Journal of Perceptual Imaging® 6: 1–10, 2023.

Efficient Coding in Human Vision as a Useful Bias in Computer Vision and Machine Learning

Philipp Grüning and Erhardt Barth

Institute for Neuro- and Bioinformatics, University of Lübeck, Ratzeburger Allee 160, 23562 Lübeck, Germany
E-mail: erhardt.barth@uni-luebeck.de

Abstract. *Interdisciplinary research in human vision has greatly contributed to the current state-of-the-art in computer vision and machine learning starting with low-level topics such as image compression and image quality assessment up to complex neural networks for object recognition. Representations similar to those in the primary visual cortex are frequently employed, e.g., linear filters in image compression and deep neural networks. Here, we first review particular nonlinear visual representations that can be used to better understand human vision and provide efficient representations for computer vision including deep neural networks. We then focus on i2D representations that are related to end-stopped neurons. The resulting E-nets are deep convolutional networks, which outperform some state-of-the-art deep networks. Finally, we show that the performance of E-nets can be further improved by using genetic algorithms to optimize the architecture of the network.* © *2023 Society for Imaging Science and Technology.*
[DOI: 10.2352/J.Percept.Imaging.2023.6.000402]

1. INTRODUCTION

Given the 75th IS&T anniversary special issue of JPI, we here give an overview of selected contributions to the topic of efficient image representations.

Linear approaches such as Fourier methods and wavelets have been very successful in modeling early visual processing and also in designing algorithms for image processing, such as image compression, denoising, etc. But even in the early days, these linear methods did not work alone, as they always required rather simple nonlinearities such as a threshold or quantization. However, certain visual phenomena, e.g., gain control and end-stopping, required new types of, often more complex, nonlinear models.

We will here review some of these approaches and mainly focus on particular nonlinear representations related to the concepts of intrinsic dimension, curvature, and end-stopping. A similar approach has been taken in the presentation at the 25th anniversary of the HVEI conference [13]. Here we broaden the topic to include machine learning and provide, in Section 4, some related and recent unpublished results regarding the design of deep neural networks.

In machine learning, we need to adapt the bias to the problem at hand and keep it as strong as necessary and as weak as possible. Therefore, the main challenge in machine learning is to find an appropriate bias. For example, convolutional neural networks (CNNs) perform better than multilayer perceptrons (MLPs) in image-based object recognition because they have an additional bias, which is adequate for dealing with images (they have sparse and shared weights as neurons do). Theoretically, an MLP could be trained to have a weight-matrix that is a convolution matrix. But if convolutions are enforced as in CNNs, i.e., introduced as a bias, the convolution emerges with probability one whereas with MLPs the probability will be very small. Historically, this additional bias was inspired by vision research, leading to, for example, the neocognitron [15] and the LeNet [30]. Here, we look at further useful biases that are based on interdisciplinary research in human vision. Although some tend to believe that everything could be learned from data, it is a fact that learning is only possible with extra-evidential factors [51], i.e., with an appropriate bias.

In our view, the result of learning is determined by (i) the raw data (a particular training set), and (ii) the rest - which we would call the bias, i.e. everything - other than the data - that influences the result (including how data are selected). So the bias is a mixed bag full of "things" out of which some things are trivial, others are well known, some are bad, and yet others can be a valuable contribution that improves the results of decision making. Finding the latter is the challenge that we refer to. We could have maybe used another term such as "model" but in our view, it seems important to use the term bias because all the different terms, such as inductive-, implicit-, cognitive-bias are all biases, and it is important to realize that the bias as such is not something to avoid. It is what we need to make decisions and the challenge is to find a good bias, one which helps solving the problem. This is why mechanisms found in human vision are so useful because they define a bias adapted to the problem of vision. The bias must also be adapted to the amount of data. This is why in the early days of machine vision we had to introduce a stronger bias, e.g., by defining specific hand-crafted features, whereas today with more data and learning capacity we need a more generic bias. From a machine-learning perspective, we choose a machine and then the learning process selects one function out of the set of all possible functions. One could therefore argue that if two machines, that can theoretically approximate any function, would lead to different results

Received Nov. 22, 2022; accepted for publication July 5, 2023; published online Sept. 15, 2023. Associate Editor: Adar Pelah.

2575-8144/2023/6/000402/10/$00.00

it is the learning that determines the result. However, that would be misleading because it obviously is the choice of the machine that is critical and makes a big difference. Therefore, we believe that distinctions, such as between model- and inductive- or learning bias [43], are hard to make.

2. THEORETICAL BACKGROUND

In the following, we present a few ways of approaching the goal of efficient coding from different perspectives.

2.1 *Linear Filters*

Linear filters, e.g., wavelets, have been used extensively to compute useful image representations and to construct models of visual representations. Linear transforms, or linear layers in a network, are computed by multiplying the input vector with a matrix, i.e., $\vec{x}_{LF} = \boldsymbol{W}\vec{x}$ where $\vec{x} \in \Re^N$ is the vector that represents the image, or an image block, $\boldsymbol{W}$ is a weight matrix, and $\vec{x}_{LF}$ is the filtered image or image block.

When the matrix $\boldsymbol{W}$ is pre-defined, i.e., not learned, such filters are now named handcrafted filters. In terms of efficient representations, linear transforms can be used to decorrelate signals. In case of higher-order statistical dependencies, however, linear transforms can only reduce very specific higher-order dependencies, and therefore nonlinear transforms are needed [58].

2.2 *Unsupervised Learning of Representations*

As an alternative to predefined filters; representations, including linear ones, can be learned based on a representative set of sample data $\vec{x}_i$, $i = 1, \ldots, p$. For the learning procedure, optimization criteria are required. One useful strategy is to maximize the efficiency of the representation, e.g., by reducing the dimension of the data vector, i.e., we look for a mapping $\vec{x} \in \Re^N \longmapsto \vec{y} \in \Re^M$, $M < N$. Possible criteria are (i) minimal loss of information (generic criterion), or (ii) better pattern-recognition performance. For biological systems, criteria such as reduced neural activity can be important. Ideally, the vector $\vec{y}$ contains the features of the data $\vec{x}$ that are the most relevant (for a more or less specific set of tasks). However, properties of the data vector $\vec{y}$ other than the reduced dimension may be of interest, and the challenge is to find generic criteria for data representation that turn out to be useful for various tasks.

From a geometric perspective, one aims to find the subspace to which the data may be confined. Linear subspaces can be found by using linear transforms such as the Principal Component Analysis (PCA) or the Independent Component Analysis (ICA). Linear transforms, however, fail if the subspace is, e.g., a curved manifold.

