Learning rich optical embeddings for privacy-preserving lensless image classification

Eric Bezzam
Audiovisual Communications Laboratory
École Polytechnique Fédérale de Lausanne
eric.bezzam@epfl.ch

Martin Vetterli
Audiovisual Communications Laboratory
École Polytechnique Fédérale de Lausanne
martin.vetterli@epfl.ch

Matthieu Simeoni
Center for Imaging
École Polytechnique Fédérale de Lausanne
matthieu.simeoni@epfl.ch

Abstract

By replacing the lens with a thin optical element, lensless imaging enables new applications and solutions beyond those supported by traditional camera design and post-processing, e.g. compact and lightweight form factors and visual privacy. The latter arises from the highly multiplexed measurements of lensless cameras, which require knowledge of the imaging system to recover a recognizable image. In this work, we exploit this unique multiplexing property: casting the optics as an encoder that produces learned embeddings directly at the camera sensor. We do so in the context of image classification, where we jointly optimize the encoder's parameters and those of an image classifier in an end-to-end fashion. Our experiments show that jointly learning the lensless optical encoder and the digital processing allows for lower resolution embeddings at the sensor, and hence better privacy as it is much harder to recover meaningful images from these measurements. Additional experiments show that such an optimization allows for lensless measurements that are more robust to typical real-world image transformations. While this work focuses on classification, the proposed programmable lensless camera and end-to-end optimization can be applied to other computational imaging tasks.

1 Introduction

Advances in imaging hardware, fabrication techniques, and computational methods have enabled novel camera design strategies that go beyond mimicking the human eye. *Lensless imaging* is one of those approaches, replacing a lens (and necessary focusing distances) with a thinner and potentially inexpensive optical element and a computational image formation step [6]. A variety of applications in virtual/augmented reality, wearables, and robotics can benefit from the low-cost and compact form factor that the lensless imaging paradigm has to offer.

The optical element in such systems is typically a passive or programmable mask placed at a short distance from the sensor. The resulting measurements are highly multiplexed, as seen in Figures 1a to 1d, due to a system response, i.e. *point spread function* (PSF), of large support unlike that of a lens. Figures 1e and 1f show the PSFs of typical lensless encoders, namely a caustic pattern of a height-varying phase mask [1, 5] and a diffracted coded aperture (CA) mask [9, 11, 2].

The majority of contributions in lensless imaging have focused on improving the computational methods to go from raw measurements to demultiplexed images, e.g. from Figures 1a to 1d to

Preprint. Under review.

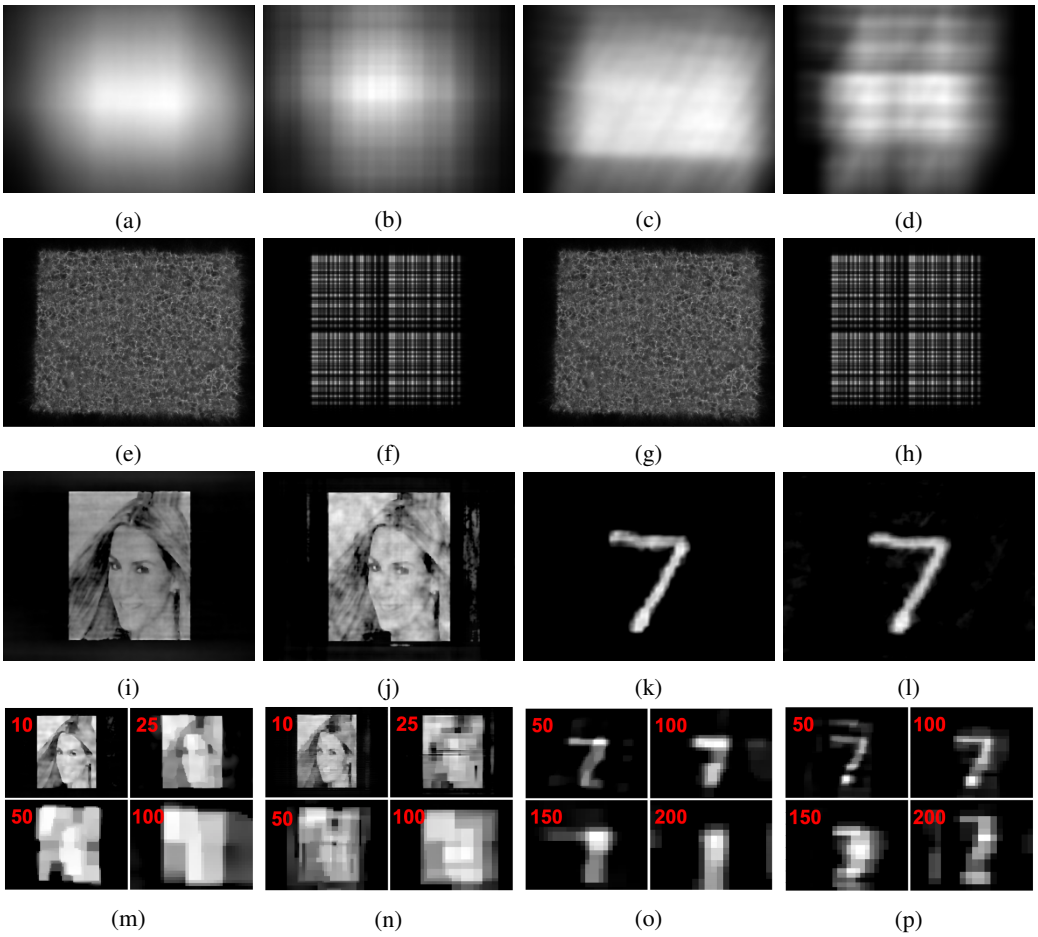


Figure 1: Discerning content from lensless camera raw measurements (top row) is next to impossible, motivating privacy-preserving imaging with such cameras. However, with sufficient knowledge about the camera (e.g. a point spread function, second row) and an appropriate computational algorithm, one is able to recover an estimate of the underlying object (third row, using ADMM [7] and a total variation prior). If the raw sensor measurement is under-sampled, it becomes increasingly difficult for classical recovery algorithms to recover an meaningful estimate of the underlying object. (Bottom row) PSF and raw measurements are downsampled by the factor in the top left corner, simulating a sensor of lower resolution prior to reconstruction.

Figures 1i to 1l. Data-driven techniques and deep learning have had an influential role in this progress, yielding faster reconstruction times and improved reconstruction quality [17, 22, 26] with respect to classical techniques based on system inversion [11, 2, 14] and convex optimization [15, 1, 5].

While machine learning advances have been readily incorporated in lensless imaging reconstruction and classification tasks [34, 25], the design of the optical element itself remains rather heuristic-based. Criteria such as sparsity, a large number of directional filters, high contrast, a delta-like autocorrelation, or designs to simplify the computational recovery [9, 11, 2, 1, 5] have been used to tackle the task of PSF engineering, independent of the down-stream task. The potential to *jointly* optimize the optical encoding and a digital post-processing has been successfully demonstrated in other computational tasks, albeit with lenses, for extended depth-of-field [33, 29], super-resolution [33], classification [8], 3-D imaging [20, 10], and hyperspectral imaging [35, 24].

In this paper, we apply end-to-end optimization to a lensless camera to jointly learn (1) a programmable mask pattern prior to the sensor measurement *and* (2) the subsequent digital processing. As well as exploiting edge components for compute and the compactness of lensless cameras, privacy-preserving classification is one of the key motivations for this approach. The multiplexed measurements of lensless cameras have been touted to maintain visual privacy [6, 23, 30, 31] as they

contain hardly any perceivable features. However, a malicious user with access to the camera can still recover an image of the underlying object through a couple measurements and clever post-processing. The objective of this work is to jointly optimize the optical encoding and the digital classifier in order to significantly reduce the size of the sensor "embedding" by exploiting this multiplexing characteristic. As the sensor resolution decreases, it becomes increasingly difficult for lensless imaging reconstruction techniques to recover a meaningful image, as demonstrated in Figures 1m to 1p. Jointly optimizing this one-to-many mapping for a particular task, e.g. classification, has the potential to produce richer embeddings, much like digital encoders [13], with a lower resolution sensor, all the while maintaining performance on the task at hand and enhancing visual privacy.

Contributions In this work, we exploit multiplexing properties of lensless cameras in order to learn privacy-preserving embeddings by training the imaging system end-to-end. Concretely, we determine the optimal pattern for a programmable component prior to the sensor, i.e. an amplitude spatial light modulator (SLM), in order to perform image classification.

To the best of our knowledge, one recent work has applied end-to-end optimization for lensless imaging with passive masks [31]; however none with programmable components. Using such components can help reduce model mismatch through hardware-in-the-loop (HITL) [28] or equivalently, physics-aware training [37]. Moreover, the re-programmability of an SLM means the end-to-end optimized camera does not have to be relegated to a single application or setting. It can be updated after deployment or conveniently reconfigured for a different task or in the case of a malicious user.

Our experiments on handwritten digit classification demonstrate the potential of significantly reducing the embedding at the sensor, as our end-to-end approach consistently performs better than lensless cameras with a fixed encoder. Moreover, we show that jointly learning the SLM pattern with the classification task is more robust to typical image transformations: shifting, rescaling, rotating, perspective changes. We are unaware of any other work that has studied the consequences of such effects on lensless imaging.

Our end-to-end approach is based upon an imaging system that can be put together from cheap and accessible components, totaling at around 100 USD. As an SLM, we use a low-cost liquid crystal display (LCD), as in [39, 15], which costs about 20 USD. To the best of our knowledge, we are the first to employ such a device in an end-to-end optimization for computational optics, as opposed to commercial SLMs which cost a few thousand USD. Our differentiable digital twin of the imaging system models incoherent, polychromatic propagation with the selected LCD component, using the bandlimited angular spectrum method (BLAS) [21] to account for diffraction.

In order to foster reproducibility, we open source the following under the GNU General Public License v3.0: wave propagation simulation[1] and training software[2] Moreover, we have previously released a package to interface with the baseline and proposed cameras.[3]

2 Problem statement

End-to-end approaches for optimizing optical components, also known as *deep optics* [36] is a recent trend enabled by improved fabrication techniques and the continual development of more powerful and efficient hardware and libraries for machine learning. It is motivated by faster and cheaper inference for edge computing (taking advantage of the speed of light) and a desire to co-design the optics and the computational algorithm to obtain optimal performance for a particular application.

An encoder-decoder perspective is often used to frame such end-to-end approaches, casting the optics as the encoder and the subsequent computational algorithm as the decoder, as shown in Figure 2, and can be formulated as the following optimization problem minimized for a labeled dataset $\{\boldsymbol{x}_i, \boldsymbol{y}_i\}_{i=1}^N$:

$$\hat{\boldsymbol{\theta}}_E, \hat{\boldsymbol{\theta}}_D = \underset{\boldsymbol{\theta}_E, \boldsymbol{\theta}_D}{\operatorname{argmin}} \sum_{i=1}^{N} \mathcal{L}\Big(\boldsymbol{y}_i, \underbrace{D_{\boldsymbol{\theta}_E, \boldsymbol{\theta}_D}\big(\overbrace{O_{\boldsymbol{\theta}_E}(\boldsymbol{x}_i)}^{\text{embedding } \boldsymbol{v}_i}\big)}_{\text{decoder output } \hat{\boldsymbol{y}}_i}\Big). \tag{1}$$

[1] `https://github.com/ebezzam/waveprop`

[2] `https://github.com/ebezzam/LenslessClassification`

[3] `https://github.com/LCAV/LenslessPiCam`

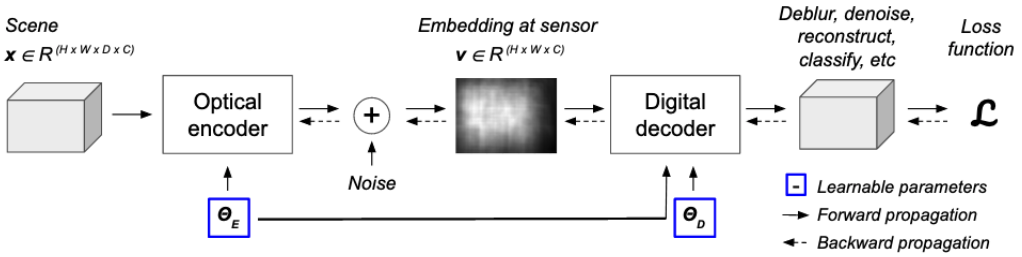


Figure 2: Encoder-decoder perspective of cameras for end-to-end optimization. The scene could be four-dimensional (height, width, depth, and color) whereas the embedding measured at the sensor is at most three-dimensional (height, width, and color).

$O_{\boldsymbol{\theta}_E}(\cdot)$ is the optical encoder, including additive noise, that outputs the sensor embedding $\boldsymbol{v}_i$ of an input $\boldsymbol{x}_i$. The encoder encapsulates propagation in free space and through all optical components prior to the sensor. While this component can be simulated via a digital twin, the hardware itself can be used to produce physical realizations of $\boldsymbol{v}_i$. Moreover, if the encoder parameters $\boldsymbol{\theta}_E$ of the physical system can be modified, the device itself can be used for forward propagation, and a differentiable digital model for backpropagating the error between the ground truth $\boldsymbol{y}_i$ and the decoder output $\hat{\boldsymbol{y}}_i$ that arose from $\boldsymbol{v}_i$ that came directly from the device. This is the essence of HITL / physics-aware training. In some cases, the hardware can also be used for backpropagation [38].

