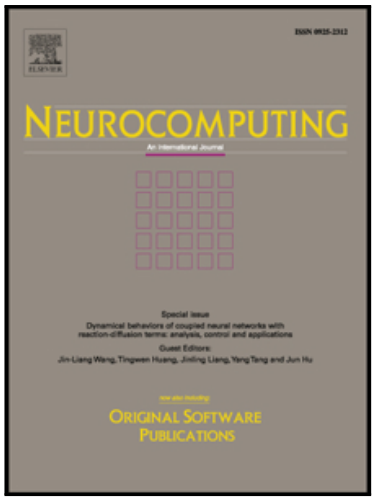
Journal Pre-proof



LSHR-Net: a hardware-friendly solution for high-resolution computational imaging using a mixed-weights neural network

Fangliang Bai, Jinchao Liu, Xiaojuan Liu, Margarita Osadchy, Chao Wang, Stuart J. Gibson

PII: S0925-2312(20)30571-3
DOI: https://doi.org/10.1016/j.neucom.2020.04.010
Reference: NEUCOM 22165

To appear in: *Neurocomputing*

Received date: 15 November 2019
Revised date: 18 February 2020
Accepted date: 3 April 2020

Please cite this article as: Fangliang Bai, Jinchao Liu, Xiaojuan Liu, Margarita Osadchy, Chao Wang, Stuart J. Gibson, LSHR-Net: a hardware-friendly solution for high-resolution computational imaging using a mixed-weights neural network, *Neurocomputing* (2020), doi: https://doi.org/10.1016/j.neucom.2020.04.010

LSHR-Net: a hardware-friendly solution for high-resolution computational imaging using a mixed-weights neural network

Fangliang Bai[a], Jinchao Liu[b], Xiaojuan Liu[d,e], Margarita Osadchy[c], Chao Wang[d], Stuart J. Gibson[a]

[a]*School of Physical Sciences, University of Kent, Canterbury, Kent, UK, CT2 7NH*
[b]*College of Artificial Intelligence, Nankai University, Tianjin, China, 300071*
[c]*Department of Computer Science, University of Haifa, Mount Carmel, Haifa, Israel*
[d]*School of Engineering and Digital Arts, University of Kent, Canterbury Kent, UK, CT2 7NT*
[e]*School of Physics and Optoelectronics Engineering, Shandong University of Technology, Zibo, China, 255049*

Abstract

Recent work showed neural-network based approaches to reconstructing images from compressively sensed measurements offer significant improvements in accuracy and signal compression. Such methods can dramatically boost the capability of computational imaging hardware. However, to date, there have been two major drawbacks: (1) the high-precision real-valued sensing patterns proposed in the majority of existing works can prove problematic when used with computational imaging hardware such as a digital micromirror sampling device and (2) the network structures for image reconstruction involve intensive computation, which is also not suitable for hardware deployment. To address these problems, we propose a novel hardware-friendly solution based on mixed-weights neural networks for computational imaging. In particular, learned binary-weight sensing patterns are tailored to the sampling device. Moreover, we proposed a recursive network structure for low-resolution image sampling and high-resolution reconstruction scheme. It reduces both the required number of measurements and reconstruction computation by operating convolution on small intermediate feature maps. The recursive structure further reduced the model size, making the network more computationally efficient when deployed with the hardware. Our method has been validated on benchmark datasets and achieved state of the art recon-

struction accuracy. We tested our proposed network in conjunction with a proof-of-concept hardware setup.

Keywords:
single pixel camera, computational imaging, neural network, image reconstruction, super resolution, binary weights

1. Introduction

In the context of structural signal recovery, the task of image reconstruction from the compressive sampling has been closely associated with computational imaging [1] using a single pixel camera [2, 3]. Single pixel camera architectures are of particular interest when imaging outside the visible range of the electromagnetic spectrum in cases where detector technology is expensive or difficult to manufacture. This approach to image acquisition involves illuminating an object scene using a sampling device which produces structured light in the form of 2D pseudo-random patterns. For each pattern, the intensity of the back scattered light is measured by a single pixel photo-detector. In the computational imaging paradigm [2], each measurement corresponds to the inner product between a sensing pattern and the image to be reconstructed. This can be formulated as:

$$y = \Phi x + e \tag{1}$$

where $x \in \mathbb{R}^n$ is the image rearranged as a vector, $\Phi \in \mathbb{R}^{m \times n}$, $m \ll n$, are m random sensing patterns (also concatenated into vector form), $e \in \mathbb{R}^m$ are measurement errors and $y \in \mathbb{R}^m$ are the measurements. The number of sensing patterns m can be much fewer than the total number of pixels n comprising the reconstructed image, resulting in a measurement ratio of $R = \frac{m}{n}$.

A digital micro-mirror device (DMD) is widely used as the sampling component in single pixel camera architectures and for coded aperture imaging [4, 5, 6, 7, 8, 9]. It contains a 2d array of micro-mirrors (hence the name) and each micro-mirror can be positioned at one of two angles to be in either an activated or inactivated state. When the array is illuminated by a uniform light source, shifting the micromirrors between states produces different binary sensing patterns, such as random Bernoulli, Hadamard, which are projected onto the object scene of interest. Given an incident, uniform,

light source, shifting mirrors between states produces different binary sensing patterns, such as random Bernoulli, Hadamard, which are used to illuminate the object scene of interest.

To reconstruct signals/images from compressively sampled measurements, Compressed sensing (CS)[10, 11], to be exact sparse optimization methods such as NESTA[12], ADMM [13] etc. have been proposed and have become the predominant algorithms using in a variety of applications. However, one major drawback of these numerical nonlinear optimization methods is that they often take a few minutes to recover a single large image at good quality.

Deep neural networks (DNNs) have become prevalent in a broad range of image processing tasks [14, 15, 16, 17, 18]. Specifically, DNN has been shown to achieve favorable results in image recovery [19]. Motivated by this success in image reconstruction tasks, DNNs were subsequently investigated for image reconstruction problems based on compressively sensed image data, [20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30]. These neural network based solutions were reported to outperform the state-of-the-art in compressed sensing algorithms in terms of speed, accuracy and data compression.

Although a variety of different network architectures were proposed, few were deliberately designed to be adaptable to the sensing hardware. To date, there have been two issues that remain to be solved. First, the real-valued sensing patterns of all existing neural network implementations for this application were stored in 32-bit floating-point format. Although high-precision sensing patterns can be used for software simulation of image sampling on modern GPUs, this is not a realistic representation of sampling using structured light sensing hardware, where instead binary patterns are used to reduce sampling complexity. Second, previous methods assumed that the sensing patterns and the reconstructed images have the same resolution. Therefore, the size of the recovered image is dependent on the size of the sensing patterns (for dense-connection based methods) or the number of convolutional patch-sampling operations (for convolutional-based methods). For large images, these methods result in large intermediate feature maps and increase the number of operations required for recovering an image. This is because the number of sampling measurements and convolutional computations depends on the size of the feature maps. In addition, when the patterns are loaded in hardware, such as a DMD, the maximum reconstruction resolution will be limited by the size of the mirror array (which is fixed) used in the sensing device.

The limitations of previous methods motivated us to design a hardware-

friendly deep learning solution, incorporating binary sensing patterns to reconstruct high-resolution images. Previous papers have highlighted the importance of integrating the DNN solutions with hardware [28, 29]. In this respect, we go one step further than previous work and provide evidence that our architecture performs well with imaging hardware. We propose a new network architecture that:

1. Uses a mixed-weights network with sparse binary patterns which lends itself naturally to hardware implementation and can be trained in an end-to-end manner. Unlike floating-point numbers, binary patterns are appropriate for both sampling and measuring hardware. Specifically, the sparse binary patterns can be represented on a DMD without the need for any additional modulation and require less on-board memory usage. Our approach effectively increases the light intensity sensitivity of the single pixel camera (the photo-diode) and the analogue to digital conversion range, compared with methods based on real-valued sensing patterns.
2. Uses a novel sensing-reconstruction scheme, which we term low-resolution sensing with high-resolution reconstruction (LSHR), to directly reconstruct high-resolution images from low-resolution sampled measurements. Given a pattern generated by a DMD of fixed size, the network reconstructs a high-resolution image which has more pixels than the number of micro-mirrors in the array. This low-throughput sampling scheme results in smaller feature maps, and therefore, fewer computational operations are required. Hence, it is more efficient than previously reported methods for use with hardware imaging set-ups.
3. Has a residual-correction sub-net that consists of a chain of recursive residual blocks, where weights are shared between different blocks. Compared with previous methods, our structure further reduces the model size, making it ideal for the limited onboard memory capacity of the hardware (e.g. single pixel camera) while yielding higher reconstruction PSNR accuracy.
4. Achieves state-of-the-art results on benchmark datasets and has been validated on proof-of-concept hardware.

The remainder of this paper is organized as follows: In Section 2, we review the related work on sensing patterns. We describe the design of our proposed network in Section 3. In Section 4 we show software simulation results for our model and compare them with existing methods. In Section 5, we present

the work of integrating the model with hardware. Finally, in Section 6 we conclude our discussion and suggest potential future directions for the work.

2. Related work

The concept of neural network based image reconstruction was first implemented using a fully-connected network [20]. Thereafter, the problem was approached using convolutional neural networks which avoid the fixed size input image constraint. We organized the related methods [20, 24, 25, 22, 21, 23, 26, 27, 31, 32, 28, 29, 30] into three categories according to the type of sensing pattern used (randomly generated, learned and binary) and discuss relevant prior work below.