2.3 *Sparse Coding*

A sparse representation is one where only a few components of the data vector are different from zero. The principle of sparse coding, as introduced by Olshausen and Field [36, 37] is implemented as an optimization problem with the cost function $E = -E_{\text{preserveinformation}} - \lambda E_{\text{sparseness}}$, where the individual energy terms are the mean reconstruction error $E_{\text{preserveinformation}} = -\|\boldsymbol{W}^{-1}\vec{x}_{SC} - \vec{x}\|^2$ and a sparseness term that favors small values for the components of $\vec{x}_{SC}$. The goal is to learn the representation $\vec{x}_{SC}$ and the basis functions, i.e., the rows of $\boldsymbol{W}^{-1}$ by solving the nested optimization problem $\min_{\boldsymbol{W}^{-1}}(\min_{\vec{x}_{SC}} E)$. Note that the set of basis functions can be over-complete, i.e., one can have more basis functions than dimensions of the input space. Moreover, the basis functions need not be orthogonal.

2.4 *Gain Control and Divisive Normalization*

Lyu and Simoncelli [32] have shown that in case of natural images, the probability density function is often better modeled as elliptical than as factorial and have introduced Radial Gaussianisation as a method that can find representations with independent components in case of elliptical symmetric densities (ESD), i.e., densities for which the points of constant probability are ellipsoids. Here the limitation of linear transforms is that they cannot make the components of signals with ESD more independent after whitening, whereas radial gaussianisation, defined as $\vec{x}_{rg} = g(\|\vec{x}_{\text{wht}}\|)\vec{x}_{\text{wht}}/\|\vec{x}_{\text{wht}}\|$, can ($\vec{x}_{\text{wht}}$ is the whitened signal and $g(.)$ is chosen such that x_{rg} is Gaussian). Moreover, it can be shown that divisive normalization is a good approximation to radial gaussianisation and is related to cortical gain control. The gain of a neuron is often controlled by the activity of neighboring neurons in order to better cope with the high dynamic range of the input. In case of divisive normalization, the output of a neuron is divided by the average response of neighboring neurons.

2.5 *Intrinsic Dimension and i2D Operators*

Let an image be modeled by a function $f : \boldsymbol{R}^2 \to \boldsymbol{R}$. Given an open region Ω, for all $(u, v) \in \Omega$, either (a) $f(u, v) =$ constant; or (b) $f(u, v) = g(au + bv)$, for some g, a, b; or (c) f varies along all directions. The image f is said to locally have intrinsic dimension 0, 1, or 2, respectively (*i0D*, *i1D*, *i2D* for short).

To find an efficient representation, we now search for a $\vec{x}_{i2D} = F(\vec{x})$ that is equal to zero if $\vec{x}$, now defined as the vector that contains all the pixel values $f(u, v)$ in Ω, is *i0D* or *i1D*. $\vec{x}_{i2D}$ should be different from zero, if $\vec{x}$ is *i2D*. Such a transformation is called *i2D* transform or *i2D* operator.

The concept of intrinsic dimension can easily be extended to more dimensions, such that, in case of videos, the intrinsic dimension *inD* varies from $n = 0, \ldots, 3$.

Why should a nonlinear *i2D* transform be better than the linear transformations discussed above? Since linear transformations cannot be *i2D* transforms [55, 56] any linear transform will fail to create zero output coefficients for all *i0D* and *i1D* inputs although the coefficients could be zero without loss of information. In differential geometry, the curvature measures the deviation from flatness and only flat surfaces can be mapped to a plane. This provides some insight for understanding why information is concentrated at (curved) *i2D* regions of an image [34, 55, 56] and a number of applications have confirmed this fact [49, 50].

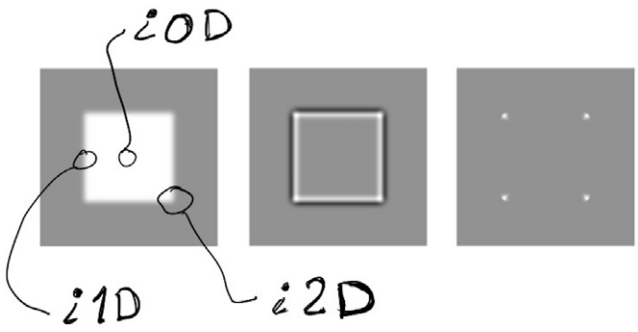

Figure 1. The statistics of natural images are well captured by the square image on the left (*i0D* regions are most frequent and *i2D* regions the least frequent). Representations in the retina already attenuate *i0D* regions by isotropic lateral inhibition (middle) and end-stopped cells would represent the square by its corners only (right) leading to a very sparse representation that nevertheless fully determines the original square.

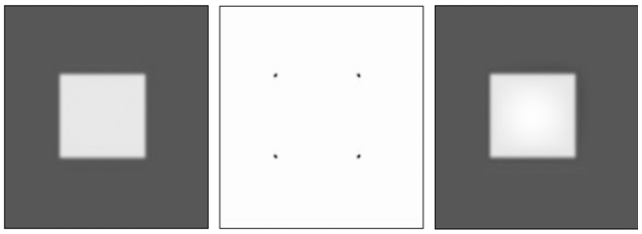

Figure 2. As shown in Ref. [34], an image (example shown left) can be reconstructed from its *i2D* representation (edge curvature shown in the middle). The reconstructed image is shown on the right. This demonstrates the redundancy of *i0D* and *i1D* regions.

Moreover, it has been shown that *i0D* and *i1D* regions are more frequent in natural images than *i2D* regions [58]. Therefore, an *i2D* transform can produce, without loss of information, a higher degree of sparsification than a linear transform.

Differential geometry has also led to methods for the design of *i2D* operators and models of end-stopped cells. In geometry, to obtain the Gaussian curvature that is zero for *i0D* and *i1D* regions, the main curvatures must be multiplied [12]. This can be generalized to a more general framework in which the outputs of orientation-selective filters of different orientations are multiplied [55]. The multiplication, however, is not essential; in a more general framework nonlinearities are needed that create some kind of AND combination of the two filters. For example, the corners in Figure 1 and Figure 2 can be detected by the logical combination of "horizontal edge filter AND vertical edge filter".

2.6 *Supervised Learning of Representations*

In deep networks, representations that emerge in the early layers are learned by backpropagating the classification errors, i.e., the representations are optimized for the classification task at hand. Task-specific representations can obviously be better for a specific task but applications often benefit from less specialized networks that can solve multiple problems [40]. How specific a learned representation is, depends on the data used to train it. With transfer learning, networks are pretrained on a benchmark, such as ImageNet, with a large database and then adapted to a specific problem with fewer data [46]. However, as seen in the Introduction, not everything can be learned effectively and thus a useful bias has the potential to improve the performance of the network.