$D_{\boldsymbol{\theta}_E,\boldsymbol{\theta}_D}(\cdot)$ is the digital decoder, which can perform a whole slew of tasks: deblurring, denoising, image reconstruction, classification, etc. It has its own set of parameters $\boldsymbol{\theta}_D$ and can optionally make use of the optical encoder parameters, e.g. for physics-based learning [20]. Its output is fed to a loss function $\mathcal{L}(\cdot)$ along with the ground-truth output $\boldsymbol{y}_i$.

In Section 3 we present the hardware for our proposed lensless imaging system and how we model the digital twin for our optical encoder. In Section 4, as we explain our task, we present the architecture of our digital decoder, the loss function, and the labeled data $\{\boldsymbol{x}_i, \boldsymbol{y}_i\}_{i=1}^N$ for our experimental setup.

3 Proposed solution for lensless classification

Our proposed camera design is motivated by the benefits of lensless cameras (compact, low-cost, privacy-preserving) and programmability.

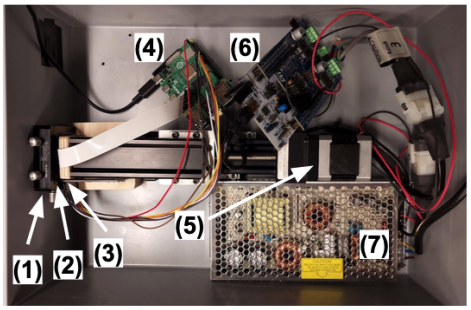
To this end, a transmissive SLM serves as the only optical component in our encoder, specifically an off-the-shelf LCD driven by the ST7735R device which can be purchased for $\$20$.[4] It can be wired to a Raspberry Pi ($\$35$) with the Raspberry Pi High Quality 12.3 MP Camera ($\$50$) as a sensor, totaling our design to just $\$105$.

An experimental prototype of the proposed design with the aforementioned components can be seen in Figure 3. The prototype includes an adjustable aperture and a stepper motor for programmatically setting the distance between the SLM and the sensor, both of which can be removed to produce a more compact design, similar to Figure A.3b of a lensless camera with a fixed diffuser.

Digital twin of optical encoder End-to-end optimization requires a sufficiently accurate and differentiable simulation of the physical setup. Our digital twin of the imaging system shown in Figure 3 accounts for wave-based image formation for spatially incoherent, polychromatic illumination, as is typical of natural scenes. A simulation based on wave-optics is necessary to account for diffraction due to the small SLM features and for wavelength-dependent propagation.

We adopt a common assumption from Fourier optics, namely that image formation is a linear shift-invariant (LSI) system between two parallel planes for a given wavelength [12]. This implies that, there exists an impulse response, i.e. a PSF, that can be convolved with the *scaled* scene in order to obtain its image at a given distance and for a specific wavelength. This convolution relationship is described in Appendix A.1.

[4] `https://www.adafruit.com/product/358`



(1) Adjustable aperture.
(2) ST7735R SLM.
(3) HQ camera sensor.
(4) Raspberry Pi to control SLM and sensor.
(5) Stepper motor.
(6) Microcontroller for stepper motor.
(7) Power source for stepper motor.

Figure 3: Experimental prototype of programmable, amplitude SLM-based camera.

Therefore, our digital twin modeling amounts to obtaining a PSF that encapsulates propagation from a given plane in the scene to the sensor plane. There are two ways to obtain this PSF: measuring it with a physical setup or simulating it. For end-to-end approaches, a differentiable simulator is typically necessary in order to backpropagate the error to update the optical encoder parameters. In Appendix A.5 we describe our modeling of this PSF for an SLM placed at a short distance in front of the sensor, as is the case for our imaging device. The learnable parameters $\boldsymbol{\theta}_E$ of our optical encoder with the ST7735R component include: the SLM pixel amplitude values $\{w_k\}_{k=1}^K$ and the distance between the SLM and the image plane d_2. While both can be optimized in an end-to-end fashion, in this work we concentrate on optimizing $\{w_k\}_{k=1}^K$ jointly with the digital decoder parameters $\boldsymbol{\theta}_D$.

4 Experiments

In this section, we apply our proposed camera and end-to-end optimization to handwritten digit classification (MNIST) [19]. Equation (1) can be slightly modified to

$$\hat{\boldsymbol{\theta}}_E, \hat{\boldsymbol{\theta}}_D = \underset{\boldsymbol{\theta}_D, \boldsymbol{\theta}_E}{\operatorname{argmin}} \sum_{i=1}^{N} \mathcal{L}\Big(y_i, \underbrace{D_{\boldsymbol{\theta}_D}\overbrace{\big(O_{\boldsymbol{\theta}_E}(\boldsymbol{x}_i)\big)}^{\text{embedding } \boldsymbol{v}_i}}_{\text{decoder output } \hat{\boldsymbol{p}}_i}\Big), \tag{2}$$

as our decoder $D_{\boldsymbol{\theta}_D}(\cdot)$ does not require information from the encoder in order to classify digits. The original $\{\boldsymbol{x}_i\}_{i=1}^N$ coming from MNIST are (28×28) images of handwritten digits and $\{y_i\}_{i=1}^N$ are labels from 0 to 9. $\{\boldsymbol{x}_i\}_{i=1}^N$ are simulated and resized to the dimensions of the PSF, as described in Appendix A.2, and the decoder outputs $\{\hat{\boldsymbol{p}}_i\}_{i=1}^N$ are length-10 vectors of scores for each label.

We conduct two experiments for evaluating the effectiveness of jointly optimizing the optical encoder and the classification task:

1. Section 4.1: reduce the dimension of the embedding $\boldsymbol{v}_i$ at the sensor and study its impact on classification performance. A lower resolution embedding at the sensor corresponds to enhanced visual privacy, as demonstrated in the introduction with Figures 1m to 1p.
2. Section 4.2: apply common real-world image transformations (shifting, rescaling, rotating, perspective changes) to evaluate the robustness of the proposed camera and the end-to-end optimization to such deformations.

$D_{\boldsymbol{\theta}_D}(\cdot)$ takes on one of two architectures in our experiments: multi-class logistic regression or a two-layer fully-connected neural network (FCNN), which are detailed in Appendix A.7.1 and Appendix A.7.2 respectively. In training both architectures, we use a cross entropy loss between the ground truth labels and the outputs of the decoder, and train for 50 epochs with a batch size of $N = 200$ and the Adam optimizer [18]. More information on the training hyperparameters and compute hardware can be found in Appendix A.7.

We use the provided train-test split of MNIST: $60'000$ training and $10'000$ test examples. Each example is simulated as per the approach described in Appendix A.2 for an object-to-camera distance of $40\,\mathrm{cm}$, a signal-to-noise ratio of $40\,\mathrm{dB}$, and an object height of $12\,\mathrm{cm}$ (unless specified otherwise). We compare six imaging systems in our experiments, and for each camera, a PSF is needed to

perform this simulation. Below is a brief description of each camera and how we obtain its PSF for an object-to-camera distance of $40\,\mathrm{cm}$:

- *Lens*: measured PSF for the camera shown in Figure A.3a with the lens focused at $40\,\mathrm{cm}$.
- *CA* (coded aperture): a binary mask is generated by taking the outer product of a maximum length sequence (MLS), as is done in [2]. A simulation of its diffraction pattern is used as the PSF.
- *Diffuser*: measured PSF for the camera shown in Figure A.3b, where the diffuser is placed roughly $4\,\mathrm{mm}$ from the sensor. The diffuser is double-sided tape as in the DiffuserCam tutorial [4]. In [3], the authors demonstrate the effectiveness of this simple diffuser for imaging when used with the Raspberry Pi High Quality Camera.
- *Fixed SLM (m)*: measured PSF for the proposed camera shown in Figure 3 for a randomly programmed pattern. The mask-to-sensor distance is programmatically set to $4\,\mathrm{mm}$ via the stepper motor to match the distance of the diffuser-based camera.
- *Fixed SLM (s)*: simulated PSF for the proposed camera, using the approach described in Appendix A.5 for a random set of SLM amplitude values and a mask-to-sensor distance of $4\,\mathrm{mm}$.
- *Learned SLM*: simulated PSF for the proposed camera that is obtained by optimizing Equation (2) for the SLM weights and then simulating the corresponding PSF using the approach described in Appendix A.5 for a mask-to-sensor distance of $4\,\mathrm{mm}$. During training, the PSF changes at each batch as the SLM values are updated after backpropagation.

More details such as the components for the measured PSFs and simulation details can be found in Appendix A.6. For the fixed optical encoders, the embeddings $\{\boldsymbol{v}_i\}_{i=1}^N$ can be pre-computed with the approach described in Appendix A.2. The resulting augmented dataset is normalized (according to the augmented training set statistics) prior to optimizing the classifier $D_{\boldsymbol{\theta}_D}(\cdot)$. For *Learned SLM*, we apply batch normalization [16] and a ReLu activation to the sensor embedding prior to passing it to the classifier. At inference, the parameters of batch normalization are fixed.

4.1 Varying embedding dimension

Table 1 reports the best test accuracy for each optical encoder, for a varying sensor embedding dimension and for two digital classification architectures: logistic regression and two-layer FCNN. The test accuracy curves can be found in Appendix A.8.

While all approaches decrease in performance as the embedding dimension reduces, *Learned SLM* is the most resilient as quantified by Table 2. The performance gap between *Learned SLM* and fixed lensless encoders, as shown in Table 1, decreases when a two layer FCNN is used. However, the benefits of learning this multiplexing are still evident for a very low embedding dimension of (3×4).

Moreover, the benefits of lensless multiplexing (for both fixed and learned encoders) can be clearly observed as the sensor dimension decreases and the *Lens*' performance deteriorates. Figure 4 provides some insight into this as downsampled measurements of *Lens* can consist of a single pixel. On the other hand, the multiplexing property of lensless cameras leads to much richer measurements, for both

Table 1: MNIST accuracy on test set, simulated accordingly.

Classifier $\rightarrow$	Logistic regression				Single hidden layer, 800 units			
Embedding $\rightarrow$ *Encoder* $\downarrow$	24×32 =768	12×16 =192	6×8 =48	3×4 =12	24×32 =768	12×16 =192	6×8 =48	3×4 =12
Lens	92.3%	74.8%	42.8%	18.4%	97.7%	83.0%	41.8%	18.8%
CA	74.1%	74.2%	64.4%	59.1%	97.3%	96.3%	91.0%	69.9%
Diffuser	91.0%	81.6%	72.6%	48.5%	95.8%	95.7%	92.8%	77.2%
Fixed SLM (m)	92.7%	91.5%	82.1%	68.7%	97.2%	97.1 %	95.9%	84.2%
Fixed SLM (s)	92.6%	91.5%	84.9%	65.8%	97.3%	97.4%	95.9%	86.4%
Learned SLM	**94.2%**	**92.9%**	**91.9%**	**83.0%**	**97.9%**	**97.7%**	**96.6%**	**90.3%**

Table 2: Relative drop in performance due to embedding compression from (24×32) to (3×4).

	Logistic regression	Single hidden layer, 800 units
Lens	$80.0\,\%$	$80.8\,\%$
CA	$20.2\,\%$	$28.2\,\%$
Diffuser	$46.7\,\%$	$19.4\,\%$
Fixed SLM (m)	$25.9\,\%$	$13.4\,\%$
Fixed SLM (s)	$28.9\,\%$	$11.2\,\%$
Learned SLM	$\mathbf{11.9}\%$	$\mathbf{8.42}\%$

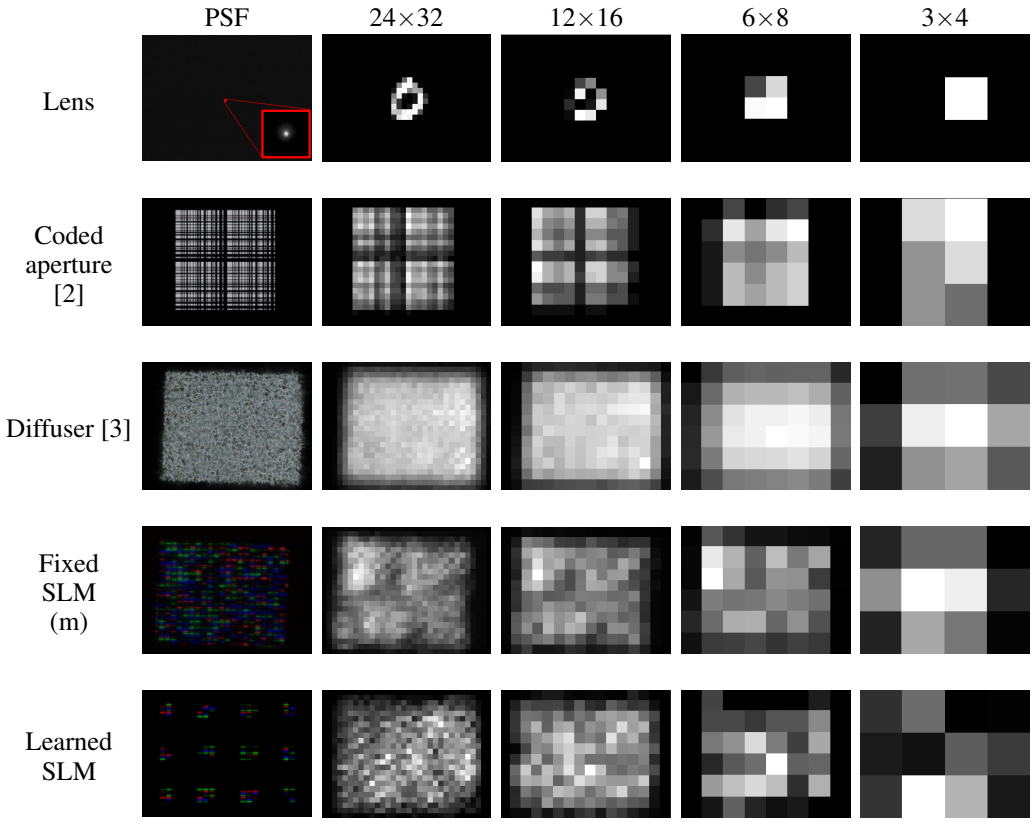


Figure 4: PSFs of the baseline and proposed camera systems, and example sensor embeddings that are simulated with the approach described in Appendix A.2. *Lens*, *Diffuser*, and *Fixed SLM (m)* used measured PSFs while *Coded aperture* and *Learned SLM* used simulated ones. *Fixed SLM (s)* is not shown as it is similar to *Fixed SLM (m)*. The PSF for *Learned SLM* is unique per model and embedding dimension. The one shown above was optimized for an embedding dimension of (3×4) and a two-layer fully connected neural network. All of the learned PSFs can be seen Appendix A.11.

