Article

Supervised and Unsupervised End-to-End Deep Learning for Gene Ontology Classification of Neural In Situ Hybridization Images [†]

Ido Cohen [1], Eli (Omid) David [1,][*] and Nathan S. Netanyahu [1,2,3]

[1] Department of Computer Science, Bar-Ilan University, Ramat-Gan 5290002, Israel; cido15@gmail.com (I.C.); nathan@cs.biu.ac.il (N.S.N.)
[2] Gonda Brain Research Center, Bar-Ilan University, Ramat-Gan 5290002, Israel
[3] Center for Automation Research, UMIACS, University of Maryland at College Park, College Park, MD 20742, USA
[*] Correspondence: mail@elidavid.com
[†] This paper is an extended version of our paper published in the 26th International Conference on Artificial Neural Networks (ICANN 2017), Alghero, Italy, 11–14 September, 2017.

check for updates

Abstract: In recent years, large datasets of high-resolution mammalian neural images have become available, which has prompted active research on the analysis of gene expression data. Traditional image processing methods are typically applied for learning functional representations of genes, based on their expressions in these brain images. In this paper, we describe a novel end-to-end deep learning-based method for generating compact representations of *in situ hybridization* (ISH) images, which are invariant-to-translation. In contrast to traditional image processing methods, our method relies, instead, on deep *convolutional denoising autoencoders* (CDAE) for processing raw pixel inputs, and generating the desired compact image representations. We provide an in-depth description of our deep learning-based approach, and present extensive experimental results, demonstrating that representations extracted by CDAE can help learn features of functional *gene ontology categories* for their classification in a highly accurate manner. Our methods improve the previous state-of-the-art classification rate (Liscovitch, et al.) from an average AUC of 0.92 to 0.997, i.e., it achieves 96% reduction in error rate. Furthermore, the representation vectors generated due to our method are more compact in comparison to previous state-of-the-art methods, allowing for a more efficient high-level representation of images. These results are obtained with significantly downsampled images in comparison to the original high-resolution ones, further underscoring the robustness of our proposed method.

Keywords: deep learning; convolutional neural networks; denoising autoencoders; ISH images; gene categorization

1. Introduction

A very large volume of high-spatial resolution imaging datasets is available these days in various domains, calling for a wide range of exploration methods based on image processing. One such dataset has become recently available in the field of neuroscience, thanks to the Allen Institute for Brain Science. This dataset contains *in situ hybridization* (ISH) images of mammalian brains, in unprecedented amounts, which has motivated new research efforts [1–3]. ISH is a powerful technique for localizing specific nucleic acid targets within fixed tissues and cells; it provides an effective approach for obtaining temporal and spatial information about gene expression [4]. Images now reveal highly complex patterns of gene expression varying on multiple scales.

Developing analytical tools for discovering gene interactions from such data remains an open challenge due to various reasons, including difficulties in extracting canonical representations of gene activities from images, and inferring statistically meaningful networks from such representations. The challenge in analyzing these images is both in extracting the patterns that are most relevant functionally, and in providing a meaningful representation that would allow neuroscientists to interpret the extracted patterns.

One of the aims at finding a meaningful representation for such images is to carry out classification to *gene ontology* (GO) categories. GO is a major bioinformatics initiative to unify the representation of gene and gene product attributes across all species [5]. More specifically, it aims at maintaining and developing a controlled vocabulary of gene and gene product attributes and at annotating them. This task is far from done; in fact, several gene and gene product functions of many organisms have yet to be discovered and annotated [6]. Gene function annotations, which are associations between a gene and a term of controlled vocabulary describing gene functional features, are of paramount importance in modern biology. They are used to design novel biological experiments and interpret their results. Since gene validation through in vitro biomolecular experiments is costly and tedious, deriving new computational methods and software for predicting and prioritizing new biomolecular annotations would make an important contribution to the field [7]. In other words, deriving an effective computational procedure that predicts reliably likely annotations, and thus speeds up the discovery of new gene annotations, would be very useful [8].