3. OVERVIEW OF SOME EARLIER RESULTS

3.1 *Sparse Coding*

In 1996, Olshausen and Field published several influential papers [14, 36, 37]. The first paper [14] relates to earlier work on efficient coding that analyzed the statistics of natural images. The focus was on the $1/f$ fall-off of the amplitude spectra, which was related to the sparseness of structure in images, as opposed to self-similarity. Moreover, the main properties of cortical neurons (localization, frequency selectivity, and orientation) are related to the principle of sparse coding: the steeper the fall-off of the amplitude spectrum, the fewer the number of active neurons tuned to high frequencies. The second paper [37] focused on the learning of sparse representations and proposed to describe images with a sparse set of learned basis functions drawn from an overcomplete dictionary. A later paper [35] extended the principle of sparse coding to the coding of motion signals. Bilinear generative image models have been considered since previous sparse codes, based on linear superposition only, cannot optimally encode objects that move or change due to other sources of variation. The idea is that, similar to the visual coding in "what" and "where" streams, the representations of object shape can be separated from object transformations and learned independently. In more general terms, representations are now learned in different subspaces. Related ideas have been discussed in [52].

A number of extensions and applications of sparse coding have been published (see, for example, the IEEE Transactions on Selected Topics in Signal Processing [44]). However, to our knowledge, the first application of sparse coding to a technical problem of pattern recognition has been presented in Ref. [29].

3.2 *Gain Control and Divisive Normalization*

Using divisive normalization, one can better predict human recognition performance, e.g., the detection of nodules in radiographs [31]. In 2007 Lyu and Simoncelli [33] presented an invertible divisive normalization transform with applications to local contrast enhancement and image compression. A few years earlier, Valerio and Navarro [47] already showed that an invertible divisive normalization stage can remove higher-order statistical dependencies beyond what is possible with linear filters. Both the linear wavelet decomposition stage and the nonlinear divisive normalization stage reduced the mutual information by a factor of six each. Bio-inspired generalized divisive normalization has been used for an

end-to-end optimized image compression model [1]. In Ref. [60] the authors discuss divisive normalization in the context of a multi-stage linear-non-linear codec. Burg et al. [9] use divisive normalization to predict responses of neurons in the macaque V1.

3.3 *Intrinsic Dimension and i2D Operators*

The first paper on intrinsic dimension appeared at the IS&T conference HVEI [56] and already presented a number of relevant topics such as the concept of intrinsic dimension, the limits of linear filters, the related differential-geometry framework, related biological phenomena such as end-stopping and bug detectors, generalized *i2D*-detector equation based on $\Sigma\Pi$ structures, the compensation equation, *i2D*-detectors as AND operations on oriented filters, and the topological invariance of integrated *i2D* activities. Later the concepts have been extended to a theory of *i2D* selectivity based on nonlinear Volterra-Wiener systems and *i2D*-operators have been related to higher-order statistical dependencies and the polyspectra of natural images [26, 59].

In Ref. [6] it could be shown that simple lateral inhibition, i.e. isotropic operations as found in the retina, can lead to *i2D* selectivity when implemented in a deep or in a recurrent network with simple nonlinearities (ON/OFF rectification, now called ReLU). A detailed retinal model was presented, which, a few years later, has been used to model the rather complex topological sensitivity of the bug detector in the frog's eye and human topological sensitivity attributed to early visual processing [4].

The computational power of such linear-non-linear (LNL) structures has been further analyzed about ten years later [60] and it has been shown how LNL networks can reduce the statistical dependencies in natural images. An early description of an LNL structure with ReLUs is given in Ref. [61]. The LNL ideas have been meanwhile confirmed by the success of certain deep-learning strategies, where a number of layers are learned by unsupervised methods (often leading to sparse representations) and only the final layer is trained for making decisions [8]. However, a unified theoretical framework of LNL sandwiches and their relation to efficient representations, intrinsic dimension, sparse coding, and deep learning is still missing. Note, however, that the simple INROG (Iterated Nonlinear Ratio of Gaussians) operator presented in Ref. [6] already provided a deep-network structure with increasingly sparse *i2D* features and divisive normalization.

Further results have been the reconstruction of images from only *i2D* features [3], and the mathematical proof that *i2D* regions are unique [34].

The geometrical aspects of extensions to video have been presented in Refs. [7, 57] and later used to model the selectivity of MT neurons to multiple motions and the perception of global motion induced by terminator motion [5]. Finally, the framework has led to a state-of-the-art saliency model [50] and performance of several state-of-the-art action recognition algorithms was significantly improved by a preprocessing step that restricted action recognition to such salient video regions [49].

4. EFFICIENT CODING IN DEEP NETWORKS

4.1 *Barlow Twins*

Based on the tradition that insights from neuroscience can help improve technical applications, the Facebook AI team has developed a novel and quite influential deep-network architecture inspired by the concepts of efficient coding [53]. Besides using the classification error, the learning of representations is enhanced by an additional term that reduces the correlations between the components of the representation vector. The name of the novel network has been chosen to give credit to Barlow, who pioneered the idea of efficient coding in the brain [2]. Barlow twins are a good example of how a useful bias can improve the performance of a deep network.

4.2 *Sparse Coding and Hyperselectivity*

Deep-learning networks are the champions on most image-based recognition benchmarks. There remains, however, a severe limitation due to the fact that the typical neuron models used in deep networks can be activated by a large set of image patterns that are outside the training set. What that implies is, that even if one succeeded in training a self-driving car such that it will make no mistakes in its natural environment, one would be able to confuse the car by providing some unnatural patterns chosen with the purpose to confuse it. In this context, it has been shown that sparse-coding neurons are hyperselective, i.e., the set of patterns that can excite them is smaller, and can thus, provide increased robustness, e.g., robustness against adversarial attacks [38, 59, 61]. The E-net units of Section 4.4 are also hyperselective [21]. A further application in deep networks is to use sparse coding for regularization, see for example [45].

4.3 *Divisive Normalization in Deep Networks*

More recently, it has been shown that the principle of divisive normalization can be incorporated into the design of deep networks. A study including CNNs for image classification and image super-resolution, as well as language modeling by recurrent neural networks (RNNs), can be found in [41], showing that divisive normalization can improve image upscaling (super-resolution).

Pan et al. [39] show that a brain-inspired weighted normalization can improve image classification for shallower networks. In Ref. [10], the authors improve a VOneNet (an ANN that aims to better mimic primate V1 than conventional CNNs and is more robust against corrupted data) with divisive normalization [11].

