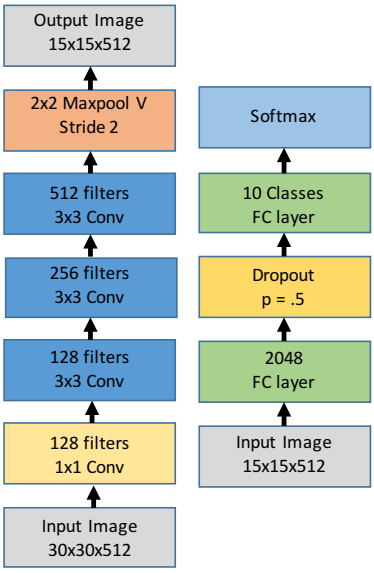

Fig. 6: Final feature section (Left) and the classification section (Right) of the network.

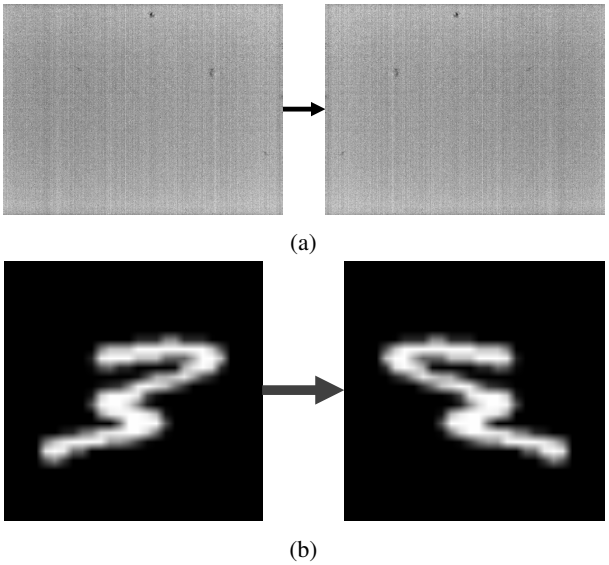

Fig. 7: (a) Horizontally flipped lensless image. (b) Shows what the data would look like if the network was using human-centric images as input.

V. Results

A. Network Configurations

We tested a lot of different configurations for our final network. Our earlier experimental networks actually achieved higher accuracy than some of our later networks. Our later networks employed ideas such as Alpha Dropout and SELU activation functions [19]. We noticed that using extra learnable parameters actually hindered our models ability to learn. The highest accuracy we found using these new methods of learning was 82%.

Our final network was able achieve a 97% accuracy and converged in fewer epochs. Meaning, the network was able to

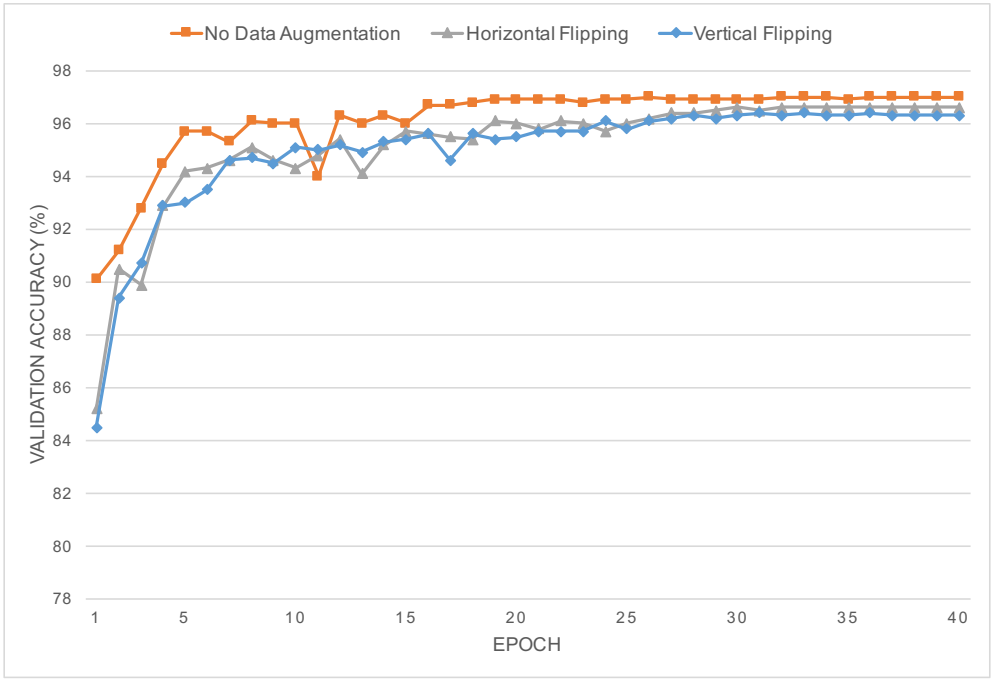

Fig. 8: Comparison between results for no data augmentation, horizontal flipping and vertical flipping.