Past methods for analyzing brain images need to reference a brain atlas, and are based on smooth nonlinear transformations [9,10]. These types of analyses may be insensitive to fine local patterns, like those found in the layered structure of the cerebellum (The cerebellum is a region of the brain. It plays an important role in motor control, and has some effect on cognitive functions [11].), or to a spatial distribution. In addition, most machine vision approaches address the challenge of providing human interpretable analysis. Conversely, in bioimaging, usually the goal is to reveal features and structures that are hardly seen even by human experts. For example, one of the new functions that follow this approach is presented in [12], using a histogram of local *scale-invariant feature transform* (SIFT) [13] descriptors on several scales.

Recently, several machine learning algorithms have been designed and implemented to predict GO annotations [14–18]. Following our preliminary work [19], we pursue in this paper an *artificial neural network* (ANN) with many layers (also known as *deep learning*), in order to achieve functional representations of neural ISH images.

Indeed, to obtain a compact feature representation of these ISH images, we further explore here *autoencoders* (AE) and *convolutional neural networks* (CNN). Specifically, we demonstrate that the *convolutional autoencoder* (CAE) is most appropriate for the task at hand. (This is similar to the work of Krizhevsky and Hinton [20], who used deep autoencoders to create short binary codes for content-based images.) Subsequently, we use this representation to learn features of functional GO categories for every image, invoking a simple L_2-regularized logistic regression classifier, as in [12]. As a result, each image is represented as a lower-dimensional vector whose components correspond to meaningful functional annotations. As pointed out in [12], the resulting representations define similarities between ISH images which could be explained, hopefully, by such functional categories.

Our experimental results further demonstrate that the representation obtained using the novel architecture of a so-called *convolutional denoising autoencoder* (CDAE) outperforms the previous state-of-the-art classification rate [12]; specifically, it improves the average AUC from 0.92 to 0.997, achieving 96% reduction in error. The method operates on input images that are downsampled significantly with respect to the original ones to make it computationally feasible.

2. Background

2.1. Biological Background

Mammalian brains vary in size and complexity (Figure 1) and are composed of billions of neurons and glia. The brain is organized in highly complex anatomical structures. This is why it has remained a fascinating challenge for many years.
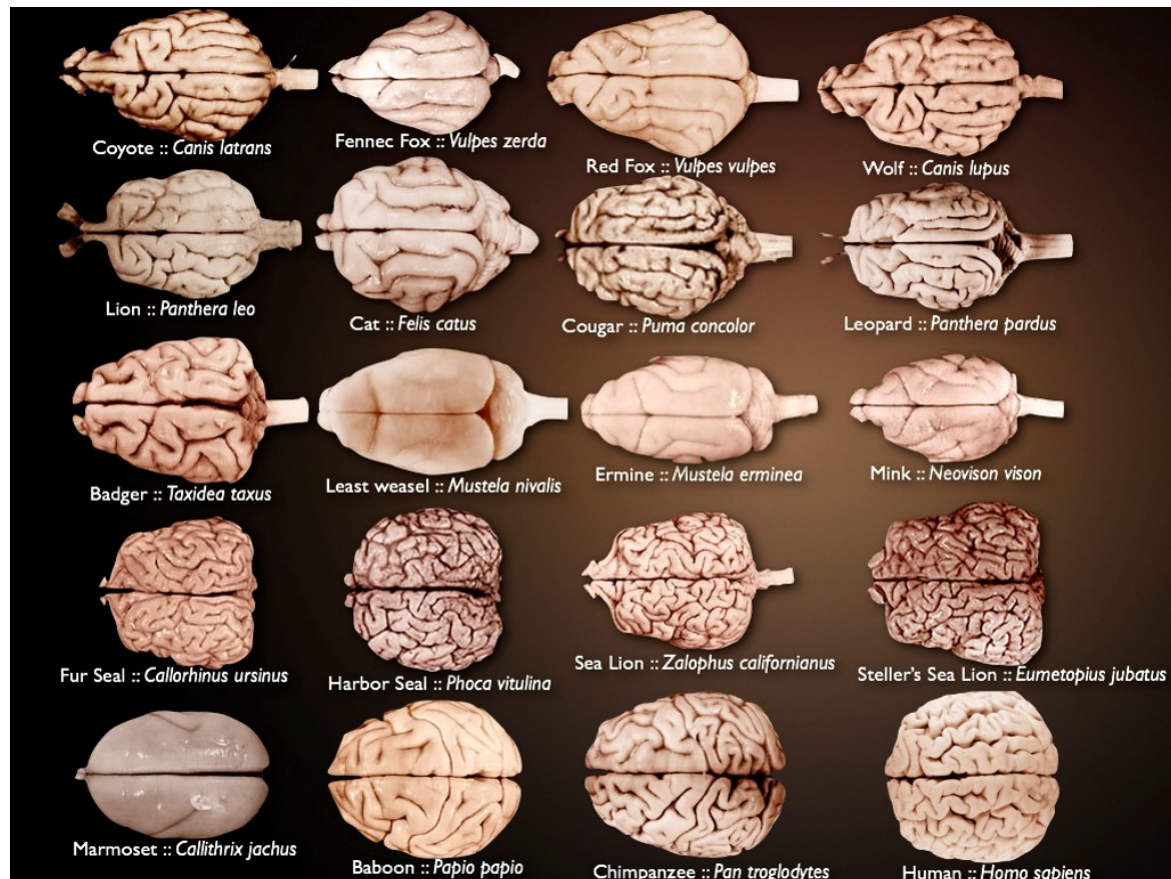

