Learning the Truncation Index of the Kronecker Product SVD for Image Restoration

Salina Bermudez[†]

Project advisor: Rosemary Renaut[‡]

Abstract. The image processing task of the recovery of an image from a noisy or compromised image is an ill-posed inverse problem. To solve this problem, it is necessary to incorporate prior information about the smoothness, or the structure, of the solution, by incorporating regularization. Here, we consider linear blur operators with an efficiently-found singular value decomposition. Then, regularization is obtained by employing a truncated singular value expansion for image recovery. In this study, we focus on images for which the image blur operator is separable and can be represented by a Kronecker product such that the associated singular value decomposition is expressible in terms of the singular value decompositions of the separable components. The truncation index k can then be identified without forming the full Kronecker product of the two terms. This report investigates the problem of learning an optimal k using two methods. For one method to learn k we assume the knowledge of the true images, yielding a supervised learning algorithm based on the average relative error. The second method uses the method of generalized cross validation and does not require knowledge of the true images. The approach is implemented and demonstrated to be successful for `Gaussian`, `Poisson` and `salt and pepper` noise types across noise levels with signal to noise ratios as low as 10. This research contributes to the field by offering insights into the use of the supervised and unsupervised estimators for the truncation index, and demonstrates that the unsupervised algorithm is not only robust and computationally efficient, but is also comparable to the supervised method.

Key words. Kronecker Product; Singular Value Decomposition; Inverse Problem; Regularization

MSC codes. 65F22

1. Introduction.

Image restoration is a fundamental problem in image processing and refers to the process of recovering an original image from its noisy or compromised version [8]. The image recovery process for contaminated images is, however, an ill-posed problem. In the context of image restoration, this means that basic inversion will not provide a recovered image that is acceptable as an approximation to the unknown true image due to the susceptibility of the problem to errors in the data [1, 6, 5]. Without additional considerations, any image that is recovered will be contaminated by noise, and unusable for interpretation. Here, we regularize the inversion algorithm by truncating the singular value decomposition (SVD) of the blurring matrix yielding a truncated singular value expansion for the solution [3, 4].

The image blurring problem is an example of a linear problem described by

$$\boldsymbol{A}\boldsymbol{x}_{\text{true}} = \boldsymbol{b}_{\text{true}}. \tag{1.1}$$

In the image blurring case $\boldsymbol{A}$ is a matrix of size $m \times n$ that estimates the discrete blurring operator of the discrete image $\boldsymbol{x}_{\text{true}}$ of length n. The discrete blurred image $\boldsymbol{b}_{\text{true}}$ is of length m. We will assume that the system is consistent with $m = n$. Then, even when $\boldsymbol{A}$ is

[†]School of Mathematical and Statistical Sciences, Arizona State University, AZ, (smbermu1@asu.edu,).
[‡]School of Mathematical and Statistical Sciences, Arizona State University, AZ, (renaut@asu.edu).

invertible, the solution of (1.1) is still challenging because $\boldsymbol{A}$ is ill-conditioned. Supposing $\boldsymbol{b}_{\text{true}}$ is contaminated by noise $\boldsymbol{\xi}$, yielding $\boldsymbol{b} = \boldsymbol{b}_{\text{true}} + \boldsymbol{\xi}$, then, the direct solution given by

$$\boldsymbol{x} = \boldsymbol{A}^{-1}\boldsymbol{b} = \boldsymbol{A}^{-1}\boldsymbol{b}_{\text{true}} + \boldsymbol{A}^{-1}\boldsymbol{\xi} = \boldsymbol{x}_{\text{true}} + \boldsymbol{A}^{-1}\boldsymbol{\xi}, \tag{1.2}$$

is contaminated by the $\boldsymbol{A}^{-1}\boldsymbol{\xi}$ term that depends on the noise in the data and the conditioning of $\boldsymbol{A}$. Consequently, it is not guaranteed that $\boldsymbol{x}$ provides a useful estimate for $\boldsymbol{x}_{\text{true}}$.

We adopt the notation that the SVD is given by $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top$. The matrices $\boldsymbol{U} \in \mathcal{R}^{n\times n}$ and $\boldsymbol{V} \in \mathcal{R}^{n\times n}$ are orthogonal and their columns form the left and right singular vectors, $\boldsymbol{u}_i$ and $\boldsymbol{v}_i$, for the matrix $\boldsymbol{A}$. The matrix $\boldsymbol{\Sigma}$ is diagonal, with entries $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > 0$, where $r \leq n$ is the rank of $\boldsymbol{A}$. When $r = n$, $\boldsymbol{A}$ is invertible and the solution of (1.2) is given by the expansion

$$\boldsymbol{x} = \sum_{i=1}^{n} \frac{\boldsymbol{u}_i^\top \boldsymbol{b}}{\sigma_i}\boldsymbol{v}_i = \sum_{i=1}^{n} \frac{\boldsymbol{u}_i^\top \boldsymbol{b}_{\text{true}}}{\sigma_i}\boldsymbol{v}_i + \sum_{i=1}^{n} \frac{\boldsymbol{u}_i^\top \boldsymbol{\xi}}{\sigma_i}\boldsymbol{v}_i. \tag{1.3}$$

We see that (1.3) contains the contributions from the error $\boldsymbol{\xi}$. For the terms with $|\boldsymbol{u}_i^\top \boldsymbol{\xi}| > \sigma_i$, we can expect their contributions to the solution to increase. In particular, if $|\boldsymbol{u}_i^\top \boldsymbol{\xi}| > \sigma_i$ while $|\boldsymbol{u}_i^\top \boldsymbol{b}_{\text{true}}| < \sigma_i$, the contribution to the solution for the i^{th} term in (1.3) is dominated by the coefficient due to the error rather than the true data. This is investigated using the Picard plot as described in [4, 6], and as will be illustrated in subsection 4.2.

To improve the naive solution given by (1.3) the most basic form of regularization is to use a truncated version of the SVD, $\boldsymbol{A} = \boldsymbol{U}(:, 1:k)\boldsymbol{\Sigma}(1:k, 1:k)\boldsymbol{V}(:, 1:k)^\top$ where the notation indicates that we use k columns from each of $\boldsymbol{U}$ and $\boldsymbol{V}$, and the leading block diagonal component of $\boldsymbol{\Sigma}$ of size $k \times k$. This corresponds to setting $\sigma_{k+1} = \sigma_{k+2} = \cdots = \sigma_r = 0$ and gives the truncated expansion for the solution

$$\boldsymbol{x}_k = \sum_{i=1}^{k} \frac{\boldsymbol{u}_i^\top \boldsymbol{b}}{\sigma_i}\boldsymbol{v}_i. \tag{1.4}$$

The number of terms to include can be estimated by using a tolerance tol which defines $k \leq r$ as the smallest index for which $\sigma_k \geq \text{tol} > \sigma_{k+1}$, and defines *regularization by truncation* to find $\boldsymbol{x}$ from $\boldsymbol{b}$.

We focus on image restoration problems of the form (1.1), where matrix $\boldsymbol{A}$ is a spatially invariant blurring operator for the true image. Moreover, we suppose that the blur is separable in the space directions x and y which define the coordinates of the image with respect to a reference space. Then, $\boldsymbol{A}$ is a Kronecker product (KP) operator given by $\boldsymbol{A} = \boldsymbol{A}_{\text{r}} \otimes \boldsymbol{A}_{\text{c}}$, and the SVD for $\boldsymbol{A}$ is expressed in terms of the SVD for each of $\boldsymbol{A}_{\text{c}}$ and $\boldsymbol{A}_{\text{r}}$ [8].