learn the necessary features quicker, in less time. While other gradient optimizers were tested such as Adam and Adagrad they did not converge accordingly [17], [20]. We found that in all of our experiments, stochastic gradient decent was the best optimizer and allowed our network to converge to its global minimum.

B. Experimental Data Augmentation

Data augmentation is a widely used technique to artificially create more samples of data while training [21]. We conducted a small test that horizontally and vertically flips the input data at a rate of 50% shown in Figure 7. Theoretically, creating more data samples should allow our network to learn more features and prevent overfitting in the network, but this idea has been applied only to human-centric images, that we know of. In our tests, we found that the network performed slightly worse. Since these images are indecipherable by humans, we are not surprised that some data augmentation would hinder our network's ability to learn the necessary features to make accurate predictions. The margin between the results is minimal as shown in Figure 8.

VI. Conclusion

Our results show promise for lensless imaging and a cost effective alternative for cameras in machine-based decision making.

References

[1] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2014.

[2] C. Szegedy, S. Ioffe, and V. Vanhoucke, "Inception-v4, inception-resnet and the impact of residual connections on learning," *CoRR*, vol. abs/1602.07261, 2016.

[3] M. S. Asif, A. Ayremlou, A. C. Sankaranarayanan, A. Veeraraghavan, and R. G. Baraniuk, "Flatcam: Thin, bare-sensor cameras using coded aperture and computation," *CoRR*, vol. abs/1509.00116, 2015.

[4] P. R. Gill and D. G. Stork, "Lensless ultra-miniature imagers using odd-symmetry spiral phase gratings," in *Imaging and Applied Optics*, p. CW4C.3, Optical Society of America, 2013.

[5] G. Kim, K. Isaacson, R. Palmer, and R. Menon, "Lensless photography with only an image sensor," *CoRR*, vol. abs/1702.06619, 2017.

[6] G. Kim, S. Kapetanovic, R. Palmer, and R. Menon, "Lensless-camera based machine learning for image classification," *CoRR*, vol. abs/1709.00408, 2017.

[7] Y. LeCun and C. Cortes, "MNIST handwritten digit database," 2010.

[8] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *CoRR*, vol. abs/1804.02767, 2018.

[9] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, (USA), pp. 807–814, Omnipress, 2010.

[10] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Neurocomputing: Foundations of research," ch. Learning Representations by Back-propagating Errors, pp. 696–699, Cambridge, MA, USA: MIT Press, 1988.

[11] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *CoRR*, vol. abs/1502.03167, 2015.

[12] M. Lin, Q. Chen, and S. Yan, "Network in network," *CoRR*, vol. abs/1312.4400, 2013.

[13] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2006.

[14] R. S. Sutton and A. G. Barto, *Introduction to Reinforcement Learning*. Cambridge, MA, USA: MIT Press, 1st ed., 1998.

[15] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, 2014.

[16] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," in *NIPS-W*, 2017.

[17] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014.

[18] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, "On the importance of initialization and momentum in deep learning," in *Proceedings of the 30th International Conference on International Conference on*

Machine Learning - Volume 28, ICML'13, pp. III–1139–III–1147, JMLR.org, 2013.

[19] G. Klambauer, T. Unterthiner, A. Mayr, and S. Hochreiter, "Self-normalizing neural networks," *CoRR*, vol. abs/1706.02515, 2017.

[20] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," Tech. Rep. UCB/EECS-2010-24, EECS Department, University of California, Berkeley, Mar 2010.

[21] S. C. Wong, A. Gatt, V. Stamatescu, and M. D. McDonnell, "Understanding data augmentation for classification: when to warp?," *CoRR*, vol. abs/1609.08764, 2016.