Networks based on pre-generated (static) patterns. A stacked denoising auto-encoder (SDA) was previously implemented [20] comprising fully-connected layers. It was trained with measurements acquired by sensing images with pre-generated random Gaussian patterns. Inspired by SDA, ReconNet [24] was subsequently proposed. It improved the accuracy by extending the network with additional convolutional layers of different kernel sizes. However the fully-connected layer caused heavy computation and large model size, the sensing area was constrained to small patches of the original image. In the post-processing step, the reconstructed small patches were concatenated to form the whole image. The BM3D [33] was then applied to smooth the edges between patches. The performance of the ReconNet was further improved by DR^2-Net [25]. Here the convolutional layers were replaced with residual blocks which make the network easier to train. But the sensing was still done in small patches. In contrast to previous methods that used fixed (pre-generated) Gaussian sensing patterns, DeepInverse [22] used real time generation of random patterns for sampling images.

Networks based on learned patterns. Some of the work described in the previous paragraph has been modified such that the sensing patterns adapt to a particular set of images through a learning process. The SDA was further adapted to learn the patterns with a fully-connected layer that inputs an image x directly into the network. The fully-connected layer was trained to obtain the measurements y when presented with x. This operation can be represented as $y = \sigma(Wx + b)$ where the $\sigma(\cdot)$ is an activation function and W and b are the weights and bias of the fully-connected layer. A similar structure to SDA was also proposed that employed a fully-connected neural network to implement the block-based compressed sensing [21]. The model

was trained to jointly optimize the sensing patterns and the network weights. DeepInverse was also optimized resulting in a new model named DeepCodedec [23]. It had an encoder-decoder architecture. The network was trained to take measurements from images using several convolutional layers. Unlike SDA, it gradually reduced the dimension of the intermediate feature maps prior to generating the measurements. The efficiency was improved by applying convolutional layers. The ReconNet was also further improved using learned patterns [26] and [28]. Before training, the fully-connected layer was initialized with random Gaussian patterns. It was then updated during the training. For testing the network, the trained patterns were fixed to perform the sensing. The results showed further improvements in reconstruction accuracy due to learning the patterns. However, the fully-connected layer caused intensive computation and blocking artifacts to appear in the reconstructed images. To deal with the aforementioned limitations, the authors proposed two networks, [27] and [29], that sensed images with a convolutional layer with a small stride step to avoid the blocking artifacts.

Networks based on a binary matrix. Neural networks with binary weights were initially designed for image classification tasks, [34, 35]. A network for video reconstruction, using binary patterns, was described in [31]. The network applied a 3D binary sampling matrix to down-sample a sequence of the temporal video frames and learned a non-linear rule, mapping between the measurements and the reconstructed frames via fully-connected layers. In more recent network, DeepBinaryMask [32], followed the same strategy of using a binary down-sampling matrix for sensing video frames but introduced a learning procedure for generating the masks. However, their work focused on temporal compression which is functionally different from the spatial compression task which is the focus of our work. Inspired by the SDA, a network with an improved architecture was proposed to implement the CS image reconstruction [30]. Differently from previous reconstruction methods, it is initial reconstruction consisted of multiple 1×1 convolutions and a reshape operation. The 1×1 convolution, in principle, is functionally equivalent to a fully-connected layer, which fixed the reconstructed image size. After the convolution, the reconstructed 1D vector was reshaped into an initial 2D image. In this work, they experimentally tested their model with binary weights and bipolar weights for image sampling. However, the simple replacement of sampling patterns did not involve the optimization of the overall network. The reported results indicated that the reconstruction accuracy of these two types of weights was sub-optimal compared with their

floating-points-based model.

In Section 3, we describe our own network architecture, which aims to solve the aforementioned limitations of the existing methods.

3. Overview of the proposed network

In this section, the network structure is explained in detail. The architecture is shown in Figure 1. It is functionally divided into two parts, i.e. the ***image reconstruction sub-net***, and the ***residual correction sub-net***.

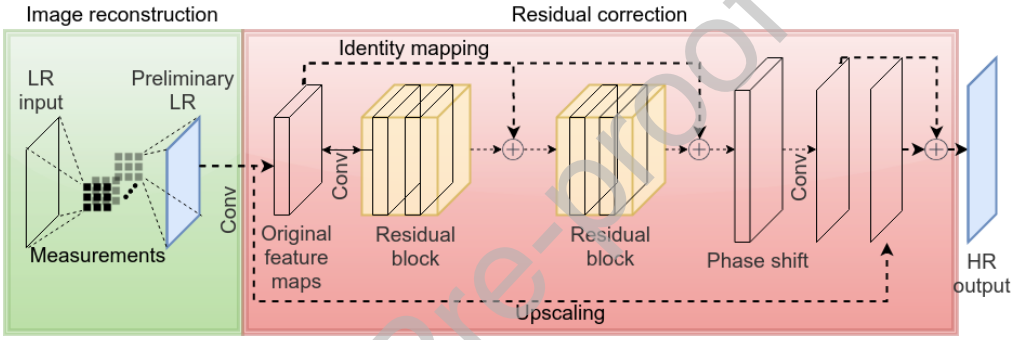


Figure 1: The schematic of the proposed network. Our Network has two parts: one that performs image reconstruction and a second part that determines the residual correction. For the image reconstruction part, the network compressively senses the low-resolution input image with static or learned binary patterns and reconstructs the preliminary image. After that, the residual correction sub-net extracts the features from the preliminary image and corrects the reconstruction error using a sequence of recursive residual blocks. Each of these blocks is connected to the original feature maps through identity branches to gradually learn the errors. Then the preliminary image and residuals maps are upscaled through two branches and combined element-wise to generate the final output image.

Our LSHR scheme assumes an object scene is sampled with low-resolution patterns. In practical applications, ground truth, high-resolution, images are not known a priori. During the training stage, we use the original images as our ground-truth and resample these at low resolution for the purposes of simulating image quality typical of current single pixel imaging systems. These low resolution and ground truth image pairs are used to train our network.

The image reconstruction sub-net samples the low-resolution input images with binary patterns to generate the measurements. From those measurements, the transposed convolution layer learns a non-linear mapping to

generate a low-resolution version of the reconstructed image. After that, the residual correction sub-net learns the detail corrections and up-scales the image to the final high-resolution size with a phase shift operation. Together these two parts are able to reconstruct the high-resolution image directly from the low-resolution sampling.

3.1. Image reconstruction sub-net

The image reconstruction sub-net learns both the binary patterns and how to reconstruct the image from the measurements. During the training, the sampling process of the computational imaging is done using a convolutional layer where the convolutional kernels act as the digital mirror array and the kernel values (weights) act as binary patterns. When the trained model is integrated with the hardware, the learned kernel values can be uploaded to the digital mirror array to do the sampling and the measurements of the back scattered light intensity are sent back to the network to reconstruct the image.

The schematic of the image reconstruction sub-net is shown in Figure 2. The sampling and reconstruction can be formulated as

$$\begin{aligned} \tilde{x} &= \mathcal{F}_d(y, W_r) + b \\ &= \mathcal{F}_d(\mathcal{F}(\varphi(x), W_b), W_r) + b \end{aligned} \tag{2}$$

where $\tilde{x}$ is the reconstructed preliminary image. The $\mathcal{F}_d(\cdot)$ is the transposed convolution with W_r and b are the real-valued kernels and bias respectively. The $\varphi(\cdot)$ down-scales the original images for simulating the sampling process. The measurements y are generated by the convolution $\mathcal{F}(\cdot)$ of image x and the binary kernels W_b where each kernel corresponds to a sensing pattern. In our work, we studied two approaches to generate the binary patterns, i.e. the pre-generated and learned patterns. We describe these in detail below and compare their performance (Section 4).

Randomly pre-generated binary weights. In this approach, the patterns were randomly generated and remained static during the training. Before the training, we initialized the binary weights from the random Bernoulli distribution with $\Pr(1) = 0.5$. The distribution was applied to each kernel independently. During the training process, we updated the weights for the rest of the network. In this approach, the network was trained to fit to a specific set of static binary patterns. In our experiments, we compared

this scheme with the learned binary weights to study the benefit of weight optimization during the training.

Learned binary weights. The kernels were initialized with real-valued weights following the uniform distribution within range $[-1, 1]$. This ensured the initialized weights were equally assigned to positive and negative values. Since the real-valued weights were necessary for the network optimizer during training, these were used for gradient calculation. These were then mapped to binary values and applied to the sensing kernels for forward propagation. The binarization scheme is,

$$w_b = \begin{cases} 1 & if \;\; w_r > 0, \\ 0 & if \;\; w_r \leq 0, \end{cases} \tag{3}$$

where the w_b are the 0, 1 binary weights and the w_r are the real-valued weights. Note that in our network, only the binary kernels were involved in

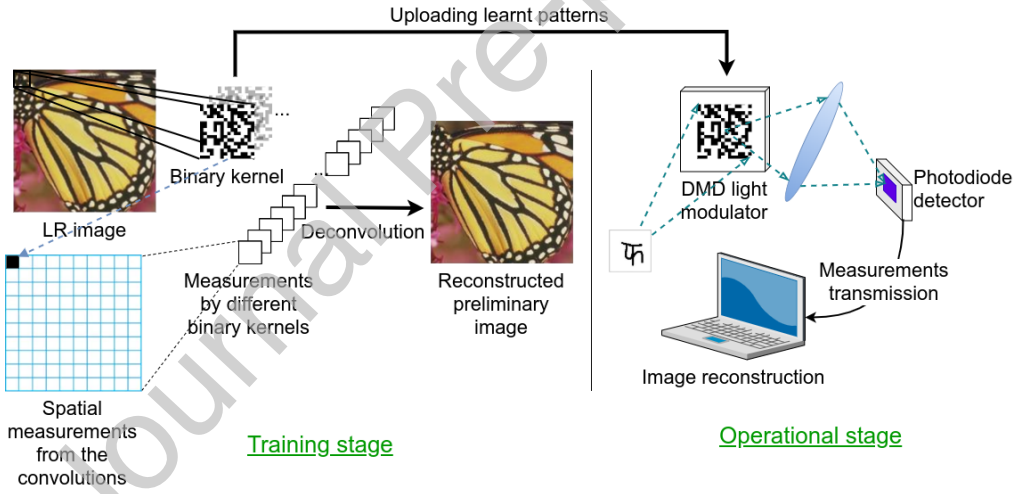


Figure 2: The operation of the image reconstruction part. Training stage: the low-resolution image is sampled with binary kernels using a convolutional layer. Each convolution operation generates a measurement, shown as the black element. By sliding the binary kernel through the image, the measurement map for the corresponding binary kernel is generated. After convolution with all binary kernels, the transposed convolutional layer is used to reconstruct the low-resolution preliminary image from the measurements. Operational stage: the learned patterns are uploaded to the DMD hardware to do the sampling, the measurements recorded by the photodiode detector are send back to network to compute the reconstructed image.

