Journal of Imaging Science and Technology® 64(6): 060408-1–060408-11, 2020.

Limitations of CNNs for Approximating the Ideal Observer Despite Quantity of Training Data or Depth of Network

Khalid Omer▲
Wyant College of Optical Sciences, The University of Arizona, 1630 E University Blvd, Tucson, AZ 85719
E-mail: komer@email.arizona.edu

Luca Caucci
Department of Medical Imaging, The University of Arizona, 1501 N Campbell Ave, Tucson, AZ 85724

Meredith Kupinski
Wyant College of Optical Sciences, The University of Arizona, 1630 E University Blvd, Tucson, AZ 85719

Abstract. *The performance of a convolutional neural network (CNN) on an image texture detection task as a function of linear image processing and the number of training images is investigated. Performance is quantified by the area under (AUC) the receiver operating characteristic (ROC) curve. The Ideal Observer (IO) maximizes AUC but depends on high-dimensional image likelihoods. In many cases, the CNN performance can approximate the IO performance. This work demonstrates counterexamples where a full-rank linear transform degrades the CNN performance below the IO in the limit of large quantities of training data and network layers. A subsequent linear transform changes the images' correlation structure, improves the AUC, and again demonstrates the CNN dependence on linear processing. Compression strictly decreases or maintains the IO detection performance while compression can increase the CNN performance especially for small quantities of training data. Results indicate an optimal compression ratio for the CNN based on task difficulty, compression method, and number of training images.* © *2020 Society for Imaging Science and Technology.*
[DOI: 10.2352/J.ImagingSci.Technol.2020.64.6.060408]

1. INTRODUCTION AND RELATED WORK

This work compares the detection performance of single and multi-layer CNNs (sCNN/mCNN) as well as state-of-the-art deep layered CNN architectures with the Bayesian ideal observer (IO). Performance is quantified by the area under (AUC) the receiver operating characteristic (ROC) curve. The IO maximizes AUC as well as other detection task figures of merit [1]. The AUC of the IO evaluates system performance and can be utilized as a comparative tool for quantifying observer-model detection performance [2]. For many tasks, the IO can be well approximated by a CNN given an adequate quantity of training data [3].

Reith and Wandell [4] compare the performance of CNN, IO, and support vector machines (SVM) for various image texture types in a binary classification task. When trained on textured patterns, the neural network approached IO performance and outperformed SVM [5]. Reith and Wandell also investigate the performance of two- and one-dimensional randomization of the image texture pixel positions. This transformation of randomization pixel positions is linear and can be calculated by a matrix–vector product, where the matrix is a scrambled version of the identity matrix. Since the IO is invariant to multiplication by a non-singular matrix, the IO remains constant throughout the various degrees of pixel randomization. The neural network performance with two-dimensional (2D) randomization across the image plane is less than SVM but IO. Randomization results in longer range correlation between pixels. When trained on column randomized image textures, the neural network performed better than the SVM but below the IO. For column randomization, correlation between pixels along the rows remained the same, which results in higher network performance as compared to a 2D randomization. Additionally, Zhang et al. [6] reports the training error of neural networks trained under various modifications of the input images and labels. They have shown that 2D randomization of image pixels requires additional training time for the training error to converge.

A related study on adversarial examples showed that additive perturbation can lead to misclassification with high confidence while maintaining the visible appearance of the original image [7]. Including adversarial examples in the training set can avoid misclassification and be used as a regularization technique [8].

In this work, the relationship between linear image transforms and classification performance is investigated on image textures. One of the transformation matrix that is populated by elements randomly sampled from a uniform distribution over [0, 1]. This transformation randomizes the pixel position and weight while the prior work discussed [4, 6] only randomizes position. The linear transformation in this work alters the image appearance and correlation structure unlike the retained visual features in adversarial examples.

▲ IS&T Member.

Received July 13, 2020; accepted for publication Nov. 25, 2020; published online Dec. 24, 2020. Associate Editor: Yaohua Xie.
1062-3701/2020/64(6)/060408/11/$25.00

This work makes three contributions:

(1) The detection performance of IO is invariant to full-rank transforms while the CNN performance can change. This work demonstrates linear transformations that reduce the CNN to near-guessing performance for large quantities of training images and increased network depth,
(2) Utilize another full-rank linear transformation, which depends on the covariance matrix of the classes, to subsequently increase the CNN performance on the same images which were reduced to near-guessing,
(3) Utilize this same linear transformation, which depends on the covariance matrix of the two classes, to increase the CNN performance through compression.

2. MATHEMATICAL BACKGROUND

An image detection task from Gaussian distributed, zero class mean image data allows closed-form analytic expressions for the IO and optimal linear compression. When the covariance matrices of these two classes are unequal, the data is called heteroscedastic and the images have the appearance of different textures. Two image texture classes are simulated on a 64×64 pixel grid with dimensionality $M = 4096$. A given image is denoted by the $M \times 1$ vector $\mathbf{g}$. The covariance matrix of each class is chosen to be circulant with a Gaussian kernel and parameterized by a scalar-valued correlation length, denoted by σ. Changing the correlation length yields image textures of varying spatial textures. According to Geirhos et al., the CNN learning is biased toward image texture as opposed to image shape [9]. The synthetic images used in this article have an analytic expression for the IO since the covariance matrices are exactly known. Real images are not used since they would require estimations of the theoretical IO to compare network performance.

The M-dimensional vector $\mathbf{g}$ will represent the input to the classifier, which will classify this vector as either belonging to the population corresponding to the probability density function (PDF) $pr_1(\mathbf{g})$ or the population corresponding to the PDF $pr_2(\mathbf{g})$. This vector may be a direct image, a reconstructed image, or the raw data being produced by an imaging system. This work utilizes synthetic images of known covariance matrices which has an analytic expression of the IO to be the theoretical performance limit of the detection task.

The IO uses the log-likelihood ratio

$$\lambda(\mathbf{g}) = \ln[\Lambda(\mathbf{g})] = \ln[pr_1(\mathbf{g})] - \ln[pr_2(\mathbf{g})] \tag{1}$$

as a decision variable and compares the result to a threshold [1]. Here, $\Lambda(\mathbf{g})$ is the likelihood ratio $pr_1(\mathbf{g})/pr_2(\mathbf{g})$. If the decision variable is above the threshold, then the data vector is assigned to $pr_1(\mathbf{g})$, and otherwise it is assigned to $pr_2(\mathbf{g})$.

In imaging applications, the dimension M of the vector $\mathbf{g}$ (e.g. the number of pixels) is very large. Assuming for $i = 1, 2$ that $pr_i(\mathbf{g})$ is a Gaussian PDF, with zero mean and covariance matrix $\boldsymbol{K}_i$, the log-likelihood ratio is

$$\lambda(\mathbf{g}) = \mathbf{g}\boldsymbol{K}_2^{-1}\mathbf{g}^\dagger - \mathbf{g}\boldsymbol{K}_1^{-1}\mathbf{g}^\dagger. \tag{2}$$

Here, scale factors and terms which do not depend on $\mathbf{g}$ have been ignored. Implementing the log-likelihood ratio, even with these PDF assumptions, incurs two major challenges. The first challenge is computational; even for Gaussian PDFs two $M \times M$ covariance matrices $\boldsymbol{K}_i$ need to be inverted, which may not be feasible if, for example, the input images contain millions of pixels. The second challenge is that, if the image statistics are estimated from data, which is often the case, a very large number of samples are required for reliable estimates. For example, in the Gaussian case, the number of samples must be greater than or equal to M to achieve invertible estimates for the $\boldsymbol{K}_i$. The number of samples needs to be at least an order of magnitude greater than M to get reliable estimates. This provides a motivation for trying to reduce the dimension of the data vector before implementing the log-likelihood ratio.