Learned SLM and the fixed lensless encoders. In Appendix A.9, we show example reconstructions for the varying embedding dimensions to show how visual privacy is enhanced by these lower resolution sensor embeddings even with knowledge of the PSF.

The PSFs corresponding to *Learned SLM* for the different embedding dimensions and the two classifiers can be found in Appendix A.11.1. It is interesting to note the PSFs for the embedding dimension of (3×4) (also visible in the last row of Figure 4). They resemble the small kernels of convolutional neural networks (CNNs) which has motivated the design of amplitude masks in other end-to-end optimization tasks [8, 31]. In our optimization, these small kernels were not set as an

Table 3: MNIST accuracy on randomly transformed test set, simulated accordingly.

Embedding dimension ↓	*Encoder* ↓	Original	Shift	Rescale	Rotate	Perspective
$(24 \times 32) = 768$	Lens	$97.7\,\%$	$\mathbf{84.4}\%$	$84.8\,\%$	$94.4\,\%$	$83.0\,\%$
	CA	$97.3\,\%$	$22.8\,\%$	$82.1\,\%$	$90.9\,\%$	$31.6\,\%$
	Diffuser	$95.8\,\%$	$42.8\,\%$	$77.8\,\%$	$89.7\,\%$	$63.2\,\%$
	Fixed SLM (m)	$97.2\,\%$	$50.9\,\%$	$83.0\,\%$	$93.2\,\%$	$77.4\,\%$
	Fixed SLM (s)	$97.3\,\%$	$48.8\,\%$	$84.3\,\%$	$93.7\,\%$	$78.0\,\%$
	Learned SLM	$\mathbf{97.9}\%$	71.7%	$\mathbf{92.0}\%$	$\mathbf{95.0}\%$	$\mathbf{83.7}\%$
$(6 \times 8) = 48$	CA	$91.0\,\%$	$14.9\,\%$	$71.8\,\%$	$76.8\,\%$	$20.1\,\%$
	Diffuser	$92.8\,\%$	$26.7\,\%$	$69.7\,\%$	$84.9\,\%$	$43.9\,\%$
	Fixed SLM	$95.9\,\%$	$29.5\,\%$	$78.3\,\%$	$89.3\,\%$	$58.7\,\%$
	Fixed SLM (sim.)	$95.9\,\%$	$29.0\,\%$	$77.9\,\%$	$89.7\,\%$	$57.8\,\%$
	Learned SLM	$\mathbf{96.6}\%$	$\mathbf{59.3}\%$	$\mathbf{88.4}\%$	$\mathbf{93.1}\%$	$\mathbf{73.4}\%$

explicit constraint, but the resizing to a (3×4) sensor embedding may explain why we observe 12 equally-spaced sub-masks.

4.2 Robustness to common image transformations

The MNIST dataset is size-normalized and centered, making it ideal for training and testing image classification systems but is not representative of how images may be taken in-the-wild. In this experiment, we evaluate the robustness of lensless encoders to common image transformation, namely we study the effects of:

- *Shift*: while maintaining an object height of $12\,\mathrm{cm}$ and an object-to-camera distance of $40\,\mathrm{cm}$, shift the image in any direction along the object plane such that it is still fully captured by the sensor.
- *Rescale*: while maintaining an object-to-camera distance of $40\,\mathrm{cm}$, set a random height uniformly drawn from $[2\,\mathrm{cm}, 20\,\mathrm{cm}]$.
- *Rotate*: while maintaining an object height of $12\,\mathrm{cm}$ and an object-to-camera distance of $40\,\mathrm{cm}$, uniformly draw a rotation angle from $[-90°, 90°]$.
- *Perspective*: while maintaining an object height of $12\,\mathrm{cm}$ and an object-to-camera distance of $40\,\mathrm{cm}$, perform a random perspective transformation via PyTorch's `RandomPerspective` with $100\,\%$ probability and a distortion factor of 0.5.[5]

Both the train and test set of MNIST are augmented with the approach described in Appendix A.2 along with each of the above image transformations, i.e. one new dataset per transformation. The same image transformation distribution is used in both training and testing. An illustration of the various image transformations for each camera can be found in Appendix A.12.

Table 3 reports the best test accuracy for each optical encoder and for each image transformation when using a two-layer FCNN classifier, see Appendix A.7.2 for architecture. In the top half of the table, we evaluate the impact of each image transformation for an embedding dimension of (24×32) to see how lensless imaging techniques fare against a lensed camera. The main difficulty for lensless cameras is shifting as the whole sensor no longer captures multiplexed information, as shown in Appendix A.12.1. This leads to a significant reduction in classification accuracy for all lensless approaches. *Learned SLM* is able to cope with shifting much better than the fixed encoding strategies for lensing imaging, most likely because it is able to adapt its multiplexing for such perturbations. For the remaining image transformations, *Learned SLM* is able to outperform the lensed camera, while all of the fixed lensless encodings exhibit worse performance than the lensed camera.

In the bottom half of Table 3, we evaluate the impact of each image transformations for an embedding dimension of (6×8), for which all lensless approaches exhibited satisfactory performance (above

[5] `RandomPerspective` documentation: `https://pytorch.org/vision/main/generated/torchvision.transforms.RandomPerspective.html`

Table 4: Relative drop in performance due to image transformations.

Embedding dimension ↓	*Encoder* ↓	Shift	Rescale	Rotate	Perspective
$(24 \times 32) = 768$	Lens	**13.6**%	13.2 %	3.38 %	15.0 %
	CA	76.6 %	15.6 %	6.58 %	67.5 %
	Diffuser	55.3 %	18.8 %	6.37 %	34.0 %
	Fixed SLM (m)	47.6 %	14.6 %	4.12 %	20.4 %
	Fixed SLM (s)	49.8 %	13.4 %	3.70 %	19.8 %
	Learned SLM	26.8 %	**6.03**%	**2.96**%	**14.5**%
$(6 \times 8) = 48$	CA	83.6 %	21.1 %	15.6 %	77.9 %
	Diffuser	71.2 %	24.9 %	8.51 %	52.7 %
	Fixed SLM (m)	69.2 %	18.4 %	6.88 %	38.8 %
	Fixed SLM (s)	69.8 %	18.8 %	6.47 %	39.7 %
	Learned SLM	**38.6**%	**8.49**%	**3.62**%	**24.0**%

90%) on the original dataset and more "protection" against post-processing recovery, as shown in Appendix A.9. We do not consider *Lens* for this embedding dimension as it performed poorly for the simulated dataset without any transformations. Once again, *Learned SLM* is more robust to image transformations, in particular shifts and perspective changes. Table 4 quantifies the reduction in classification performance due to each of the image transformations, with *Learned SLM* being the least affected among the lensless imaging approaches.

Figure A.10 shows the PSFs corresponding to the *Learned SLM* masks for the two embedding dimensions and the various image transformations. It is worth noting that the masks for the embedding dimension of (6×8) trained with image transformations are denser than the mask that was obtained without image transformations (Figure A.9g). This may be a result of a need for more degrees-of-freedom to account for the higher complexity in the input space due to these distortions.

5 Conclusion

We have introduced a low-cost and programmable lensless imaging system that can produce robust, privacy-preserving classification results. This is achieved through an end-to-end training that jointly optimizes (1) an optical encoding that produces highly multiplexed embeddings directly at the sensor and (2) the architecture that classifies these privacy-preserving measurements. Our experiments on handwritten digit classification show that the proposed design and training strategy outperforms lensless systems that employ a fixed optical encoding, and is more resilient to common real-world image transformations. Moreover, jointly training the optical encoder and the digital decoder allows one to reduce the sensor resolution, further enhancing the visual privacy of the measurements. Adding to the security of the camera is the ability to re-configure the optical encoder if a malicious user obtains information that can be used to decode the sensor embeddings.

For future work, we plan exploit the programmability aspect of our proposed camera with real world data, namely employ hardware-in-the-loop (HITL) techniques as this has been shown to reduce model mismatch [28, 37]. Moreover, a programmable mask allows for time-multiplexed measurements that can be used as additional features for an imaging or classification task [14, 35].

Limitations Training end-to-end is expensive due to optical wave propagation simulation. This could be alleviated by using the hardware itself to perform the forward propagation, as is done in HITL, but this approach comes with its own limitations as forward propagation cannot be parallelized for a batch of training examples.

Relying on physical devices for computation can also have drawbacks. They are more susceptible to degradation (due to usage and over time) than purely digital computations. Moreover, device tolerances can lead to unwanted differences between two seemingly identical setups. Such differences may be more prominent for low-cost components such as the cheap LCD used in this paper, as opposed to commercial SLMs.

Acknowledgments and disclosure of funding

We thank Sepand Kashani for his input and insight at the initial stages of the project, Julien Fageot and Karen Adam for their feedback and discussions, and Arnaud Latty and Adrien Hoffet for their help in building the experimental prototype.

This work was in part funded by the Swiss National Science Foundation (SNSF) under grants CRSII5 193826 "AstroSignals - A New Window on the Universe, with the New Generation of Large Radio-Astronomy Facilities" (M. Simeoni) and 200 021 181 978/1 "SESAM - Sensing and Sampling: Theory and Algorithms" (E. Bezzam).

References

[1] Nick Antipa, Grace Kuo, Reinhard Heckel, Ben Mildenhall, Emrah Bostan, Ren Ng, and Laura Waller. Diffusercam: lensless single-exposure 3d imaging. *Optica*, 5(1):1–9, Jan 2018.

[2] M. Salman Asif, Ali Ayremlou, Aswin Sankaranarayanan, Ashok Veeraraghavan, and Richard G. Baraniuk. Flatcam: Thin, lensless cameras using coded aperture and computation. *IEEE Transactions on Computational Imaging*, 3(3):384–397, 2017.

[3] Eric Bezzam, Sepand Kashani, Martin Vetterli, and Matthieu Simeoni. LenslessPiCam: A hardware and software platform for lensless computational imaging with a Raspberry Pi, 2022.

[4] C. Biscarrat, S. Parthasarathy, G. Kuo, and N. Antipa. Build your own diffusercam: Tutorial, 2018.

[5] Vivek Boominathan, Jesse K. Adams, Jacob T. Robinson, and Ashok Veeraraghavan. Phlatcam: Designed phase-mask based thin lensless camera. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(7):1618–1629, 2020.

[6] Vivek Boominathan, Jacob T Robinson, Laura Waller, and Ashok Veeraraghavan. Recent advances in lensless imaging. *Optica*, 9(1):1–16, 2022.

[7] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Found. Trends Mach. Learn.*, 3(1):1–122, jan 2011.

[8] Julie Chang, Vincent Sitzmann, Xiong Dun, Wolfgang Heidrich, and Gordon Wetzstein. Hybrid optical-electronic convolutional neural networks with optimized diffractive optics for image classification. *Scientific reports*, 8(1):1–10, 2018.

[9] Wanli Chi and Nicholas George. Optical imaging with phase-coded aperture. *Opt. Express*, 19(5):4294–4300, Feb 2011.

[10] Diptodip Deb, Zhenfei Jiao, Alex Bo-Yuan Chen, Misha Ahrens, Kaspar Podgorski, and Srinivas C Turaga. Programmable 3d snapshot microscopy with fourier convolutional networks, 2022.

[11] Michael J. DeWeert and Brian P. Farm. Lensless coded-aperture imaging with separable Doubly-Toeplitz masks. *Optical Engineering*, 54(2):1 – 9, 2015.

[12] J.W. Goodman. Introduction to Fourier optics, 2005.

[13] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

[14] Yi Hua, Shigeki Nakamura, M. Salman Asif, and Aswin C. Sankaranarayanan. Sweepcam — depth-aware lensless imaging using programmable masks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(7):1606–1617, 2020.

[15] Gang Huang, Hong Jiang, Kim Matthews, and Paul Wilford. Lensless imaging by compressive sensing. In *2013 IEEE International Conference on Image Processing*, pages 2101–2105, 2013.

[16] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, page 448–456. JMLR.org, 2015.

[17] Salman S. Khan, Adarsh V. R., Vivek Boominathan, Jasper Tan, Ashok Veeraraghavan, and Kaushik Mitra. Towards photorealistic reconstruction of highly multiplexed lensless images. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[18] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[19] Yann LeCun. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998.

[20] Eric Markley, Fanglin Linda Liu, Michael Kellman, Nick Antipa, and Laura Waller. Physics-based learned diffuser for single-shot 3d imaging. In *NeurIPS 2021 Workshop on Deep Learning and Inverse Problems*, 2021.

[21] Kyoji Matsushima and Tomoyoshi Shimobaba. Band-limited angular spectrum method for numerical simulation of free-space propagation in far and near fields. *Opt. Express*, 17(22):19662–19673, Oct 2009.