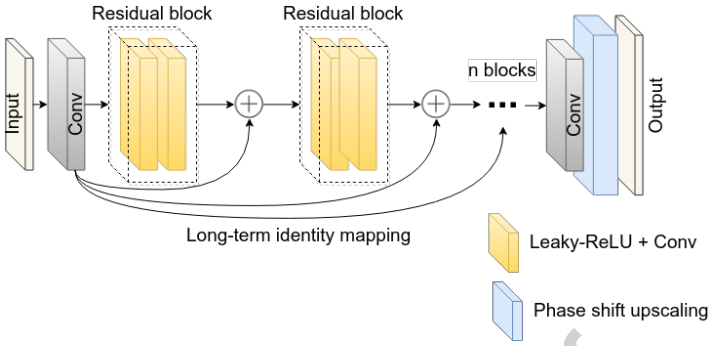


Figure 3: The schematic of the residual correction sub-net. The network feeds in the reconstructed preliminary image as input node and then extracts the original features. The feature maps are then passed to the recursive residual blocks, shown as dashed green lines. Each residual block has an identity branch that connects the original features with its output. Thereafter the residuals and the original features are added, element-wise, to generate the input to the next residual block. For each residual block, we applied leaky ReLU as the pre-activation function. At the end of the network, an extra convolutional layer and an upscaling layer is added to generate the residual output.

the convolution operations. In addition, we clipped the real-valued weights to fit within the range $[-1, 1]$. This ensured the effective binarizaiton mapping since the very large values out of the range did not have significant impact on the binarization process. We also applied an ℓ_2 norm regularization to the weights to avoid the risk of gradient explosion.

3.2. Residual correction sub-net

Taking the output of the image reconstruction sub-net as input, the residual correction sub-net predicts the fine details resulting in a high-resolution output image. The schematic of the residual correction sub-net is shown in the red block in Figure 1. This sub-net has two branches: up-scaling and residual mapping. During the training, the upscaling branch interpolates the intermediate input image to the required size of the high-resolution output. The residual mapping branch learns the reconstruction residual (fine details) between the upscaled intermediate input image and the original ground truth image using the long-term recursive residual blocks. The outputs of the two branches are added element-wise to reconstruct the final high-resolution image. In the remainder of the section, we describe the long-term recursive residual blocks and the image upscaling processes.

The conventional residual block is formulated as $\hat{a} = \mathcal{R}(a) = \mathcal{F}(a, W) +$

$h(a)$ where a and $\hat{a}$ are the input and output of the residual block, W indicates the weights of the residual block, $\mathcal{F}(a, W)$ learns the residual mapping between the input and the output and $h(a)$ is the identity mapping function. Our approach differs from the conventional residual block formulation. All of our blocks have skip connections with the intermediate reconstructed images, which we refer to as *long-term connections*. Each block share weights, forming a recursive chain. The sequence of the blocks in our network is shown in Figure 3. We used two convolutional layers with a pre-activation function in each block. For the identity mapping, we connected the feature maps associated with the low resolution input (generated by the first convolutional layer) to the output of each block. This long-term connection directly related these features with the outputs of the deep residual blocks. This can be formulated as

$$\begin{aligned} \hat{a}^j &= \mathcal{R}^j(\hat{a}^{j-1}) = \mathcal{F}(\hat{a}^{j-1}, W^j) + h(a^0) \\ \mathcal{F}(\hat{a}^{j-1}, W^j) &= W_2^j \sigma(W_1^j \sigma(\hat{a}^{j-1})) \end{aligned} \tag{4}$$

where $\mathcal{R}^j$ is the residual mapping function of the j-th block, a^0 is the initial features, and $\hat{a}^j$ is the output of j-th block. W^j is the weight and σ is the Leaky ReLU activation function [36]. The ith-layer in each block shared the same weights W_i where $i \in 1, 2$. This formed a recursive structure and reduced the total amount of model parameters significantly.

The image upscaling was implemented at the end of the residual correction sub-net. After the residual mapping branch extracted the residual from the preliminary low-resolution image, we applied a phase shift layer [37] to enlarge the size of the learned residual by a factor of s to have high-resolution residual features. We set the network such that the high-resolution residual features have the same number of channels (one for grayscale and three for RGB) as the final image. In the up-scaling branch, we also enlarged the image size by s with the phase shift operation. Then the residual and the image were added, element-wise, to generate the output image in the high-resolution. In our experiment, we set the upscaling factor s as 2.

3.3. Network training

The details of the network structure used in our experiment are illustrated in Figure 4. The network structure code can be downloaded at our *GitHub repository*. The proposed network consists of two functionally different sub-nets which contain different types of weights respectively. A straightforward strategy, used in previous work, to train such a heterogeneous network, is

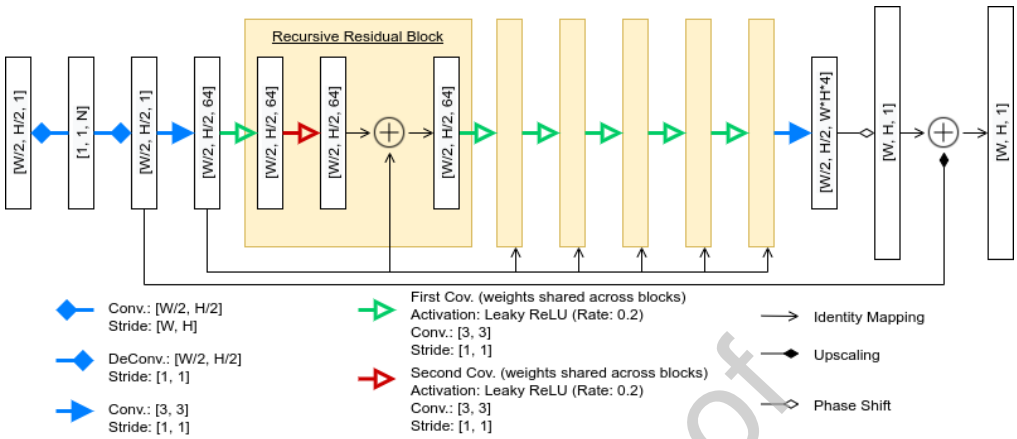


Figure 4: The details of the network structure. The diagram illustrates the feature map convolution, in which we take an image of size $H = 32$ and $W = 32$ as input. The image sampling is done at downscaled resolution and the output is at original resolution. The first yellow block illustrate the inner structure of the recursive residual blocks, which were simplified in the later blocks.

to train the two parts separately in a pipeline manner. Hence, the image-reconstruction sub-net is first trained and then used as a pre-trained model for training the whole network. This approach can be viewed as either a two-step training strategy or as a semi-decoupled strategy [25]. In contrast, we trained the heterogeneous network with pure end-to-end learning. These two parts of the network were trained jointly with a separate learning-rate update scheme for each. Specifically, for the image reconstruction sub-net, we set a larger initial learning rate with faster decay. This encouraged a rapid updating of the binary weights in the early stages of training and a slower update in the later stages, facilitating the residual correction sub-net to recover the fine image details. For the the residual correction sub-net, we initialized a relatively small learning rate with a slower decay rate since the residual correction for the details is more difficult to learn.

Denoting the original image as x, we aim to train the whole network f to reconstruct the high-resolution image $\tilde{x} = f(x, W)$, where W denotes the weights of the model. We associated the loss function with the output of both sub-nets (parts), i.e. the reconstructed low-resolution image and the upscaled high-resolution image, to train the network. In contrast to the common ℓ_2-norm loss function, used in previous work, we trained the network using the Charbonnier loss function, which is a variant of the ℓ_1-norm function. Given

$\tilde{x}^s$ the generated image at s upscaling factor, then our loss function is written as

$$\mathcal{L}(x, \tilde{x}, W) = \frac{1}{N} \sum_{i=1}^{N} \sum_{s=1}^{S} \omega \alpha(x_i^s - \tilde{x}_i^s) + \frac{\lambda}{2N} \sum_{W} w^2 \tag{5}$$

where N is the batch size and $\alpha(\mu) = \sqrt{\mu^2 + \epsilon}$ denotes the Charbonnier penalty. The second term is the ℓ_2-norm regularization for the weights. Our experiments indicated that images generated using the Charbonnier loss function were usually sharper than the results obtained using an ℓ_2 norm loss function. We accumulated the loss of both sub-nets. The ground truth image x_i^1 was generated by downsizing the original image using the bicubic interpolation method. The scalar weight ω controls the influence of each x_i^s in the loss function. In our experiment, we set $\omega = 2^s$ for each part. This multi-loss function forms a supervision scheme that can control the residual training at each part of the network.