In this paper, we investigate the problem of learning a good choice for k to use in (1.4). In section 2 we introduce the image restoration problem using the Kronecker product and the SVD for the Kronecker product. To find k we use two different approaches; these are described in section 3. Numerical experiments given in section 4 are used to validate the use of the two different functions. The robustness of each function with respect to different blurring operators, and varying degrees of noise contamination and noise sources is demonstrated. Conclusions and future extensions of the research are given in section 5.

Main Contributions In this work we demonstrate that we can use unsupervised learning to give the truncation index for the SVD used in image restoration by minimizing the generalized cross validation criterion applied for a relatively small training set of contaminated images. We show (i) that we can obtain the index using the minimization of a given function without calculating the function for all possible k, (ii) that it is possible to learn the optimal truncation index using knowledge of true solutions to minimize an overall error, and (iii) that, alternatively, it is possible to learn the index without knowledge of true solutions using the method of generalized cross validation. Moreover, the experiments indicate that the approach is robust across blur types, noise types, and signal to noise ratios[1].

We note that this work can be seen as an application of a general framework for learning the regularization operator for the solution of inverse problems that was presented in [10]. Here our regularization approach is described by the truncated singular value decomposition, and the optimization is carried out not only with regard to minimizing the error of the solutions via supervised learning but extends to unsupervised learning using generalized cross validation. This technique has applications in many disciplines such as medical, astronomical, and traffic imaging.

2. Mathematical Details of Image Restoration and the Kronecker Product SVD.

Here we follow the derivation given in [8] which describes the image blurring process for two dimensional images. In subsection 2.1 we give the basic background for the Kronecker product following [8, Chapter 1] and then obtain the SVD following [8, Chapter 4].

2.1. Image Restoration.

Before examining the SVD we reframe (1.1) for the two dimensional case. We suppose that (1.1) is replaced by the two dimensional formulation in which we assume that the true image and blurred images are given by $\boldsymbol{X}_{\text{true}} \in \mathcal{R}^{N\times N}$, and $\boldsymbol{B}_{\text{true}} \in \mathcal{R}^{N\times N}$, respectively, and $\boldsymbol{A}$ is obtained from the discretization of a blur that is separable. For $\boldsymbol{A} = \boldsymbol{A}_{\text{r}} \otimes \boldsymbol{A}_{\text{c}}$, where $\boldsymbol{A}_{\text{r}}$ and $\boldsymbol{A}_{\text{c}}$ denote the blurring across rows and columns, respectively,

$$\boldsymbol{B}_{\text{true}} = \boldsymbol{A}_{\text{c}}\boldsymbol{X}_{\text{true}}\boldsymbol{A}_{\text{r}}^{\top}. \tag{2.1}$$

Matrices $\boldsymbol{A}_{\text{c}}$ and $\boldsymbol{A}_{\text{r}}$ are each assumed to be of size $N\times N$ requiring only $2N^2$ entries for storage. This is equivalent to the one dimensional formulation (1.1) when both $\boldsymbol{X}_{\text{true}}$ and $\boldsymbol{B}_{\text{true}}$ are consistently reshaped as one dimensional vectors of length $n = N^2$ and $\boldsymbol{A}$ is explicitly formed from its KP, yielding the matrix $\boldsymbol{A}$ of size $N^2 \times N^2$ with $n^2 = N^4$ entries. From here on, we assume that no reshaping of the images is applied and that the matrix $\boldsymbol{A}$ is never explicitly formed from its KP. Furthermore, consistent with the contaminated representation for the one dimensional case given in (1.2), we have the contaminated image $\boldsymbol{B} = \boldsymbol{B}_{\text{true}} + \boldsymbol{\Xi}$, and using the properties of the KP when $\boldsymbol{A}$ is invertible, we have

$$\boldsymbol{X} = \boldsymbol{A}_{\text{c}}^{-1}\boldsymbol{B}(\boldsymbol{A}_{\text{r}}^{\top})^{-1} = \boldsymbol{A}_{\text{c}}^{-1}(\boldsymbol{B}_{\text{true}} + \boldsymbol{\Xi})(\boldsymbol{A}_{\text{r}}^{\top})^{-1}, \tag{2.2}$$

which contains the contribution due to $\boldsymbol{\Xi}$.

[1] See https://github.com/sbermudez01/Salina-Bermudez-SIURO-Manuscript-Code.git for the MATLAB® code.

2.2. The Kronecker Product SVD. For (2.2), we assume that $\boldsymbol{A}_{\mathrm{c}} = \boldsymbol{U}_{\mathrm{c}}\boldsymbol{\Sigma}_{\mathrm{c}}\boldsymbol{V}_{\mathrm{c}}^{\top}$ and $\boldsymbol{A}_{\mathrm{r}} = \boldsymbol{U}_{\mathrm{r}}\boldsymbol{\Sigma}_{\mathrm{r}}\boldsymbol{V}_{\mathrm{r}}^{\top}$ are the SVDs for $\boldsymbol{A}_{\mathrm{c}}$ and $\boldsymbol{A}_{\mathrm{r}}$, respectively. Then, using the properties of the Kronecker product, the SVD for $\boldsymbol{A}$ is given by

$$\boldsymbol{A} = (\boldsymbol{U}_{\mathrm{r}}\boldsymbol{\Sigma}_{\mathrm{r}}\boldsymbol{V}_{\mathrm{r}}^{\top}) \otimes (\boldsymbol{U}_{\mathrm{c}}\boldsymbol{\Sigma}_{\mathrm{c}}\boldsymbol{V}_{\mathrm{c}}^{\top}) = (\boldsymbol{U}_{\mathrm{r}} \otimes \boldsymbol{U}_{\mathrm{c}})(\boldsymbol{\Sigma}_{\mathrm{r}} \otimes \boldsymbol{\Sigma}_{\mathrm{c}})(\boldsymbol{V}_{\mathrm{r}} \otimes \boldsymbol{V}_{\mathrm{c}})^{\top} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{\top}, \tag{2.3}$$

see [3, 8]. Further, we can rewrite (2.2), again using the properties of the KP, as

$$\boldsymbol{X} = \boldsymbol{V}_{\mathrm{c}}\boldsymbol{\Sigma}_{\mathrm{c}}^{-1}\left(\boldsymbol{U}_{\mathrm{c}}^{\top}\boldsymbol{B}\boldsymbol{U}_{\mathrm{r}}\right)\boldsymbol{\Sigma}_{\mathrm{r}}^{-1}\boldsymbol{V}_{\mathrm{r}}^{\top}. \tag{2.4}$$

Introducing the matrix $\hat{\boldsymbol{B}} = \boldsymbol{U}_{\mathrm{c}}^{\top}\boldsymbol{B}\boldsymbol{U}_{\mathrm{r}}$ as the set of coefficients of the image $\boldsymbol{B}$ we see that (2.4) is equivalent to

$$\boldsymbol{X} = \boldsymbol{V}_{\mathrm{c}}(\boldsymbol{\Sigma}_{\mathrm{c}}^{-1}\hat{\boldsymbol{B}}\boldsymbol{\Sigma}_{\mathrm{r}}^{-1})\boldsymbol{V}_{\mathrm{r}}^{\top}. \tag{2.5}$$