Lastly, the well-known AlexNet employs local response normalization (LRN), dividing the activation of a neuron at position (x, y) by the activation of adjacent neurons [28].

4.4 *E-nets Based on End-stopped Model Neurons*

Feature-Product networks (FP-nets) inspired by end-stopped cortical cells and the concept of intrinsic dimension have

been introduced in Ref. [19]. FP-nets contain FP-units that multiply the outputs of two learned filters. In Ref. [21] it could be shown that FP-nets outperform state-of-the-art deep networks, such as the ResNet and MobileNet-V2, on the Cifar-10 and ImageNet benchmarks. Moreover, FP-nets have been shown to be less sensitive to adversarial attacks and JPEG artifacts. Furthermore, the learned model neurons are end-stopped to different degrees, and they provide sparse representations with an entropy that decreases with hyperselectivity.

It turned out, however, that not the multiplication but the AND-type combination of two filters is the essential component of such networks that we therefore now call E-nets (the E for end-stopping) that contain E-blocks, which perform the AND combination of two learned filters [16]. Indeed, it has been shown that the two filters could be combined via the minimum operation, which simplifies the computations without reducing performance [17]. In a further study, the multiplication could be replaced with logarithmic addition [20].

In Ref. [18] it has been shown that E-nets can predict subjective image quality and it has been argued that this may be due to the fact that the E-nets are inspired by vision.

As shown in Section 4.5, E-nets can be further improved by placing the end-stopped units at strategic positions that are found with genetic algorithms.

4.5 *Finding Optimal E-net Architectures*

As depicted in Figure 3, state-of-the-art CNNs for image classification are similar in their overall composition. Their main building element is a convolution layer, followed by a nonlinearity. A deep CNN can be viewed as a long sequence of such layers that extract increasingly complex features [54]. These features range from lines and edges with specific orientations to corners (*i2D* signals) and more complex shapes such as eyes, tires, or even whole objects.

A CNN can be divided into *stacks* that contain *blocks* containing specific arrangements of different *layers*. To create an E-net, specific blocks of a state-of-the-art CNN are exchanged with E-blocks. We here focus on the CNN ResNet [23] and the benchmark Cifar-10 [27]. The ResNet is composed of three stacks with N pyramid blocks each [22] - see Fig. 3. In Refs. [21] and [17], exchanging the first block of each stack yielded an E-net with improved classification accuracy and increased robustness. The above *design rule* that defines which blocks are exchanged is based on the observation that end-stopping occurs in rather early stages (areas V1 and V2 of the visual cortex). Accordingly, E-blocks should be positioned early in the E-net. ResNets use residual connections that pass-through low-level-features from earlier convolution layers to the entire network. Thus, a stack can be seen as a sub-network that also processes low-level information at a specific scale [48]. Thus, in each stack, an early block should be replaced with an E-block. In our experiments, we refer to this model as *default model*.

However, the impact of the design rule has not been estimated systematically, mainly because of the heavy computational workload: for a CNN with M blocks, there are 2^M possible E-nets.

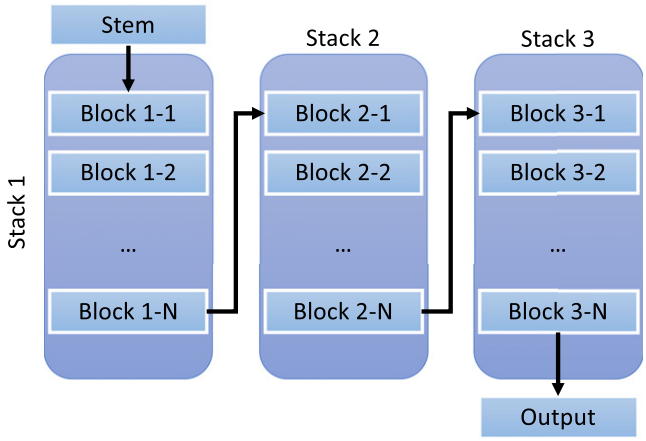

Figure 3. Typical structure of a CNN. An input image is first processed by a stem which is a convolution layer, possibly subsampling the input. Afterward, the stem's output is processed by a sequence of stacks. Each stack contains a certain number of blocks consisting of a specific sequence of convolution layers; these sequences usually define a model. In most cases, the beginning of a stack subsamples its input. For classification, the output of the last stack, a tensor of shape $H \times W \times D$, is averaged into a D-dimensional vector. Lastly, a linear layer transforms the vector into a C-dimensional vector, each entry representing a class score. This vector can be transformed into a probability distribution (e.g., by the Softmax function). As an example, the schema shows the structure of a ResNet used for the Cifar-10 dataset consisting of three stacks with N blocks each. Note that other architectures may have a different number of stacks, and each stack may consist of different numbers of blocks.

Thus, we need to solve a discrete search problem with a vast search space, where each query (i.e., training a model to determine the test error) can take several hours. An exhaustive search is thus not feasible for larger networks.

However, using a genetic algorithm is a straightforward approach to speed up the search. The algorithm mimics the process of natural selection: a population of entities (in our case models) is tested for their fitness (in our case the test score), and only the fittest entities of this population survive to pass on their genes (modeled as *bitstrings*) to another population, creating new genes by recombination and mutation. This iteration step is repeated several times such that better entities will eventually prevail.

The pseudo-code for the genetic algorithm is shown in Algorithm 1 in the Appendix. We ran different genetic algorithms to obtain 216 ($N = 5$), 72 ($N = 7$), and 180 ($N = 9$) models. We started with $N = 5$, ran $N = 7$, and finished with $N = 9$. We obtained a number of E-ResNets that performed better than the baseline on average (in the Appendix, we provide the results in Table B.1, Table B.2, and Table B.3 for $N = 5, 7, 9$, respectively).

