Error-Resilient Analog Image Storage and Compression with Analog-Valued RRAM Arrays: An Adaptive Joint Source-Channel Coding Approach

Xin Zheng[1*], Ryan Zarcone[2], Dylan Paiton[3], Joon Sohn[1], Weier Wan[1], Bruno Olshausen[3+] and H. -S. Philip Wong[1#]
[1]Department of Electrical Engineering, Stanford University, Stanford, CA, 94305, USA,
[2]Biophysics Graduate Group, [3]Vision Science Graduate Group, UC Berkeley, Berkeley, CA, 94720, USA.
E-mail: [*]xzheng3@stanford,edu, [+] baolshausen@berkeley.edu, [#] hspwong@stanford.edu

Abstract – We demonstrate by experiment an image storage and compression task by directly storing analog image data onto an analog-valued RRAM array. A joint source-channel coding algorithm is developed with a neural network to encode and retrieve natural images. The encoder and decoder adapt jointly to the statistics of the images and the statistics of the RRAM array in order to minimize distortion. This adaptive joint source-channel coding method is resilient to RRAM array non-idealities such as cycle-to-cycle and device-to-device variations, time-dependent variability, and non-functional storage cells, while achieving a reasonable reconstruction performance of ~ 20 dB using only 0.1 devices/pixel for the analog image.

I. Introduction

Much of today's data (e.g. video, audio, and images) are inherently analog. To store and compress these analog data onto digital memory, analog-digital data conversion (source coding) and digital compression (channel coding) are usually performed separately. Analog, non-volatile memory (NVM), such as RRAM and PCM, offer opportunities to directly store multi-dimensional analog data. The challenge is to perform reliable storage and retrieval of analog signals with non-ideal NVM devices. In this paper, we present a joint source-channel coding algorithm with a neural network to store and compress natural images onto an analog-valued RRAM array. Through jointly learning the data and memory statistics, our source-channel coding algorithm finds an optimal use of the memory's intrinsic data storage capacity [1]. Using this algorithm, we demonstrate by experiment that natural images can be reliably stored and retrieved with an analog-valued RRAM array, while having additional desirable properties of being resilient to array-level non-idealities such as cycle-to-cycle and device-to-device variations, time-dependent variability, and non-functional storage cells. Our work presents a way to use imperfect NVMs whereby cost and fabrication advantages are utilized while the non-idealities of the device technology are circumvented. This approach shows that it is fruitful to customize the way we use the device technology to suit the task at hand.

II. Joint Source-Channel Coding Algorithm

To learn a mapping from natural images to the RRAM array, we constructed a multilayer autoencoder neural network (**Fig. 1**), analogous to a denoising autoencoder [2]. With a succession of linear/nonlinear operations, the encoder transforms an input image into a set of resistances to be written to the array. To retrieve the stored image, the decoder transforms the read-resistances into a reconstruction. Specifically, the encoder and decoder weights are parameterized as filter convolutions. For the non-linearities, or "activation functions", we used divisive normalization (Equation 3 in [3]), a population nonlinearity that implements a local form of gain control.

In order to train the network, we constructed a differentiable model of the RRAM channels, with two sources of channel noise: (1) a uniform noise induced by the write process and (2) a "sparse" noise induced by device failure. The latter was implemented during network training by randomly choosing a certain percentage of the devices and then sending them to the low resistance state (LRS) (i.e. setting R_T to 0). Thus, the noise model for each device was:

$$\log R_M(R_T) = \log R_T \cdot (1-\eta) + \log \sigma \cdot \epsilon,, \epsilon \sim U(-0.5, 0.5) \quad (1)$$

where σ is the write process acceptance range, $U(-0.5, 0.5)$ is the uniform distribution function and $\eta \in \{0,1\}$ is the sparse noise parameter (by default $\eta = 0$, but $\eta = 1$ when the device fails). More importantly, this channel model is not specific to the type of memory device. The model can be adapted to any analog programmable memory device (even those with highly nonlinear input/output functions [4]).

For training, the network weights were learned by backpropagating the gradient of the objective function:

$$C = \langle \|X - \hat{X}\|^2 + \lambda \left[\max\left(0, \log \frac{R_T}{R_{T_{MAX}}}\right) + \max\left(0, \log \frac{R_{T_{MIN}}}{R_T}\right) \right] \rangle \quad (2)$$

The first term corresponds to the squared error between the original image, X, and its reconstruction, $\hat{X}$. The second term is a cost that penalizes the network for using values outside the range of acceptable R_T (λ being a scalar hyper-parameter that weights this term's importance). $\langle . \rangle$ indicates an average over an image-batch during training. The training set consisted of ~1.2×10^5 images from the 2016 ImageNet test set [5] and the Flickr Creative Commons set [6].