[22] Kristina Monakhova, Joshua Yurtsever, Grace Kuo, Nick Antipa, Kyrollos Yanny, and Laura Waller. Learned reconstructions for practical mask-based lensless imaging. *Opt. Express*, 27(20):28075–28090, Sep 2019.

[23] Thuong Nguyen Canh and Hajime Nagahara. Deep compressive sensing for visual privacy protection in flatcam imaging. In *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, pages 3978–3986, 2019.

[24] Pedram Pad, Simon Narduzzi, Clément Kündig, Engin Türetken, Siavash A. Bigdeli, and L. Andrea Dunbar. Efficient neural vision systems based on convolutional image acquisition. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12282–12291, 2020.

[25] Xiuxi Pan, Xiao Chen, Tomoya Nakamura, and Masahiro Yamaguchi. Incoherent reconstruction-free object recognition with mask-based lensless optics and the transformer. *Opt. Express*, 29(23):37962–37978, Nov 2021.

[26] Xiuxi Pan, Xiao Chen, Saori Takeyama, and Masahiro Yamaguchi. Image reconstruction with transformer for mask-based lensless imaging. *Opt. Lett.*, 47(7):1843–1846, Apr 2022.

[27] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

[28] Y. Peng, S. Choi, N. Padmanaban, and G. Wetzstein. Neural Holography with Camera-in-the-loop Training. *ACM Trans. Graph. (SIGGRAPH Asia)*, 2020.

[29] Samuel Pinilla, Seyyed Reza Miri Rostami, Igor Shevkunov, Vladimir Katkovnik, and Karen Eguiazarian. Hybrid diffractive optics design via hardware-in-the-loop methodology for achromatic extended-depth-of-field imaging, 2022.

[30] Yamin Sepehri, Pedram Pad, Pascal Frossard, and L. Andrea Dunbar. Privacy-preserving image acquisition using trainable optical kernel. *CoRR*, abs/2106.14577, 2021.

[31] Wanxin Shi, Zheng Huang, Honghao Huang, Chengyang Hu, Minghua Chen, Sigang Yang, and Hongwei Chen. Loen: Lensless opto-electronic neural network empowered machine vision. *Light: Science & Applications*, 11(1):1–12, 2022.

[32] P.Y. Simard, D. Steinkraus, and J.C. Platt. Best practices for convolutional neural networks applied to visual document analysis. In *Seventh International Conference on Document Analysis and Recognition, 2003. Proceedings.*, pages 958–963, 2003.

[33] Vincent Sitzmann, Steven Diamond, Yifan Peng, Xiong Dun, Stephen Boyd, Wolfgang Heidrich, Felix Heide, and Gordon Wetzstein. End-to-end optimization of optics and image processing for achromatic extended depth of field and super-resolution imaging. *ACM Trans. Graph.*, 37(4), jul 2018.

[34] Jasper Tan, Li Niu, Jesse K. Adams, Vivek Boominathan, Jacob T. Robinson, Richard G. Baraniuk, and Ashok Veeraraghavan. Face detection and verification using lensless cameras. *IEEE Transactions on Computational Imaging*, 5(2):180–194, 2019.

[35] Edwin Vargas, Julien N. P. Martel, Gordon Wetzstein, and Henry Arguello. Time-multiplexed coded aperture imaging: Learned coded aperture and pixel exposures for compressive imaging systems. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 2692–2702, October 2021.

[36] Gordon Wetzstein, Aydogan Ozcan, Sylvain Gigan, Shanhui Fan, Dirk Englund, Marin Soljačić, Cornelia Denz, David AB Miller, and Demetri Psaltis. Inference in artificial intelligence with deep optics and photonics. *Nature*, 588(7836):39–47, 2020.

[37] Logan G Wright, Tatsuhiro Onodera, Martin M Stein, Tianyu Wang, Darren T Schachter, Zoey Hu, and Peter L McMahon. Deep physical neural networks trained with backpropagation. *Nature*, 601(7894):549–555, 2022.

[38] Tiankuang Zhou, Lu Fang, Tao Yan, Jiamin Wu, Yipeng Li, Jingtao Fan, Huaqiang Wu, Xing Lin, and Qionghai Dai. In situ optical backpropagation training of diffractive optical neural networks. *Photon. Res.*, 8(6):940–953, Jun 2020.

[39] A. Zomet and S.K. Nayar. Lensless imaging with a controllable aperture. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 1, pages 339–346, 2006.

A Appendix

A.1 Modeling incoherent polychromatic wave propagation between two planes

Convolutional relationship Figure A.1a illustrates the physical setup assumed by our simulation: the scene of interest is at a fixed distance d_1 from the optical encoder, which itself is at a distance d_2 from the image plane. We adopt a common assumption from *scalar diffraction theory*, namely that image formation is a linear shift-invariant (LSI) system [12]. This implies that there exists an impulse response, i.e. a point spread function (PSF), that can be convolved with the input scene to obtain the output image. In optics, this convolutional relationship is between the *scaled* scene and the image, namely

$$U_2(x, y; \lambda) = \int_{\mathbb{R}} dr \int_{\mathbb{R}} ds\, h(x - r, y - s; \lambda) \left[\frac{1}{|M|} U_0\Big(\frac{r}{M}, \frac{s}{M}; \lambda\Big) \right], \tag{3}$$

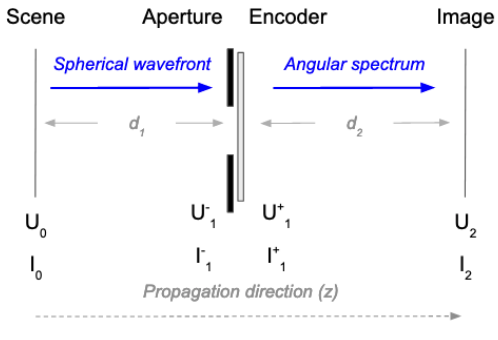
where U_0 and U_2 are the wave fields, i.e. complex amplitudes, at the scene and the image planes respectively, h is the PSF, and $M = -d_2/d_1$ is a magnification factor that also accounts for inversion [12]. Note that this convolution is dependent on the wavelength λ.

This LSI assumption significantly reduces the computational load for simulating optical wave propagation, as the convolution theorem and the fast Fourier transform (FFT) algorithm can be used to efficiently evaluate the wave field at the image plane via the spatial frequency domain. An aperture, or some form of cropping, helps to enforce the LSI assumption in order to avoid new patterns from emerging at the sensor for lateral shifts at the scene plane. This is particularly necessary for encoders with a large support, e.g. those of lensless cameras.

For lenses, the PSF in Equation (3) can be approximated by the Fraunhofer diffraction pattern (scaled Fourier transform) of the aperture function. For an arbitrary mask, the simplifications resulting from a lens are not possible and a more exact diffraction model is needed to predict the image pattern, e.g. Fresnel propagation or the angular spectrum method [12].

Incoherent, polychromatic illumination The convolutional relationship in Equation (3) is for coherent illumination, e.g. coming from a laser. However, illumination from natural scenes typically consists of diffuse or extended sources which are considered to be *incoherent*. In such cases, impulses at the image plane vary in a statistically independent fashion, thus requiring them to be added on an intensity basis [12]. In order words, for incoherent illumination, the convolution of Equation (3) should be expressed with respect to intensity:

$$I_2(x, y; \lambda) = \int_{\mathbb{R}} dr \int_{\mathbb{R}} ds\, p(x - r, y - s; \lambda) \left[\frac{1}{|M|^2} I_0\Big(\frac{r}{M}, \frac{s}{M}; \lambda\Big) \right], \tag{4}$$



(a)

(b)

Figure A.1: (a) Propagation setup. Not drawn to scale for visualization purposes. (b) Example physical measurement setup.

where I_0 and I_2 are the intensities at the scene and image planes respectively, with image intensity defined as the average instantaneous intensity:

$$I(x, y; \lambda) = \langle |U(x, y; \lambda, t)|^2 \rangle, \tag{5}$$

and the intensity PSF is proportional to the squared modulus of the coherent illumination PSF, i.e. of Equation (3):

$$p(x, y) \propto |h(x, y)|^2. \tag{6}$$

For polychromatic simulation, each wavelength has to be simulated independently. Converting this multispectral data to RGB is typically done in two steps: (1) mapping each wavelength to the XYZ coordinates defined by the International Commission on Illumination[6] and (2) converting to red-green-blue (RGB) values based on a reference white.[7]

A.2 Simulating propagation with an image of the desired scene

When simulating the propagation between two planes, as shown in Figure A.1a, one may wish that the scene in the I_0 plane corresponds to the content of a digital image. This section describes how to process an image such that its output corresponds to (1) a scene at a distance d_1 from the camera, (2) content from the original image having a height h_{obj}, and (3) a measurement taken by a camera of a known PSF.

An RGB image can be interpreted as image intensities at three wavelengths: red, green, and blue [33]. We use the following wavelengths for red, green, and blue respectively: $640\,\text{nm}$, $550\,\text{nm}$, and $460\,\text{nm}$. For a grayscale image, such as images from MNIST [19], the same data can be used across channels, or it can be convolved with a grayscale version of the PSF.

Concretely, given an image $\boldsymbol{x} \in \mathbb{R}^{H \times W \times C}$ with C channels and an intensity PSF $\boldsymbol{p} \in \mathbb{R}^{H_{\text{PSF}} \times W_{\text{PSF}} \times C}$, the simulation of a sensor measurement $\boldsymbol{v} \in \mathbb{R}^{DH_{\text{PSF}} \times DW_{\text{PSF}} \times C}$ (with an optional downsampling factor $D \geq 1$) can be summarized by the following steps:

1. Resize $\boldsymbol{x}$ to the PSF's dimension to obtain $\boldsymbol{x}_r \in \mathbb{R}^{H_{\text{PSF}} \times W_{\text{PSF}} \times C}$, while preserving $\boldsymbol{x}$'s original aspect ratio and scaling it to correspond to a desired object height (or width). The details of this rescaling are explained in Appendix A.3.
2. Convolve each channel of $\boldsymbol{x}_r$ with the corresponding PSF channel to obtain $\boldsymbol{y} \in \mathbb{R}^{H_{\text{PSF}} \times W_{\text{PSF}} \times C}$. Due to large convolution kernels, this is typically best to do in the spatial frequency domain, where convolution corresponds to an element-wise multiplication, and the FFT algorithm can be used to efficiently move between domains.
3. If $D \neq 1$, downsample the convolution output to the sensor resolution. We apply bilinear interpolation for this resizing.
4. Add noise at a desired signal-to-noise ratio (SNR). More on this in Appendix A.4.

Having a faithful estimate of the intensity PSF $\boldsymbol{p}$ is the most vital part of the above simulation. For a fixed optical encoder, if a physical setup is available, the best approach may be to simply measure the PSF by placing a point source (e.g. a white LED behind a pinhole as shown in Figure A.3c) at the desired distance and taking the resulting measurement as the intensity PSF. Some post-processing may be necessary to remove sensor noise and balance color channels. For encoders that have no parametric function, e.g. pseudo-random diffusers [1], measuring the PSF may be the only viable option.

For encoders that have a parametric function, e.g. with lenses and/or SLMs, it is possible to simulate the intensity PSF. This is in fact necessary for most end-to-end optimization techniques, unless forward-/back-propagation are done directly with hardware [38]. Even for a parametric encoder, it can be useful to measure the PSF (if a physical setup is available) in order to reduce mismatch due to model assumptions / simplifications [5]. In Appendix A.5, we describe our modeling of the PSF of an SLM at a particular wavelength, and explain how we account for the specifications of the ST7735R component and the Raspberry Pi High Quality Camera.

[6] `https://en.wikipedia.org/wiki/CIE_1931_color_space`

[7] `http://www.brucelindbloom.com/index.html?Eqn_RGB_XYZ_Matrix.html`

Using physical measurements The above simulation of Figure A.1a seeks to replicate the physical measurement setup shown in Figure A.1b, namely projecting the image of the desired scene on a display at a distance d_1 from the camera. While such a measurement would produce more realistic results, it can be very time-consuming for an *entire dataset* of images. If this dataset is to be used for a task with a fixed optical encoder, e.g. a lens or diffuser, it may be worth the time and effort as the measurement only has to be done once. For a task that seeks to optimize the optical encoder in an end-to-end fashion, new measurements would have to be performed during training whenever updates are made to the optical encoder. This is highly impracticable for optical encoders that require precise fabrication [33, 5, 20]. In the case of programmable optical encoders, alternating between physical measurements and updating the optical encoder lends itself to hardware-in-the-loop / physics-aware training [28, 37]. This has the potential to reduce model-mismatch but at the cost of longer training, due to acquisition time and a lack of parallelization. This technique is outside the scope of this work.