As shown in Figure 4, the models found by the genetic algorithm outperform the baseline and also previous E-nets [17, 21]. Note that the improvements over the baseline are considerable, given that the original ResNet already

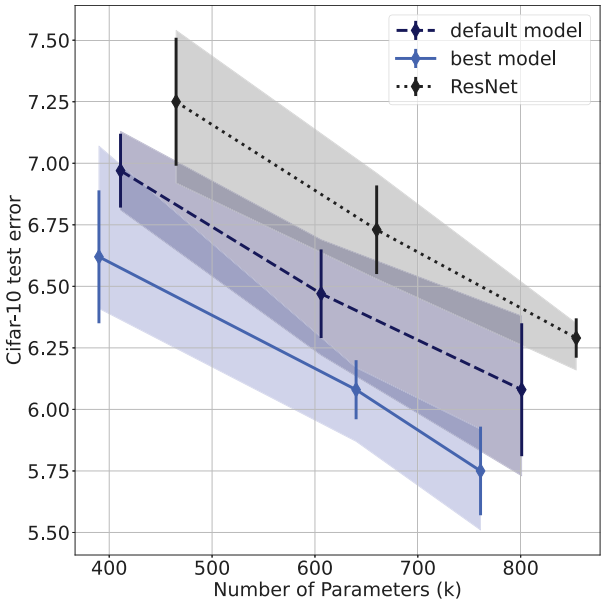

Figure 4. Cifar-10 test error comparison. Each diamond shows a particular model's number of parameters in thousands (x-axis), and its mean test error after 200 epochs averaged over five runs (y-axis), with error bars indicating the standard deviation. The transparent area shows the range from the smallest to the largest test error. The baseline results are depicted in black with a dotted line, and the dark blue and dashed line shows the results for the default model. The results for the best E-net configurations found by the genetic algorithm are shown in light blue with a solid line.

performs very well on Cifar-10, and we are just exchanging a few blocks of the ResNet with E-blocks.

5. CONCLUSIONS

From a geometric perspective, the goal of efficient representation is to find the tiny sub-manifolds in which the natural images, or a particular set of images, lie. The statistical view is that one obtains more efficient representations the more redundancies are removed.

We have reviewed a number of approaches that can lead to more efficient representations and have shown how some of them can be used to design better-performing deep networks. E-nets are a particular class of deep networks that are inspired by end-stopped visual neurons and the concept of intrinsic dimension. We have reviewed some of the results obtained with E-nets. We have also presented novel results that show how E-net architectures can be further optimized by using genetic algorithms. The main idea behind E-nets is to learn pairs of filters, which are then combined with a nonlinearity that generates AND terms, e.g., by multiplication or minimum operation. Current E-nets are just one particular way of implementing this general idea, which is inspired by end-stopped neurons and the concept of curvature in differential geometry. As argued in the Introduction, MLPs could in principle be trained to become CNNs but in practice, this is very unlikely. The same argument applies when comparing CNNs with E-nets. In [21] we could show that CNNs do indeed learn neurons that are end-stopped to a certain degree but including the E-blocks leads to a significant increase regarding the degree of end-stopping. Conversely, every E-neuron in the E-net could in principle be trained to become an "ordinary" neuron of a CNN if the angle between the two filters is zero. And indeed this happens with about 15–50% of the neurons depending on how deep the layer is [21].

We have seen that all the aspects of efficient coding discussed here, i.e., linear filters, sparse coding, divisive normalization, and end-stopping, have contributed to the design of better deep neural networks in machine learning. Thus, one can conclude that innovative approaches to computer vision and neural networks have been inspired by interdisciplinary studies of human vision. In a reciprocal view, the fact that insights from vision research may lead to algorithms that really work can be a benefit for vision research by providing insights for building more robust human-vision models. It is this type of interaction that has led to the initial design of deep neural networks, which have then later been boosted by increased computing power, large amounts of data, and big investments.

One could argue that the E-nets, especially when optimized as shown here, are not really mimicking human vision. Yes, we are only taking inspiration not rebuilding the brain - once the benefits of learning AND-combined pairs of filters are understood, we can move on and see what is possible in a technical sense.

However, the inspiration that we take is more than just some inspiration, it is one effective way to address the main challenge in machine learning and AI, i.e., we need useful biases that we cannot derive from the data.

APPENDIX A. CNN STRUCTURE AND E-NETS

CNNs typically start with a single convolution layer called a stem. Then, the input is further processed by several blocks. Often, a block is the defining substructure of a network consisting of particular sequences of convolution layers and further layers such as batch normalization [25]. For example, the MobileNet-V2 [42] mainly consists of several mobile inverted blocks, the DenseNet [24] of DenseBlocks, and there are many different block structures for the ResNet [23] – see Han et al. [22] for an overview. Analogously, stacks are structures that contain several blocks. Stacks are separated by subsampling operations, e.g., by using a strided convolution layer or a max-pooling layer.

E-nets are created by substituting entire blocks of a network with E-blocks. We describe a particular model design by using bitstrings. For example, the default model for an E-ResNet with three blocks per stack ($N = 3$) is represented by the bitstring "100 100 100" with a "1" indicating an E-block, and a "0" a standard block. Spaces separate stacks. We refer to the "000 000 000" string, i.e., the original ResNet, as the *baseline* - see [17] for details.

APPENDIX B. GENETIC ALGORITHM

The pseudo-code for the genetic algorithm is shown in Algorithm 1, its parameters are:

Algorithm 1 Genetic E-net search

Require: $\mathcal{S}_0, P_{and}, P_{mut}, N_{mut}, N_{rec}, N_{rand}, N_{iter}$
  $\mathcal{S}_T \leftarrow \{\}$
  $k \leftarrow 0$
  while $k < N_{iter}$ **do**
    Train each model $m_j \in \mathcal{S}_k$, compute test error y_j
    $\mathcal{S}_T \leftarrow \mathcal{S}_T \cup \mathcal{S}_k$
    $m_{1st}, m_{2nd} \leftarrow$ 2 models with the min. y out of $\mathcal{S}_T$
    $\mathcal{S}_{rec} \leftarrow \text{REC}(\mathcal{S}_T, P_{and}, P_{mut}, N_{rec}, m_{1st}, m_{2nd})$
    $\mathcal{S}_{mut} \leftarrow \text{MUTATE}(\mathcal{S}_T \cup \mathcal{S}_{rec}, P_{mut}, N_{mut}, m_{1st})$
    $\mathcal{S}_{rand} \leftarrow \text{MUTATE}(\mathcal{S}_T \cup \mathcal{S}_{rec} \cup \mathcal{S}_{mut}, \frac{1}{2}, N_{rand}, m_{zero})$
    $k \leftarrow k + 1$
    $\mathcal{S}_k \leftarrow \mathcal{S}_{rec} \cup \mathcal{S}_{mut} \cup \mathcal{S}_{rand}$
  end while

- The probability of using an and- instead of an or-combination when recombining two bitstrings (P_{and}).
- The mutation probability (P_{mut}).
- How many mutated survivors are in the new population (N_{surv}), see Algorithm 3.
- How many recombined survivors (N_{rec}) are in the new population, see Algorithm 2.
- How many entities are drawn randomly for the new population (N_{rand}).