III. Analog Value Storage with RRAM Array

To demonstrate by experiment this analog data storage method, we fabricated a CMOS-integrated TiN/HfOx/Pt 1K 1T1R array (**Fig. 2**). **Fig. 3** shows a cross-section schematic of the RRAM Stack: Pt (30 nm) / HfOx (5 nm) / TiN (50 nm). **Fig. 4** shows typical DC (**Fig. 4 (a)**) and AC (**Fig. 4 (b)**) cell characteristics in the 1T1R array with analog storage capability. We encoded 448 analog resistances, R_T (10 KΩ to 100 MΩ), from 64×64 pixels (8 bits per pixel) images onto

978-1-7281-1987-8/18/$31.00 ©2018 IEEE

448 cells, a ratio of 1 device per 10 pixels. We set the write acceptance range σ to 2. Ref. [7] showed that for a two-terminal RRAM device, any resistance level within its dynamic range can be achieved with high accuracy by applying a combination of incremental step RESET and SET pulse trains. Here we extend this write algorithm to 1T1R device programming. A double-direction incremental step pulse programming strategy (DD-ISPP) was used to precisely program cell resistances into the acceptance range, in spite of cycle-to-cycle and device-to-device variations. **Fig. 5 (a)** shows the DD-ISPP sequence. The device was initially SET to the LRS (around 10 kΩ). A full DD-ISPP writing pulse parameter set is listed in **Fig. 5 (b).** The cell resistance was verified after each write pulse. The sequence continued until one out of three conditions was satisfied: (i) the programmed resistance fell into the acceptance range; (ii) the WL voltage exceeded the maximum value (5V for RESET, 1.2V for SET); (iii) the total number of write pulses reached the maximum number (100). If the final programmed resistance was still outside the acceptance range, we SET it back to the initial LRS (corresponding to the sparse noise in the RRAM model of Section II). **Fig. 5 (c) (d)** show an example writing with DD–ISPP, where RESET over-programming is fixed by the SET pulse train. Targeting at different R_T (100 KΩ, 300 KΩ, 1 MΩ, and 3 MΩ) with the acceptance range of 2, **Fig. 6** shows cycle-to-cycle (**Fig. 6 (a)**) and device-to-device (**Fig. 6 (b)**) resistance distributions written by DD-ISPP, both following the same uniform distribution centered at R_T with boundaries defined by the acceptance range (this being well-modeled by the uniform noise term in the RRAM model). **Fig. 7** shows the log-scale 448 resistance values encoded by the network (R_T, **Fig. 7 (a)**) and stored on the array (R_M, **Fig. 7 (b)**).

IV. Image Reconstruction with Device Non-ideality

Device non-ideality brings different challenges to the system robustness of analog RRAM storage than digital (binary) storage. Specifically, there are two main kinds of device non-ideality that degrade analog array storage capability: error cells and resistance relaxation.

Error Cells – The full resistance dynamic range (max. and min. achievable resistance values) of different cells within an array may differ. For a specific cell, if the target resistance assumed by the algorithm is outside the full dynamic range of the cell by a distance larger than the acceptance range, error values are generated. Our experimental array had an error rate of 0.2% (1 out of 448). Different images could have different number of error bits based on the targeted resistance from the encoder. In order to achieve reliable reconstruction results, the algorithm needs to tolerate the worst-case scenario (maximum error rate). We trained the network with an estimated maximum error rate of 2%. Note that this is a very high error rate for a product technology and we used this as a way to illustrate the error resiliency of our methodology. To test the algorithm's error-resiliency, we explored the relation between reconstruction mean squared error (MSE) and storage error rate. Various error rates (0.5%, 1%, 2%, 5%, and 10%) were dialed in by randomly selecting a set of cells and SETTING them to LRS (**Fig. 8**). **Fig. 9** shows reconstructions (top) for the original image and corresponding MSE (bottom) for the different error rates (all images are scaled to have pixel values of mean = 0, variance = 1). Interestingly, even though the network was trained with 2% error, the algorithm was able to tolerate error up to 5% before suffering from serious degradation.