A.3 Rescaling image to PSF resolution for a desired object height

The goal of this step in the simulation of Appendix A.2 is to rescale a digital image $\boldsymbol{x} \in \mathbb{R}^{H \times W \times C}$ such that its convolution with a digital PSF $\boldsymbol{p} \in \mathbb{R}^{H_{\text{PSF}} \times W_{\text{PSF}} \times C}$ corresponds to the setup in Figure A.1a for an object of height h_{obj}. For such a configuration, namely a scene-to-encoder distance of d_1 and an encoder-to-image distance of d_2, the object height *at the sensor* is given by:

$$h_{\text{sensor}} = h_{\text{obj}}(d_2/d_1) = |M| h_{\text{obj}}. \tag{7}$$

For our simulation we are interested in the number of pixels that this height corresponds to. If our PSF was measured for the above distances with a sensor resolution of $(H_{\text{sensor}} \times W_{\text{sensor}})$ pixels and a pixel pitch of Δ,[8] the sensor will have captured a PSF for a scene of the following physical dimensions:

$$(h_{\text{scene}} \times w_{\text{scene}}) = \Big(\frac{\Delta H_{\text{sensor}}}{|M|} \times \frac{\Delta W_{\text{sensor}}}{|M|} \Big). \tag{8}$$

Consequently, the object height *in pixels* is approximately given by:

$$H_{\text{pixel}} = \text{round}\Big(\frac{h_{\text{obj}} H_{\text{PSF}}}{h_{\text{scene}}} \Big). \tag{9}$$

Therefore, to rescale the original input image $\boldsymbol{x} \in \mathbb{R}^{H \times W \times C}$ to the PSF resolution, while preserving its aspect ratio and scaling it such that it corresponds to the desired object height, we need to perform the following steps:

1. Resize $\boldsymbol{x}$ to $\big(\text{round}(SH) \times \text{round}(SW) \times C\big)$ where $S = (H_{\text{pixel}}/H)$.
2. Pad above to $(H_{\text{PSF}} \times W_{\text{PSF}} \times C)$.

The resulting image $\boldsymbol{x}_r \in \mathbb{R}^{H_{\text{PSF}} \times W_{\text{PSF}} \times C}$ can then be convolved with the PSF to simulate a propagation as in Figure A.1a.

A.4 Adding noise at a desired signal-to-noise ratio

Different types of noise can be added during simulation. In practice, read noise at a sensor follows a Poisson distribution with respect to the input. However, as this distribution is typically not differentiable with respect to the optical encoder parameters, Gaussian noise is used instead [33, 20].

In order to add generated noise to a signal and obtain a desired signal-to-noise ratio (SNR), the generated noise must be scaled appropriately. SNR (in dB) is defined as:

$$\text{SNR} = 10 \log_{10}(\sigma_S^2/\sigma_N^2), \tag{10}$$

where σ_S^2 is the clean image variance and σ_N^2 is the generated noise variance. For a target SNR T, the generated noise can be scaled with the following factor:

$$k = \sqrt{\frac{\sigma_S^2}{\sigma_N^2 10^{(T/10)}}}. \tag{11}$$

[8] The sensor may have a different resolution than the PSF used in simulation as we may wish to downsample the measured PSF for a lighter computational load, especially if we are simulating a large dataset during training. Note that this simplification is possible for PSF's with a broad support such as those of diffusers and SLMs, but doing so with a PSF with a very small support, e.g. a lens, can significantly hurt the simulation quality.

In our simulation, we generate noise following a Poisson distribution, as we do not backpropagate through noise generation to the optical encoder parameters.

A.5 Point spread function modeling for a spatial light modulator

Our modeling of the PSF for propagation through a spatial light modulator (SLM) is similar to that of [33], namely for each wavelength λ, we simulate the propagation in Figure A.1a:

1. *From the scene to the optical element*: propagation is modeled by spherical wavefronts. Assuming a point source at the scene plane U_0, we have the following wave field at the aperture plane:
$$U_1^-(x, y; z = d_1, \lambda) = \exp\Big(j\frac{2\pi}{\lambda}\sqrt{x^2 + y^2 + d_1^2}\Big), \tag{12}$$
where d_1 is the distance between the scene and the camera aperture.
2. *At the optical element*: the wave field is multiplied with a potentially complex-valued mask pattern $M(x, y)$ corresponding to the SLM:
$$U_1^+(x, y; z = d_1, \lambda) = U_1^-(x, y; z = d_1, \lambda)M(x, y). \tag{13}$$
Note that an infinitesimally small distance is assumed between the opening of the aperture U_1^- and the exit of the SLM U_1^+.
3. *To the sensor*: free-space propagation according to scalar diffraction theory [12] as light is diffracted by the optical element. We employ the bandlimited angular spectrum method (BLAS) which produces accurate simulations for both near- and far-field [21]. This yields the following wave field PSF at the sensor plane:
$$U_2(x, y; z = d_1 + d_2, \lambda) = \mathcal{F}^{-1}\Big(\mathcal{F}\big(U_1^+(x, y; z = d_1, \lambda)\big)H(u, v; z = d_2, \lambda)\Big), \tag{14}$$
where $\mathcal{F}$ and $\mathcal{F}^{-1}$ denote the spatial Fourier transform and its inverse, u, v are spatial frequencies for x, y, and the free-space frequency response $H(u, v; z = d_2, \lambda)$ according BLAS is given by:
$$H(u, v; z = d_2, \lambda) = e^{j\frac{2\pi}{\lambda}d_2\sqrt{1-(\lambda u)^2-(\lambda v)^2}}\text{rect}\Big(\frac{u}{2u_{\text{limit}}}\Big)\text{rect}\Big(\frac{v}{2v_{\text{limit}}}\Big), \tag{15}$$
where the bandlimiting frequencies are given by
$$u_{\text{limit}} = \frac{\sqrt{(d_2/S_x)^2 + 1}}{\lambda}, \quad v_{\text{limit}} = \frac{\sqrt{(d_2/S_y)^2 + 1}}{\lambda}, \tag{16}$$
and $(S_x \times S_y)$ are the physical dimensions of the propagation region, in our case the physical dimensions of the sensor.
4. As we are simulating incoherent light, we require the squared modulus of the wave field PSF:
$$P(x, y; z = d_1 + d_2, \lambda) = \big|U_2(x, y; z = d_1 + d_2, \lambda)\big|^2. \tag{17}$$
Note that we assume the intensity PSF to be equal to the squared modulus of the wave field PSF, as is done in [33]. In theory, they are simply proportional and multiple realizations would be needed to estimate this statistical quantity [12].

A.5.1 Modeling the spatial light modulator

A key component in the above PSF simulation is modeling the complex-valued mask $M(x, y)$ associated with the SLM. Two assumptions are commonly made in its modeling:

- The mask is assumed to be either a phase transformation, i.e. $|M(x, y)| = 1$, or an amplitude transformation, i.e. $M(x, y) \in \mathbb{R}$.
- The mask is discretized according to the SLM resolution. This approximation neglects *deadspace* (or equivalently the fill factor) of individual pixels.

We model the SLM as a superposition of apertures for each adjustable pixel:

$$M(x,y) = \sum_{k}^{K} w_k A(x - x_k, y - y_k), \tag{18}$$

where the complex-valued weights $\{w_k\}_{k=1}^{K}$ satisfy $|w_k| \leq 1$, the coordinates $\{(x_k, y_k)\}_{k=1}^{K}$ are the centers of the SLM pixels, and the aperture function $A(\cdot)$ is assumed to be identical for each SLM pixel. This model takes into account deadspace but assumes that no stray light passes between the pixels.

While numerical discretization may not be able to perfectly sample $M(x,y)$ to account for arbitrary shifts of $A(\cdot)$ in Equation (18), these shifts can be accounted for in the spatial frequency domain (u,v):

$$\mathcal{F}(M(x,y)) = M(u,v) = A(u,v) \sum_{k}^{K} w_k e^{jux_k} e^{jvy_k}. \tag{19}$$

Equation (13) for the wave field at the exit of the SLM then becomes

$$U_1^+(x,y;z=d_1,\lambda) = U_1^-(x,y;z=d_1,\lambda)\mathcal{F}^{-1}(M(u,v)). \tag{20}$$

While this allows for arbitrary shifts, it requires an additional FFT and can be expensive when tracking gradients in order to optimize the SLM weights w_k.

A cheaper way to account for deadspace is to discretize $M(x,y)$ at a finer resolution than that of the SLM, and only modulate those pixels which fall within the individual SLM pixel apertures (by setting the appropriate w_k value). While optimizing the SLM weights in our end-to-end approach (for the experiments in Section 4), we adopt this latter simplification which is much more tractable when tracking gradients.

Other parameters that have been modeled for SLMs in holography and that are applicable to the context of imaging include: non-linear mapping between voltage-to-phase/amplitude of the individual SLM pixels, Zernike coefficients to model deviations from theoretical propagation models, and content-dependent undiffracted (stray) light [28].

A.5.2 Specifics for the ST7735R component and the Raspberry Pi High Quality Camera

As the ST7735R component is originally intended to serve as a color display, it has an interleaved pattern of red, green, and blue filters as shown in Figure A.2a,[9] which can be modeled as a wavelength-dependent version of Equation (18)

$$M(x,y;\lambda) = \sum_{c\in\{R,G,B\}} F_c(\lambda) \sum_{k_c}^{K_c} w_{k_c} A(x - x_{k_c}, y - y_{k_c}), \tag{21}$$

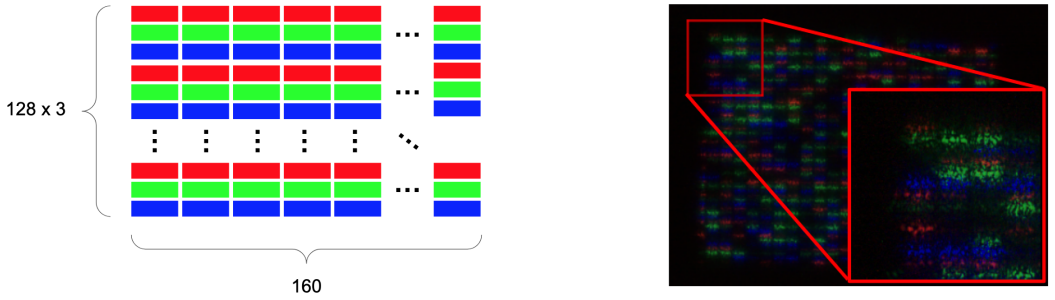
where:

- $\{F_c(\cdot)\}_{c\in\{R,G,B\}}$ is the wavelength-response of each color filter,
- $\left\{\{w_{k_c}\}_{k_c=1}^{K_c}, c \in \{R,G,B\}\right\}$ are *real*-valued weights for the red, green, and blue sub-pixels, and $\left\{\{(x_{k_c}, y_{k_c})\}_{k_c=1}^{K_c}, c \in \{R,G,B\}\right.$ are their respective centers.

The display has a resolution of 128×160 color pixels, with three sub-pixels per color pixel as shown in Figure A.2a. The dimension of each sub-pixel is $(0.06\,\mathrm{mm} \times 0.18\,\mathrm{mm})$ and the dimension of the entire screen is $(28.03\,\mathrm{mm} \times 35.04\,\mathrm{mm})$.[10] If we assume a uniform spacing of sub-pixels, this corresponds to a pixel pitch of roughly $(0.073\,\mathrm{mm} \times 0.22\,\mathrm{mm})$ and a fill-factor of 82%, namely deadspace of 18% around each sub-pixel.

[9] More information can be found on the device driver datasheet: `https://cdn-shop.adafruit.com/datasheets/ST7735R_V0.2.pdf`

[10] ST7735R breakout board datasheet: `https://cdn-shop.adafruit.com/datasheets/JD-T1800.pdf`



(a) (b)

Figure A.2: Visualizing pixel layout of the ST7735R component. (a) Red, green, blue color filter arrangement. (b) Zooming into section of a measured point spread function for a random pattern.

The Raspberry Pi High Quality Camera[11] uses the Sony IMX477R back-illuminated sensor which has the following specifications: 3040×4056 pixel resolution, 7.9mm sensor diagonal, and a pixel size of $(1.55\mu\text{m} \times 1.55\mu\text{m})$, which corresponds to full sensor dimensions of $(4.71\,\text{mm} \times 6.29\,\text{mm})$. As the display of the ST7735R component is larger than the sensor, we use a subset of its pixels that covers the sensor area. From the pixel pitch of the ST7735R component determined above $(0.073\,\text{mm} \times 0.22\,\text{mm})$, it can be concluded that the number of SLM sub-pixels that overlap the sensor is around 64×22. Moreover, for enforcing the LSI assumption described in Appendix A.1, we crop the SLM such that 80% of sensor surface is exposed, which corresponds to $51 \times 22 = 1122$ SLM sub-pixels. This is the number of SLM sub-pixels that we optimize in Section 4

A.6 Detailed description about baseline and proposed point spread functions

For the experiments in Section 4, our baseline and proposed imaging systems use the Raspberry Pi High Quality Camera, either for the PSF measurement or in simulating the PSF. Measured PSFs are obtained by placing a white LED behind a pinhole aperture, as shown in Figure A.3c, at the target distance $(40\,\text{cm})$, and measuring the response in an environment with no external light.

Simulated PSFs are obtained by using the approach described in Appendix A.5. Unless noted otherwise, the scene, encoder, and images planes (U_0, U_1^+, U_2 respectively in Figure A.1a) for simulating the PSFs take on the size and resolution of the Raspberry Pi High Quality Camera: 3040×4056 pixel resolution, and a pixel size of $(1.55\mu\text{m} \times 1.55\mu\text{m})$.