The linear data transformation is implemented by an $L \times M$ dimensional matrix $\boldsymbol{T}$ via the equation $\mathbf{t} = \boldsymbol{T}\mathbf{g}$. Using terminology from the perception literature, each row of $\boldsymbol{T}$ is referred to as a channel [10]. In this article, $\mathbf{t}$ will be called a channelized image or a channelized data vector. The number L is the dimension of the channelized data and satisfies $L \ll M$. Channelized data are preferable for mathematical observers since calculating a decision variable usually involves the estimation of parametric likelihoods [11]. In the channelized representation, this estimation can be much more accurate given common constraints on finite training data [12, 13]. Computational needs are also lower because the inverse of a covariance matrix, required for likelihood evaluations, is now $L \times L$ instead of $M \times M$.

The channels are linearly independent when $\boldsymbol{T}$ is a full-rank matrix. Consider zero-mean normal distributions for the channelized data conditional on the two populations

$$pr_i(\mathbf{t}) = \frac{\exp\left[-\frac{1}{2}\mathbf{t}^\dagger(\boldsymbol{T}\boldsymbol{K}_i\boldsymbol{T}^\dagger)^{-1}\mathbf{t}\right]}{\sqrt{(2\pi)^L \det(\boldsymbol{T}\boldsymbol{K}_i\boldsymbol{T}^\dagger)}} \tag{3}$$

for $i = 1, 2$. Note that the covariance of the channelized data are related to the covariance of the image data by $\boldsymbol{C}_i = \boldsymbol{T}\boldsymbol{K}_i\boldsymbol{T}^\dagger$, where $\boldsymbol{C}_i$ is the $L \times L$ covariance matrix of the channelized data.

The covariance matrix eigenanalysis is rarely practical for modern imaging systems since an image is comprised of several million elements. Fukunaga and Koontz were the first to suggest covariance matrix eigendecomposition for detection and classification tasks [14]. With certain eigenspectrum assumptions, the Fukunaga–Koontz transform (FKT) is the low-rank approximation to the optimal classifier for zero mean, heteroscadastic, and normally distributed data [15, 16]. An adaptation of FKT widely used in pattern recognition, is called a tuned basis function (TBF) [17]. When the quantity of training data is limited, linear compression using FKT can increase detection performance of the CNN.

The "curse of dimensionality" is evident from a performance peak at an optimal compression, followed by a decrease [18]. The CNN performance on the linearly compressed images is heavily dependent upon the compression matrix. For IO, full-rank linear transformations do not change detection performance regardless of the specific compression matrix.

This FKT method uses a matrix $\boldsymbol{T}$ to transform the data so that $\boldsymbol{T}(\boldsymbol{K}_1 + \boldsymbol{K}_2)\boldsymbol{T}^\dagger$ equals the identity matrix. This equality guarantees that both covariance matrices of the transformed data will have the same eigenvectors. Furthermore, the sum of the two eigenspectra, when eigenvalues associated with the same eigenvector are added, is equal to one. Consequently, the FKT can be helpful for detection because it makes the *strongest* eigenvectors of one class the *weakest* eigenvectors of the other.

Principal component analysis (PCA) is a commonly used method for dimensionality reduction and is similar to FKT as both methods utilize eigenspectrum analysis to construct a transformation matrix [19]. However, the PCA does not consider a two-class problem and instead diagonalizes a single object class covariance matrix.

3. PERFORMANCE EVALUATION METHODS

Formation of the CNN architecture is done through the TensorFlow Keras API in Python. Tensor operations are executed utilizing 4 Nvidia 1080Ti graphic cards. Network construction is a heuristic model based on prior knowledge regarding the image ensemble. For example, kernel size selection can be estimated based on the correlation length. As the correlation length increases, the spatial variability in the image decreases, allowing for a larger kernel size to be used without affecting detection performance. The diagram of sCNN and mCNN architectures can be found in Figures 1 and 2, respectively. Architecture design of sCNN and mCNN consist of one and four convolutional blocks, respectively. The convolutional block for sCNN and the initial block for mCNN has a convolutional layer containing 32 non-zero padded convolutional kernels of size 2×2 of stride 1. Subsequent convolution blocks for mCNN contain 12 non-zero padded convolutional kernels of size 4×4. The convolutional blocks for mCNN and sCNN utilize batch normalization and Leaky ReLU for regularization [20]. A flattened layer occurs after the last convolutional block and is the largest contribution of network parameters, with sCNN having more parameters than mCNN. Both networks have an additional dense layer of size 128×1 which also has batch normalization and Leaky ReLU. The initial convolution block and the last dense layer for both mCNN and sCNN have dropout set to 0.75 as an additional degree of regularization. All architectures utilize binary cross-entropy as the loss function. For the mCNN, dropout on the first and last layer are zero for $\sigma_1 = 1.0$ and $\sigma_2 = 0.4$ pixels. These large differences between correlation length require adjustments to optimize the network architecture. Dropout is set to zero to increase AUC for this long correlation length difference. Batch size is approximately 10% of the training set but is slightly varied to increase performance depending on task difficulty and number of training images.

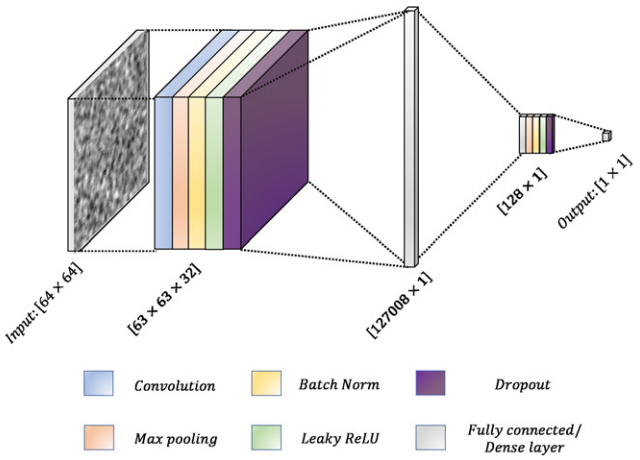


Figure 1. Architecture design for the single-layer CNN (sCNN). The non-zero padded convolution block and last dense layer includes several regularization techniques to avoid overfitting. sCNN has more parameters than mCNN primarily due to the large number of connections between the convolution block and the flattened layer.

When trained, the channelized array $(L \times 1)$ is reshaped into a $(\sqrt{L} \times \sqrt{L})$ matrix to allow for 2D convolution to occur. L varies from 16 to 4096, where L must be integer squares. The value of 16 was chosen to be the minimum L value in order to preserve the sCNN throughout the experiment, values smaller than 16 would require the kernel size of 2×2 to become a 1×1 to allow for the minimum stride $= 1$.

The number of trainable parameters for the sCNN is equal to

$$N_s = 737 + 4096\left(\sqrt{L} - 2\right)^2, \tag{4}$$

and for mCNN equals

$$N_m = 3533 + 1536\left(\sqrt{L} - 14\right)^2. \tag{5}$$

When the image is linearly transformed, L is the number of linear channels. Equations (4) and (5) are also valid for unprocessed images and full-rank transformations. Here, $L = M$ represents the number of pixels of the unprocessed image and the total number of linear channels for a full-rank transformation. Image compression decreases the number of parameters utilized in the CNN. For the sCNN, L values between 16 to 4096 have a trainable parameter range between $N_s = 803,553$ to $N_s = 15,745,761$. The mCNN is trained only when $L = M$, and uses $N_m = 3,843,533$ parameters. When $L > 44$, sCNN has more trainable parameters than mCNN. The sCNN has more parameters than mCNN due to the flattening on the last convolutional layer followed by a fully connected/dense layer. The number of trainable parameters in the CNN is related to the product of: number of parameters in last layer, number of kernels, and number of fully connected parameters.