Resistance Change After Programming – Our DD-ISPP method is capable of programming cell resistances efficiently within the acceptance range given by the target. However, when decoding from resistance values read from the array, time-dependent variability (TDV), which cannot be suppressed with write-verification methods [8][9], may cast the resistance values outside of acceptance ranges and degrade reconstructions. TDV must be properly considered when evaluating image reconstruction performance. After storing the resistances onto the array, resistance drift over time was monitored (**Fig. 10**). **Fig. 11 (a)** shows the final resistances programmed with DD-ISPP. Within 1 second after programming, we read the resistance again (**Fig. 11 (b)**) and observed some outliers due to read noise and short-term relaxation [8]. **Fig. 11 (c)** shows that more outliers appear 1 minute after programming. A broader distribution is observed when reading resistance 12 hours after programming (**Fig. 11 (d)**). **Fig. 12** shows the reconstructions (top) of the original image and their corresponding MSE (bottom) over time. **Fig 12 (a)** shows compression-only, where R_T=R_M. **Fig. 12 (b)-(e)** correspond to the reconstruction results from the resistance read values from the array shown in **Fig. 11 (a)-(d)**. Our scheme is resilient to outliers, where "verify-read", "direct read" and "1 minute after write" show similar MSE and resistance distribution broadening after 12-hours only causes a small increase in MSE.

It should also be noted that at this compression rate of 1 device/10 pixels and 1 bit/ channel, JPEG would produce a very poor reconstruction. As shown previously [4], if the goal is simply to perform compression with a set of additive-noise storage devices (neglecting the more realistic device failures) the neural-network algorithm is able to outperform JPEG.

V. Conclusion

Key achievements: (1) A joint source-channel coding algorithm was developed, adapting to RRAM array characteristics to store analog data directly onto the analog-valued RRAM array. (2) An image storage and compression task was demonstrated by experiments with an RRAM array. (3) Our method empirically showed error-resilience against device non-idealities such as error cells, cycle-to-cycle and device-to-device variations, and resistance relaxation.

Acknowledgment

Work supported in part by ASCENT, one of the six centers in JUMP, ENIGMA from the NSF/SRC, E2CDA, NSFGRFP, and the Stanford SystemX Alliance and Stanford NMTRI.

REFERENCES

[1] JH. Engel, *et al. IEDM* (2014)
[2] P. Vincent *et al. ICML* (2008)
[3] J. Balle *et al. ICLR* (2017)
[4] R. Zarcone, *et al. DCC* (2018)
[5] J. Deng *et al. CVPR* (2009)
[6] B. Thomee *et al, Communications of the ACM*, 59, 2, pp. 64-73 (2016)
[7] F. Alibart, *et al. Nanotechnology*, 23, 7, p.075201 (2012)
[8] A. Fantini, *et al. IEDM* (2015)
[9] S. Ambrogio, *et al. IEEE TED*, 62, 11, pp.3812-3819 (2015)

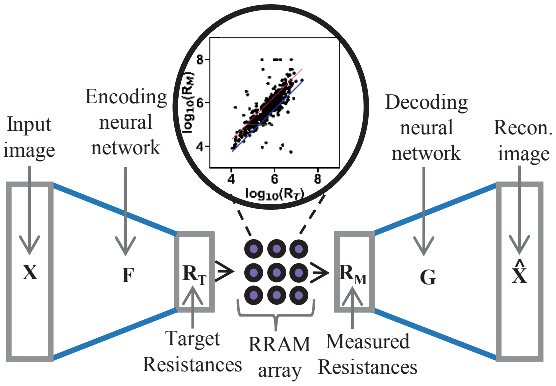


Fig. 1 Diagram of the autoencoder architecture. The input image, X, is transformed through the encoding neural network, F, into a set of target write resistances, R_T. These resistances are then written to the device array (purple circles). The devices are then read, yielding measured resistances R_M. R_M are passed through the decoding neural network, G, yielding the reconstructed image , $\hat{X}$.