4. Experiments

We conducted a series of tests to study the performance of the network. First, we evaluated the image reconstruction quality (see Section 4.3) on three datasets. Our learned and fixed-pattern binary models showed the first and second highest peak signal-to-noise ratio (PSNR) compared to the four methods reviewed in Section 2. In Section 4.4, we analyze how fixed and learned patterns affected the model training process. Finally, in Section 4.5, we assess the reconstruction efficiency of the network in comparison with other tested methods.

4.1. Datasets

We used the DIV2K image dataset [38] for training and validation. We applied data augmentation to the training images. Specifically, we randomly cropped 50 small patches of size 256×256 from each of the 800 images, that comprise the DIV2K dataset, to generate $40,000$ training images. In addition, we randomly applied flipping and rotation to the original patches. We used the cropped image patches as ground truth images for the high-resolution output.

Three datasets were used to test the model's performance. First we used a benchmark dataset of 11 test images, which has been used in existing work, to evaluate the reconstructed image quality and compare it with the

results of previous methods. Secondly, we evaluated the proposed method on a much larger dataset – the test set of ILSVRC2017, comprising 50000 natural images from 1000 classes [39]. It is known that natural images are often approximately sparse in the domain of the discrete cosine transformation (DCT) and the wavelet transform [40], and CS is an efficient method for approximate recovery of such images. Since our method is an alternative to CS, we have also evaluated the performance of our structured signal recovery method with images of various levels of sparsity. For this experiment, we generated a DCT-sparse version of the ILSVRC2017 test set and we controlled the sparsity of the DCT coefficients as follows: Each image was first transferred into the DCT domain where the coefficients were reordered based on their magnitude, then we set 5 percentage threshold cases for coefficient magnitude such that 100%, 20%, 10%, 5% or 1% of the coefficients were retained and all other coefficients were set to zero.

4.2. Setting network parameters and hyperparameters

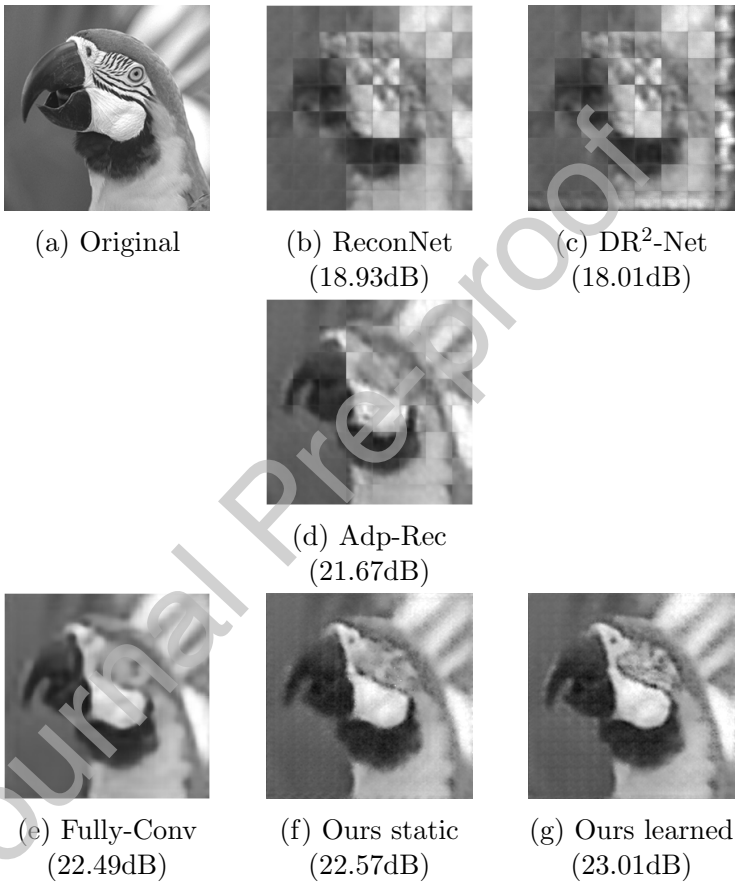
For the image reconstruction sub-net, we used 16×16 patterns for both the sensing kernels and the transposed convolution kernels. For the residual blocks, the kernel size for the convolutional layers was 3×3 and we used leaky ReLU activation with leaky rate $p = 0.2$. We used 64 channels for each of the convolutional layers.

The network was trained with a batch size of 16 using the Adam optimizer for 300 epochs. For the image reconstruction sub-net, we set the initial learning rate and the decay rate to 1×10^{-4} and 0.25 respectively. For the residual correction sub-net, we set the initial learning rate and the decay rate to 1×10^{-5} and 0.75 respectively. We set the decay step to $200,000$. The proposed method was trained on an NVidia GeForce GTX 1080Ti GPU.

In our experiment, we trained the network with different measurement ratios, $R = \frac{m}{N}$, of 0.01, 0.10 and 0.25, where m is the number of sampling kernels and N is the number of pixels in the sensing images. Accordingly, the number of binary kernels for the 128×128 benchmark sampling images are 164, 1638 and 4096.

4.3. Image reconstruction results

We evaluated our model on the benchmark dataset and compared the results with seven recently proposed methods: ReconNet [24], DR^2-Net [25], Adp-Rec [26], Fully-Conv [27], 2FC2Res [28], Fully-Block Net [29], and CSNet$^+$ [30]. To be consistent with previous work, we used the PSNR as the metric.



(a) Original (b) ReconNet (18.93dB) (c) DR^2-Net (18.01dB)

(d) Adp-Rec (21.67dB)

(e) Fully-Conv (22.49dB) (f) Ours static (22.57dB) (g) Ours learned (23.01dB)

Figure 5: The reconstruction result of the tested methods, including two of ours, at the compression ratio of $R = 0.01$.

(a) Original (b) ReconNet (19.04dB) (c) DR^2-Net (21.09dB)

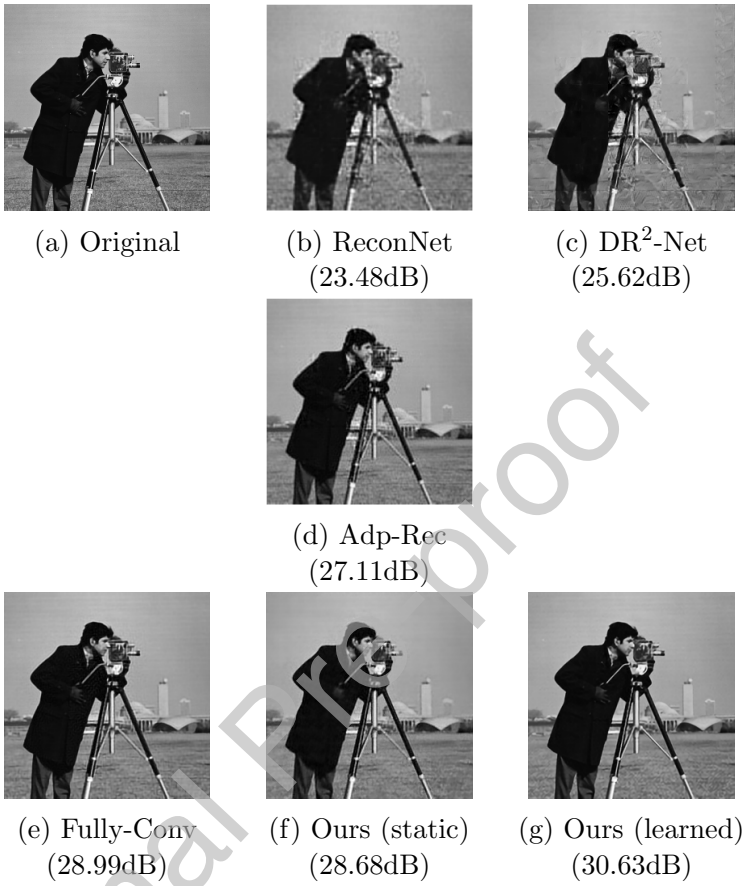
(d) Adp-Rec (23.83dB)

(e) Fully-Conv (24.98dB) (f) Ours (static) (24.34dB) (g) Ours (learned) (24.66dB)

Figure 6: The reconstruction result of the tested methods, including two of ours, at the compression ratio of $R = 0.10$.

The comparison results are summarized in Table 3. From the table, it can be seen that our network with learned patterns achieved the highest average PSNR at all three measurement ratios. Note the comparison with the Fully-Block Net and CSNet$^+$ follows protocols that were reported in their work. Our model with learned patterns indicates better results using the same protocol.

The example images reconstructed by different methods at measurement ratios of 0.01,0.10 and 0.25 are shown in Figures 5, 6 and 7 respectively. Our model reconstructed more details than other methods, resulting in images that are visually sharper. At the lowest measurement ratio 0.01, the



(a) Original (b) ReconNet (23.48dB) (c) DR^2-Net (25.62dB)

(d) Adp-Rec (27.11dB)

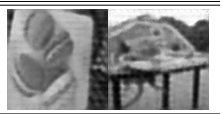
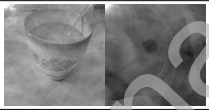
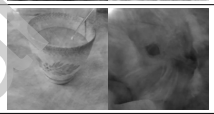
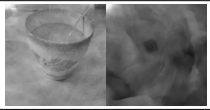
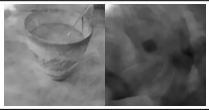
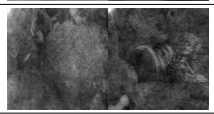
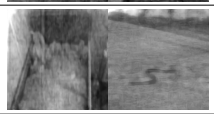
(e) Fully-Conv (28.99dB) (f) Ours (static) (28.68dB) (g) Ours (learned) (30.63dB)

Figure 7: The reconstruction result of the tested methods, including two of ours, at the compression ratio of $R = 0.25$.

block effect is not observed in the output images generated by the Fully-Conv network and our network. This is because both methods used the convolutional layer rather than the fully-connected layer to implement the sensing. Therefore the network could be trained in an end-to-end fashion and post-processing was not required to smooth the output images. At the measurement ratio of 0.10, the blocking effect can be eliminated for all methods since a sufficient number of measurements were acquired. At the highest measurement ratio 0.25, the Fully-Conv network is visually comparable to our method but our learned-weights model still achieved a higher PSNR value.