Figure 1. Different mammalian brains, varying in size and complexity (source: National Museum of Health and Medicine's University of Wisconsin and Michigan State Comparative Mammalian Brain Collections [21]).

The dataset of the Allen Institute for Brain Science contains a significant repository of ISH images of mammalian brains for many research fields. ISH is a powerful technique for localizing specific nucleic acid targets within fixed tissues and cells. It provides an effective approach for obtaining temporal and spatial information about gene expression [4]. ISH images reveal highly complex patterns of gene expression varying on multiple scales, as shown in Figure 2. This information calls for a wide range of exploration methods based on image processing.

Finding meaningful representations for such images is one of the main goals which could be used for classification of *gene ontology* (GO) categories. GO has the aim of unifying the representation of gene and gene product (A gene product is the biochemical material, either RNA or protein, resulting from expression of a gene.) attributes across all species [5]. GO categories are shaped in the form of a *directed acyclic graph* (DAG), in which each category has relevant subcategories, and vice versa. An example can be seen in Figure 3.
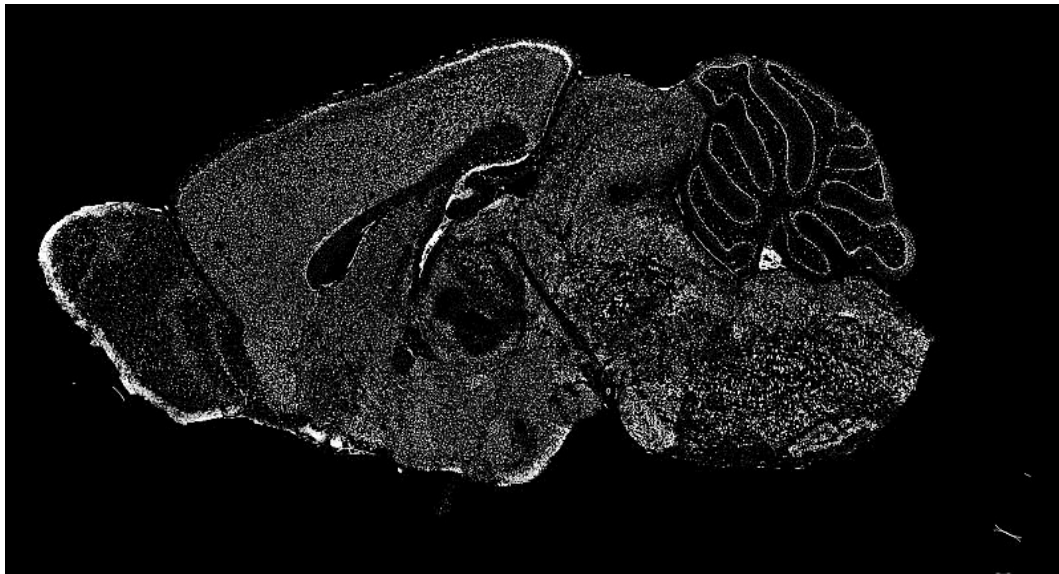
Figure 2. Typical in situ hybridization (ISH) image of a mammalian brain used for our research (in gray scale).
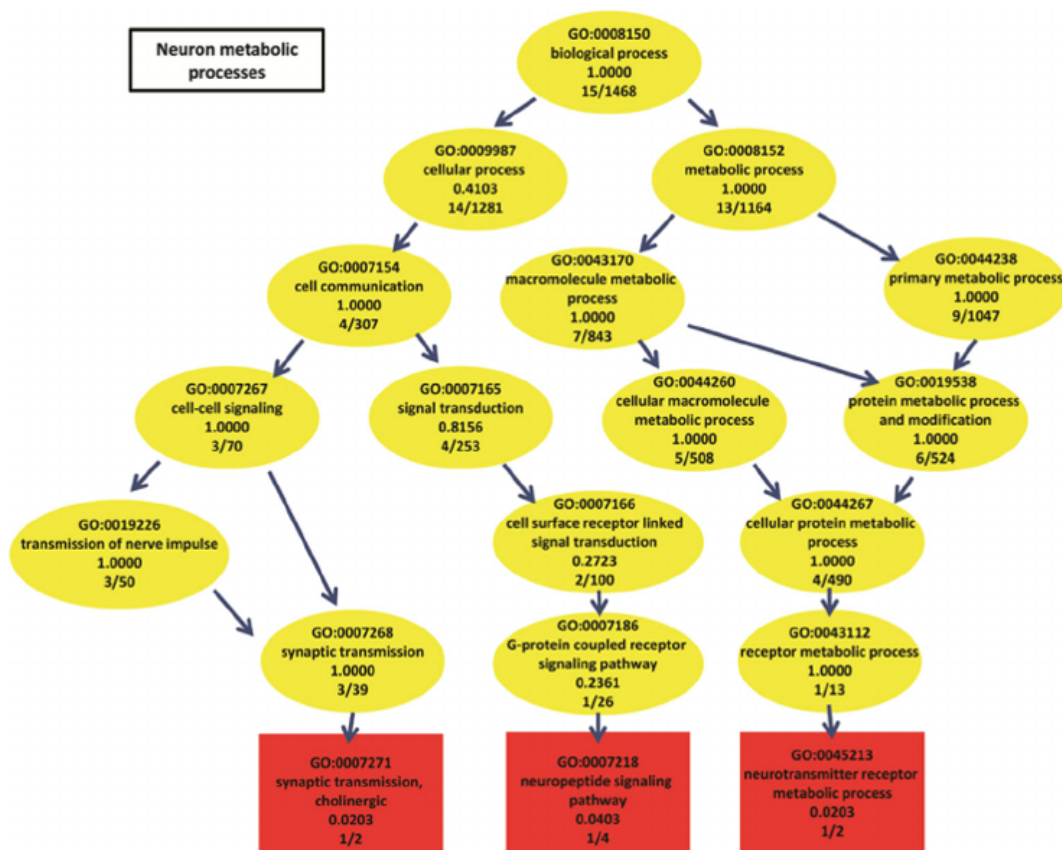

Figure 3. Neuron metabolic processes represented by a directed acyclic graph (source: The European Bioinformatics Institute (EMBL–EBI)).