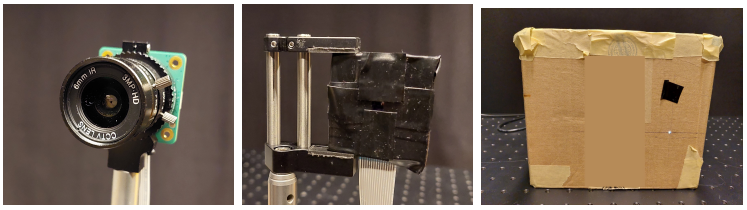
Below are technical details regarding each PSF:

- *Lens*: measured PSF for the camera shown in Figure A.3a, which has a 6mm wide angle lens[12] focused at $40\,\text{cm}$. The lens and its objective have a thickness of $34\,\text{mm}$, and the lens is $7.53\,\text{mm}$ from the sensor.
- *CA* (coded aperture): a binary mask is generated by (1) generating a length-63 maximum length sequence (MLS) binary array,[13] (2) repeating the sequence to create a 126-length sequence, and (3) computing the outer product with itself to create a 126×126 matrix. These are the same steps for generating a coded aperture mask, as in [2], except that we use a shorter MLS sequence (63 instead of 255) to obtain a feature size of $30\,\mu\text{m}$ (as in [2]). The mask covers 80% of the sensor height (as *Fixed SLM (s)* and *Learned SLM* below). For the PSF, we simulate the mask's diffraction pattern for a distance of $d_2 = 0.5\,\text{mm}$, matching the distance in [2].
- *Diffuser*: measured PSF for the camera shown in Figure A.3b, where the diffuser is placed roughly $4\,\text{mm}$ from the sensor. The diffuser is double-sided tape as in the DiffuserCam

[11] Raspberry Pi High Quality Camera datasheet: `https://cdn-shop.adafruit.com/product-files/4561/4561+Raspberry+Pi+HQ+Camera+Product+Brief.pdf`

[12] 6mm Wide Angle Lens for Raspberry Pi HQ Camera datasheet: `https://cdn-shop.adafruit.com/product-files/4563/4563-datasheet.pdf`

[13] Using the SciPy function `max_len_seq`: `https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.max_len_seq.html`



(a) (b) (c)

Figure A.3: Baseline cameras: (a) lensed and (b) diffuser-based. (c) Point source for measuring PSF.

tutorial [4]. In [3], the authors demonstrate the effectiveness of this simple diffuser when used with the Raspberry Pi High Quality Camera. It is less than $1\,\mathrm{mm}$ thick and is placed roughly $4\,\mathrm{mm}$ from the sensor.

- *Fixed SLM (m)*: measured PSF for the proposed camera shown in Figure 3 for a random pattern. With a stepper motor, the mask-to-sensor distance is programmatically set to $4\,\mathrm{mm}$ to match the distance of the diffuser-based camera.
- *Fixed SLM (s)*: simulated PSF for the proposed camera, using the approach described in Appendix A.5 for a random set of SLM amplitude values and a mask-to-sensor distance of $4\,\mathrm{mm}$. The aperture is set such that SLM pixels covering 80% of the sensor surface area are exposed. This corresponds to $51 \times 22 = 1122$ SLM sub-pixels as described in Appendix A.5.2.
- *Learned SLM*: simulated PSF for the proposed camera that is obtained by optimizing Equation (2) for the SLM weights, and simulating the corresponding PSF with the approach described in Appendix A.5 for a mask-to-sensor distance of $4\,\mathrm{mm}$. Like *Fixed SLM (s)*, SLM pixels that cover 80% of the sensor surface area are used, corresponding to $51 \times 22 = 1122$ SLM pixels. As the SLM values are updated after backpropagation during training, the resulting PSF is different for each batch. Moreover, when simulating the PSF with the approach described in Appendix A.5, the downsampling factor is set to $D = 8$ (resolution of 380×507), as computing a full-scale PSF at each batch leads to much longer training times. With the compute hardware described in Appendix A.7, it takes around $3\,\mathrm{min}$ to simulate the entire MNIST dataset ($70'000$ examples) for a downsampling factor of $D = 8$, whereas it takes around $250\,\mathrm{min}$ to simulate the same dataset at full resolution (3040×4056).

As the PSF for *Learned SLM* is of a lower dimension than the rest of the PSFs (downsampled by a factor of 8), when simulating each example in the dataset with the approach described in Appendix A.2, we first downsample the other PSFs (except *Lens*) by a factor of 8, such that the intensity PSF also has a resolution of 380×507. Note that this cannot be done for the *Lens* PSF due to its very compact support; so we retain an intensity PSF $\boldsymbol{p} \in \mathbb{R}^{3040 \times 4056 \times 3}$ when simulating *Lens*' examples.

A.7 Training details, hyperparameters, hardware, and network architectures

Experiments and classifiers in Section 4 were run on a Dell Precision 5820 Tower X-Series (08B1) machine with an Intel i9-10900X $3.70\,\mathrm{GHz}$ CPU and two NVIDIA RTX A5000 GPUs. PyTorch [27] was used for dataset preparation and training.

As the task consists of multi-label classification (10 digits from 0 to 9), we use a cross entropy loss in optimizing Equation (2):

$$\mathcal{L}(y, \hat{\boldsymbol{p}}) = -\log \frac{\exp(\hat{\boldsymbol{p}}_{c=y})}{\sum_{c=0}^{9} \exp(\hat{\boldsymbol{p}}_c)}, \tag{22}$$

where $y \in [0, 10)$ are the ground truth labels, and $\hat{\boldsymbol{p}} = D_{\boldsymbol{\theta}_D}\left(O_{\boldsymbol{\theta}_E}(\boldsymbol{x})\right) \in \mathbb{R}^{10}$ are the predicted scores for a given input $\boldsymbol{x}$ that passes through the optical encoder $O_{\boldsymbol{\theta}_E}(\cdot)$ and the digital decoder $D_{\boldsymbol{\theta}_D}(\cdot)$.

All classifiers are trained for 50 epochs and with a batch size of $N = 200$. A large batch size is used to accelerate the training of *Learned SLM*, i.e. to minimize the number of PSF updates per epoch and to parallelize FFT convolutions with the PSF. As the other approaches have a fixed optical encoder (and therefore fixed PSF), the FFT convolutions only need to be done once prior to training. When

the two GPUs are used, it takes approximately $15\,\mathrm{min}$ for the fixed-encoder classifiers to train and $7.5\,\mathrm{h}$ for the end-to-end optical encoder digital classifier architecture to train. This significantly larger training time for the end-to-end approach is because PSF simulation and dataset augmentation has to be done during training, while this can be pre-computed for the fixed encoders.

In the following sub-sections, we describe the two classifier architectures used in Section 4, namely the $D_{\boldsymbol{\theta}_D}(\cdot)$ in Equation (2). For the fixed optical encoders, the embeddings $\{\boldsymbol{v}_i\}_{i=1}^N$ that are inputted to the classifiers are pre-computed with the approach described in Appendix A.2. The resulting augmented dataset is normalized (according to the augmented training set statistics). For *Learned SLM*, we apply batch normalization [16] and a ReLu activation to the sensor embedding prior to passing it to the classifier. At inference, the parameters of batch normalization are fixed.

A.7.1 Logistic regression for Section 4.1

The classifier performs the following steps:

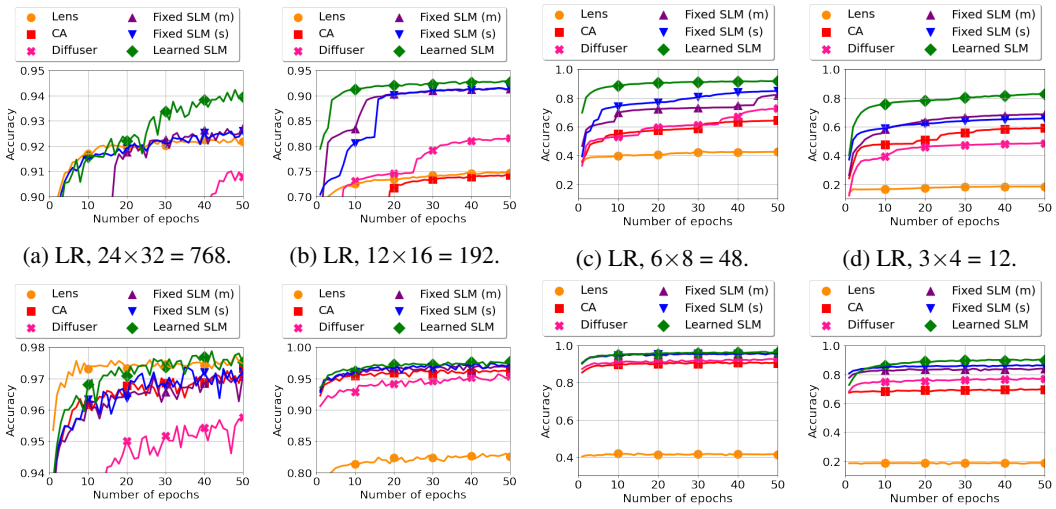
1. Flatten sensor embedding.
2. Fully connected linear layer to 10 classes.
3. Softmax decision layer.

A.7.2 Two-layer fully connected neural network for Sections 4.1 and 4.2

The classifier performs the following steps:

1. Flatten sensor embedding.
2. Fully connected linear layer to hidden layer of 800 units, as in [32]
3. Batch normalization.
4. ReLu activation.
5. Fully connected linear layer to 10 classes.
6. Softmax decision layer.

A.8 Test accuracy curves for experiment on varying embedding dimension - Section 4.1



(a) LR, 24×32 = 768. (b) LR, 12×16 = 192. (c) LR, 6×8 = 48. (d) LR, 3×4 = 12.

(e) NN, 24×32 = 768. (f) NN, 12×16 = 192. (g) NN, 6×8 = 48. (h) NN, 3×4 = 12.

Figure A.4: MNIST test accuracy curves while varying the sensor embedding dimension. Top row is for a logistic regression (LR) classifier, bottom row for a fully connected neural network (NN) with a single hidden layer of 800 units. The input dimension is indicated in the sub-figure caption.

A.9 Example reconstructions of low-dimensional embeddings

Although lensless measurements exhibit visual privacy in their raw measurements, with sufficient knowledge about the camera (e.g. a point spread function) and an appropriate computational algorithm, one is able to recover an estimate of the underlying object. In this section, we apply a common approach for recovering an image estimate from the raw measurements, namely solving the following inverse problem [1]:

$$\hat{\boldsymbol{x}} = \underset{\boldsymbol{x} \geq 0}{\operatorname{argmin}} \frac{1}{2} \|\boldsymbol{y} - \boldsymbol{H}\boldsymbol{x}\|_2^2 + \tau \|\Psi \boldsymbol{x}\|_1, \tag{23}$$

where $\hat{\boldsymbol{x}}$ is the image estimate, $\boldsymbol{y}$ is the raw measurement, $\boldsymbol{H}$ models the cropped convolution (aperture followed by optical encoder), $\boldsymbol{x}$ is the underlying image, and $\boldsymbol{\Psi}$ maps $\boldsymbol{x}$ into a domain in which it is sparse. As MNIST data is sparse in pixels, $\boldsymbol{\Psi}$ could be the identity matrix. However, our experiments found that the finite difference operator obtains better results.

Example reconstructions solving Equation (23) with 10 iterations of ADMM [7] for the setup in Section 4.1 (object height of $12\,\mathrm{cm}$, $40\,\mathrm{cm}$ from the camera) can be seen in Figure A.5 for an embedding dimension of 760×1014 (downsampling Raspberry Pi HQ Camera resolution by 4). This represents a typical resolution for lensless imaging, where such imaging systems are able to recover an accurate estimate of the underlying image. Note that the height of the digit in *Lens* is larger than the rest due to a larger mask-to-sensor. Conversely, *Coded aperture* has a very small mask-to-sensor distance and therefore a very small object height at the sensor.

For the reconstructions in Figure A.6 (except *Lens* as the image is directly form on the sensor), we use a much lower dimensional sensor resolution, namely $24 \times 32 = 768$ as in the experiments of Section 4.1 (downsampling Raspberry Pi HQ Camera resolution by around 126). Equation (23) is solved with 100 iterations of ADMM [7]. *Coded aperture* produces poor results as its mask-to-sensor distance is just $0.5\,\mathrm{mm}$ as in the proposed design of [2]. As a result, the object height at the sensor is $12\,\mathrm{cm} \times |M| = 12\,\mathrm{cm} \times (0.5\,\mathrm{mm}/40\,\mathrm{cm}) = 0.15\,\mathrm{mm}$, which is about 100 pixels on the Raspberry Pi HQ sensor. As the sensor resolution is downsampled by around 126, the corresponding reconstruction (and underlying image) is contained within a single pixel. For the remaining lensless approaches, the reconstructed digit is of much poorer quality than downsampling by a factor of 4 (Figure A.5). Nonetheless, some features of digits can be distinguished, e.g. 7, 1, 0, and 4 for *Fixed SLM (m)*.

Figure A.7 shows example reconstructions for an embedding dimension of $6 \times 8 = 48$. For this sensor resolution, the measurements exhibit higher visual privacy as it is not possible to discern distinguishable features of digits from the images recovered by ADMM (nor from *Lens*). Despite this inability to recover distinguishable features, all lensless approaches achieve above 90% classification on the *raw measurement*, see Table 1 for the two-layer neural network classifier.

(a) Lens. (b) Coded aperture. (c) Diffuser. (d) Fixed SLM (m). (e) Fixed SLM (s).

Figure A.5: Example reconstruction for an embedding dimension of 760×1014.