But now, observing that $\hat{\boldsymbol{B}}$ is equivalent to the reshaped vector of coefficients for the image, and that $\boldsymbol{\Sigma}_{\mathrm{r}} \otimes \boldsymbol{\Sigma}_{\mathrm{c}}$ yields diagonal $\boldsymbol{\Sigma}$ of size $N^2 \times N^2$, we can also summarize the entries for $\boldsymbol{\Sigma}$ using the array given by the outer product

$$\boldsymbol{S} = \boldsymbol{\sigma}_{\mathrm{c}}\boldsymbol{\sigma}_{\mathrm{r}}^{\top} \tag{2.6}$$

$$\boldsymbol{S}_{ij} = (\boldsymbol{\sigma}_{\mathrm{c}})_i(\boldsymbol{\sigma}_{\mathrm{r}})_j, \quad 1 \le i, j \le N, \tag{2.7}$$

where $\boldsymbol{\sigma}_{\mathrm{c}} = \texttt{diag}(\boldsymbol{\Sigma}_{\mathrm{c}})$ and $\boldsymbol{\sigma}_{\mathrm{r}} = \texttt{diag}(\boldsymbol{\Sigma}_{\mathrm{r}})$. Here `diag` extracts the diagonal entries from a matrix, and index ij extracts the ij^{th} entry from a 2D array. Equivalently, assuming $\boldsymbol{\Sigma}_{\mathrm{r}} \otimes \boldsymbol{\Sigma}_{\mathrm{c}}$ is invertible, so that all the singular values are strictly positive, we can use $\boldsymbol{R}$ of size $N \times N$ to indicate the array with entries $1/((\boldsymbol{\sigma}_{\mathrm{c}})_i(\boldsymbol{\sigma}_{\mathrm{r}})_j)$. Then, without reshaping, we have

$$\boldsymbol{\Sigma}_{\mathrm{c}}^{-1}\hat{\boldsymbol{B}}\boldsymbol{\Sigma}_{\mathrm{r}}^{-1} = \boldsymbol{R} \odot \hat{\boldsymbol{B}} \text{ and} \tag{2.8}$$

$$\left(\boldsymbol{\Sigma}_{\mathrm{c}}^{-1}\hat{\boldsymbol{B}}\boldsymbol{\Sigma}_{\mathrm{r}}^{-1}\right)_{ij} = \frac{\hat{\boldsymbol{B}}_{ij}}{(\boldsymbol{\sigma}_{\mathrm{c}})_i(\boldsymbol{\sigma}_{\mathrm{r}})_j} = \frac{(\hat{\boldsymbol{B}}_{\mathrm{true}})_{ij} + \hat{\boldsymbol{\Xi}}_{ij}}{(\boldsymbol{\sigma}_{\mathrm{c}})_i(\boldsymbol{\sigma}_{\mathrm{r}})_j}, \tag{2.9}$$

where $\odot$ denotes the elementwise Hadamard product, as indicated in (2.9). As in (1.3) we have an expansion for $\boldsymbol{X}$ which will be contaminated with the error terms if $|\hat{\boldsymbol{\Xi}}_{ij}| > (\boldsymbol{\sigma}_{\mathrm{c}})_i(\boldsymbol{\sigma}_{\mathrm{r}})_j$ when $|\hat{\boldsymbol{\Xi}}_{ij}| > |\hat{\boldsymbol{B}}_{ij}|$, where we consistently define $\hat{\boldsymbol{B}}_{\mathrm{true}} = \boldsymbol{U}_{\mathrm{c}}^{\top}\boldsymbol{B}_{\mathrm{true}}\boldsymbol{U}_{\mathrm{r}}$ and $\hat{\boldsymbol{\Xi}} = \boldsymbol{U}_{\mathrm{c}}^{\top}\boldsymbol{\Xi}\boldsymbol{U}_{\mathrm{r}}$.

To obtain the truncated solution that is equivalent to (1.4) for the one dimensional case we need to truncate $\boldsymbol{\Sigma}$. To truncate the entries in $\boldsymbol{\Sigma}$ we introduce the array for entries of the truncated singular value matrix

$$(\boldsymbol{S}_{\mathrm{trunc}})_{ij} = \begin{cases} (\boldsymbol{\sigma}_{\mathrm{c}})_i(\boldsymbol{\sigma}_{\mathrm{r}})_j & (\boldsymbol{\sigma}_{\mathrm{c}})_i(\boldsymbol{\sigma}_{\mathrm{r}})_j > \texttt{tol} \\ 0 & \text{otherwise,} \end{cases} . \tag{2.10}$$

Then, consistent with the notation for the inverse entries from $\boldsymbol{S}$, we use $\boldsymbol{R}_{\mathrm{trunc}}$ as the array with the entries of the pseudoinverse for $\boldsymbol{\Sigma}^{\dagger}$

$$(\boldsymbol{R}_{\mathrm{trunc}})_{ij} = \begin{cases} \frac{1}{(\boldsymbol{\sigma}_{\mathrm{c}})_i(\boldsymbol{\sigma}_{\mathrm{r}})_j} & (\boldsymbol{\sigma}_{\mathrm{c}})_i(\boldsymbol{\sigma}_{\mathrm{r}})_j > \texttt{tol} \\ 0 & \text{otherwise.} \end{cases} \tag{2.11}$$

where † indicates the pseudoinverse of a matrix [3]. Combining the terms (2.5) and (2.11) we obtain the the truncated solution

$$\boldsymbol{X}_{\text{trunc}} = \boldsymbol{V}_{\text{c}}(\boldsymbol{R}_{\text{trunc}} \odot \hat{\boldsymbol{B}})\boldsymbol{V}_{\text{r}}^{\top}, \tag{2.12}$$

which is efficiently implemented by matrix operations for matrices of sizes $N \times N$ rather than $N^2 \times N^2$, and depends on `tol` in (2.10).

In (2.10) we see that the truncation of the singular values depends on the products of the singular values in $\boldsymbol{\sigma}_{\text{c}}$ and $\boldsymbol{\sigma}_{\text{r}}$, and not on each of these independently. Reordering the elements $(\boldsymbol{\sigma}_{\text{c}})_i(\boldsymbol{\sigma}_{\text{r}})_j$ from large to small, corresponding to the ordered singular values for $\boldsymbol{\Sigma}$, (2.12) corresponds to identifying an index k such that the first k singular values are greater than `tol` and the remaining ordered singular values for index greater than k are ignored in forming (2.12). Assuming that $\boldsymbol{A}$ is invertible, there are N^2 choices for truncated solutions (2.12) indexed by truncation index k given by $\boldsymbol{X}_{\text{trunc}}(k)$, $k = 1, \ldots, N^2$. We note that each choice of k corresponds to a choice for `tol` where k will increase as `tol` decreases, and we expect that the solutions (2.12) depend on k via (2.11).

3. Identification and Validation of Optimal Truncation Indices. We establish how to learn an optimal choice for k with respect to minimization of an objective function $\Phi(k)$ with different choices for Φ and with the definition

$$k_{\text{OPT}} = \underset{1 \le k \le N^2}{\text{argmin}}\, \Phi(k). \tag{3.1}$$

Here, we consider two possible approaches for determining k_{OPT} in (3.1) in a learning framework when we have more than one image to restore. Suppose that we have a set of T contaminated images $\boldsymbol{B}(t)$, indexed by t, $t = 1, \ldots, T$. These correspond to T true images $\boldsymbol{X}_{\text{true}}(t)$, $t = 1, \ldots, T$. In the first instance we suppose that $\boldsymbol{X}_{\text{true}}(t)$ are given and that we can learn k_{OPT} to minimize a measure of the image error. In the second case, we suppose that $\boldsymbol{X}_{\text{true}}(t)$ are not given and it is of interest to learn the value for k_{OPT} in an unsupervised learning framework, namely without using $\boldsymbol{X}_{\text{true}}(t)$. This leads to our second method that uses the method of generalized cross validation (GCV). This is a classical statistical technique with extensive discussion in the literature [2, 6]. Fundamentally, the aim of GCV is find a good estimate for k_{OPT} that is valid if the image is restored with a missing data point, considered over all possible missing data points. In both cases we can validate against known images to test the ability of the unsupervised method to provide stable indices for k_{OPT}, and we can test the results against data that were not used in the training, namely with a set of testing images. A third method was used in early testing for the unsupervised learning using the normalized cummulative periodogram (NCP) [7, 13], but this method did not perform as well and results are not reported. In particular, the NCP requires the choice of an additional parameter, for tuning the reliability with respect to the noise in the data, and therefore the approach is in general less robust.