Thousands of GO categories exist, and the task is far from done; in fact, several gene and gene product functions of many organisms have yet to be discovered and annotated [6]. Gene function annotations, which are associations between a gene and a term of controlled vocabulary describing gene functional features, are of paramount importance in modern biology. They are used to design novel biological experiments and interpret their results. With that said, gene validation through in vitro

biomolecular experiments is costly and lengthy. It is a multi-stage process, which starts by revealing the gene product and then searching for it in known databases. If it appears in the database, progress is possible; otherwise, a series of laboratory studies are needed to reconstruct the gene. Sometimes, it is a short process, but it could also be extremely tedious. For this reason, deriving new computational methods and software for predicting and prioritizing new biomolecular annotations would make an important contribution to the field [7]. In other words, as previously noted, deriving an effective computational procedure that predicts reliably likely annotations, and thus speeding up the discovery of new gene annotations, would be very useful [8].

2.2. FuncISH: Learning Functional Representations

ISH images of mammalian brains reveal highly complex patterns of gene expression varying on multiple scales. In [12], the authors present *FuncISH*, a learning method of functional representations of ISH images, using a histogram of local descriptors on several scales. They first represent each image as a collection of local descriptors of SIFT features [13,22]. Next, they construct a standard *bag-of-words* description of each image, giving a 2004-dimensional representation vector for each gene. Finally, given a set of predefined GO annotations of each gene, they train a separate classifier for each known biological category, using the SIFT bag-of-words representation as an input vector. Specifically, they construct a set of 2081 L_2-*regularized logistic regression* classifiers. The GO categories considered are those for which the number of annotations range between 15 to 500. The lower limit is picked to provide enough positive examples for testing, while the higher limit is chosen to preclude the resulting semantic explanations from being too general. A scheme taken from [12] representing the work flow is presented in Figures 4 and 9.
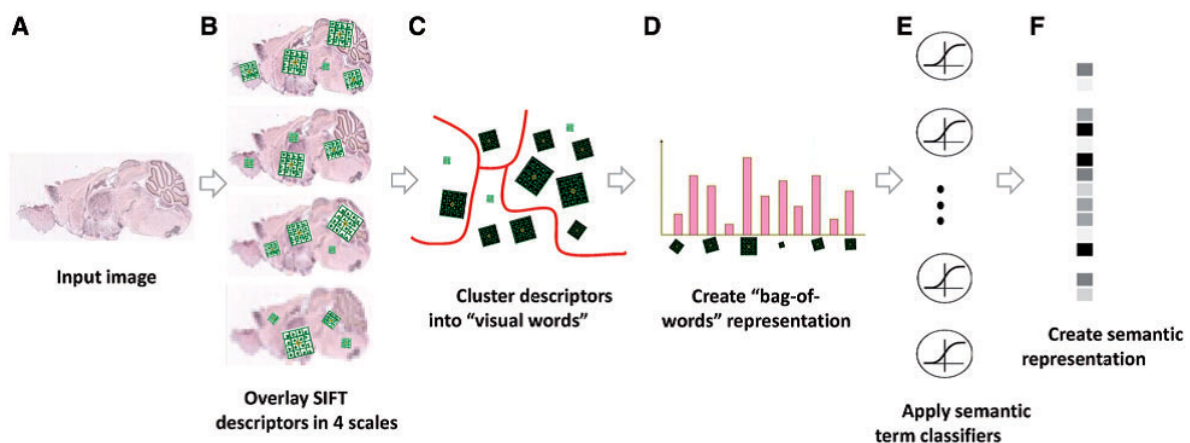

Figure 4. Illustration of the image processing pipeline: (**A**) original image in pixel grayscale indicating level of gene expression; (**B**) local SIFT descriptors are extracted from image at four resolutions; (**C**) descriptors from all 16,351 images are clustered into 500 representative "visual words" for each resolution level using *k*-means; (**D**) each image is represented as a histogram counting the occurrences of visual words; (**E**) L_2-regularized logistic regression classifiers applied to 2081 GO categories; and (**F**) final 2081 dimensional image representation (source: Liscovitch et al. [12]).