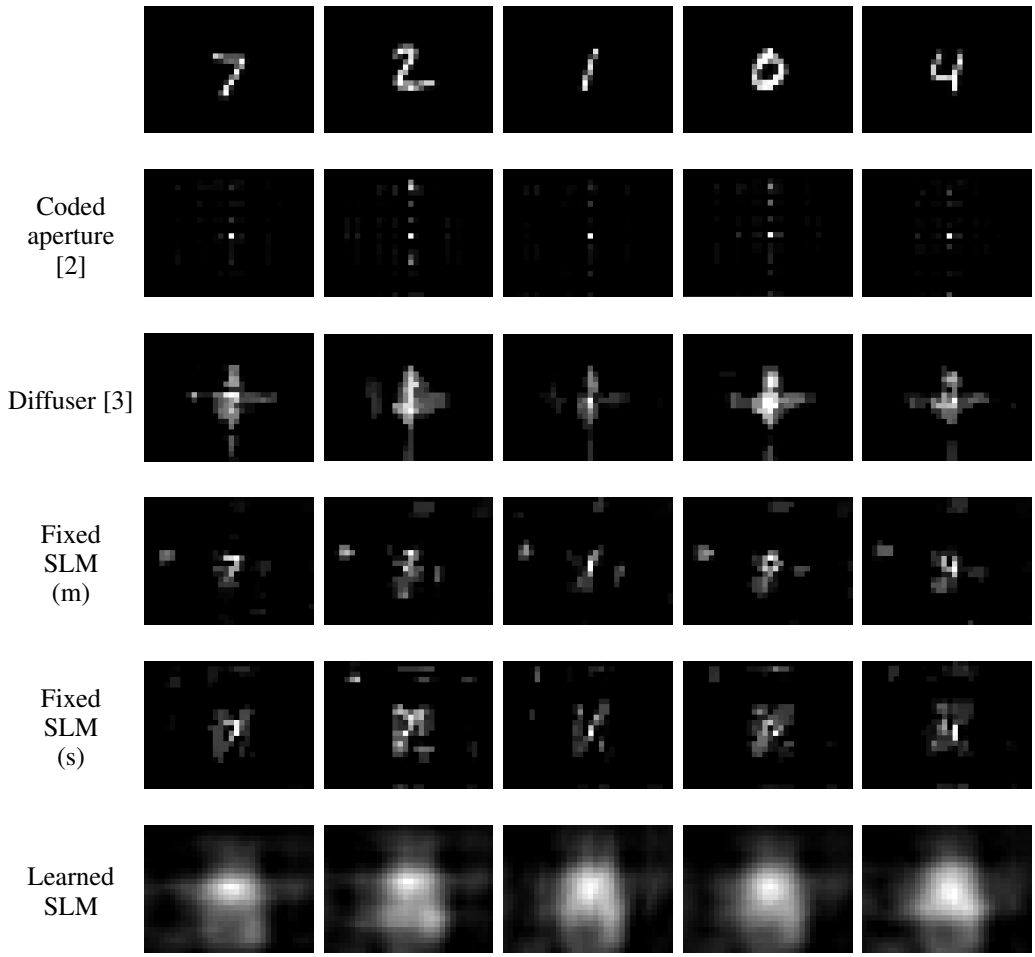


Figure A.6: Example reconstructions (except for *Lens*) for an embedding dimension of $24 \times 32 = 768$, which corresponds to a downsampling of around 126 along each dimension. Reconstructions are obtained using ADMM to solve Equation (23) for 100 iterations.

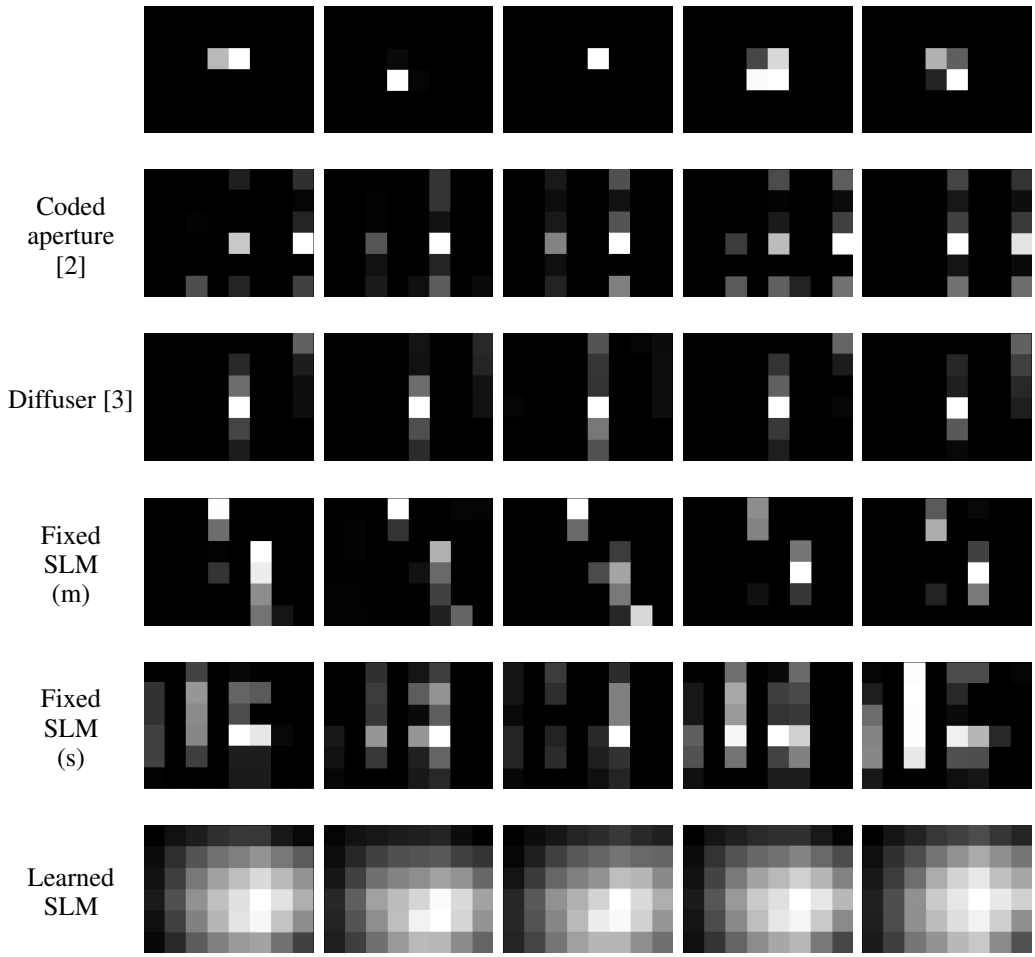
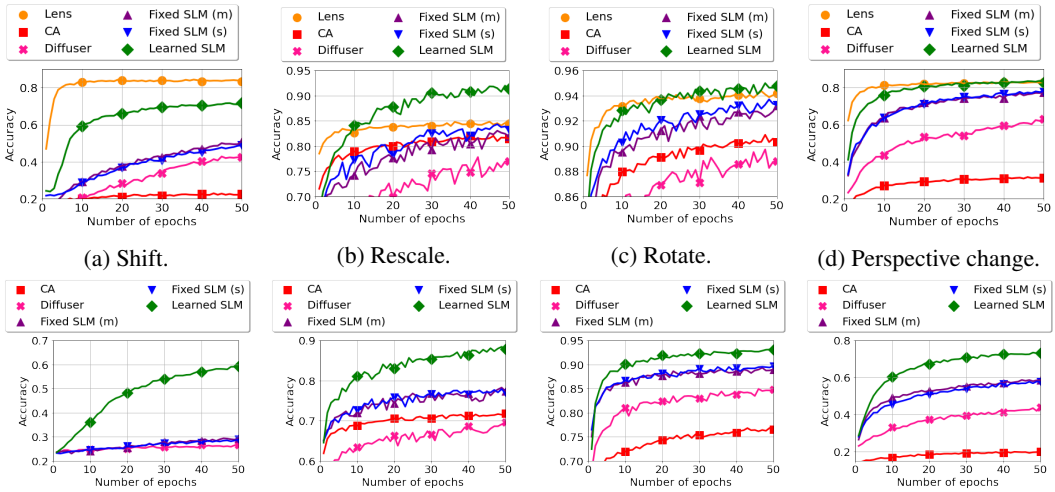


Figure A.7: Example reconstructions (except for *Lens*) for an embedding dimension of $6 \times 8 = 48$, which corresponds to a downsampling of around 507 along each dimension. Reconstructions are obtained using ADMM to solve Equation (23) for 100 iterations.

A.10 Test accuracy curves for experiments on robustness to image transformations- Section 4.2



(a) Shift. (b) Rescale. (c) Rotate. (d) Perspective change.

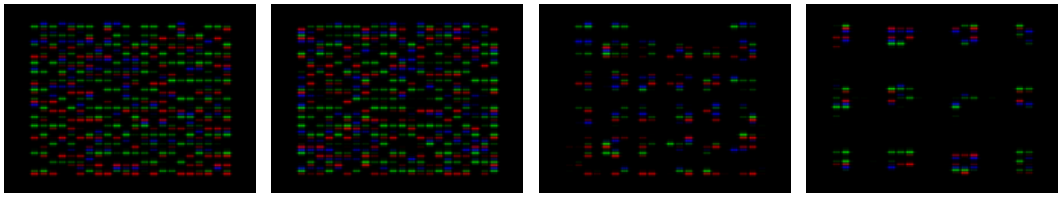
(e) Shift. (f) Rescale. (g) Rotate. (h) Perspective change.

Figure A.8: MNIST test accuracy curves for various image transformations. Top row is for an embedding dimension of $24 \times 32 = 768$, bottom row for an embedding dimension of $6 \times 8 = 48$. The classifier architecture is as described in Appendix A.7.2. The image transformation is indicated in the sub-figure caption.

A.11 PSFs of learned SLM masks

The PSFs corresponding to the SLM masks determined from the end-to-end optimizations of Section 4.1 and Section 4.2 can be found in this section.

A.11.1 Varying embedding dimension experiment - Section 4.1

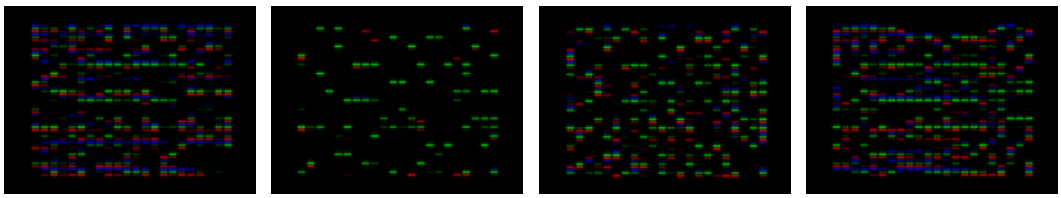


(a) LR, 24×32 = 768. (b) LR, 12×16 = 192. (c) LR, 6×8 = 48. (d) LR, 3×4 = 12.

(e) NN, 24×32 = 768. (f) NN, 12×16 = 192. (g) NN, 6×8 = 48. (h) NN, 3×4 = 12.

Figure A.9: PSFs of learned masks. Top row is for a logistic regression (LR) classifier, bottom row for a fully connected neural network (NN) with a single hidden layer of 800 units. The input dimension is indicated in the sub-figure caption.

A.11.2 Robustness to image transformations experiment - Section 4.2



(a) Shift, 24×32. (b) Rescale, 24×32. (c) Rotate, 24×32. (d) Perspective, 24×32.

(e) Shift, 6×8. (f) Rescale, 6×8. (g) Rotate, 6×8. (h) Perspective, 6×8.

Figure A.10: PSFs of learned masks for Section 4.2 experiment. Top row is for an embedding dimension of 24×32 = 768, while the bottom row if for an embedding dimension of 6×8 = 48. The type of transformation is indicated in the sub-figure caption.

A.12 Visualizing image transformation effects

In this section, we visualize the various image transformations effects that are applied in the experiments of Section 4.2, namely the raw measurements and its corresponding reconstruction with Equation (23) for the lensless approaches. By comparing to Figure A.6, we can see how the image transformations affect the capability of recovering the underlying image. Note that the raw measurements (and not the recovered images) are passed to classifier.

In what follows, we focus on the embeddings of *Fixed SLM (m)* and *Learned SLM*. For all transformations, we observe that it is difficult to recover discernible features from the recovered images of *Learned SLM*'s measurements. Nonetheless, its raw measurements produce better classification results than the other (fixed) lensless encoders (see Table 3).

A.12.1 Shift

Figures A.11 and A.12 show the raw embeddings and reconstructed outputs for *Fixed SLM (m)* and *Learned SLM* under random shifts. We can see how shifting the object at the scene plane results in a shift in the raw measurement. As a result, the sensor loses multiplexed information with respect to objects that are centered, see Figure 4. This loss of multiplexed information may explain why there is a sudden drop in classification performance for *Shift* in Table 4 for lensless approaches.

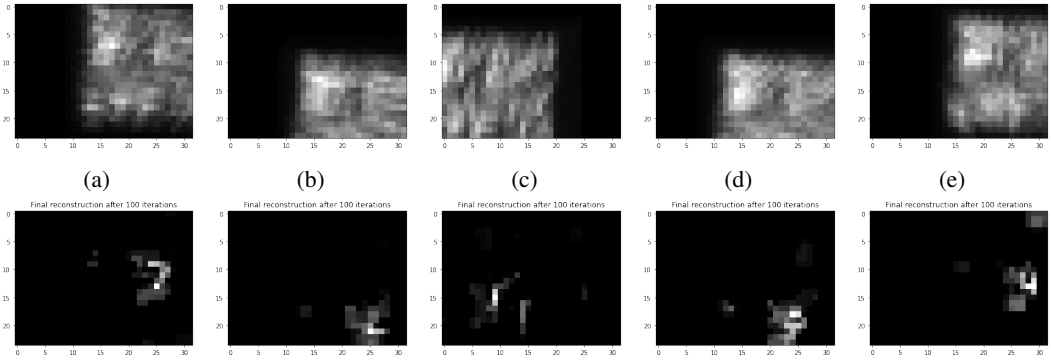


Figure A.11: Example reconstructions (100 iterations of ADMM to solve Equation (23)) for *Fixed SLM (m)* in the presence of random shifts, for an embedding dimension of 24×32 (downsampling of around 126 along each dimension). (Top) raw measurements and (bottom) corresponding reconstruction.