Additionally, the binary classification performance analysis includes state-of-the-art CNN architectures VGG-

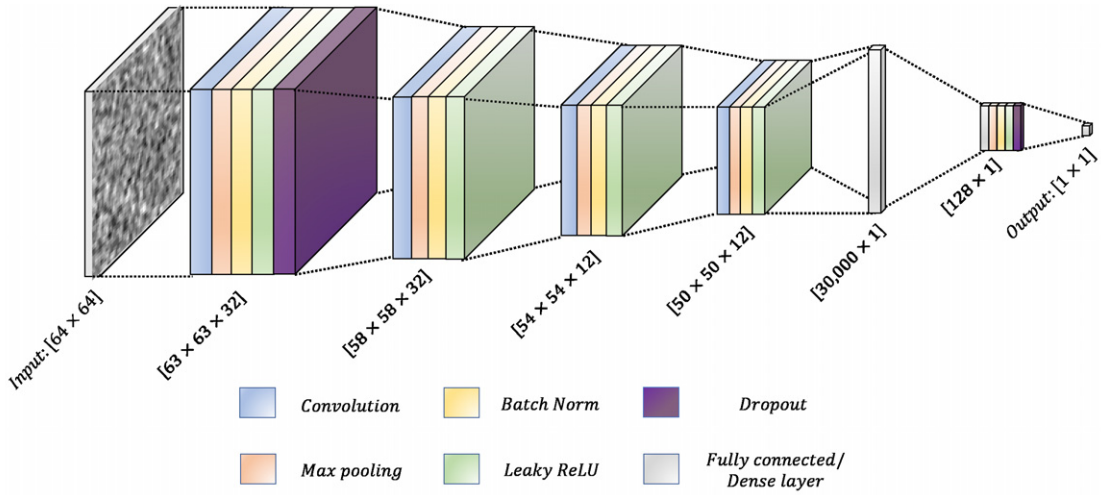


Figure 2. Architecture design for the multi-layer CNN (mCNN). The first non-zero padded convolution block and last dense layer includes several regularization techniques to avoid overfitting. mCNN has less parameters compared to sCNN due to the smaller final convolution block and flattened layer.

16 [21], ResNet-50 [22], and DenseNet-121 [23] on the channelized images $\tilde{\mathbf{g}}$. The selection of these networks were based on the increase in complexity as compared to sCNN and mCNN with varying structural design.

3.1 *Generating Images*

A simple covariance model is chosen to provide an analytic solution for the IO and FKT. Two classes of images are simulated on a 64×64 pixel grid; $M = 64^2$. The covariance matrix of each class is chosen to be circulant with a Gaussian kernel and parameterized by a correlation length as in

$$[\boldsymbol{K}_i]_{\mathbf{n},\mathbf{m}} = \exp\left(\frac{-((n_x - m_x)\text{mod}64)^2 - ((n_y - m_y)\text{mod}64)^2}{2\sigma_i^2}\right), \quad (6)$$

where σ_i is the correlation length of the i^{th} class in units of pixels. The 2D vectors $\mathbf{n}$ and $\mathbf{m}$ are pixel indices. A short correlation length yield images of higher frequency content than images of longer correlation length; see Figure 3.

3.2 *Channelized Images: Linear Data Transformation*

Multiplying an $L \times M$ matrix and the $M \times 1$ vector $\mathbf{g}$ reduces the dimension of the data to $L \times 1$ and the compression ratio is L/M. The matrix is fully populated of rank L. In this work the IO and CNN detection performance are compared for three linear data reduction methods: (1) FKT compression denoted by the matrix $\boldsymbol{T}$, (2) linear compression without any prior knowledge denoted by $\mathbf{R}$, and (3) FKT compression subsequent to uncompressed $\mathbf{R}$ (i.e. $L = M$) denoted by $\tilde{\boldsymbol{T}}$. FKT is the optimal linear compression method given the constraint that images are Gaussian distributed, zero class mean, and heteroscadastic. On the other extreme, linear compression with no prior knowledge is implemented by populating a matrix with samples from a uniform distribution. The channelized data are still zero-mean Gaussian distributed with covariance matrices: $\boldsymbol{T}\boldsymbol{K}_i\boldsymbol{T}^\dagger$, $\mathbf{R}\boldsymbol{K}_i\mathbf{R}^\dagger$, or $\tilde{\boldsymbol{T}}\mathbf{R}\boldsymbol{K}_i\mathbf{R}^\dagger\tilde{\boldsymbol{T}}^\dagger$.

For $L = M$, the matrices $\boldsymbol{T}$, $\mathbf{R}$, and $\tilde{\boldsymbol{T}}$ are full rank, invertible, and $\mathbf{g}$ has not been compressed. The data-processing inequality guarantees that information to discriminate the classes has not been increased by post-processing [24]. For $L = M$ these operators are invertible and therefore information has not been lost when the images are not compressed. Figures 3, 4, 5, and 6 are example images with two different correlation lengths and uncompressed channelized images using matrices $\boldsymbol{T}$, $\mathbf{R}$, and $\tilde{\boldsymbol{T}}$, respectively.

The FKT matrix is populated by L eigenvectors of $\boldsymbol{K}_2^{-1}\boldsymbol{K}_1$ with corresponding eigenvalues κ_l. The IO AUC can be maximized when these eigenvectors are chosen to have the L largest values of $\kappa_l + \kappa_l^{-1}$ [16]. Then the compressed data is

$$\mathbf{t} = \boldsymbol{T}\mathbf{g}, \quad (7)$$

where $\boldsymbol{T}$ is an $L \times M$ matrix and FKT makes the *strongest* eigenvectors of one class and the *weakest* eigenvectors of the other. The FKT ensures that $\boldsymbol{T}\mathbf{K_1}\mathbf{T}^\dagger + \boldsymbol{T}\mathbf{K_2}\mathbf{T}^\dagger = \mathbf{I}$, where $\mathbf{I}$ is the identity matrix. Consequently, when the variance in a given pixel is low for one class, it is high for the other. This variance difference is visible in samples of $\mathbf{t}$ for $L = M$ case given in Fig. 4.

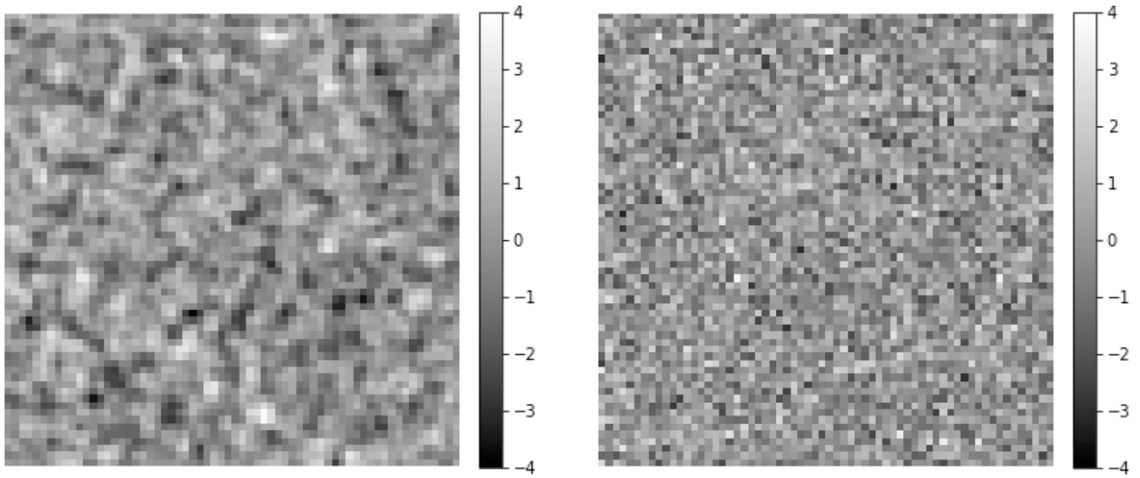
Without any prior knowledge of $pr(\mathbf{g})$, compression can be implemented by

$$\mathbf{r} = \mathbf{R}\mathbf{g}. \quad (8)$$

Here, $\mathbf{R}$ is an $L \times M$ matrix populated by elements sampled from a uniform distribution over [0, 1]. For the special case of $L = M$, no compression takes place which is denoted by $\tilde{\mathbf{g}} = \mathbf{R}\mathbf{g}$. Uncompressed channelized images $\tilde{\mathbf{g}}$ are shown in Fig. 5.