In general, with any of these approaches it is important to note that there are some potential disadvantages of the learning process. There must be sufficient data of similar image quality and type in terms of noise level and the image blur, here we assume $T = 40$ images are available and we use 20 images to the train the algorithm and 20 images for testing.

The size of the image need not be a factor, provided that any given image can be partitioned to the size of the smallest image in the data set.

We describe the methods used to identify k_{OPT} in subsections 3.1 to 3.2, the approaches to minimize their objective functions in subsection 3.3, and the approach to validate the choice of k_{OPT} in subsection 3.4.

3.1. Supervised Learning Using Uncorrupted True Images. Given a known uncorrupted image $\boldsymbol{X}_{\text{true}}(t)$ we can evaluate the error between $\boldsymbol{X}_{\text{true}}(t)$ and $\boldsymbol{X}_{\text{trunc}}(t,k)$ as a function of k, $k = 1, \ldots, N^2$. We define the absolute relative error $\varepsilon(t,k)$ for a given image t and a given truncation index k by

$$\varepsilon(t,k) = \frac{\|\boldsymbol{X}_{\text{trunc}}(t,k) - \boldsymbol{X}_{\text{true}}(t)\|_F}{\|\boldsymbol{X}_{\text{true}}(t)\|_F}. \tag{3.2}$$

Here we use the definition of the Frobenius norm, $\|\boldsymbol{Y}\|_F^2 = \sum_{ij} y_{ij}^2$ where $\boldsymbol{Y}$ is a matrix with entries y_{ij}. Consequently, $\varepsilon(t,k)$ measures the total relative error of the true image from the approximated image for each pixel as a function of k for fixed t. This leads to the error function defined over images $t = 1, \ldots, T_{\text{train}}$, for a training set of size T_{train},

$$\Phi_{\text{MRE}}(k) = \frac{1}{T_{\text{train}}} \sum_{t=1}^{T_{\text{train}}} \varepsilon(t,k) = \frac{1}{T_{\text{train}}} \sum_{t=1}^{T_{\text{train}}} \frac{\|\boldsymbol{X}_{\text{trunc}}(t,k) - \boldsymbol{X}_{\text{true}}(t)\|_F}{\|\boldsymbol{X}_{\text{true}}(t)\|_F} \tag{3.3}$$

$$= \frac{1}{T_{\text{train}}} \sum_{t=1}^{T_{\text{train}}} \frac{\|\boldsymbol{R}_{\text{trunc}} \odot \hat{\boldsymbol{B}} - \hat{\boldsymbol{X}}(t)\|_F}{\|\boldsymbol{X}_{\text{true}}(t)\|_F}. \tag{3.4}$$

Here we have used (2.12) to rewrite $\boldsymbol{X}_{\text{trunc}}(t,k)$ and introduced $\hat{\boldsymbol{X}} = \boldsymbol{V}_{\text{c}}^\top \boldsymbol{X}_{\text{true}} \boldsymbol{V}_{\text{r}}$ yielding the efficient representation depending on $\boldsymbol{R}_{\text{trunc}} \odot \hat{\boldsymbol{B}}$ as a function of k. Minimizing $\Phi_{\text{MRE}}(k)$ as a function of k using the definition in (3.1) to give k_{OPT}, leads to k_{MRE} that is optimal with respect to minimizing the absolute relative error over all training images. We denote this method by MRE (for mean relative error).

3.2. Using Generalized Cross Validation. The second process to learn the truncation index k_{OPT} uses the GCV method [3, 5, 8]. This is an unsupervised method and is formulated for a single set of data in terms of the residual $\|\boldsymbol{A}\boldsymbol{x}(k) - \boldsymbol{b}\|_2$, weighted by the number of terms not used to generate $\boldsymbol{x}(k)$. For a single image, as shown on [8, page 79, Section 6.5], the GCV function admits a concise expression given by

$$\mathcal{G}(k) = \frac{1}{(N^2 - k)^2} \sum_{j=k+1}^{N^2} |\hat{B}_j(\mathrm{t})|^2, \tag{3.5}$$

which leads to

$$\Phi_{\text{GCV}}(k) = \frac{1}{T_{\text{train}}} (N^2 - k)^2 \sum_{t=1}^{T_{\text{train}}} \sum_{j=k+1}^{N^2} |\hat{B}_j(\mathrm{t})|^2, \tag{3.6}$$

taken over all images $t = 1, \ldots, T_{\text{train}}$. Now minimizing $\Phi_{\text{GCV}}(k)$ as a function of k defines k_{GCV} as optimal with respect to the GCV function calculated for all training images. Notice that due to the orthogonality of $\boldsymbol{V}_{\text{c}}$ and $\boldsymbol{V}_{\text{r}}$ the expression is given just in terms of the coefficients $\hat{B}_j(\text{t})$ for each image t. Moreover, in (3.6) we can switch the order of the summation so that we can form a summed set of coefficients across all images, $|\hat{B}_{\text{all}}|^2 = \sum_{t=1}^{T_{\text{train}}} |\hat{B}(\text{t})|^2$ which is independent of k. The sum over j is then very efficient for any k. Given a choice of k_{OPT} we can also calculate the error given by (3.2) as a quantitative measure of the quality of the restoration.

3.3. Methods to Minimize the Functions. In each case we consider two options to minimize Φ_{MRE} and Φ_{GCV}. This yields four different choices for k_{OPT}. Here k_{MRE}^{exh} and k_{GCV}^{exh} are the indices given by evaluating $\Phi(k)$ over all discrete choices for k and then finding the actual minimum of the discrete set $\Phi(k)$, $k = 1, \ldots, N^2$ in each case. This is the *exhaustive* case, hence the superscript *exh*. We can verify whether $\Phi(k)$ at the chosen point k_{OPT}^{exh} is a discrete local or global minimum by looking at its graph of the discrete set $(k, \Phi(k))$, $k = 1, \ldots, N^2$. For the second approach we use the MATLAB® function `fminbnd` to automatically find the minimum of the function $\Phi(k)$, but treated as a continuous function in k. We can plot the output of the iterated values $(k_i, \Phi(k_i))$, $i = 1, \ldots, k_{\text{conv}}$, where k_{conv} is the k_i at which the algorithm converges to the provided tolerance, to assess the performance of `fminbnd` for finding the minimum index. Practically, we select k_{OPT} as the nearest integer to k_{conv}. Because Φ is a function of only one variable, this approach is successful, as will be shown in the presented results.

3.4. Validation Approach. Given a choice of truncation index k_{OPT} provided by one of the methods, it is important to validate the use of k_{OPT} for images that are not part of the training set. Given the images $\boldsymbol{B}(t)$, $t = 1, \ldots, T$, $T_{\text{train}} < T$ images are used to learn k_{OPT} for each Φ. The remaining images $t = T_{\text{train}} + 1, \ldots, T$ are used for validation. For each image we can calculate $\varepsilon(t, k)$, for a given $k = k_{\text{OPT}}$, as defined in (3.2) and therefore

$$\rho(k) = \frac{1}{T_{\text{test}}} \sum_{t=T_{\text{train}}+1}^{T} \varepsilon(t, k) \tag{3.7}$$

is a measure of the average relative error over $T_{\text{test}} = T - T_{\text{train}}$ images that were not used for training. Notice that by (3.4), $\rho(k)$ calculated for the training data alone is just $\Phi_{\text{MRE}}(k)$ and can be used to compare the average relative errors of the restored images from the training and testing data sets.