Algorithm 2 Create a set by recombining two strings

procedure REC($\mathcal{S}_T, P_{and}, P_{mut}, N_{rec}, m_{1st}, m_{2nd}$)
  $\mathcal{S}_{rec} \leftarrow \{\}$
  while $|\mathcal{S}_{rec}| < N_{rec}$ **do**
    for $i \in \{0, ..., |m_{1st}| - 1\}$ **do**
      $r \leftarrow \mathcal{U}(0, 1)$
      if $r \leq P_{and}$ **then**
        $m^i_{out} \leftarrow m^i_{1st} \wedge m^i_{2nd}$
      else
        $m^i_{out} \leftarrow m^i_{1st} \vee m^i_{2nd}$
      end if
    end for
    $m_{out} \leftarrow \text{MUTATE-STRING}(m_{out}, P_{mut})$
    if $m_{out} \notin \mathcal{S}_T$ **then**
      $\mathcal{S}_{rec} \leftarrow \mathcal{S}_{rec} \cup \{m_{out}\}$
    end if
  end while
end procedure

m_{zero} is the all-zero string. $\mathcal{U}(0, 1)$ denotes the uniform distribution of values between 0 and 1. $\mathcal{S}_T$ is the set of trained models.

A critical factor of the algorithm's efficiency is the starting population $\mathcal{S}_0 = \{m_0, m_1, \ldots\}$, where m_i is a bitstring describing a model. Running the algorithm in sequence for model sizes $N = 5, 7, 9$, the starting population of each genetic algorithm was handpicked based on the results of the previous model. For $N = 5$, we used results from previous experiments on $N = 3$ models.

We ran the proposed genetic algorithm to find better-performing E-ResNets for 5, 7, and 9 blocks per stack, using the same hyperparameters $P_{\text{and}} = 0.7$, $P_{\text{mut}} = 0.05$, $N_{\text{surv}} = 2$, $N_{\text{recomb}} = 3$, $N_{\text{rand}} = 1$. We evaluated twelve training runs in each iteration step, i.e., six different models for two different seeds. We chose the population size based on our hardware setup: four GeForce RTX 2080 GPUs with 11 GB RAM each. For the smaller models, we were able to run three training sessions concurrently on one GPU. However, this was not possible for $N = 9$.

Algorithm 3 Create a mutated set

procedure MUTATE($\mathcal{S}_T, P_{mut}, N_{mut}, m$)
  $\mathcal{S}_{mut} \leftarrow \{\}$
  while $|\mathcal{S}_{mut}| < N_{mut}$ **do**
    $m_{out} \leftarrow \text{MUTATE-STRING}(m, P_{mut})$
    if $m_{out} \notin \mathcal{S}_T$ **then**
      $\mathcal{S}_{mut} \leftarrow \mathcal{S}_{mut} \cup \{m_{out}\}$
    end if
  end while
end procedure

For the genetic algorithm results, see Table B.1 for five blocks ($N = 5$), Table B.2 for seven blocks, and Table B.3 for nine blocks, respectively. Most notably, for the larger models ($N = 7$ and $N = 9$), we obtained E-ResNets that had a significant margin of almost one percent between the baseline mean test error and the E-net test error.

B.1 *Evaluation with More Runs*

To increase the statistical validity of the above findings, we compared the best E-ResNet, the baseline, and the default model on Cifar-10 for five runs instead of only two. We used

Table B.1. Best ten E-ResNets ($N = 5$) found by the genetic algorithm. We do not provide results for the baseline and the default model for this particular experiment (i.e., the particular seeds). However, based on previous results [17], the baseline has a mean test error of 7.2% and the default-model a mean test error of 6.9%.

Rank	Bitstring	Cifar-10 test error		
		mean	min	max
1	00011 01000 00010	6.433	6.352	6.514
2	00010 00010 00010	6.456	6.368	6.544
3	00011 00000 00010	6.467	6.398	6.536
4	10100 01000 01000	6.548	6.538	6.558
5	00111 01000 00011	6.554	6.288	6.820
6	00101 00010 00000	6.572	6.456	6.688
7	01011 01000 00010	6.587	6.582	6.592
8	00010 00001 00000	6.602	6.526	6.678
9	00011 00000 01010	6.603	6.444	6.762
10	10011 01100 00110	6.607	6.542	6.672

Table B.2. Best ten E-ResNets ($N = 7$) found by the genetic algorithm. In addition, we report the default model and baseline results.

Rank	Bitstring	Cifar-10 test error		
		mean	min	max
1	0000011 0100000 0000001	5.932	5.864	6.000
2	0000011 0100000 0000000	5.973	5.804	6.142
3	0000011 0101000 0000000	6.005	5.842	6.168
4	0000011 1100000 0001000	6.027	5.752	6.302
5	0001101 0001000 0000000	6.037	5.890	6.184
6	0000011 0101000 0100000	6.037	5.814	6.260
7	0101111 0001000 0010000	6.037	6.026	6.048
8	0000101 0100000 0000010	6.059	5.976	6.142
9	0000011 0100000 0100001	6.077	5.970	6.184
10	0001011 1001000 0000000	6.096	5.982	6.210
48	1000000 1000000 1000000	6.304	6.270	6.338
67	0000000 0000000 0000000	6.970	6.938	7.002

Table B.3. Best ten E-ResNets ($N = 9$) found by the genetic algorithm. In addition, we report the baseline results.

Rank	Bitstring	Cifar-10 test error		
		mean	min	max
1	000110010 111000000 000000000	5.548	5.430	5.666
2	001110011 001000000 000000000	5.600	5.564	5.636
3	001110010 001010000 001000000	5.627	5.436	5.818
4	001110010 011010000 001000000	5.655	5.448	5.862
5	001110010 010001000 001000000	5.657	5.600	5.714
6	001110010 011001000 001000000	5.668	5.602	5.734
7	000110010 001000000 000000000	5.676	5.514	5.838
8	001100000 010001010 001000000	5.684	5.608	5.760
9	001110010 011000000 000000000	5.723	5.618	5.828
10	001110000 011001000 001000000	5.728	5.696	5.760
159	000000000 000000000 000000000	6.549	6.450	6.648

each of the three best models based on the Cifar-10 genetic algorithm results.

Algorithm 4 Mutation of one string

```
procedure MUTATE-STRING(m, P_mut)
    for i ∈ {0, ..., |m| − 1} do
        r ← U(0, 1)
        if r ≤ P_mut then
            m^i ← ¬m^i
        end if
    end for
end procedure
```

Fig. 4 shows the test error curves for the baseline, the default model, and the best models obtained from the genetic algorithm. Here, for each number of blocks, we selected the best-performing of the three tested configurations (for the results of all models, see Table B.4). The experiment, using more seeds, did not reproduce the high error differences from the genetic algorithm results. Thus, sampling more runs for evaluation shows a downside of the genetic algorithm search: to process more E-net configurations, we reduce the number of runs for each E-net, decreasing the statistical validity of the results. Thus, the genetic algorithm overfits to models that perform particularly well on a few different runs and would likely select a different optimal model when using different random seeds. However, this reduction in performance was to be expected.