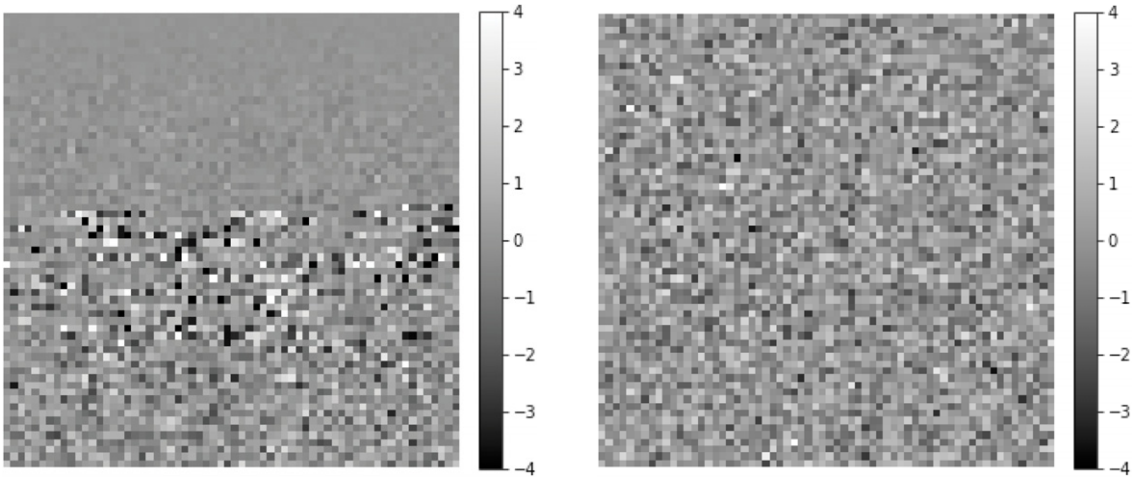
The FKT applied to the $\tilde{\mathbf{g}}$ image set is denoted by

$$\tilde{\mathbf{t}} = \tilde{\boldsymbol{T}}\tilde{\mathbf{g}}. \quad (9)$$



(a) Sample from $pr_1(\mathbf{g})$ (b) Sample from $pr_2(\mathbf{g})$

Figure 3. Example images of g for classes: (a) $\sigma_1 = 1.00$ and (b) $\sigma_2 = 0.40$ pixels. The visual difference between the two classes is distinct; the $\sigma_1 = 1.00$ image spatial structure varies less than the $\sigma_1 = 0.40$ image.



(a) Sample from $pr_1(\mathbf{t})$ (b) Sample from $pr_2(\mathbf{t})$

Figure 4. Example channelized images t from FKT for classes: (a) $\sigma_1 = 1.00$ and (b) $\sigma_2 = 0.40$ pixels. The FKT constrains pixels of high variance in one class to have low variance in the other. Here $\mathsf{t} = \boldsymbol{T}\mathsf{g}$ and example images g are shown in Fig. 3. Visible differences between the classes are present in both g and t but the correlation structure is different.

Here, $\mathbf{R}$ is now full rank of size $M \times M$ and $\tilde{\boldsymbol{T}}[\mathbf{R}\boldsymbol{K}_1\mathbf{R}^\dagger + \mathbf{R}\boldsymbol{K}_2\mathbf{R}^\dagger]\tilde{\boldsymbol{T}}^\dagger = \mathbf{I}$ is the FKT constraint. Compression is achieved via the $L \times M$ matrix $\tilde{\boldsymbol{T}}$ when $L < M$. Once again, when the variance in a given pixel is low for one class it is high for another. This variance difference is visible in Fig. 6.

3.3 *Estimating Observer Performance*

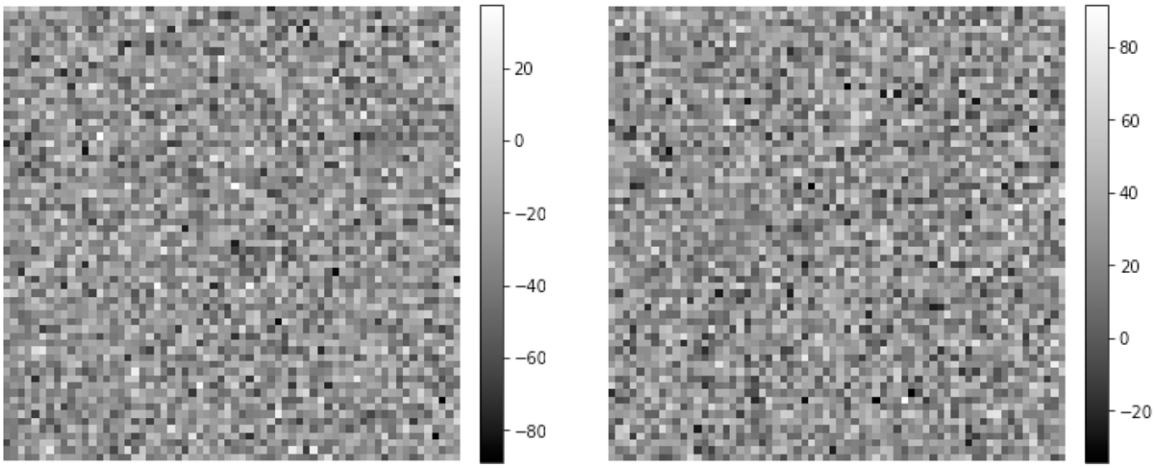
The last node of the network consists of a sigmoid activation function. This sigmoid produces an estimate of the posteriors $pr(i|\mathbf{g})$ [25]. For the case of equal prevalence $pr(i = 1) = pr(i = 2)$, the posteriors are

$$pr(i = 1|\mathbf{g}) = \frac{1}{1 + \Lambda(\mathbf{g})}, \tag{10}$$

where $\Lambda(\mathbf{g})$ is the likelihood ratio defined in Eq. (1) and $pr(i = 2|\mathbf{g}) = 1 - pr(i = 1|\mathbf{g})$. The posteriors are evaluated on a set of testing images to generate an ROC curve; the AUC is estimated by trapezoidal integration. The average values of the AUC estimates did not change for more than 20,000 testing images (10,000 $pr(i = 1|\mathbf{g})$ and 10,000 $pr(i = 2|\mathbf{g})$), so this quantity was selected as an asymptotic estimate.

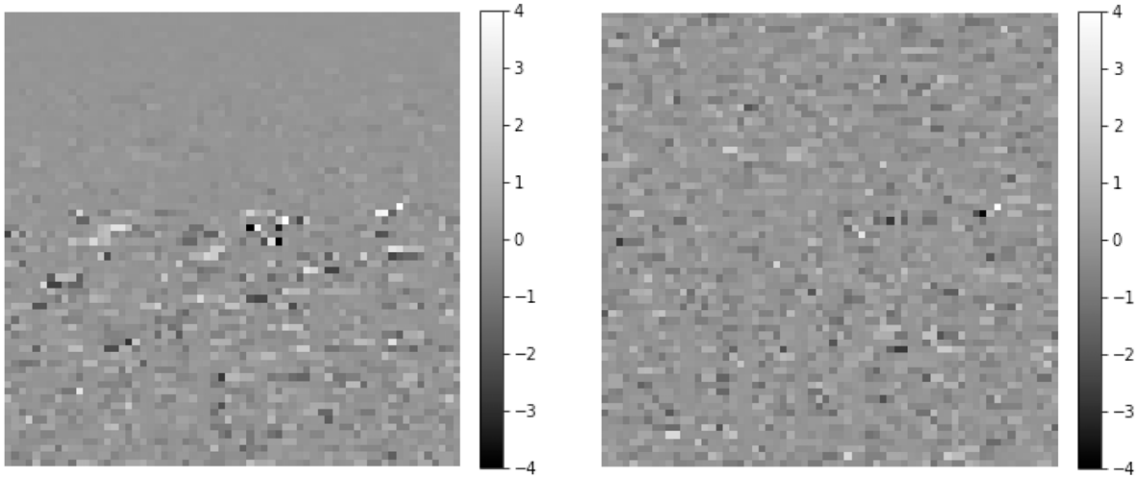
4. RESULTS

These results compare the performance of CNN architectures to IO using various forms of image sets. The detection task performance are computed from four image data sets: original image data $\mathbf{g}$, image data compressed by FKT $\mathbf{t}$, linear compression with no prior knowledge $\mathbf{r}$, and FKT compression subsequent to a non-compressive (i.e. $L = M$) invertible linear transform denoted by $\tilde{\mathbf{t}}$. The detection performance for a specific model and type of data set is



(a) Sample from $pr_1(\tilde{\mathbf{g}})$ (b) Sample from $pr_2(\tilde{\mathbf{g}})$

Figure 5. Example channelized images $\tilde{\mathbf{g}}$ from full-rank linear transform for classes: (a) $\sigma_1 = 1.00$ pixels and (b) $\sigma_2 = 0.40$ pixels. Here $\tilde{\mathbf{g}} = \mathsf{R}\mathbf{g}$ and images $\mathbf{g}$ are shown in Fig. 3. There is no visible difference between the two classes after this full-rank linear transform R.