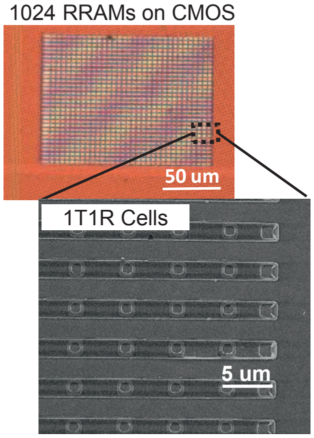


Fig. 2 SEM image of the 1K 1T1R analog-valued RRAM array used for storing image in this work

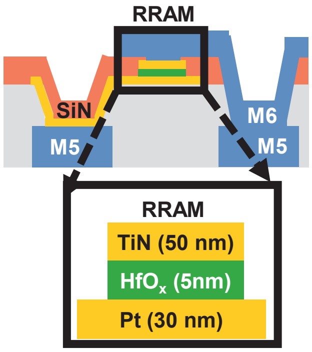


Fig. 3 The cross-section schematic of RRAM stack: Pt (30 nm) / HfOx (5 nm) / TiN (50 nm). SiN : passivation layer. Al : M7 layer

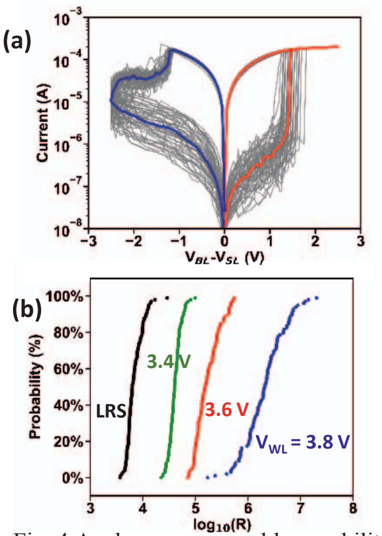
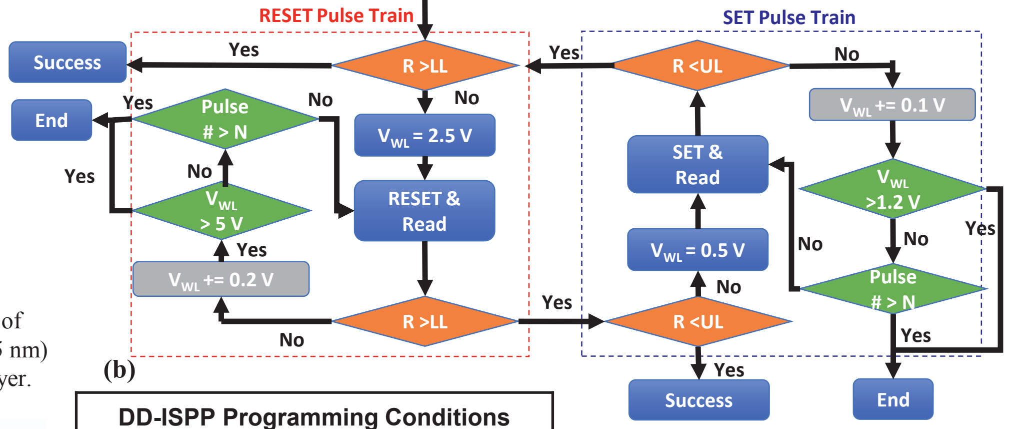


Fig. 4 Analog programmable capability of RRAM cell (a) Typical I-V curves of one cell in RRAM array. Median set (red) and reset (blue) of 50 set (V_{WL} = 1.2 V, V_{SL} = 0 V) /reset (V_{WL} = 4.5 V, V_{BL} = 0 V) cycles. (b) Multiple resistance levels achieved by pulse RESET with different V_{WL} (Pulse Width = 100 ns, V_{SL} = 3.5 V, V_{BL} = 0 V).

LL: Lower Limit of Acceptance Range
UL: Upper Limit of Acceptance Range
N: Maximum number of write pulses



(b)

DD-ISPP Programming Conditions

	RESET	SET
V_{WL} Start (V)	2.5	0.5
V_{WL} Stop (V)	5	1.2
V_{WL} Step (V)	0.2	0.1
V_{BL} (V)	0	2.5
V_{SL} (V)	3.5	0
Pulse Width (ns)	100	50