4. Description of Numerical Experiments and Results.

4.1. Numerical Experiments. The numerical images used in the experiments are selected from multiple online sample image databases including NASA's California Institute of Technology, the University of Southern California, Stanford University and an online site bogotobogo.[2] The original images are set to a gray scale and the same size. The images are then

[2]The selected testing and training images are available to the public courtesy of NASA, JPL-Caltech, SU, USC, and bogotobogo [11, 12, 14, 15].

blurred and corrupted by the addition of noise, dependent on noise type and the amount of noise added, as described in subsections 4.1.1 and 4.1.2.

4.1.1. Image Blur. All the images are contaminated using a two dimensional Gaussian blur [8, 3.2] given in each spatial dimension by the function

$$p(x) = \exp\left(-\frac{1}{2}\left(\frac{x}{\alpha}\right)^2\right), \tag{4.1}$$

with entries less than $1e{-}4$ set to 0. We assume that each image is on the domain $-1 \leq x, y \leq 1$ for all images with $N \times N$ pixels. Then, $p(x)$ is discretized for the given image and normalized such that the $\sum_i p(x_i) = 1$. Two choices of blur are considered, a symmetric blur with $\boldsymbol{A}_{\rm c} = \boldsymbol{A}_{\rm r}$, with $\alpha = .01$ and a blur that is more elongated in one direction, $\boldsymbol{A}_{\rm c} \neq \boldsymbol{A}_{\rm r}$ with $\boldsymbol{A}_{\rm r}$ formed using $\alpha = .02$, but keeping the same $\boldsymbol{A}_{\rm c}$. The point spread functions for these blurs are illustrated in Figure 1 for the case with $N = 512$. In the presented formulation we assume zero boundary conditions, so that $\boldsymbol{A}_{\rm r}$ and $\boldsymbol{A}_{\rm c}$ are Toeplitz [8]. An example from the image set, `peppers512x512.tif`, is shown on the left in Figure 2, with the effect of blurring shown in the middle and right in Figure 2. We observe that the asymmetric blur distorts the image significantly. For the presentation of the results of the restoration we will, therefore, mainly show the restored images for the case with the symmetric blur. We will, however, still use the asymmetric case for testing the estimation of $k_{\rm OPT}$, in terms of the relative errors that are obtained over all data by the different methods.

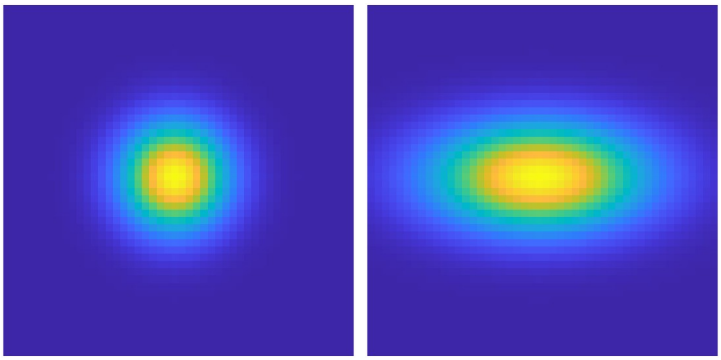

Figure 1. *The point spread function for symmetric and asymmetric blur, left and right plot respectively, for the case with $N = 512$, zoomed into the middle region with from indices 233 to 280, in both directions.*

4.1.2. Noise Contamination and the Signal to Noise Ratio. Given the blurred image $\boldsymbol{B}_{\rm true}$, noise $\boldsymbol{\Xi}$ is added to the given image based on obtaining images with comparable signal to noise ratios defined by

$$\text{SNR} = 10 \log\left(\frac{\|\boldsymbol{B}_{\rm true}\|_F^2}{\|\boldsymbol{\Xi}\|_F^2}\right). \tag{4.2}$$

To obtain the comparable noise levels for the different n oise d istributions, w e g enerate a contaminated image $\boldsymbol{B}$ by using the Image Processing toolbox function $\boldsymbol{B}$ =`imnoise`($\boldsymbol{B}_{\rm true}$,

Figure 2. *Original, uncontaminated image* $\boldsymbol{X}_{\text{true}}$ *from the testing set, on gray scale where* $N = 512$*, on the left.* $\boldsymbol{B}_{\text{true}}$ *images with symmetric and asymmetric blur, middle and right respectively.*

'`noisetype`'), where `noisetype` is chosen as either `gaussian`, `poisson` or `salt & pepper`. Then, to obtain the scaling of the error added to $\boldsymbol{B}$ we use

$$\boldsymbol{\Xi} = (\boldsymbol{B} - \boldsymbol{B}_{\text{true}})10^{-(\frac{\text{SNR}}{20})}\frac{\|\boldsymbol{B}_{\text{true}}\|_F}{\|\boldsymbol{B} - \boldsymbol{B}_{\text{true}}\|_F}, \tag{4.3}$$

and set $\boldsymbol{B} = \boldsymbol{B}_{\text{true}} + \boldsymbol{\Xi}$. This is non standard, particularly for Gaussian noise, but it allows the use of the same approach for all noise types and generates consistent SNR of each image. For the `salt & pepper` case we note that we can change the density for the noise by using the density parameter of impacted pixels. For the experiments we use the default that 5% of pixels are contaminated. Hence, rather than noise level we generate the results in terms of the SNR. These effects are shown in Figures 3 and 4 for the case of low (SNR = 10) and high (SNR = 40) SNR, respectively, with noise types `gaussian`, `poisson` and `salt & pepper` noise from left to right in each image, as applied to the image obtained with symmetric blur as given in the middle Figure 2.

Figure 3. *Images* $\boldsymbol{B}$ *with symmetric blur and* `gaussian`*,* `poisson` *and* `salt&pepper` *noise, left to right, respectively, for the case with* $N = 512$ *and SNR* $= 10$*.*

4.2. Experimental results. For the example image with Poisson noise and SNR = 40 as shown in the middle plot in Figure 4, we find the naive solution which is obtained from (2.5).

Figure 4. *Images* $\boldsymbol{B}$ *with symmetric blur and* `gaussian`, `poisson` *and* `salt&pepper` *noise, left to right plots respectively, for the case with* $N = 512$ *and* $SNR = 40$.

This is illustrated on the right in Figure 5. The Picard plots for all images in Figure 4 are given in Figure 5, from left to right. The Picard plot gives the singular values, σ_k, the coefficients $|\hat{\boldsymbol{B}}_k|$ and their ratios, and shows that the ratios of the coefficients to the singular indices grow in magnitude for large index k when the noise is such that $|\hat{\boldsymbol{B}}_k| > \sigma_k$, and corresponds to the point of instability when restoring the image, [5]. Across all noise types the plots suggest that we may estimate the truncation index to generate the truncated solution indicated in (1.4) and (2.12). For clarity in the Picard plots, the entries are plotted every 100 points. From this plot we can estimate that it might be appropriate to form a truncated solution, say with $k = 6500$, using (2.12). This point is given by the solid grey line in each plot. This estimated value is applied to all three cases of noise for the image $\boldsymbol{B}$ shown in Figure 4 to verify accuracy for all types. These solutions are illustrated in Figure 6. With these results we can confirm that a truncated solution is far more reliable and accurate than the naive restoration. But, in general, we want to identify an optimal k without analysis of each Picard plot. In the Picard plots the thick solid blue line indicates the truncation index which gives the actual least relative error for the given contaminated case.