Table B.4. Cifar-10 test results averaged over five runs. We used the three best models (E-ResNets with a specific design rule) found with the genetic algorithm (Table B.1, Table B.2, and Table B.3). Furthermore, for each number of blocks, we evaluated the baseline and the default model.

Bitstring	Cifar-10 test error			
	mean	std	min	max
001110010 001010000 001000000	5.75	0.18	5.51	5.92
000110010 111000000 000000000	5.90	0.16	5.65	6.10
001110011 001000000 000000000	6.01	0.17	5.78	6.22
100000000 100000000 100000000	6.08	0.27	5.73	6.38
000000000 000000000 000000000	6.29	0.08	6.16	6.35
0000011 0100000 0000000	6.08	0.12	5.87	6.17
0000011 0101000 0000000	6.13	0.18	5.82	6.29
0000011 0100000 0000001	6.23	0.25	5.92	6.51
1000000 1000000 1000000	6.47	0.18	6.22	6.69
0000000 0000000 0000000	6.73	0.18	6.53	6.96
00011 01000 00010	6.62	0.27	6.41	7.07
00011 00000 00010	6.74	0.25	6.46	7.10
00010 00010 00010	6.88	0.40	6.43	7.45
10000 10000 10000	6.97	0.15	6.81	7.13
00000 00000 00000	7.25	0.26	6.92	7.54

REFERENCES

[1] J. Ballé, V. Laparra, and E. P. Simoncelli, "End-to-end optimized image compression," *5th Int'l. Conf. on Learning Representations, ICLR 2017* (OpenReview.net, Amherst, MA, 2017).

[2] H. Barlow, "Possible principles underlying the transformation of sensory messages," Sensory Communication **1**, 217–234 (1961).

[3] E. Barth, T. Caelli, and C. Zetzsche, "Image encoding, labelling and reconstruction from differential geometry," CVGIP: Graph. Models Image Process. **55**, 428–446 (1993).

[4] E. Barth, M. Ferraro, and C. Zetsche, "Global topological properties of images derived from local curvature features," in *Visual Form 2001. Lecture Notes in Computer Science*, edited by C. Arcelli, L. P. Cordella, and G. S. di Baja (Springer, Cham, 2001), pp. 285–294.

[5] E. Barth and A. B. Watson, "A geometric framework for nonlinear visual coding," Opt. Express **7**, 155–185 (2000).

[6] E. Barth and C. Zetzsche, "Endstopped operators based on iterated nonlinear center-surround inhibition," Proc. SPIE **3299**, 67–78 (1998).

[7] E. Barth, C. Zetzsche, and G. Krieger, "Curvature measures in visual information processing," Open Syst. Inf. Dyn. **5**, 25–39 (1998).

[8] Y. Bengio, "Learning deep architectures for AI," Found. Trends Mach. Learn. **2**, 1–127 (2009) also published as a book. Now Publishers, 2009.

[9] M. F. Burg, S. A. Cadena, G. H. Denfield, E. Y. Walker, A. S. Tolias, M. Bethge, and A. S. Ecker, "Learning divisive normalization in primary visual cortex," PLOS Comput. Biol. **17**, e1009028 (2021).

[10] A. Cirincione, R. Verrier, A. Bic, S. Olaiya, J. J. DiCarlo, L. Udeigwe, and T. Marques, "Implementing divisive normalization in CNNs improves robustness to common image corruptions," *SVRHM 2022 Workshop @ NeurIPS* (OpenReview.net, Amherst, MA, 2022).

[11] J. Dapello, T. Marques, M. Schrimpf, F. Geiger, D. Cox, and J. J. DiCarlo, "Simulating a primary visual cortex at the front of cnns improves robustness to image perturbations," Advances in Neural Inf. Process. Systems **33**, 13 073–13 087 (2020).

[12] M. P. Do Carmo, *Riemannian Geometry* (Birkhäuser, Boston, 1992).

[13] M. Dorr, E. Vig, and E. Barth, "Efficient image representations and features," Proc. SPIE **8651**, 86510R (2013).

[14] D. J. Field, B. A. Olshausen, and N. Brady, "Wavelets, blur, and the sources of variability in the amplitude spectra of natural scenes," Proc. SPIE **2657**, 108–119 (1996).

[15] K. Fukushima, "Neocognitron: a self organizing neural network model for a mechanism of pattern recognition unaffected by shift in position," Biol. Cybern. **36**, 193–202 (1980).

[16] P. Grüning, "E-nets as novel deep networks", dissertation (University of Lübeck, 2022).

[17] P. Grüning and E. Barth, "Bio-inspired min-nets improve the performance and robustness of deep networks," *SVRHM 2021 Workshop@ NeurIPS* (OpenReview.net, Amherst, MA, 2021).

[18] P. Grüning and E. Barth, "FP-nets for blind image quality assessment," J. Percept. Imag. **4**, 010402-1–010402-13 (2021).

[19] P. Grüning, T. Martinetz, and E. Barth, "Feature products yield efficient networks," Preprint arXiv:2008.07930, (2020).

[20] P. Grüning, T. Martinetz, and E. Barth, "Log-nets: Logarithmic feature-product layers yield more compact networks," *Artificial Neural Networks and Machine Learning – ICANN 2020* (Springer, Cham, 2020), pp. 79–91.

[21] P. Grüning, T. Martinetz, and E. Barth, "FP-nets as novel deep networks inspired by vision," J. Vis. **22**, 8–8 (2022).

[22] D. Han, J. Kim, and J. Kim, "Deep pyramidal residual networks," *Proc. IEEE Conf. on Computer Vision and Pattern Recognition* (IEEE, Piscataway, NJ, 2017), pp. 5927–5935.

[23] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *Proc. IEEE Conf. on Computer Vision and Pattern Recognition* (IEEE, Piscataway, NJ, 2016), pp. 770–778.

[24] 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.

[25] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *ICML'15: Proc. 32nd Int'l. Conf. on Machine Learning* (PMLR, Cambridge, MA, 2015), Vol. 37, pp. 448–456.

[26] G. Krieger, C. Zetzsche, and E. Barth, "Nonlinear image operators for the detection of local intrinsic dimensionality," *Proc. IEEE Workshop Nonlinear Signal and Image Processing* (IEEE, Piscataway, NJ, 1995), pp. 182–185.