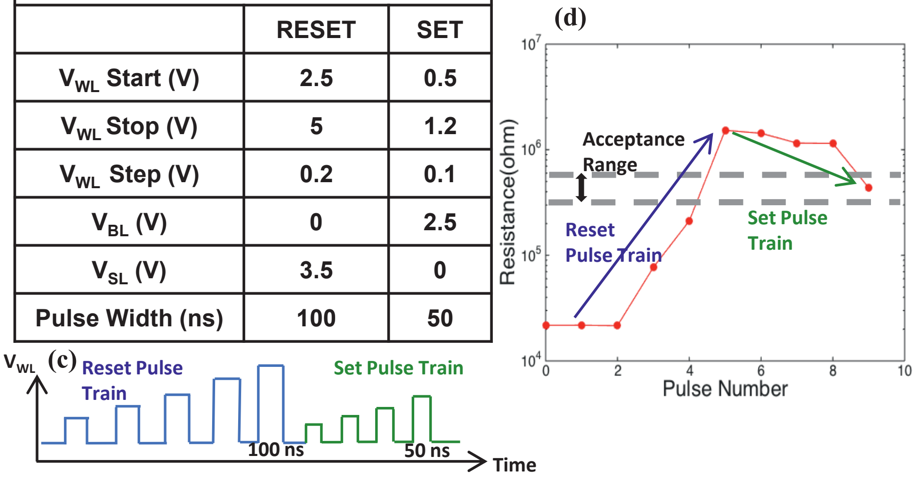


Fig. 5 DD - ISPP Scheme (a) Flow chart showing how to use DD-ISPP to fine tune the resistance into acceptance range using incremental step RESET pulse train and incremental step SET pulse train (b) Table: SET and RESET incremental pulse train parameters used in DD-ISPP. (c) -(d) Example writing with DD – ISPP showing the over-programming can be fixed by programming in the opposite direction, and starting from minimal voltage when changing the direction can minimize programming across the range and thus save energy. (c) write pulse train waveform (d) measured resistance as a function of pulse number.

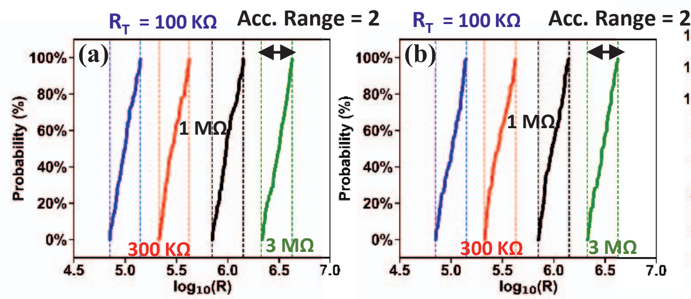


Fig. 6 Resistance generated from DD-ISPP scheme follows uniform distribution $\log_{10}(R_M) = \log_{10}(R_T)+\log_{10}(2)\bullet U(-0.5,0.5)$, where U(-0.5,0.5) is the uniform distribution function. (a) 1 cell, 100 cycles for each R_T (b)100 cell, 1 cycle for each R_T

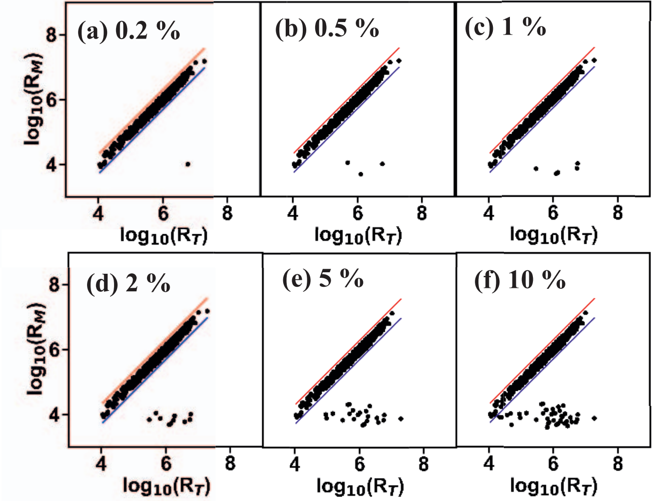
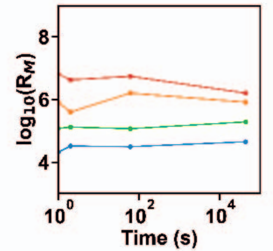
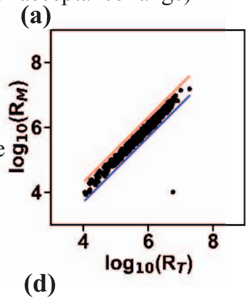


Fig. 8 (a) Writing error rate of 0.2% is achieved in experiment (b)-(f)Various writing error rates (0.5%, 1%, 2%, 5%, and 10%) are dialed into the measurement by randomly selecting a portion (error rate) of cells and SET them to LRS. (red line: upper limit of acceptance range, blue line: bottom limit of acceptance range)