The difference between the results relating to the static patterns and the learned patterns, of our network, is significant at the measurement ratio of 0.01. The learned-patterns model achieved better average PSNR and reconstructed more detail. This implies that learning binary weights can help preserve more detail for the same measurement ratio and make the model converge faster, thereby reducing the training time.

Table 1: The sample images from reconstruction of the large scale test dataset. The rows denotes the reconstruction at different sparsity in DCT domain. The fist row is the reconstruction of the original images. The second to last rows showing the reconstruction of the sparsity-controlled images. Specifically, the sparsity of the images are at 100%, 20%, 10%, 5% and 1% of the original images. The first column shows the ground truth images and the second to last column show the reconstruction at compression ratio of 0.25, 0.1 and 0.01.

Sparsity	Raw image	Reconstruction		
		$R = 0.25$	$R = 0.1$	$R = 0.01$
original				
20%				
10%				
5%				
1%				

Next, we evaluated the model on the ILSVRC2017 test dataset. Figure 8 shows the mean PSNR values of the reconstructed images from ILSVRC2017 test set. The mean PSNR values produced by our method in a large-scale test are similar to those produced on a small benchmark set, indicating good generalization. Furthermore, PSNR values increase with increased sparsity. This indicates that the model performs well also on DCT-sparse images.

We also found that the PSNR of the reconstructed images, at three mea-

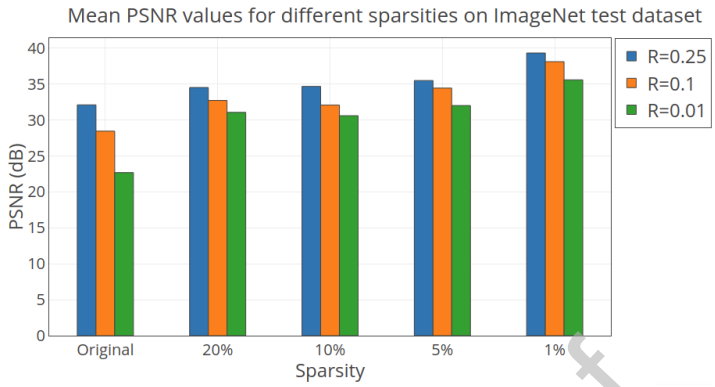


Figure 8: The evaluation of the learned binary model on the ILSVRC2017 test set. The trained learned-binary model was tested on the original ILSVRC2017 test set and the sparsified images in three measurement ratios (R = 0.01, 0.1 and 0.25). For the dataset, we controlled the sparsity of the images in the DCT domain. Specifically, we fixed the sparsity of the original images in the DCT domain such that 20%, 10%, 5% and 1% of the original DCT coefficients were retained. The results show that the trained model works well on the large-scale image dataset, indicating the ability of the model to generalize. It is also observed that the mean PSNR values increase with increasing sparsity. This denotes that the model also performs well on DCT-sparse images.

surement ratios, tend to be similar when we increase the sparsity of the image in the DCT domain. We present examples of reconstructed images in Table 1.

4.4. Model training analysis with fixed and binary sampling schemes

First, we analyzed the training efficiency by monitoring the validation loss in both sampling schemes. We found that training with the learned patterns produced a faster loss reduction for all three measurement ratios (as shown in Figure 9) than training with fixed patterns. When the measurement ratio was increased, the discrepancy between the losses of the two networks also increased. Furthermore, the network with learned patterns yielded a lower final loss, than the fixed patterns network, especially for R of 0.1 and 0.25. Even though the learned patterns network showed some instability compared to the fixed patterns network [1], it still is beneficial since it can be trained

[1] In the static scheme, the sampling patterns were not involved in the calculation of back-propagation. Only the real-valued weights in the rest of the network were updated. In the learned scheme, the binary weights were updated in each step. The binarization function

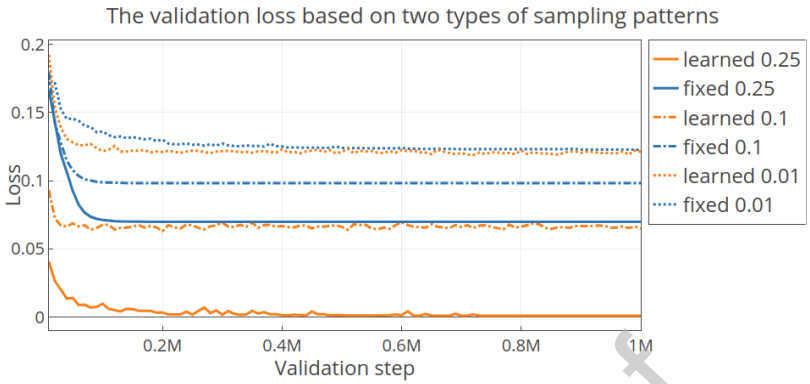


Figure 9: The validation losses of models with static (blue lines) and learned (orange lines) binary patterns. Each pattern type was validated for three measurement ratios (R = 0.01, 0.1 and 0.25). The validation loss with learned patters drops faster than that with the static patterns. The losses of both models at R = 0.01 are close at the end of training, but for higher measurement ratio the difference is large.

more quickly.

Next, we analyzed the sparsity of learned patterns by exploring the percentage of valid pixels (with value 1) in the patterns during pattern update. In compressive sampling theory, we typically use a small number of dense sensing patterns (equal numbers of ones and zeros) in contrast with a raster scan sensing in which each pattern is maximally sparse (contains one on pixel) and records the intensity of single pixel values one at a time. Conversely the sparse patterns are more efficient for single pixel imaging hardware as they require less on-board memory usage. Our approach effectively adapts the sparsity of patterns according to the measurement ratios and hence finds an optimal compromise between sensing efficiency and hardware performance. Specifically, we initialized all patterns using a single precision uniform distribution within the range $[-1, 1]$ (as required for model optimization), which were subsequently binarized to form patterns with a similar number of ones and zeros. However, the number of ones decreased dramatically during training since the model at large sampling rates does not necessarily need dense patterns. In contrast, for a relatively small measurement ratio of $R = 0.01$, the number of ones remained consistently high, which suggests that more

introduced fluctuations in the gradient calculation, which made the training progress less stable.

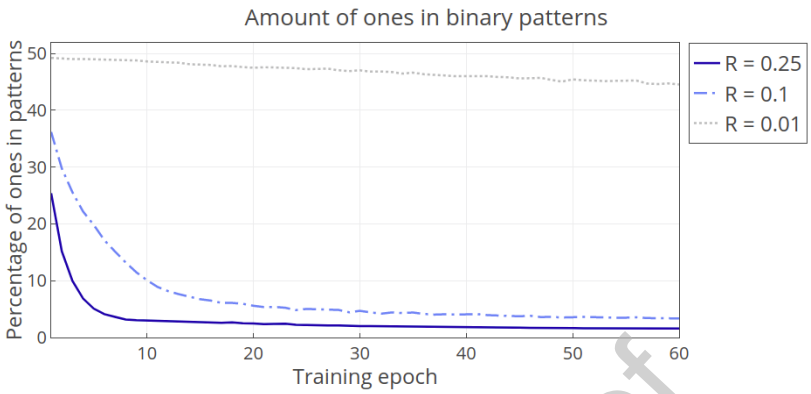


Figure 10: During training the binary patterns adapt differently for each measurement ratio. Notice that the fraction of ones contained in the binary patterns is inversely at $R = 0.25$, while for the very small measurement ratio $R = 0.01$, the fraction of ones remains constant because more information needs to be sensed by each pattern.

information was sampled by each pattern. As a result, the sampling patterns at $R = 0.1$ and $R = 0.25$ contain fewer ones compared to the patterns at $R = 0.01$, as seen in Figure 10. This variation due to R implies that the learning process can generate efficient binary sampling patterns that adapt to different measurements.

Table 2: Efficiency comparison of the tested methods in restoring an image of size 32 × 32, with the sampling measurement ratio of R=0.01. The $\mathcal{O}_{space}$ and $\mathcal{O}_{time}$ denote the space and time complexity of the reconstruction layer. The number of convolutional layers and blocks of Fully-Conv were not reported in their work.

Reconstruction efficiency and model size of 8 methods								
	Image-restoration				Residual-correction			
Name	$\mathcal{O}_{space}$	$\mathcal{O}_{time}$	# Weights	Format	# Conv layers	Structure	Share weights	Kernel size
ReconNet	1.024×10^4	1.024×10^4	1024	32-bit	6	Plain	No	32×32
DR^2-Net	1.024×10^4	1.024×10^4	1024	32-bit	12	4 Blocks	No	32×32
Adp-Rec	1.024×10^4	1.024×10^4	1024	32-bit	6	Plain	No	32×32
2FC2Res	1.024×10^4	1.024×10^4	1024	32-bit	6	2 Blocks	No	32×32
Fully-Conv	2560	2.62144×10^6	256	32-bit	-	-	No	32×32
Fully-Block Net	2560	2.62144×10^6	256	32-bit	25	12 Blocks	No	32×32
CSNet^2	1.024×10^4	1.024×10^4	1024	32-bit	12	5 Blocks	No	32×32
Ours	2560	6.5536×10^5	256	1-bit	12	6 Blocks	Yes	16×16

4.5. Analysis of the reconstruction efficiency

We analyzed the computational efficiency of the network by calculating the time and space complexity, which are introduced in the following content. The results demonstrated that our model has a good balance between the computational cost and the model size for the best image quality.