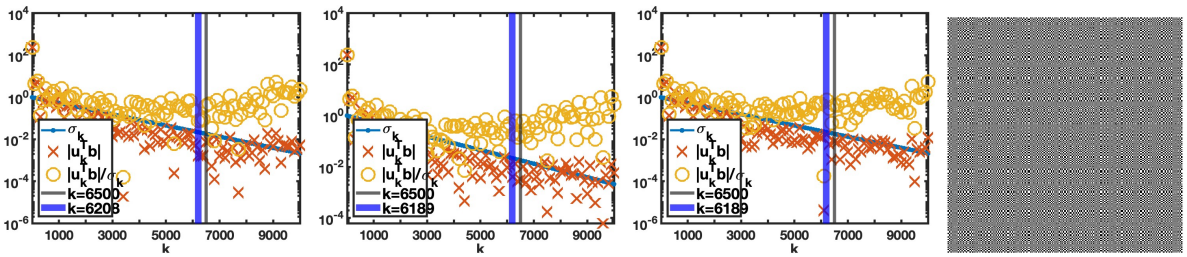

Figure 5. *The Picard plots for the symmetric blur operator with* $N = 512$ *and* $SNR = 40$*, for the noise types* `gaussian`, `poisson` *and* `salt&pepper`*, respectively, from left to right. In the last plot* X_{naive} *is the restored image of* $\boldsymbol{B}$ *for the case with* `poisson` *noise corresponding to the second Picard plot. The solid grey line is at* $k = 6500$ *and the thicker solid blue line is at* k *which gives the least error over all choices for* k.

Figure 6. *The restoration of images in Figure 4 using a truncated solution with $k = 6500$ points for* `gaussian`*,* `poisson` *and* `salt&pepper` *noise, left to right plots respectively. The relative errors for these images are* 0.0914*,* 0.0918 *and* 0.0917*, rounded to* 3 *significant digits. This compares with the optimal errors obtained for these images for the blue vertical lines indicated in the Picard plots in Figure 5,* 0.0911*,* 0.0914*, and* 0.0912*, respectively.*

In the experiments to find k_{OPT} we use a total of $T = 40$ images, separated into two sets. Training images 1 to 20, $T_{\text{train}} = 20$, are used to find the optimal k_{MRE} and k_{GCV}. Testing images 21 to 40, $T_{\text{test}} = T - T_{\text{train}} = 20$, are used to evaluate the use of k_{MRE} and k_{GCV}. Examples from each set are illustrated in Figure 7. The example image in Figure 2 is a member of the testing set.

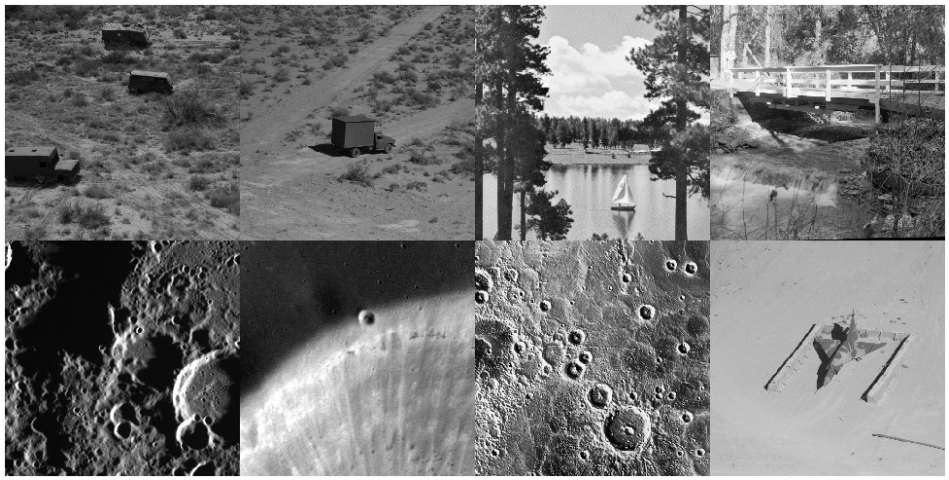

Figure 7. *Four images from the training and testing set, top and bottom row, respectively*

In Figure 8 we plot the functions $\Phi(k)$ illustrated for training the data on images of size $N = 128$ with symmetric blur, SNR $= 40$ and `gaussian` noise. The solid lines represent the *exhaustive* case used to calculate $\Phi(k)$ using all k for both functions (3.4) and (3.6). The second approach utilizes the MATLAB® function `fminbnd` to automatically find the minimum of both functions. The $\times$ indicate the iterative indices k_i used in `fminbnd` to find the minimum of $\Phi(k)$. The diamonds indicate the points at the minimum. This case illustrates that `fminbnd`

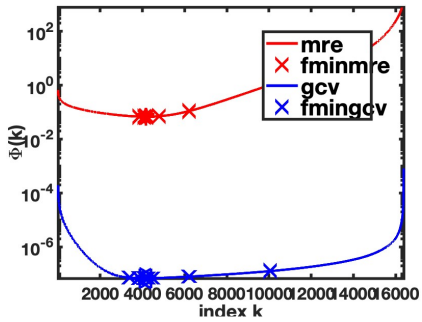

Figure 8. *Here with the solid line we show the functions (3.4) and (3.6), with red and blue, respectively, for all k. The indices k_{MRE} and k_{GCV} are found as the minimum over all the plotted points on the curves, and are indicated by the diamonds in each case. The $\times$ symbols are the k_i used in the minimization of (3.4) and (3.6), respectively, when we minimize using the MATLAB® function* `fminbnd` *to find k_{MRE} and k_{GCV}. The plot is obtained using $N = 128$ for* `gaussian` *noise with $SNR = 40$ and symmetric blurring.*

does identify the minimum of $\Phi(k)$. Due to the increased efficiency of using `fminbnd`, in our results we will only present k_{OPT} using `fminbnd`.

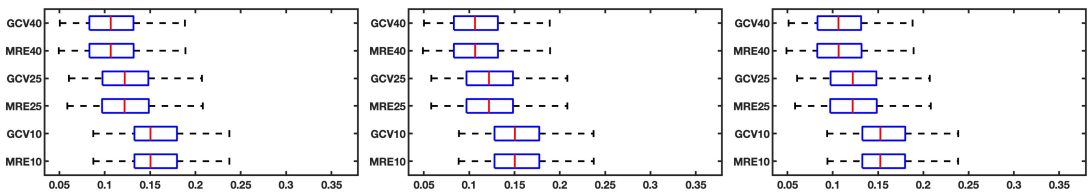

Figure 9. *Box plots for the training data for problem size $N = 512$ with noise type* `gaussian`*,* `poisson` *and* `salt&pepper`*, left to right plots respectively. In each plot the noise levels in each pair are $SNR = 40$, 25 and 10, with each pair corresponding to the results with Φ_{GCV} and then Φ_{MRE} for finding k_{GCV} and k_{MRE}, respectively, as indicated in the box plot labels.*

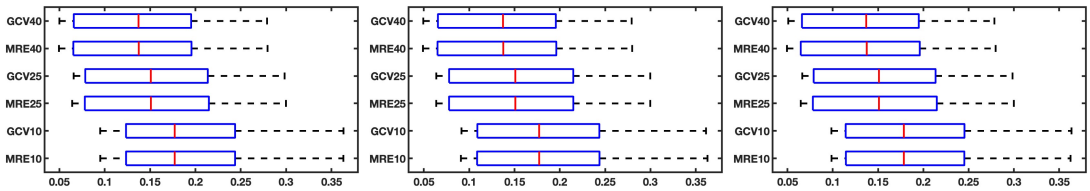

Figure 10. *Box plots for the testing data for problem size $N = 512$ with noise type* `gaussian`*,* `poisson` *and* `salt&pepper`*, left to right plots respectively. In each plot the noise levels in each pair are $SNR = 40$, 25 and 10, with each pair corresponding to the results with Φ_{GCV} and then Φ_{MRE} for finding k_{GCV} and k_{MRE}, respectively, as indicated in the box plot labels.*

4.2.1. Relative Error Box Plots. In Figures 9 and 10 we show the box plot relative errors for the experiments, as indicated in the captions. The results shown are chosen to illustrate results with size $N = 512$ for all noise levels, SNR = 10, 25 and 40, and all noise types, with symmetric blur and contrasting results for training and testing data in Figures 9 and 10, respectively. Specifically, the relative errors are calculated for the training data against the

true data, and for the testing data against the true data, yielding the two sets of box plots. For each method, Φ_{MRE} and Φ_{GCV}, three noise levels are chosen yielding the six different plots in each case, as indicated in the labels. There are then three sets of results in Figures 9 and 10 to represent the effects of each of the three noise types.