(a) Sample from $pr_1(\tilde{\mathbf{t}})$ (b) Sample from $pr_2(\tilde{\mathbf{t}})$

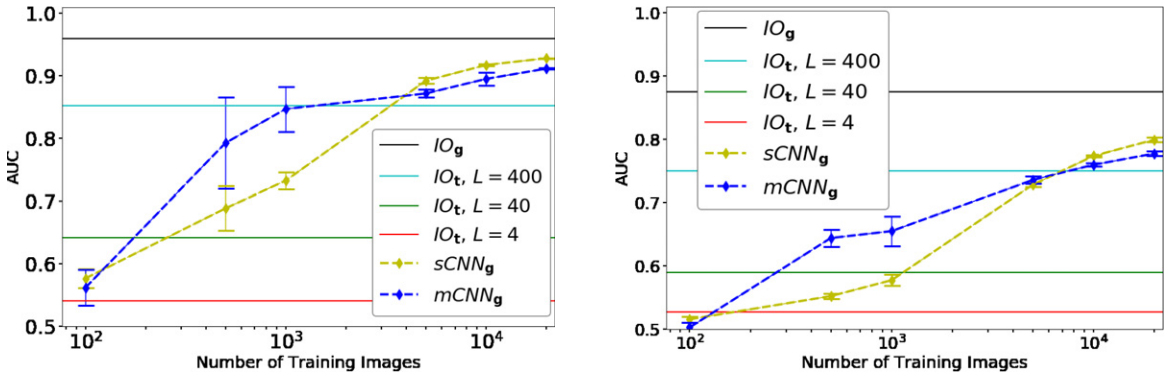
Figure 6. Example channelized images $\tilde{\mathbf{t}}$ from FKT for classes: (a) $\sigma_1 = 1.00$ pixels and (b) $\sigma_2 = 0.40$ pixels. Here $\tilde{\mathbf{t}} = \tilde{\boldsymbol{T}}\tilde{\mathbf{g}}$ and channelized images $\tilde{\mathbf{g}}$ are shown in Fig. 5. The visible differences between the images $\mathbf{g}$ (see Fig. 3) which was lost in the uncompressed channelized images $\tilde{\mathbf{g}}$ (see Fig. 5) has been restored by FKT channelized images shown in this figure.

denoted, for example by $\text{sCNN}_{\mathbf{t}}$ for a single-layer CNN on an FKT channelized image set.

In Figure 7, the AUCs are compared for: $\text{IO}_{\mathbf{g}}$, $\text{IO}_{\mathbf{t}}$, $\text{sCNN}_{\mathbf{g}}$, and $\text{mCNN}_{\mathbf{g}}$ as a function of number of training images. An easy task is evaluated in Fig. 7(a) and a more difficult task, where the correlation lengths are more similar, is reported in Fig. 7(b). The IO does not depend on the number of training images because the true value of the covariance matrices are used to estimate AUC. The AUC for $\text{IO}_{\mathbf{t}}$ is compared for different numbers of FKT channels: $L = 4$, 40, and 400. As expected, the AUC of $\text{IO}_{\mathbf{t}}$ increases with the number of channels. At a given L, the AUC of $\text{IO}_{\mathbf{t}}$ is lower for the more difficult task in Fig. 7(b) as compared to Fig. 7(a).

The gap between the AUCs of $\text{IO}_{\mathbf{t}}$ for the easy and more difficult task closes as the number of channels decreases. At maximum compression, for $\text{IO}_{\mathbf{t}} : L = 4$ AUC $= 0.54$ for the easy task and AUC $= 0.52$ for the more difficult task; see Figs. 7(a) and 7(b), respectively. Notably, for 100 training images $\text{mCNN}_{\mathbf{g}}$ and $\text{sCNN}_{\mathbf{g}}$ are outperformed by $\text{IO}_{\mathbf{t}} : L = 40$ for the easy task and $\text{IO}_{\mathbf{t}} : L = 4$ for the more difficult task. For a given L, the CNN requires a greater quantity of training data to meet $\text{IO}_{\mathbf{t}}$ as the task difficulty increases.

The number of trainable parameters in the sCNN and mCNN depends on the image size; see Eqs. (4) and (5). Both the sCNN and mCNN have the same hyperparameters in the first block. The number of trainable parameters for the $\text{sCNN}_{\mathbf{g}}$ and $\text{mCNN}_{\mathbf{g}}$ are: $N_s = 15,745,761$ and $N_m = 3,843,533$, respectively. Detection performance depends on both the number of independent samples and the number of trainable network parameters. The number of independent



(a) $\sigma_1 = 0.38$ and $\sigma_2 = 0.30$ pixels (b) $\sigma_1 = 0.38$ and $\sigma_2 = 0.34$ pixels

Figure 7. AUC dependence on quantity of training images for: (a) an easy task, where $IO_g AUC = 0.98$ and (b) a task of moderate difficulty, where $IO_g AUC = 0.88$. Here $\sigma_1 = 0.38$ pixels and in (a) $\sigma_2 = 0.30$ pixels and (b) $\sigma_2 = 0.34$ pixels. The upper bound on AUC, for a given number of channels (L), is given by IO_t, which is always lower for the more difficult task in (b). Here, the CNN performance strictly increases with quantity of training data. $mCNN_g$ outperforms $sCNN_g$ for 500 and 1,000 training images. For larger quantities of training images $sCNN_g$ performance is greater.

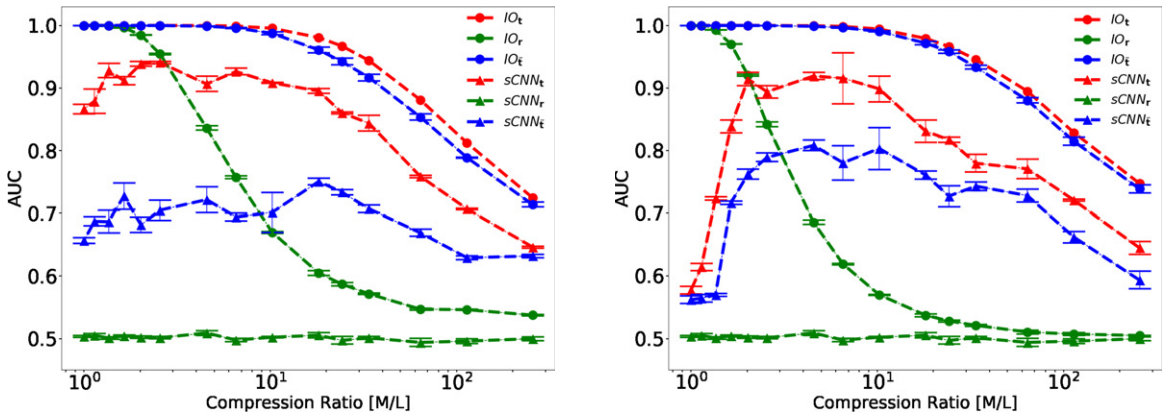
samples is the product of the number of training images and the size of an image. Above 1,000 training images, the number of independent samples exceeds the number of trainable parameters in $mCNN_g$, but is less than the number of trainable parameters in $sCNN_g$. Also above 1,000 training images, the AUC of $sCNN_g$ meets or exceeds the AUC of $mCNN_g$; see Fig. 7. At and below 1,000 training images $mCNN_g$ outperforms or equals $sCNN_g$ where the reduction of trainable parameters increases detection performance.