To determine the relative computational efficiency of our network, we compared the model size (space complexity) and the number of operations (time complexity) of our network's image reconstruction layer with the other 4 networks used in prior work (see Table 2). The comparison is based on the reconstruction of a single channel (greyscale) image of size 32×32 with a measurement ratio $R = 0.01$. The comparison is valid for any image size. The time and space complexity are formulated as **Time** $\sim \mathcal{O}(M^2 \cdot K^2 \cdot C_{in} \cdot C_{out})$ and **Space** $\sim \mathcal{O}(K^2 \cdot C_{in} \cdot C_{out})$, where M is the size of the feature map, K is the size of the kernel, C_{in} and C_{out} are number of input and output channels separately.

Our network has the smallest model size among all the tested networks and lower time complexity than the Fully-Conv network. Note that the ReconNet, DR^2-Net, Adp-Rec, and 2RC2Res perform fewer operations in the initial image reconstruction step because these networks use fully-connected layers. However, the fully-connected layer can only be trained for a specific image size, which is less practical.

For the residual-correction part, our recursive residual block with LSHR sampling scheme generates smaller intermediate feature maps and uses fewer model weights, thereby reducing the computational burden. In the Fully-Conv and Fully-Block Net networks, images were reconstructed directly back to the high-resolution size. The network then corrected the reconstruction error by applying convolution to the feature maps that had the same size as the high-resolution test image. Since the time complexity is directly related to M^2, which is the square of the image size, the computational cost of these three networks increases quadratically when the output image size is doubled. In contrast, our own network reconstructs the image at low resolution, and then convolutional operations are performed on small feature maps. These are upscaled back to the original size only at the last layer. Therefore, the number of operations performed by our network is order of $\frac{M^2}{4}$, which is four times less than the Fully-Conv and Fully-Block Net. Furthermore, the number of blocks does not affect the total number of weights since weights are shared between blocks forming a recursive residual block structure. Specifically, the weights are only shared between the first layers (or second layers)

among each of the six, two-layer, recursive residual blocks.

The last part of our analysis evaluated the performance of the network for different numbers of residual blocks in our recursive structure. The depth of the recursive residual block affects the reconstruction accuracy. It is seen in Figure 11 that the image quality increases by adding more blocks and the best performance (time and accuracy) is obtained with the 6-block structure. Adding more blocks leads to degradation of the image quality. In principle, adding more residual blocks could improve the capability of the residual mapping, but in practice, training a deeper network is harder. It is also observed in Figure 11 that the reconstruction time increases linearly with the addition of blocks. Therefore, our final model was constructed by using 6 blocks, which gave the best performance and reasonable reconstruction time. It was found that the accuracy increased and reached the best performance with 6 blocks, which was used in our final model.

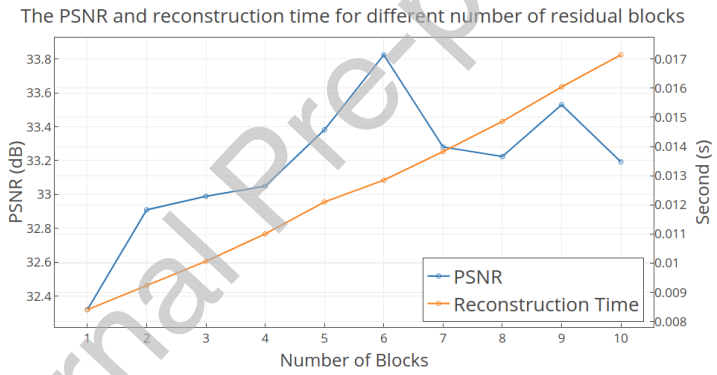


Figure 11: The average PSNR and the reconstruction time as a function of numbers of residual blocks in the recursive structure. The number of residual blocks influences the performance. The PSNR value was maximized when 6 blocks were used in the recursive structure. The average reconstruction time increased approximately linearly.

5. Implementation on hardware

In real-world applications, the signal/image sampling is usually done by optical devices which inevitably introduce noise and artifacts into the image data. Computer simulations alone provide no guarantees that an image recovery network architecture will be robust to these aspects of practical single-pixel imaging systems. Therefore it is important to validate the effi-

cacy of our LSHR-Net software solution, which uses learned binary patterns, with respect to typical single pixel imaging hardware.

Our hardware comprised a silicon planar photo-detector with a purposely designed amplifier circuit, lenses and a light projector. The photo-detector had a peak sensitivity at the wavelength of $930nm$ and its sensitive area was $93.6mm^2$. We connected the circuit to an Arduino circuit board which performed 10-bit analog-to-digital conversion (1024 scales). For evaluation purposes, we used test images from a database as an alternative to setting up unique object scenes. Test images were multiplied, in software, with each of the sampling patterns (forming modulated images) and projected using a TI DLP LightCrafter evaluation module consisting of a built-in DMD plane with a 608×684 array. The size, in pixels, of the sampling patterns was constrained by the sensitivity of the photo-detector and the analog-to-digital conversion resolution. A good practical resolution for the sampling patterns was found to be 16x16 pixels. Each of the modulated test images were focused onto the photo-detector using a set of lenses with focal length of $40mm$, $50mm$ and $100mm$. A filter with fixed attenuation was used to reduce light intensity at the photo-detector thereby avoiding saturation. We recorded the light intensity of the modulated images and sent these measurements as inputs data to the model.

For the hardware experiments, we trained our model with the MNIST dataset [41] using the same training settings described in Section 4. The network was trained with $10,000$ MNIST images. The model was evaluated using 18 randomly selected test images of handwritten digits (9 each from MNIST and the Omniglot datasets). We used the Omniglot dataset [42], which consists of a set of natural language characters, to demonstrate that the proposed method can generalize to datasets containing images that contains with different image structure from the training set.

The model reconstructed images directly from the photo-detector measurements at a super resolution size of 32×32. We evaluated performance at the same measurement ratios used in Section 4.2. Results on MNIST and Omniglot are shown in Figure 12 and 13 respectively. It is observed that the reconstruction quality of the character structure was improved by increasing the number of measurements. At the same time, artifacts in the reconstructed images can be seen. These are caused predominantly by noise in the hardware setup (e.g. by the amplifier circuit). The average SNR of the recorded measurement signal was 15.7dB. Moreover, in Figure 12 and 13 it can be seen that the reconstructed images of $R = 0.25$ are more pixelated

Figure 12: The figure shows the reconstruction results of 9 random selected MNIST handwritten numbers using the hardware measurements. The images at the top row are the original ground truth images and the images from the second to the last row are reconstructed results at $R = 0.01$, 0.1 and 0.25.

Figure 13: The figure shows the reconstruction results of 9 random selected Omniglot characters using the hardware measurements. The images at the top row are the original ground truth images and the images from the second to the last row are reconstructed results at $R = 0.01$, 0.1 and 0.25.

than those of $R = 0.1$ and $R = 0.01$. Visually, the model resulted in better reconstruction quality. This is however due to the smoothing effect which is also seen in Figure 5.

6. Conclusions

In this paper, we have proposed a hardware-friendly method for image reconstruction from compressively sensed measurements, using mixed-weights deep neural networks. The proposed method, which consists of sampling and reconstruction networks, was specially designed to ease hardware realization, particularly to integrate our work with single pixel camera. Our novel LSHR network uses trainable binary sampling patterns that can be deployed on a single pixel camera's DMD sampling array. LSHR net samples light intensity functions at low-resolution and reconstructs images with

high-resolution details. Effectively, it reduces the number of measurements at the same measurement ratio and reduces the convolutional computing cost. Hence, it improves the efficiency of the reconstruction process significantly compared with previous work. For the purpose of reducing the hardware storage requirement for image reconstruction, the reconstruction network equips long-term recursive residual blocks. It has a weights-sharing strategy that makes the trained models of our method much more compact than those of previously reported network architectures and requires less onboard storage in the imaging hardware. The experimental results on the benchmark image datasets indicate that our method yields better image quality than those reported in previous work for a number of different measurement ratios. We also implemented our method on proof-of-concept hardware and demonstrated that it can sample images as compact measurements and then recover them from the measurements successfully. Our network architecture has potential applications beyond the scope of single pixel imaging. For example, it may be adapted for similar imaging modalities such as coded aperture imaging and structured light sensing. An efficient approach to network training for different imaging modalities may involve transfer learning and this could be the focus of future work in this area. Moreover, for a specific hardware setup, fine-tuning after the initial deployment of hardware can potentially yield improvements in image quality using software alone.

Acknowledgements

We dedicate this article to memory of Craig Douglas (Seismicstuff ltd) who designed and made the amplifier circuit used in our hardware experiment.

Table 3: The PSNR of 11 test image in dB from recent six methods at three measurement ratios. The reported mean is the average PSNR value for all images. The red figures and the blue figures denote the first and second highest value among all the methods. Our network based on learned binary weights yields the highest average PSNR at all three measurement ratios.