From the box plots, we can see that the approaches for finding k_{OPT} are successful. Notice that in each figure, all three plots have the same x-axis scale so that a true comparison can be seen between the training and testing data, as the error level decreases, and for the different methods. We see immediately that the errors for the testing data are higher than for the training data and have a larger range of whiskers, meaning there is more uncertainty in the testing results. This is to be expected since the testing data are not used to find the k_{OPT}. The errors are, however, still reasonably low and we can see that both Φ_{MRE} and Φ_{GCV} are successful across all noise types and levels. This demonstrates that we can learn k_{OPT} in both a supervised and an unsupervised framework. The results indicated in the box plots are visually comparable for both supervised and unsupervised methods. This suggests that the GCV estimator is a powerful tool that can be used to find k_{OPT} from small sets of data, here just 20 images are used for training in each case.

4.2.2. Quantitative Results. To further assess the methods, the impact of using k_{OPT} to generate the restored images can be analyzed quantitatively. As previously described, we calculate the average relative error, (3.7), for the given k_{OPT} and sets of images. The results are summarized in Tables 1 to 3 for the three choices of SNR and the three noise types. For comparison we also record the time taken by each method, τ, and the number of iterations and function evaluations, IT and FE respectively. Here ρ is the mean relative error over the testing images, $t = T_{\text{train}} + 1$ to T. In Table 1 we give the results for size $N = 256$ with symmetric blur, the same case but with $N = 512$ in Table 2, and then for the case with asymmetric blur and $N = 512$ in Table 3. All results are reported for experiments using MATLAB® version R2022a running on an iMac with a 4.2 GHz Quad-Core Intel Core i7 chip.

Table 1

The results of calculating k_{OPT} using (3.4) and (3.6) and `fminbnd` *in terms of the computational cost measured in seconds, the number of iterations IT, the number of function evaluations, FE, and the average relative error ρ, for each method. Here the results are given for the problem with $N = 256$ and symmetric blur.*

Noise		K		τ		IT		FE		ρ	
Type	SNR	MRE	GCV	MRE	GCV	MRE	GCV	MRE	GCV	MRE	GCV
Gaussian	10	1426	1287	0.12	0.072	29	28	30	29	0.179	0.181
	25	2970	3063	0.11	0.031	27	32	28	33	0.124	0.123
	40	5120	5206	0.14	0.032	32	19	33	20	0.0946	0.0944
Poisson	10	1410	1410	0.099	0.039	26	22	27	23	0.176	0.176
	25	2889	3063	0.11	0.03	37	26	38	27	0.124	0.123
	40	5210	5205	0.072	0.028	20	24	21	25	0.0946	0.0946
S & P	10	1406	1276	0.1	0.033	33	30	34	31	0.178	0.18
	25	3053	3080	0.075	0.027	26	27	27	28	0.123	0.123
	40	5108	5196	0.071	0.029	24	22	25	23	0.0947	0.0945

Table 2

The results of calculating k_{OPT} using (3.4) and (3.6) and `fminbnd` *in terms of the computational cost measured in seconds, the number of iterations IT, the number of function evaluations, FE, and the average relative error ρ, for each method. Here the results are given for the problem with $N = 512$ and symmetric blur.*

Noise		K		τ		IT		FE		ρ	
Type	SNR	MRE	GCV	MRE	GCV	MRE	GCV	MRE	GCV	MRE	GCV
Gaussian	10	2007	2002	0.51	0.13	26	30	27	31	0.197	0.197
	25	3906	4120	0.38	0.13	27	24	28	25	0.16	0.159
	40	6222	6338	0.34	0.14	22	22	23	23	0.14	0.139
Poisson	10	2002	2026	0.45	0.13	38	20	39	21	0.193	0.193
	25	3898	3906	0.34	0.13	25	31	26	32	0.159	0.159
	40	6156	6273	0.35	0.13	25	29	26	30	0.14	0.139
S & P	10	2026	2012	0.35	0.13	24	26	25	27	0.196	0.196
	25	3907	4122	0.42	0.14	28	26	29	27	0.16	0.159
	40	6153	6434	0.45	0.13	33	29	34	30	0.14	0.139

Table 3

The results of calculating k_{OPT} using (3.4) and (3.6) and `fminbnd` *in terms of the computational cost measured in seconds, the number of iterations IT, the number of function evaluations, FE, and the average relative error ρ, for each method. Here the results are given for the problem with $N = 512$ and asymmetric blur.*

Noise		K		τ		IT		FE		ρ	
Type	SNR	MRE	GCV	MRE	GCV	MRE	GCV	MRE	GCV	MRE	GCV
Gaussian	10	1176	1277	0.43	0.13	25	27	26	28	0.217	0.216
	25	2223	2205	0.32	0.13	16	25	17	26	0.186	0.186
	40	3397	3507	0.39	0.16	27	30	28	31	0.169	0.169
Poisson	10	1273	1272	0.39	0.13	31	28	32	29	0.212	0.212
	25	2314	2228	0.33	0.13	22	23	23	24	0.185	0.186
	40	3377	3509	0.39	0.14	27	23	28	24	0.169	0.169
S & P	10	1272	1275	0.41	0.14	34	25	35	26	0.215	0.215
	25	2210	2231	0.38	0.13	26	27	27	28	0.186	0.186
	40	3389	3545	0.36	0.13	24	33	25	34	0.169	0.169

First of all, from all three tables Tables 1 to 3, we see that the average errors obtained by methods using Φ_{MRE} and Φ_{GCV} are comparable, regardless of blur type or image size. This confirms the observations from the box plots for the symmetric case and $N = 512$. We also see uniformly that the number of function evaluations and iterations are comparable for each method, but that the cost in terms of computational time of finding k_{MRE} is higher than that of finding k_{GCV}. To understand the difference in costs, we look back to (3.4) and (3.6) and assess that the precalculation of $|\hat{\boldsymbol{B}}_{\text{all}}|$ provides a function that is more efficient to evaluate for GCV. These observations together strongly support the use of the GCV function to find k_{OPT}. We also confirm that the approach is successful for different image sizes, see the results

in Tables 1 and 2, and different blur types, comparing Tables 2 and 3. Moreover, from the calculated estimates for k_{MRE} and k_{GCV}, comparing the methods, we see that we do not find $k_{\text{MRE}} = k_{\text{GCV}}$. This suggests that the results are not too sensitive to the choice for k_{OPT}, provided that the given function has found a minimum to the tolerances used in `fminbnd`, particularly given that the different choices for k_{OPT} yield comparable relative errors over the testing data. As would be expected the errors and the computational costs are larger for the larger problem. Moreover, we would anticipate from the image degradation seen for the image with asymmetric blur, that we would expect k_{OPT} to be smaller than for the case with symmetric blur. This is confirmed in the results. Finally, these quantitative results demonstrate that the approach is successful for all three noise types and three noise levels considered in the experiments.

We are able to conclude that despite changes in noise type, SNR indices, and blur type, the GCV method in (3.6) does not require the known data, is far less expensive to run, and gives errors that are comparable to the use of the MRE method in (3.4).

4.2.3. Illustrative Restored Images. We now assess this conclusion by plotting images that were restored using the optimal indices k_{MRE} and k_{GCV} for the original image from the testing set given in Figure 2 and an image from the training set.