(left) Fig. 10 Example resistance drift along with time after programming



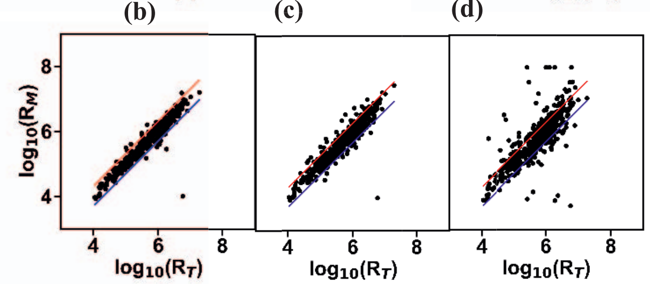


Fig. 11 Resistance change after programming (a) Resistance value after write-verify (b) Immediately read after write-verify with outliers contributed from read noise and short-term resistance relaxation. (c) More outliers appear when reading 1 minute after programming (d) Wider distribution is observed when reading resistance 12 hours after programming. (red line: upper limit of acceptance range, blue line: bottom limit of acceptance range)

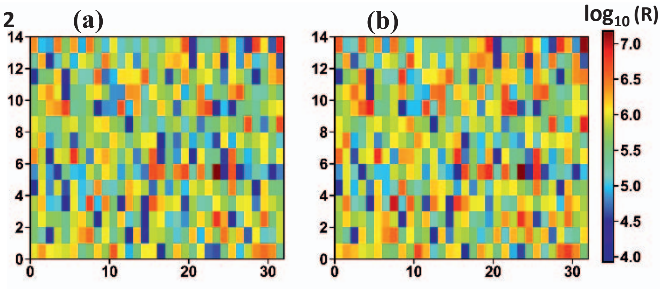


Fig. 7 448 Resistance values stored in (32×14) RRAM array (a) Target resistance log10 (R_T) (b) Programmed RRAM resistance log10 (R_M) , with error rate of 0.2%

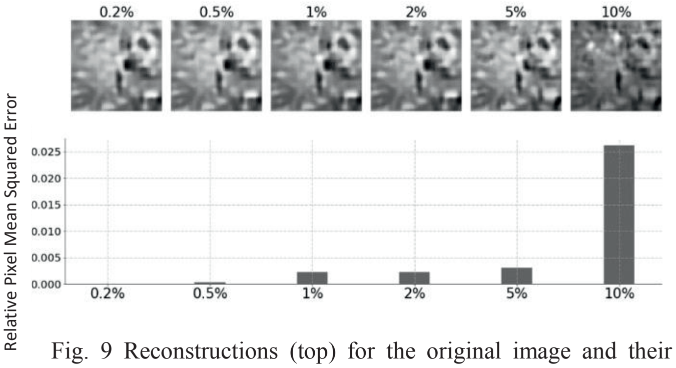


Fig. 9 Reconstructions (top) for the original image and their corresponding relative mean squared error (MSE) (bottom) to the MSE of 0.2% error rate (Fig. 12 (b)) for various error rates (0.2%, 0.5%, 1%, 2%, 5%, 10%) of devices. The network is trained with 2% noise, while it is able to tolerate an error up to 5% before suffering serious reconstruction degradation.

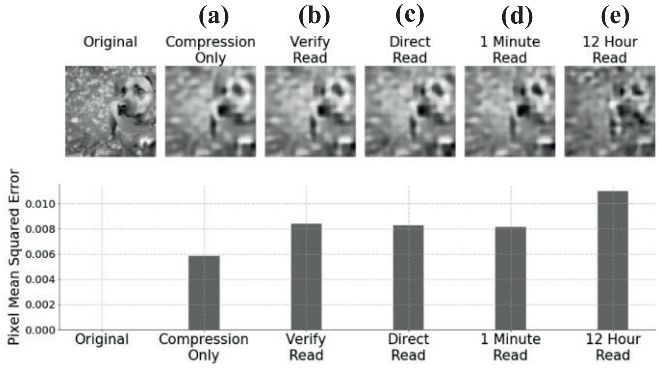


Fig. 12 Reconstructions (top) for the original image and their corresponding mean squared error (MSE) (bottom) for various conditions. (a) Compression only: where images are not passed through the devices and thus R_T=R_M, this corresponds to the case when the storage medium (the RRAM) is perfect, (b) Verify: where write-verified resistance values are used. (c) Direct read: where read noise and short-term resistance relaxation has been added. (d) 1 minute after write: where values are taken after the RRAM array has drifted for 1 minute. (e) 12 hour read: where the RRAM array has relaxed for 12 hours.