Image	Methods	measurement ratio			Image	Methods	measurement ratio		
		R=0.25	R=0.1	R=0.01			R=0.25	R=0.1	R=0.01
Barbara	ReconNet	23.58dB	22.17dB	19.08dB	Boats	ReconNet	27.83dB	24.56dB	18.82dB
	DR^2-Net	25.77dB	22.69dB	18.65dB		DR^2-Net	30.09dB	25.58dB	18.67dB
	Adp-Rec	27.40dB	24.28dB	21.36dB		Adp-Rec	32.47dB	28.80dB	21.09dB
	Fully-Conv	28.59dB	24.28dB	22.06dB		Fully-Conv	33.88dB	29.48dB	22.3dB
	2FC2Res	27.92dB	24.27dB	21.48dB		2FC2Res	33.59dB	29.12dB	21.29dB
	Ours (static)	27.52dB	24.57dB	22.03dB		Ours (static)	32.05dB	29.55dB	22.59dB
	Ours (Learned)	31.11dB	24.56dB	22.34dB		Ours (Learned)	34.13dB	29.59dB	23.31dB
Fingerprint	ReconNet	26.15dB	20.99dB	15.01dB	Cameraman	ReconNet	23.48dB	21.54dB	17.51dB
	DR^2-Net	27.65dB	22.03dB	14.73dB		DR^2-Net	25.62dB	22.46dB	17.08dB
	Adp-Rec	32.31dB	26.55dB	16.22dB		Adp-Rec	27.11dB	24.97dB	19.74dB
	Fully-Conv	32.91dB	27.36dB	16.33dB		Fully-Conv	28.99dB	25.62dB	20.63dB
	2FC2Res	32.17dB	25.92dB	16.22dB		2FC2Res	28.84dB	25.07dB	19.98dB
	Ours (static)	30.36dB	26.07dB	17.10dB		Ours (static)	28.68dB	26.53dB	20.84dB
	Ours (Learned)	33.38dB	26.40dB	17.23dB		Ours (Learned)	30.63dB	26.56dB	21.35dB
Flinstones	ReconNet	22.74dB	19.04dB	14.14dB	Foreman	ReconNet	32.08dB	29.02dB	22.03dB
	DR^2-Net	26.19dB	21.09dB	14.01dB		DR^2-Net	33.53dB	29.20dB	20.59dB
	Adp-Rec	27.94dB	23.83dB	16.12dB		Adp-Rec	36.18dB	33.51dB	25.53dB
	Fully-Conv	30.26dB	24.98dB	16.92dB		Fully-Conv	38.10dB	34.00dB	27.26dB
	2FC2Res	29.72dB	24.94dB	16.27dB		2FC2Res	38.25dB	34.29dB	25.77dB
	Ours (static)	28.00dB	24.34dB	16.81dB		Ours (static)	35.34dB	33.13dB	26.36dB
	Ours (Learned)	31.01dB	24.66dB	17.27dB		Ours (Learned)	36.91dB	33.45dB	27.13dB
Lena	ReconNet	27.47dB	24.48dB	18.51dB	House	ReconNet	29.96dB	26.74dB	20.30dB
	DR^2-Net	29.42dB	25.39dB	17.97dB		DR^2-Net	31.83dB	27.53dB	19.61dB
	Adp-Rec	31.63dB	28.50dB	21.49dB		Adp-Rec	34.38dB	31.43dB	22.93dB
	Fully-Conv	33.00dB	28.97dB	22.51dB		Fully-Conv	36.22dB	32.36dB	23.67dB
	2FC2Res	32.97dB	28.86dB	21.57dB		2FC2Res	35.35dB	31.45dB	22.92dB
	Ours (static)	31.60dB	29.37dB	23.13dB		Ours (static)	34.80dB	32.55dB	24.82dB
	Ours (Learned)	34.18dB	29.57dB	23.52dB		Ours (Learned)	36.61dB	33.73dB	25.12dB
Monarch	ReconNet	24.95dB	21.49dB	15.61dB	Peppers	ReconNet	25.74dB	22.72dB	17.39dB
	DR^2-Net	27.95dB	23.10dB	15.33dB		DR^2-Net	28.49dB	24.32dB	16.90dB
	Adp-Rec	29.25dB	26.65dB	17.70dB		Adp-Rec	29.65dB	26.67dB	19.75dB
	Fully-Conv	32.63dB	27.61dB	18.46dB		Fully-Conv	32.90dB	28.72dB	21.38dB
	2FC2Res	32.46dB	27.60dB	17.85dB		2FC2Res	32.82dB	27.52dB	20.05dB
	Ours (static)	31.51dB	28.71dB	20.09dB		Ours (static)	31.20dB	28.23dB	21.52dB
	Ours (Learned)	34.20dB	29.07dB	20.79dB		Ours (Learned)	33.51dB	28.61dB	22.10dB
Parrot	ReconNet	26.66dB	23.36dB	18.93dB	Mean	ReconNet	26.42dB	23.28dB	17.94dB
	DR^2-Net	28.73dB	23.94dB	18.01dB		DR^2-Net	28.66dB	24.32dB	17.44dB
	Adp-Rec	30.51dB	27.59dB	21.67dB		Adp-Rec	30.80dB	27.53dB	20.33dB
	Fully-Conv	32.13dB	27.92dB	22.49dB		Fully-Conv	32.69dB	28.30dB	21.27dB
	2FC2Res	31.89dB	27.93dB	21.77dB		2FC2Res	32.36dB	27.91dB	20.47dB
	Ours (static)	32.64dB	29.84dB	22.57dB		Ours (static)	31.25dB	28.44dB	21.62dB
	Ours (Learned)	34.75dB	30.18dB	23.01dB		Ours (Learned)	33.68dB	28.67dB	22.11dB
Mean$^\diamondsuit$	CSNet {0,1}	-	26.39dB	20.62dB	Mean$^\heartsuit$				
	CSNet$^+$	-	28.37dB	21.02dB		Fully-Block Net	33.57dB	28.94dB	22.12dB
	Ours (Learned)	33.68dB	28.67dB	22.11dB		Ours (Learned)	33.66dB	29.04dB	22.79dB

$\diamondsuit$ Results of CSNet0,1 and CSNet$^+$ at R = 25% were not reported in their work [30].

$\heartsuit$ The Fully-Block Net [29] was tested only on a subset of the *standard* benchmark. To be specific, seven images from the *standard* benchmark set were selected for testing. To compare with their results, we presented in the table our results on the same subset.

Declaration of interests

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Author statement

Fangliang Bai: Conceptualization, Methodology, Software, Validation, Writing - Original Draft, Writing - Review & Editing. Jinchao Liu: Conceptualization, Methodology, Validation, Writing - Original Draft, Writing - Review & Editing. Xiaojuan Liu: Investigation. Margarita Osadchy: Validation, Writing - Review & Editing. Chao Wang: Validation. Stuart J. Gibson: Conceptualization, Validation, Writing - Review & Editing, Supervision, Project administration, Funding acquisition.

References

References

[1] J. H. Shapiro, Computational ghost imaging, Phys. Rev. A 78 (2008) 061802. doi:10.1103/PhysRevA.78.061802.

[2] M. B. Wakin, J. N. Laska, M. F. Duarte, D. Baron, S. Sarvotham, D. Takhar, K. F. Kelly, R. G. Baraniuk, An architecture for compressive imaging, in: 2006 International Conference on Image Processing, 2006, pp. 1273–1276. doi:10.1109/ICIP.2006.312577.

[3] A. Sankaranarayanan, L. Xu, C. Studer, Y. Li, K. Kelly, R. Baraniuk, Video compressive sensing for spatial multiplexing cameras using motion-flow models, SIAM Journal on Imaging Sciences 8 (3) (2015) 1489–1518. doi:10.1137/140983124.

[4] M.-J. Sun, M. P. Edgar, G. M. Gibson, B. Sun, N. Radwell, R. Lamb, M. J. Padgett, Single-pixel three-dimensional imaging with time-based depth resolution, Nature communications 7 (2016) 12010.

[5] M.-J. Sun, J.-M. Zhang, Single-pixel imaging and its application in three-dimensional reconstruction: A brief review, Sensors 19 (3) (2019) 732.

[6] B. Lochocki, A. Gambín, S. Manzanera, E. Irles, E. Tajahuerce, J. Lancis, P. Artal, Single pixel camera ophthalmoscope, Optica 3 (10) (2016) 1056–1059.

[7] Z. Zhang, X. Wang, G. Zheng, J. Zhong, Hadamard single-pixel imaging versus fourier single-pixel imaging, Opt. Express 25 (16) (2017) 19619–19639. doi:10.1364/OE.25.019619.

[8] M.-J. Sun, M. P. Edgar, D. B. Phillips, G. M. Gibson, M. J. Padgett, Improving the signal-to-noise ratio of single-pixel imaging using digital microscanning, Opt. Express 24 (10) (2016) 10476–10485. doi:10.1364/OE.24.010476.

[9] A. Chiranjan, B. Duvenhage, F. Nicolls, Implementation of adaptive coded aperture imaging using a digital micro-mirror device for defocus deblurring, in: 2016 Pattern Recognition Association of South Africa

and Robotics and Mechatronics International Conference (PRASA-RobMech), IEEE, 2016, pp. 1–5.

[10] D. L. Donoho, Compressed sensing, IEEE Transactions on Information Theory 52 (4) (2006) 1289–1306. doi:10.1109/TIT.2006.871582.

[11] E. J. Candes, J. Romberg, T. Tao, Robust uncertainty principles: exact signal reconstruction from highly incomplete frequency information, IEEE Transactions on Information Theory 52 (2) (2006) 489–509. doi:10.1109/TIT.2005.862083.

[12] S. Becker, J. Bobin, E. J. Candès, Nesta: A fast and accurate first-order method for sparse recovery, SIAM J. Img. Sci. 4 (1) (2011) 1–39. doi:10.1137/090756855.

[13] S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein, Distributed optimization and statistical learning via the alternating direction method of multipliers, Foundations and Trends in Machine Learning 3 (1) (2011) 1–122. doi:10.1561/2200000016.

[14] A. Krizhevsky, I. Sutskever, G. E. Hinton, Imagenet classification with deep convolutional neural networks, in: F. Pereira, C. J. C. Burges, L. Bottou, K. Q. Weinberger (Eds.), Advances in Neural Information Processing Systems 25, Curran Associates, Inc., 2012, pp. 1097–1105.