Figure 11. *The restoration of images in Figure 4 using the function Φ_{MRE} for* `gaussian`*,* `poisson` *and* `salt&pepper` *noise, left to right plots respectively. The relative errors for these images are all* .091*, with the same rounding as in Table 2.*

The images presented in Figures 11 and 12 provide a visual evidence of the quality of the restoration of the images $\boldsymbol{B}$ in Figure 4, images from the testing set, using the truncation indices k_{MRE} and k_{GCV} found using each Φ. Also given in the captions are the relative errors for each of these images as compared to the true images. For all of the images in Figures 11 and 12 the relative errors are .091, with rounding to two significant d igits. T hese relative errors are marginally less in all cases than the relative errors reported in Figure 6 where the truncation index was found by inspection of the Picard plot, in each case. Moreover, as expected from Table 1, the results from both methods are quite similar despite the difference in truncation indices. Further, even though these images are from the testing set, their relative errors are less than the average relative errors for these cases as reported in Table 2. But from Figure 10 we can see that it is possible for images from the testing set range to have errors .091 which are on the low ends of the whiskers in the relative error box plots.

Figure 12. *The restoration of images in Figure 4 using the function Φ_{GCV} for* `gaussian`*,* `poisson` *and* `salt&pepper` *noise, left to right plots respectively. The relative errors for these images are all* .091*, with the same rounding as in Table 2.*

In Figure 13 we show the restoration of an image, `house.tiff`, from the training data set, with the different noise types and SNR = 40, to confirm the presented quantitative results. Specifically, that the testing set results are acceptable in relation to the training results. For this example, the relative errors given in the captions are all close, 0.102 with rounding for the different noise types, respectively, and larger than the reported errors for the image from the testing set. The relative error box plots for the training data allow that images may have larger relative errors for an individual case as compared to results from the testing data. Overall the images presented in Figures 11 to 13 support the results presented in the box plots and the results in Table 2, and demonstrate that the restorations are acceptable.

To show the impact of using the approach to restore images with asymmetric blur, in Figure 14 we show the restoration of the corrupted image with the three noise types for the image in the right in Figure 2, corresponding to an image in the testing set with SNR = 40. The relative errors are all close for each image, 0.118 with rounding for the different noise types.

Overall, these results are notable particularly given the simplicity of the approach which only uses truncation in the SVD and does not impose any other form of regularization. We emphasize that it is sufficient to use the unsupervised GCV method to find an acceptable truncation index for image restoration.

5. Conclusions. We have shown that the presented machine learning approach can be used to identify k_{OPT} for image restoration. Different methods for identifying k_{OPT} were considered. Our experiments considered both symmetric and asymmetric blur operators and image sizes up to $N = 512$, and demonstrated that either method to identify k_{OPT} is robust regardless of image size, blur operator, signal to noise ratios, and the `gaussian`, `poisson` and `salt&pepper` noise types. The reported experiments demonstrate that it is sufficient to use an unsupervised method to find the truncation indices, specifically the GCV function can be efficiently minimized. Moreover, the results obtained are comparable in terms of the achieved relative errors to those from the supervised MRE method. Notably, it is computationally cheaper in terms of wall clock time to use the unsupervised GCV estimator. Finally, these results are achieved using a small training set with just 20 images for images of sizes, and has

Figure 13. *The restoration of a blurred image in the training set with* $N = 512$*, using* k_{GCV} *as given in Table 2 for* `gaussian`*,* `poisson` *and* `salt&pepper` *noise, left to right plots respectively, and symmetric blur. The relative errors for these images are are all* 0.103 *with the same rounding as in Table 2.*

Figure 14. *The restoration of a blurred image in the testing set with* $N = 512$*, using* k_{GCV} *as given in Table 3 for* `gaussian`*,* `poisson` *and* `salt&pepper` *noise, left to right plots respectively, and asymmetric blur. The relative errors for these images are are all* 0.118 *with the same rounding as in Table 3.*

been tested on images from $N = 64$ to $N = 512$, with results presented for $N = 256$ and $N = 512$. Overall, determining k_{OPT} using `fminbnd` applied to minimize the GCV function is effective and efficient across blur types, image sizes, SNR levels and noise types.

We note that our investigation is not unique. In another report on optimal experimental design for inverse problems, similar supervised learning approaches were used [9]. There, Tikhonov regularization was more generally considered, as well as optimal spline and mean squared error (MSE) filters. They remarked that these optimal filters and approaches performed better than the standard GCV method for a one-dimensional problem, but did not consider the use of GCV for two dimensional problems. It would, therefore, be interesting to use the GCV approach for the more challenging problems considered in [9]. Further, future work should look at applying the approach when used with different boundary conditions or for restoring color images with independent color channels. In summary, this presented research has made a valuable contribution to the automated improvement of contaminated image data sets without knowledge of true solutions.

Acknowledgments. I thank Dr. Rosemary Renaut for leading the research project. I also thank Miandra Ellis for her help. I would like to acknowledge Aracely Delgadillo for her involvement in the research project and thank her efforts in conducting experiments and conclusions. I am appreciative of the comments of the referees which assisted with the revision of this manuscript. The author would like to acknowledge the Western Alliance to Expand Student Opportunities (WAESO), the Louis Stokes Alliance for Minority Participation (LSAMP) (Cooperative Agreement No. HRD-1619524), and the National Science Foundation (NSF)(award number DMS-1757663) for providing student support. Any opinions, findings, conclusions, or recommendations that are presented in this document are solely those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

REFERENCES

[1] R. C. Aster, B. Borchers, and C. H. Thurber. *Parameter Estimation and Inverse Problems*. Elsevier, Amsterdam, 2rd edition, 2013.
[2] G. H. Golub, M. Heath, and G. Wahba. Generalized cross-validation as a method for choosing a good ridge parameter. *Technometrics*, 21(2):215–223, 1979.
[3] G. H. Golub and C. F. Van Loan. *Matrix Computations*. Johns Hopkins Studies in the Mathematical Sciences. Johns Hopkins University Press, 2013.
[4] P. C. Hansen. The truncated SVD as a method for regularization. *BIT Numerical Mathematics*, 27(4):534–553, Dec 1987.
[5] P. C. Hansen. *Rank-Deficient and Discrete Ill-Posed Problems*. Society for Industrial and Applied Mathematics, Philadelphia, 1998.
[6] P. C. Hansen. *Discrete Inverse Problems*. Society for Industrial and Applied Mathematics, Philadelphia, 2010.
[7] P. C. Hansen, M. E. Kilmer, and R. H. Kjeldsen. Exploiting residual information in the parameter choice for discrete ill-posed problems. *BIT*, 46(1):41–59, 2006.
[8] P. C. Hansen, J. G. Nagy, and D. P. O'Leary. *Deblurring Images*. Society for Industrial and Applied Mathematics, Philadelphia, 2006.
[9] J. Chung, M. Chung, and D. P. O'Leary. *Designing Optimal Spectral Filters For Inverse Problems*. SIAM Journal on Scientific Computing 33.6 (2011): 3132- 3152.
[10] E. Haber, and L. Tenorio. *Learning regularization functionals—a supervised training approach.* Inverse Problems 19.3 (2003): 611.
[11] NASA and JPL-California Institute of Technology. Photojournal: Mercury, 2016. https://photojournal.jpl.nasa.gov/targetFamily/Mercury.
[12] N.A. Photojournal: MATLAB Demo Images, N.D. https://www.bogotobogo.com/Matlab/images/MATLAB_DEMO_IMAGES/
[13] B. W. Rust and D. P. O'Leary. Residual periodograms for choosing regularization parameters for ill-posed problems. *Inverse Problems*, 24(3):034005, 2008.
[14] Stanford University. Photojournal: Sample Still Images, 2007. https://web.stanford.edu/class/ee398b/samples.htm.
[15] University of Southern California. Photojournal: Volume 3: Miscellaneous, 1981. https://sipi.usc.edu/database/database.php.