[27] A. Krizhevsky, V. Nair, and G. Hinton, "Cifar-10 (canadian institute for advanced research)," http://www.cs.toronto.edu/ kriz/cifar.html (2021).

[28] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, edited by F. Pereira, C. Burges, L. Bottou, and K. Weinberger (Curran Associates, Red Hook, NY, 2012), Vol. 25.

[29] K. Labusch, U. Siewert, T. Martinetz, and E. Barth, "Learning optimal features for visual pattern recognition," Proc. SPIE **6492**, 64920B (2007).

[30] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," Neural Comput. **1**, 541–551 (1989).

[31] T. Luo and X. Mou, "Divisive normalization in channelized hotelling observer," Proc. SPIE **7527**, 75271K (2010).

[32] S. Lyu and E. Simoncelli, "Nonlinear extraction of independent components of natural images using radial gaussianization," Neural Comput. **21**, 1485–1519 (2009).

[33] S. Lyu and E. P. Simoncelli, "Statistically and perceptually motivated nonlinear image representation," Proc. SPIE **6492**, 649207 (2007).

[34] C. Mota and E. Barth, "On the uniqueness of curvature features," in *Dynamische Perzeption*, edited by G. Baratoff and H. Neumann, Proc. in Artificial Intelligence (Infix Verlag, Köln, 2000), Vol. 9, pp. 175–178.

[35] B. A. Olshausen, C. Cadieu, J. Culpepper, and D. K. Warland, "Bilinear models of natural images," Proc. SPIE **6492**, 649206 (2007).

[36] B. A. Olshausen and D. J. Field, "Emergence of simple-cell receptive field properties by learning a sparse code for natural images," Nature **381**, 607–609 (1996).

[37] B. A. Olshausen and D. J. Field, "Learning efficient linear codes for natural images: the roles of sparseness, overcompleteness, and statistical independence," Proc. SPIE **2657**, 132–138 (1996).

[38] D. M. Paiton, C. G. Frye, S. Y. Lundquist, J. D. Bowen, R. Zarcone, and B. A. Olshausen, "Selectivity and robustness of sparse coding networks," J. Vis. **20**, 10–10 (2020) 11.

[39] X. Pan, E. Kartal, L. G. S. Giraldo, and O. Schwartz, "Brain-inspired weighted normalization for CNN image classification," *Int'l. Conf. on Learning Representations (ICLR)* (2021).

[40] R. Ranjan, V. M. Patel, and R. Chellappa, "Hyperface: A deep multi-task learning framework for face detection, landmark localization, pose estimation, and gender recognition," IEEE Trans. Pattern Anal. Mach. Intell. **41**, 121–135 (2017).

[41] M. Ren, R. Liao, R. Urtasun, F. H. Sinz, and R. S. Zemel, "Normalizing the normalizers: Comparing and extending network normalization schemes," *5th Int'l. Conf. on Learning Representations, ICLR 2017* (2017).

[42] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," *Proc. IEEE Conf. on Computer Vision and Pattern Recognition* (IEEE, Piscataway, NJ, 2018), pp. 4510–4520.

[43] F. H. Sinz, X. Pitkow, J. Reimer, M. Bethge, and A. S. Tolias, "Engineering a less artificial intelligence," Neuron **103**, 967–979 (2019).

[44] J.-L. Starck, J. Fadili, M. Elad, R. Nowak, and P. Tsakalides, "Introduction to the issue on adaptive sparse representation of data and applications in signal and image processing," IEEE J. Sel. Top. Signal Process. **5**, 893–895 (2011) 10.

[45] X. Sun, N. M. Nasrabadi, and T. D. Tran, "Supervised deep sparse coding networks for image classification," IEEE Trans. Image Process. **29**, 405–418 (2020).

[46] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, and C. Liu, "A survey on deep transfer learning," *Int'l. Conf. on Artificial Neural Networks* (Springer, Cham, 2018), pp. 270–279.

[47] R. Valerio and R. Navarro, "Nonlinear image representation with statistical independent features: efficient implementation and applications," Proc. SPIE **5007**, 352–363 (2003).

[48] A. Veit, M. J. Wilber, and S. Belongie, "Residual networks behave like ensembles of relatively shallow networks," Adv. Neural Inf. Process. Syst. **29** (2016).

[49] E. Vig, M. Dorr, and D. Cox, "Space-variant descriptor sampling for action recognition based on saliency and eye movements," *LNCS 7578, Proc. European Conf. on Computer Vision* (Springer, Cham, 2012), pp. 84–97.

[50] E. Vig, M. Dorr, T. Martinetz, and E. Barth, "Intrinsic dimensionality predicts the saliency of natural dynamic scenes," IEEE Trans. Pattern Anal. Mach. Intell. **34**, 1080–1091 (2012).

[51] S. Watanabe, *Pattern Recognition: Human and Mechanical* (Wiley-Interscience, Hoboken, NJ, 1985).

[52] L. Wiskott and T. J. Sejnowski, "Slow feature analysis: Unsupervised learning of invariances," Neural Comput. **14**, 715–770 (2002).

[53] J. Zbontar, L. Jing, I. Misra, Y. LeCun, and S. Deny, "Barlow twins: Self-supervised learning via redundancy reduction," *CoRR*, vol. abs/2103.03230, 2021.

[54] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," *European Conf. on Computer Vision* (Springer, Cham, 2014), pp. 818–833.

[55] C. Zetzsche and E. Barth, "Fundamental limits of linear filters in the visual processing of two-dimensional signals," Vis. Res. **30**, 1111–1117 (1990).

[56] C. Zetzsche and E. Barth, "Image surface predicates and the neural encoding of two-dimensional signal variation," Proc. SPIE **1249**, 160–177 (1990).

[57] C. Zetzsche and E. Barth, "Direct detection of flow discontinuities by 3D curvature operators," Pattern Recognit. Lett. **12**, 771–779 (1991).

[58] C. Zetzsche, E. Barth, and B. Wegmann, "The importance of intrinsically two-dimensional image features in biological vision and picture coding," in *Digital Images and Human Vision*, edited by A. B. Watson (MIT Press, Cambridge, MA, 1993), pp. 109–138.

[59] C. Zetzsche and G. Krieger, "Nonlinear neurons and higher-order statistics: new approaches to human vision and digital image processing," Proc. SPIE **3644** (1999).

[60] C. Zetzsche and U. Nuding, "Nonlinear encoding in multilayer LNL systems optimized for the representation of natural images," Proc. SPIE **6492**, 649204 (2007).

[61] C. Zetzsche and F. Röhrbein, "Nonlinear and extra-classical receptive field properties and the statistics of natural scenes," Netw.: Comput. Neural Syst. **12**, 331 (2001).