[15] R. Girshick, J. Donahue, T. Darrell, J. Malik, Rich feature hierarchies for accurate object detection and semantic segmentation, in: Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition, CVPR '14, 2014, pp. 580–587.

[16] E. Shelhamer, J. Long, T. Darrell, Fully convolutional networks for semantic segmentation, IEEE Transactions on Pattern Analysis and Machine Intelligence 39 (4) (2017) 640–651.

[17] J. Gu, Z. Wang, J. Kuen, L. Ma, A. Shahroudy, B. Shuai, T. Liu, X. Wang, G. Wang, J. Cai, et al., Recent advances in convolutional neural networks, Pattern Recognition 77 (2018) 354–377.

[18] Z. Tu, W. Xie, D. Zhang, R. Poppe, R. C. Veltkamp, B. Li, J. Yuan, A survey of variational and cnn-based optical flow techniques, Signal Processing: Image Communication 72 (2019) 9–24.

[19] Y. Tai, J. Yang, X. Liu, C. Xu, Memnet: A persistent memory network for image restoration, in: 2017 IEEE International Conference on Computer Vision (ICCV), Vol. 00, 2018, pp. 4549–4557. doi:10.1109/ICCV.2017.486.

[20] A. Mousavi, A. B. Patel, R. G. Baraniuk, A deep learning approach to structured signal recovery, in: 2015 53rd Annual Allerton Conference on Communication, Control, and Computing (Allerton), 2015, pp. 1336–1343. doi:10.1109/ALLERTON.2015.7447163.

[21] A. Adler, D. Boublil, M. Elad, M. Zibulevsky, A deep learning approach to block-based compressed sensing of images, CoRR abs/1606.01519. arXiv:1606.01519.

[22] A. Mousavi, R. G. Baraniuk, Learning to invert: Signal recovery via deep convolutional networks, in: 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2017, pp. 2272–2276. doi:10.1109/ICASSP.2017.7952561.

[23] A. Mousavi, G. Dasarathy, R. G. Baraniuk, DeepCodec: Adaptive Sensing and Recovery via Deep Convolutional Neural Networks, ArXiv e-printsarXiv:1707.03386.

[24] K. Kulkarni, S. Lohit, P. Turaga, R. Kerviche, A. Ashok, Reconnet: Non-iterative reconstruction of images from compressively sensed measurements, in: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 449–458.

[25] H. Yao, F. Dai, D. Zhang, Y. Ma, S. Zhang, Y. Zhang, Dr^2-net: Deep residual reconstruction network for image compressive sensing, CoRR abs/1702.05743. arXiv:1702.05743.

[26] X. Xie, Y. Wang, G. Shi, C. Wang, J. Du, X. Han, Adaptive measurement network for cs image reconstruction, in: J. Yang, Q. Hu, M.-M. Cheng, L. Wang, Q. Liu, X. Bai, D. Meng (Eds.), Computer Vision, Springer Singapore, Singapore, 2017, pp. 407–417.

[27] J. Du, X. Xie, C. Wang, G. Shi, X. Xu, Y. Wang, Fully convolutional measurement network for compressive sensing image reconstruction, Neurocomputing 328 (2019) 105 – 112, chinese Conference on Computer Vision 2017.

[28] Z. Zhao, X. Xie, C. Wang, W. Liu, G. Shi, J. Du, Visualizing and understanding of learned compressive sensing with residual network, Neurocomputing 359 (2019) 185–198.

[29] X. Xie, C. Wang, J. Du, G. Shi, Full image recover for block-based compressive sensing, in: 2018 IEEE International Conference on Multimedia and Expo (ICME), IEEE, 2018, pp. 1–6.

[30] W. Shi, F. Jiang, S. Liu, D. Zhao, Image compressed sensing using convolutional neural network, IEEE Transactions on Image Processing 29 (2019) 375–388.

[31] M. Iliadis, L. Spinoulas, A. K. Katsaggelos, Deep fully-connected networks for video compressive sensing, Digital Signal Processing 72 (2018) 9 – 18. doi:https://doi.org/10.1016/j.dsp.2017.09.010.

[32] M. Iliadis, L. Spinoulas, A. K. Katsaggelos, Deepbinarymask: Learning a binary mask for video compressive sensing, CoRR abs/1607.03343. arXiv:1607.03343.

[33] K. Dabov, A. Foi, V. Katkovnik, K. Egiazarian, Image denoising by sparse 3-d transform-domain collaborative filtering, IEEE Transactions on Image Processing 16 (8) (2007) 2080–2095. doi:10.1109/TIP.2007.901238.

[34] M. Courbariaux, Y. Bengio, J.-P. David, Binaryconnect: Training deep neural networks with binary weights during propagations, in: C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, R. Garnett (Eds.), Advances in Neural Information Processing Systems 28, Curran Associates, Inc., 2015, pp. 3123–3131.

[35] M. Rastegari, V. Ordonez, J. Redmon, A. Farhadi, Xnor-net: Imagenet classification using binary convolutional neural networks, in: B. Leibe, J. Matas, N. Sebe, M. Welling (Eds.), Computer Vision – ECCV 2016, Springer International Publishing, Cham, 2016, pp. 525–542.

[36] K. He, X. Zhang, S. Ren, J. Sun, Delving deep into rectifiers: Surpassing human-level performance on imagenet classification, in: Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV), ICCV '15, IEEE Computer Society, Washington, DC, USA, 2015, pp. 1026–1034. doi:10.1109/ICCV.2015.123.

[37] W. Shi, J. Caballero, F. Huszar, J. Totz, A. P. Aitken, R. Bishop, D. Rueckert, Z. Wang, Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network, in: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 1874–1883.

[38] E. Agustsson, R. Timofte, Ntire 2017 challenge on single image super-resolution: Dataset and study, in: 2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), 2017, pp. 1122–1131. doi:10.1109/CVPRW.2017.150.

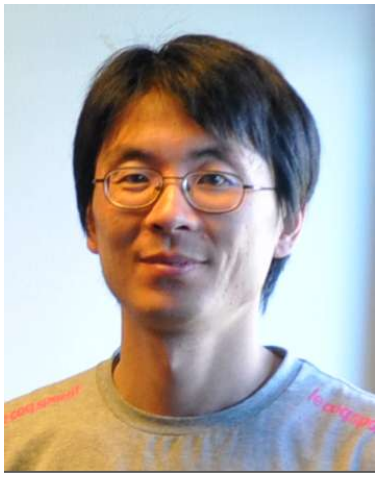
[39] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, L. Fei-Fei, ImageNet Large Scale Visual Recognition Challenge, International Journal of Computer Vision (IJCV) 115 (3) (2015) 211–252. doi:10.1007/s11263-015-0816-y.

[40] D. Taubman, M. Marcellin, JPEG2000 image compression fundamentals, standards and practice: image compression fundamentals, standards and practice, Vol. 642, Springer Science & Business Media, 2012.

[41] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proceedings of the IEEE 86 (11) (1998) 2278–2324.

[42] B. M. Lake, R. Salakhutdinov, J. B. Tenenbaum, Human-level concept learning through probabilistic program induction, Science 350 (6266) (2015) 1332–1338.

Fangliang Bai is a Ph.D. student in the School of Physical Science at University of Kent, Canterbury, UK. He received his M.S. degree in School of Software Engineering from Cranfield University in 2015. His research interests are machine learning, compressive sensing, image processing and neural networks.



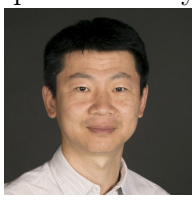
Jinchao Liu received the B.Sc. degree in automation and M.Sc. degree in control science and engineering from the Wuhan University of Technology, Wuhan, China, in 2004 and 2007, respectively, and the Ph.D. degree in mechanical engineering from the Technical University of Denmark, Kgs. Lyngby, Denmark, in 2011. He is currently an Associate Professor with the College of Artificial Intelligence, Nankai University, China. Before joining NKU, he was with VisionMetric Ltd, Canterbury, UK. His current research interests include machine learning, machine vision, robotics and using AI for scientific discovery.

Xiaojuan Liu obtained a Master of Science Degree in Nankai Univesity in 2004 and engineering PhD in Nankai Univesity

in 2007. Since 2007, she has been an associate professor working in Shandong University of Technology. The main researching interests are all-solid-state laser, fiber laser and fiber amplifier, ultra-short lasers, microwave photonic filters.

Margarita Osadchy received the PhD degree with honors in computer science in 2002 from the University of Haifa, Israel. From 2001 to 2004, she was a visiting research scientist at the NEC Research Institute and then a postdoc-

toral fellow in the Department of Computer Science at the Technion. Since 2005 she has been with the Computer Science Department at the University of Haifa. Her main research interests are machine learning, computer vision, computer security, and privacy.



Chao Wang is currently a Senior Lecturer in the School of Engineering and Digital Arts at University of Kent, where he first joined as a Lecturer in 2013. From 2011 to 2012, he was a NSERC Postdoctoral Fellow in the Photonics Laboratory, University of California, Los Angeles (UCLA), USA. He received his B.Eng degree in Opto-electrical Engineering from Tianjin University, China, in 2002, M.Sc degree in Optics from Nankai University, China, in 2005, and Ph.D degree in Electrical and Computer Engineering from the University of Ottawa, Canada, in 2011.

Stuart Gibson was appointed to the position of Lecturer in the School of Physical Sciences at the University of Kent in 2007. He is the co-inventor of the EFIT-V facial composite system which is currently used by the majority of UK police constabularies and in numerous other countries. The main theme of Dr Stuart Gibson's research

is forensic applications of digital image processing and machine learning.