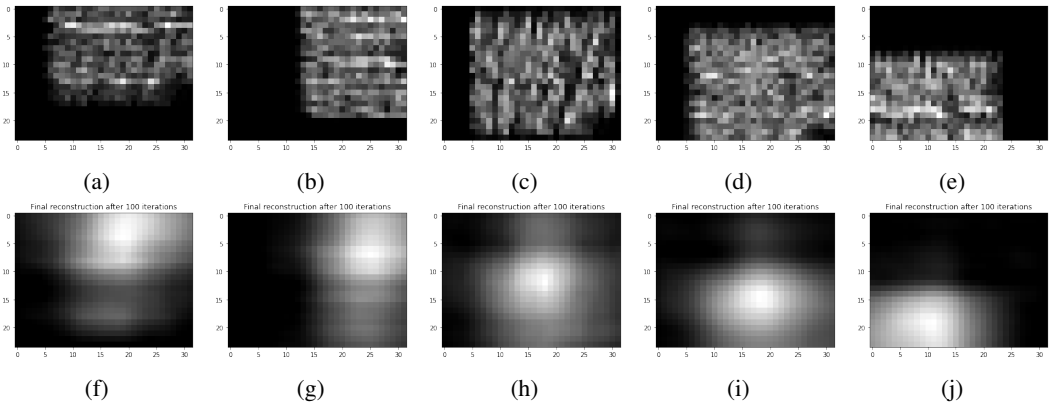


Figure A.12: Example reconstructions (100 iterations of ADMM to solve Equation (23)) for *Learned SLM* in the presence of random shifts, for an embedding dimension of 24×32 (downsampling of around 126 along each dimension). (Top) raw measurements and (bottom) corresponding reconstruction.

Figure A.13 shows that for *Fixed SLM (m)* with a larger sensor resolution, and hence more multiplexed information, a faithful image of the scene can be recovered under random shifts.

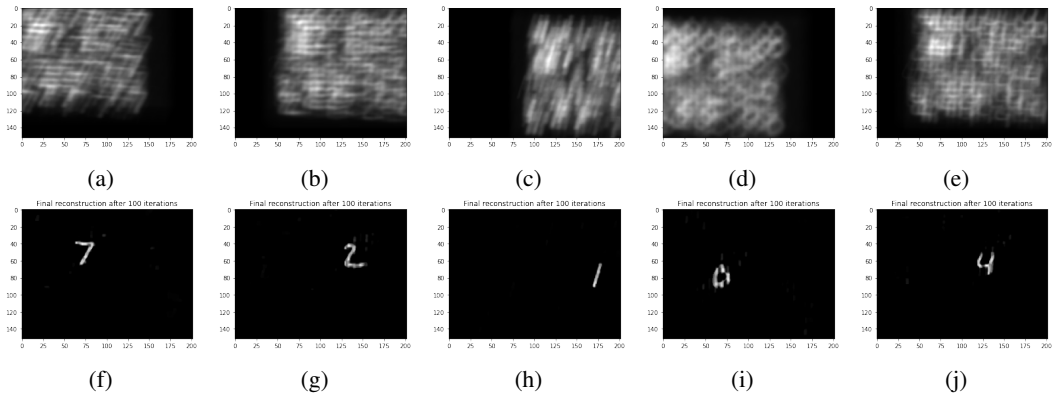


Figure A.13: Example reconstructions (100 iterations of ADMM to solve Equation (23)) for *Fixed SLM (m)* in the presence of random shifts, for an embedding dimension of 152×202 (downsampling of around 20 along each dimension). (Top) raw measurements and (bottom) corresponding reconstruction.

A.12.2 Rescale

Figures A.14 and A.15 show the effect of rescaling on raw embeddings and reconstructed outputs of *Fixed SLM (m)* and *Learned SLM*.

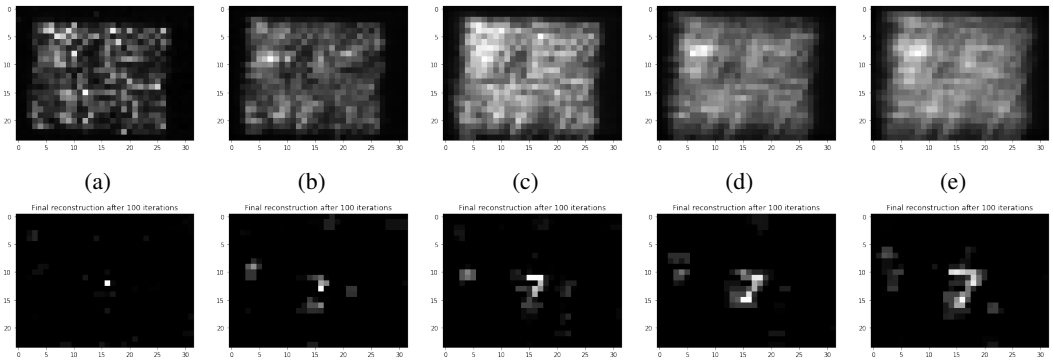


Figure A.14: Example reconstructions (100 iterations of ADMM to solve Equation (23)) for *Fixed SLM (m)*, for an embedding dimension of 24×32. The object height increases from left to right: $2\,\mathrm{cm}$, $6.5\,\mathrm{cm}$, $11\,\mathrm{cm}$, $15.5\,\mathrm{cm}$, $20\,\mathrm{cm}$. (Top) raw measurements and (bottom) corresponding reconstruction.

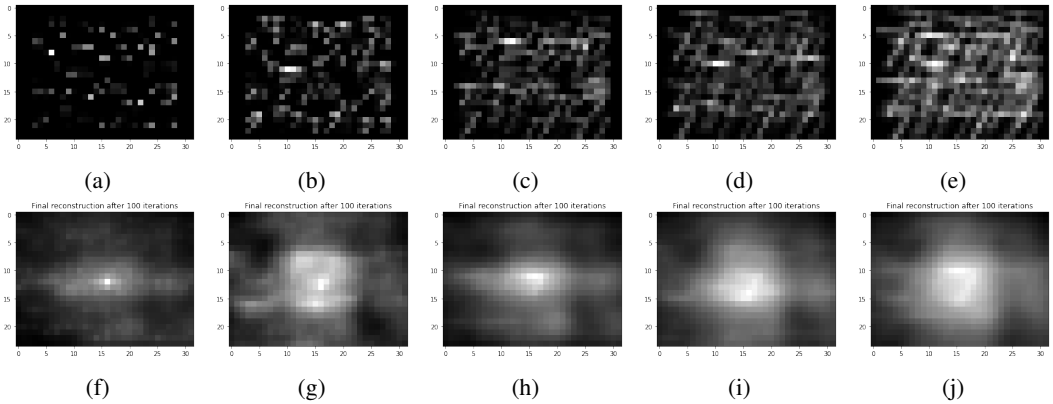


Figure A.15: Example reconstructions (100 iterations of ADMM to solve Equation (23)) for *Learned SLM*, for an embedding dimension of 24×32. The object height increases from left to right: $2\,\mathrm{cm}$, $6.5\,\mathrm{cm}$, $11\,\mathrm{cm}$, $15.5\,\mathrm{cm}$, $20\,\mathrm{cm}$. (Top) raw measurements and (bottom) corresponding reconstruction.

A.12.3 Rotate

Figures A.16 and A.17 show the effect of rotation on raw embeddings and reconstructed outputs of *Fixed SLM (m)* and *Learned SLM*.

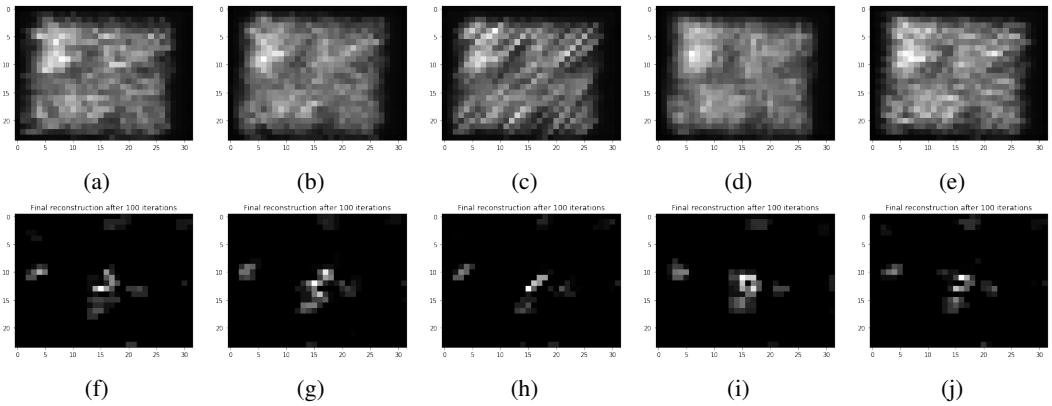


Figure A.16: Example reconstructions (100 iterations of ADMM to solve Equation (23)) for *Fixed SLM (m)* in the presence of random rotations, for an embedding dimension of 24×32. (Top) raw measurements and (bottom) corresponding reconstruction.

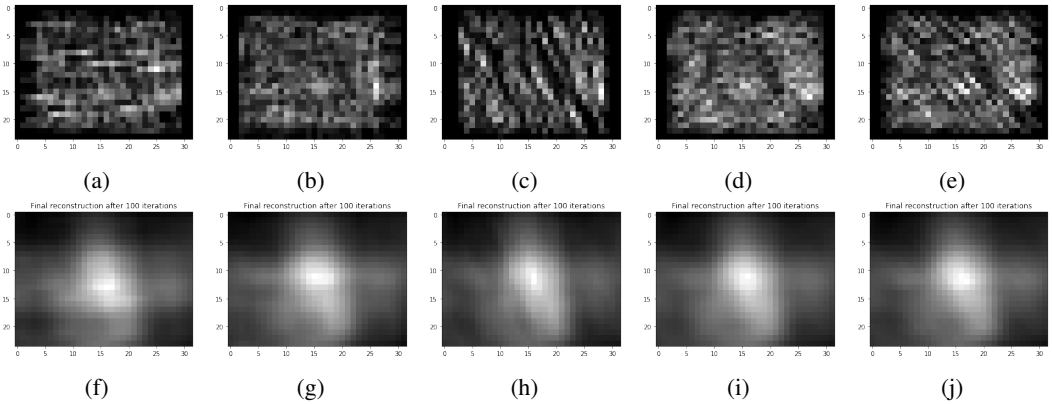
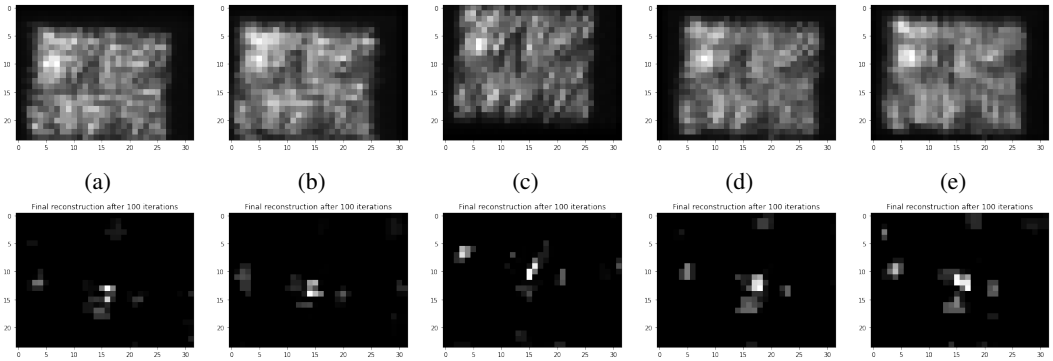


Figure A.17: Example reconstructions (100 iterations of ADMM to solve Equation (23)) for *Learned SLM* in the presence of random rotations, for an embedding dimension of 24×32. (Top) raw measurements and (bottom) corresponding reconstruction.

A.12.4 Perspective

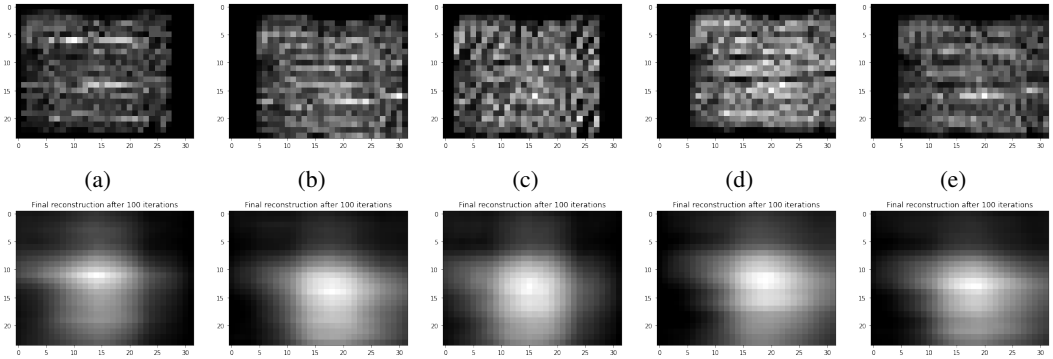
Figures A.18 and A.19 show the effect of perspective changes on raw embeddings and reconstructed outputs of *Fixed SLM (m)* and *Learned SLM*.



(a) (b) (c) (d) (e)

(f) (g) (h) (i) (j)

Figure A.18: Example reconstructions (100 iterations of ADMM to solve Equation (23)) for *Fixed SLM (m)* in the presence of random perspective changes, for an embedding dimension of 24×32. (Top) raw measurements and (bottom) corresponding reconstruction.



(a) (b) (c) (d) (e)

(f) (g) (h) (i) (j)

Figure A.19: Example reconstructions (100 iterations of ADMM to solve Equation (23)) for *Learned SLM* in the presence of random perspective changes, for an embedding dimension of 24×32. (Top) raw measurements and (bottom) corresponding reconstruction.