When the number of independent samples increases to $2,048,000$ and $4,096,000$ for 500 and 1000 training images, respectively, the mean AUC increases by 0.05 to 0.85 based on task difficulty and number of training images; see Figs. 7(a) and 7(b) for respective performance increases. The AUC of the sCNN is about 0.10 less than mCNN in this region. The task is now better estimated by the mCNN due to the magnitude of the number of independent samples and training images being equal.

When the number of training images exceeds 5000 training images, the number of independent samples is larger than $20,480,000$. The number of independent sample count is now in the same order of magnitude as the number of sCNN parameters. Here, the sCNN AUC is greater than mCNN by .02 in Figs. 7(a) and 7(b). In the limit of increased training images, the AUC difference between $sCNN_g$ and $mCNN_g$ decreases. The difference in the performance of the two networks decreases as the number of independent samples outweighs network parameters.

Figure 8 reports AUC as a function of compression ratio M/L for an easy task of: (a) short correlation lengths and (b) longer correlation lengths. The AUC mean and standard deviation are estimated from 100 training images. The IO performance is denoted by circle markers. Triangle markers denote performance of sCNN. The colors denote linear processing methods: red ($\mathbf{t}$), green ($\mathbf{r}$), and blue ($\tilde{\mathbf{t}}$) defined in Eqs. (7), (8), and (9), respectively. The IO is invariant to multiplication by a non-singular matrix. Therefore, IO AUC is equal for all three full-rank compression matrices when $L = M$. As the compression ratio increases, the IO AUC decreases at different rates for each compression matrix. Compression from FKT (red) is optimal linear compression for this zero-mean Gaussian heteroscadastic data. The AUC $= 1.0$ for the FKT compressed image (denoted by $IO_{\mathbf{t}}$ and red circles in Fig. 8) until $M/L \approx 10$, where the AUC decreases monotonically with compression ratio. The performance trend of $sCNN_{\mathbf{t}}$ matches $IO_{\mathbf{t}}$ for large compression ratios. However, given small compression $sCNN_{\mathbf{t}}$ AUC is convex and performance peaks at $M/L \approx 6$. Further compression degrades the detection performance due to the trade-off between information content and estimation from finite sample statistics [18]. This performance peak is less pronounced in Fig. 8(a) as compared to Fig. 8(b). The correlation lengths are shorter and therefore image data contains higher frequency content in Fig. 8(a) compared to Fig. 8(b). Both detection tasks are of similar difficulty level given by IO AUC $= 1.0$ for uncompressed images. However, even for uncompressed images the $sCNN_{\mathbf{t}}$ AUC for these tasks deviates; see Fig. 8(b) compared to Fig. 8(a). The longer correlation length structure is more challenging for the sCNN to learn given a low quantity of training data. This indicates that although IO AUC is a metric for detection task difficulty, it is not necessarily a predictor of the quantity of training images necessary for an IO performance approximation. The correlation structure of the data also needs to be considered.

The IO AUC decreases most rapidly when no prior knowledge is used in the compression matrix $IO_{\mathbf{r}}$ (green circles in Fig. 8). Compression is implemented by populating an $L \times M$ matrix with uniformly distributed random samples; see Eq. (8), even at low compression ratios, where $IO_{\mathbf{r}}$ AUC $= 1.0$ the $sCNN_{\mathbf{r}}$ AUC ≈ 0.50. This result is the largest disparity between CNN and IO reported in this work. The non-singular linear transform $\tilde{\mathbf{g}} = \mathbf{R}\mathbf{g}$ does not change the information content of the image data since the original image data can be recovered by $\mathbf{g} = \mathbf{R}^{-1}\tilde{\mathbf{g}}$. However,



(a) $\sigma_1 = 0.45$, $\sigma_2 = 0.35$ pixels (b) $\sigma_1 = 1.00$, $\sigma_2 = 0.98$ pixels

Figure 8. AUC mean and standard deviation are estimated from 100 training images as a function of compression ratio for an easy task of: (a) short correlation lengths and (b) longer correlation lengths. IO performance is denoted by circle markers. Triangle markers denote performance of sCNN. The colors denote linear processing methods: red (t), green (r), and blue (t̃) defined in Eqs. (7), (8), and (9), respectively. The AUC improvement with compression of $\text{sCNN}_{\mathbf{t}}$ and $\text{sCNN}_{\tilde{\mathbf{t}}}$ is less appreciable for the shorter correlation length images (a) compared to longer correlation length images (b).

the sCNN trained on $\tilde{\mathbf{g}}$ no longer recognizes the correlation differences between the two classes.

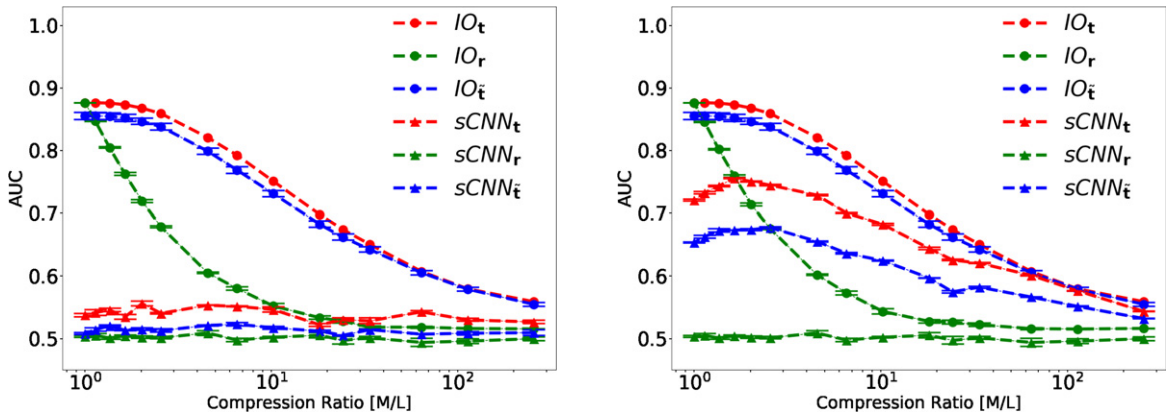
The correlation structure change due to matrix multiplication by $\mathbf{R}$ can be changed again by an FKT compression matrix applied to $\tilde{\mathbf{g}}$ images; see Eq. (9). This is denoted by $\text{IO}_{\tilde{\mathbf{t}}}$ (blue circles in Fig. 8) and closely matches the curve for $\text{IO}_{\mathbf{t}}$ (red circles in Fig. 8), with a maximum AUC difference of less than 0.01 which begins to occur when $M/L = 10$. The AUC of $\text{sCNN}_{\mathbf{t}}$ and $\text{sCNN}_{\tilde{\mathbf{t}}}$ differs by $\approx$0.10 at $M/L = 1$ for the shorter correlation length images in Fig. 8(a). For the longer correlation length images in Fig. 8(b), AUC of $\text{sCNN}_{\mathbf{t}}$ and $\text{sCNN}_{\tilde{\mathbf{t}}}$ differs by $\approx$0.01 at $M/L = 1$. Therefore, the effectiveness of FKT to restore a correlation structure which sCNN can recognize depends on the correlation structure of the original data set. For the shorter correlation length case, an increase in FKT compression on $\tilde{\mathbf{g}}$ results in AUC fluctuating near 0.7 and peaking at $M/L \approx 200$; see $\text{sCNN}_{\tilde{\mathbf{t}}}$ denoted by blue triangles in Fig. 8(a). For the longer correlation case, the AUC of $\text{sCNN}_{\tilde{\mathbf{t}}}$ is more similar to $\text{sCNN}_{\mathbf{t}}$ including the performance peak observed near $M/L = 10$.

Figure 9 shows the AUC versus compression ratio for (a) $N_{\text{train}} = 100$ and (b) $N_{\text{train}} = 10,000$ training images. The IO AUC decreased in Fig. 9 as compared to Fig. 8 because the correlation lengths are both shorter. Due to the shorter correlation lengths, IO and CNN detection task difficulty increases. For $N_{\text{train}} = 100$; see Fig. 9(a), $\text{sCNN}_{\mathbf{t}}$ peaks at $AUC \approx 0.55$. The $\text{sCNN}_{\mathbf{t}}$ AUC using additional training images in Fig. 9(b) increases to AUC = 0.72. This result proves to agree with the assessments made in Fig. 7 that an increased performance can be achieved with additional training images. Fig. 9 shows that the validity of this assessment includes compressed data sets. Additional training images increases the sensitivity to compression as seen in Fig. 9(b).

In Figure 10, the AUC of IO and mCNN are compared for an extremely easy detection task ($\sigma_1 = 1.00$, $\sigma_2 = 0.40$ in magenta) and a more difficult detection task ($\sigma_1 = 0.38$, $\sigma_2 = 0.34$ in green). See Fig. 3 for example images of the extremely easy task. The mCNN performance is reported as a function of number of training images for $\mathbf{g}$ and $\tilde{\mathbf{g}} = \mathbf{Rg}$. The IO performance does not depend on number of training images or non-singular linear transform. For both the extremely easy and more difficult detection task, $\text{mCNN}_{\mathbf{g}}$ improves monotonically with increased quantity of training images. Even if the correlation length differences are small, the CNN is learning to distinguish the images.

For the easier task $\text{mCNN}_{\tilde{\mathbf{g}}}$ AUC (magenta circles in Fig. 10) improves monotonically with increased quantity of training images. For 1,000 and 10,000 training images, AUC $\approx$ 0.65, while peaking at an AUC = 0.782 for 1,000,000 training images. For the more difficult task $\text{mCNN}_{\tilde{\mathbf{g}}}$ AUC $\approx$ 0.5 (green circles in Fig. 10) and does not improve with increased quantity of training images. This result indicates that the non-singular linear transform $\tilde{\mathbf{g}} = \mathbf{Rg}$ can render the correlation differences undetectable to the CNN unless the original images $\mathbf{g}$ differ in correlation length by a sufficiently large magnitude.

Table I shows the AUC performance trained on $\tilde{\mathbf{g}}$ when $\sigma_1 = 1.0$ and $\sigma_2 = 0.40$ using common state-of-the-art neural network architectures: VGG-16, ResNet-50, and DenseNet-121 which contains 16, 50, and 121 network layers, respectively. The CNN architectures in Table I contain more convolutional layers than sCNN/mCNN while ResNet-50 and DenseNet-121 have complexities in their architectures not found in mCNN/sCNN. All architectures excluding VGG-16 include regularization techniques such as batch normalization and dropout. VGG-16, ResNet-50, and DenseNet-121 have $14.7M$, $23.5M$, and $6.9M$ training parameters, respectively. During training, usage of early stopping was incorporated to avoid overfitting to the training set and minimize the generalization error. The VGG-16 AUC = 0.737 and AUC = 0.739 for 1000 and 10,000 images. For mCNN, AUC $\approx$ 0.65 for 1000 and 10,000 training images, as seen in Fig. 10. For 1000 training images



(a) 100 Training Images (b) 10,000 Training Images

Figure 9. AUC as a function of data compression for a task of moderate difficulty ($\sigma_1 = 0.38$, $\sigma_2 = 0.34$) and varying quantities of training images: (a) $N_{\text{train}} = 100$ and (b) $N_{\text{train}} = 10{,}000$. IO performance is denoted by circle markers and does not depend on number of training images. Triangle markers denote performance of sCNN. The colors denote linear processing methods: red (t), green (r), and blue ($\tilde{t}$) defined in Eqs. (7), (8), and (9), respectively.

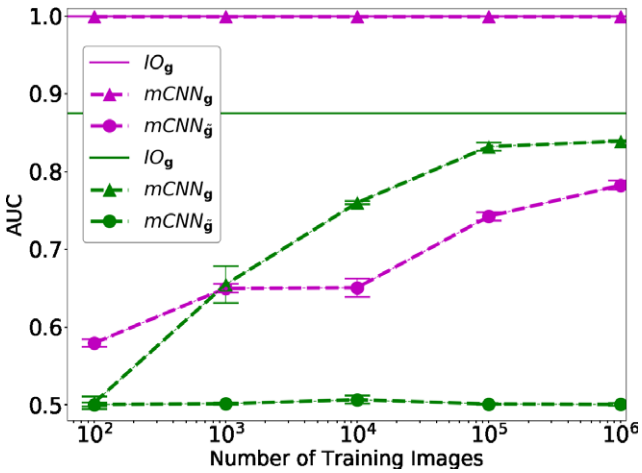


Figure 10. AUC as a function of number of training images for two classification tasks of varying difficulty. In magenta $IO_g = 1.0$ for $\sigma_1 = 1.00$, $\sigma_2 = 0.40$ (see Fig. 3 for example images) and in green $IO_g = 0.87$ for $\sigma_1 = 0.38$, $\sigma_2 = 0.34$. Performance on image data $\mathbf{g}$ is denoted by circle markers, performance on the linear transformed images $\tilde{\mathbf{g}} = \mathbf{R}\mathbf{g}$ is denoted by triangles. For the easier task $mCNN_{\tilde{g}}$ AUC (magenta circles) improves with increased quantity of training images.

VGG-16 has a larger AUC compared to ResNet-50 and only slightly larger for DenseNet-121. For 10,000 training images the AUC differences between architectures are within one standard deviation. Deeper and more complex networks such as ResNet-50 and DenseNet-121 are often attributed to learning higher level features. The complexities in ResNet-50 and DenseNet-121 consist of regularization techniques to avoid vanishing gradients that saturate and decrease network performance. Table I suggests these techniques do not contribute in increasing the performance beyond the performance of VGG-16. The cases shown in Table I still perform below the IO AUC = 1.0 indicating that a simple task for the IO is difficult even for the state-of-the-art CNN architectures.

Table I. For various state-of-the-art CNN architectures, the AUC mean and standard deviation of the detection task for $\tilde{\mathbf{g}}$ images (see Fig. 5); here $\sigma_1 = 1.0$ and $\sigma_2 = 0.40$ (see Eq. (6)) and IO AUC is 1.0. For 1000 training images, VGG-16 has a larger AUC compared to ResNet-50 and only slightly larger for DenseNet-121. For 10,000 training images the AUC differences are within one standard deviation.

CNN architecture	#of training images	AUC
VGG-16	1,000	0.737 ± 0.001
	10,000	0.739 ± 0.002
ResNet-50	1,000	0.703 ± 0.003
	10,000	0.735 ± 0.002
DenseNet-121	1,000	0.731 ± 0.004
	10,000	0.724 ± 0.015

5. CONCLUSION

In many imaging applications, CNNs performance approximates the IO detection performance given enough training data; see Figs. 7 and 10. The quantity of image data required for this approximation depends strongly on the difficulty of the detection task (i.e. the similarity of the image PDFs) and the number of measurements in a single image (M). This work demonstrates that certain linear transformations of the image data challenge this approximation.

The results for AUC versus number of training images in Figs. 7(a) and 7(b) show the AUC of sCNN, mCNN, and IO as a function of number of training images utilizing the unaltered image set $\mathbf{g}$. For 500 and 1,000 training images, the mCNN outperforms sCNN. The mCNN uses additional layers in the network architecture resulting in fewer parameters than sCNN. This result goes against some findings suggesting neural networks benefit from overparameterization [26]. The parameters in the deeper layers utilize additional convolutions that provide beneficial results for small quantities of training images [27]. With increasing quantities of training images, the difference in

performance of mCNN and sCNN decreases as learning is also strongly related to the number of training images in the data set, asymptotically approaching the IO performance [3]. Additionally, in Table I, the added layers and regularization techniques using ResNet-50 and DenseNet-121 do not increase performance compared to the shallower network VGG-16 for this detection task.

For small training sets, linear transformation in the form of compression can increase or decrease the CNN performance; see Figs. 8 and 9. Good compression makes use of prior information to preserve detection task performance. The balance between the compression ratio and sacrifice of information is determined by the task difficulty and number of training data. The work done in this article further analyzes the case where the data set is small and performance of the sCNN is increased by utilizing the FKT compared to no compression method. The FKT is the optimal low-rank approximation to the optimal classifier for zero mean, heteroscadastic, and normally distributed data. Therefore, the FKT provides the optimal data reduction method for the presented image textures. The convex shaped $\text{sCNN}_{\mathbf{t}}$ from Fig. 8 demonstrates an optimal compression ratio with performance peaking at $M/L \approx 5$. Increasing compression degrades performance of the sCNN as task difficulty increases (IO AUC decreases), known as the "curse of dimensionality" [18]. The FKT requires the covariance matrix of the two classes to meet the condition where $\boldsymbol{T}(\boldsymbol{K}_1 + \boldsymbol{K}_2)\boldsymbol{T}^\dagger$ is the identity matrix.

Furthermore, Figs. 8–10 include observer performance on image set $\tilde{\mathbf{g}}$, which uses a transformation matrix that does not depend on the prior knowledge of the two classes. This transformation matrix is sampled from a uniform distribution over [0, 1] and randomize the pixel weight and position. The CNN AUC $= 0.5$ for $\tilde{\mathbf{g}}$ even in the limit of large training data; see Figs. 7 and 8. The CNN trained on $\tilde{\mathbf{g}}$ was only able to achieve non-guessing performance when the task was extremely simple ($\sigma_1 = 1.0$ and $\sigma_2 = 0.40$); see Fig. 8. Since the matrix is full rank, the IO performance on $\tilde{\mathbf{g}}$ and $\mathbf{g}$ are equal, with a value of 1.0. For the extremely simple task, the mCNN AUC $= 1.0$ when trained on $\mathbf{g}$ for 1,000 training images. In contrast, mCNN AUC $=$ 0.65 when trained on $\tilde{\mathbf{g}}$ for the same amount of training images. This implies a greater difficulty for the network to generalize a representation for $\tilde{\mathbf{g}}$. With the state-of-the-art network architectures, performance on $\tilde{\mathbf{g}}$ images increases compared to mCNN; see Table I. However, the AUC of the state-of-the-art CNN architectures performed below IO for even a 121-layer architecture. Table I suggests that the techniques used in ResNet-50 and DenseNet-121 to improve upon shallower networks do not contribute in increasing the performance beyond VGG-16 for this detection task.

This article also presents a method to change the correlation structure of $\tilde{\mathbf{g}}$ through a subsequent FKT full-rank linear transformation that then increases CNN performance; see Figs. 8 and 9. This change in correlation structure does not alter the information in the original image since the transformation is invertible; therefore, IO performance remains unchanged up to numerical error. This linear transformation also does not increase the amount of information in the image, as stated by the data-processing inequality. However, the restoration process presumably restores the correlation structure to a pattern recognized by the CNN.

Overall, this work contributes a linear transformation that reduces the CNN to near-guessing performance even for large quantities of training images. Subsequently applying a different full-rank transformation that depends on the covariance matrix of the two classes increases the CNN performance. This work also shows that the FKT can be used to increase the CNN performance through compression.

REFERENCES

[1] H. H. Barrett and K. J. Myers, *Foundations of Image Science* (Wiley, New York, 2013).

[2] W. S. Geisler, "Contributions of ideal observer theory to vision research," Vis. Res. **51**, 771–781 (2011); Vision Research 50th Anniversary Issue: Part 1.

[3] W. Zhou, H. Li, and and M. A. Anastasio, Approximating the Ideal Observer and Hotelling Observer for binary signal detection tasks by use of supervised learning methods. arXiv:1905.06330, (2019).

[4] F. Reith and B. Wandell, A convolutional neural network reaches optimal sensitivity for detecting some, but not all, patterns. arXiv:1911.05055, (2019).

[5] S. Wolfram, "Statistical mechanics of cellular automata," Rev. Mod. Phys. **55**, 601–644 (1983).

[6] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, Understanding deep learning requires rethinking generalization. 2016. arXiv:1611.03530, Published in ICLR (2017).

[7] I. J. Goodfellow, J. Shlens, and C. Szegedy, Explaining and harnessing adversarial examples. arXiv:1412.6572, (2014).

[8] A. Chakraborty, M. Alam, V. Dey, A. Chattopadhyay, and D. Mukhopadhyay, Adversarial attacks and defences: A survey. arXiv:1810.00069, (2018).

[9] R. Geirhos, P. Rubisch, C. Michaelis, M. Bethge, F. A. Wichmann, and W. Brendel, "Imagenet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness," *Int'l. Conf. on Learning Representations* (ICLR, La Jolla, CA, 2019).

[10] D. J. Field, "Relations between the statistics of natural images and the response properties of cortical cells," J. Opt. Soc. Am. A **4**, 2379–2394 (1987).

[11] C. K. Abbey, H. H. Barrett, and M. P. Eckstein, "Practical issues and methodology in assessment of image quality using model observers," Proc. SPIE **3032**, 182–194 (1997).

[12] M. A. Kupinski, E. Clarkson, and J. Y. Hesterman, "Bias in Hotelling observer performance computed from finite data," Proc. SPIE **6515**, 65150S–65150S–7 (2007).

[13] B. D. Gallas, "Variance of the channelized-hotelling observer from a finite number of trainers and testers," Proc. SPIE **5034**, 100–111 (2003).

[14] K. Fukunaga and W. L. G. Koontz, "Application of the Karhunen-Loeve expansion to feature selection and ordering," IEEE Trans. Comput. **19**, 311–318 (1970).

[15] X. Huo, "A statistical analysis of Fukunaga-Koontz transform," IEEE Signal Process. Lett. **11**, 123–126 (2004).

[16] M. K. Kupinski and E. Clarkson, "Method for optimizing channelized quadratic observers for binary classification of large-dimensional image datasets," J. Opt. Soc. Am. A Opt. Image Sci. Vis. **32**, 549–565 (2015).

[17] A. Mahalanobis, R. R. Muise, S. R. Stanfill, and A. Van Nevel, "Design and application of quadratic correlation filters for target detection," IEEE Trans. Aerosp. Electron. Syst. **40**, 837–850 (2004).

[18] R. Bellman, *Dynamic Programming* (Princeton University Press, Princeton, NJ, 1957), Vol. XXV, p. 342.

[19] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*, Springer Series in Statistics (Springer New York Inc., New York, NY, USA, 2001).

[20] S. Ioffe and C. Szegedy, Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, arXiv:1502.03167, (2015).
[21] K. Simonyan and A. Zisserman, Very deep convolutional networks for large-scale image recognition. *CoRR*, arXiv:1409.1556, (2015).
[22] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *2016 IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)* (IEEE, Piscataway, NJ, 2016), pp. 770–778.
[23] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," *2017 IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)* (IEEE, Piscataway, NJ, 2017), pp. 2261–2269.
[24] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, Wiley Series in Telecommunications and Signal Processing (Wiley-Interscience, USA, 2006).
[25] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning* (MIT, Cambridge, MA, 2016), http://www.deeplearningbook.org.
[26] Z. Allen-Zhu, Y. Li, and Y. Liang, "Learning and generalization in overparameterized neural networks, going beyond two layers," in *Advances in Neural Information Processing Systems 32*, edited by H. Wallach, H. Larochelle, A. Beygelzimer, F. d' Alché-Buc, E. Fox, and R. Garnett (Curran Associates, Inc., San Diego, CA, 2019), pp. 6155–6166.
[27] H. W. Lin and M. Tegmark, "Why does deep and cheap learning work so well?," J. Stat. Phys. **168**, 1223–1247 (2017).