Applying [12] to the genomic set of neural mouse ISH images (available from the Allen Brain Atlas) reveals that "most neural biological processes could be inferred from spatial expression patterns with high accuracy". Despite ignoring important global location information, ∼700 functional annotations were successfully inferred, and then used to detect gene–gene similarities not captured by previous, global correlation-based methods. According to [12], combining local and global patterns of expression is an important topic for further research, e.g., the use of more sophisticated nonlinear classifiers.

Pursuing further the above classification problem poses a number of challenges. First, defining a certain set of rules that an ISH image has to conform to in order to classify it to the correct GO category. Conventional computer vision techniques, although capable of identifying shapes and objects in an

image, seem unlikely to provide effective solutions to the problem of interest. Thus, following the work in [12], we use deep learning to achieve more accurate results, due to the more effective functional feature representation of the ISH images learned by our neural network.

2.3. Convolutional Neural Networks (CNNs)

CNNs are variations of *multilayer perceptrons*, designed to process data that come in the form of multiple arrays, e.g., [23–25]. 1D, 2D, and 3D CNNs are used typically for processing signals/sequences, images/spectrograms, and video/volumetric images, respectively. There are four main components associated with CNNs, all of which exploit the properties of natural signals; these are local connections, shared weights, pooling, and the use of many layers.

A typical CNN architecture (Figure 5) is structured as a series of components from two types, convolutional layers and pooling layers. Overall, units in a convolutional layer are organized in feature maps, where each unit is connected to a local region of units in the feature maps of the previous layer, through a set of weights called a filter. This local weighted sum is then passed through an *activation function*, such as $\text{ReLU}(x) = \max(0, x)$ or $\tanh(x) = (e^x - e^{-x})/(e^x + e^{-x})$, which allows such networks to solve nontrivial problems. All units in a feature map share the same filter weights, while different feature maps in a layer use a different filter. The mathematical advantage is that the number of weights to be learned is not dependent on the number of units, but on the number of feature maps and the size of the filter. This is reasonable because in spatial data such as images, often local values are highly correlated, due to the fact that image statistics are invariant to location. In other words, if a pattern appears in one part of the image, it could appear anywhere in the image.

While the role of a convolutional layer is to detect locally connected features from the previous layer, the role of a pooling layer is to merge semantically similar features into one. Pooling layers subsample the data, thus reducing the representation size. For example, a typical pooling method, called max-pooling, selects only the maximum value of a local region of units in one (or more) feature maps (e.g., selects only the maximum value for each 2×2 region, thus decreasing the size by a factor of 4). Usually, two or three stages of convolution (using nonlinear functions) and pooling layers are stacked, followed by some fully-connected layers and a final classification layer. Designing a CNN to cope with the relative large size of our input images might be the answer, although it does not provide a solution to our second issue, namely the small number of training samples for each category.
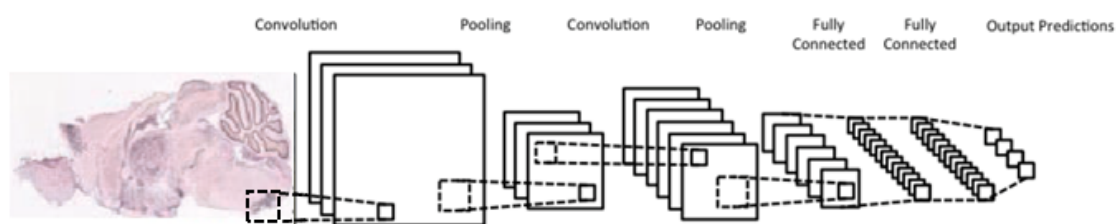

Figure 5. Typical architecture of a convolutional neural network; consists of two convolutional layers (each immediately preceded by a pooling layer), followed by two fully-connected layers, with a final classification layer.

2.4. Auto-Encoders (AEs)

CNNs are effective in a supervised framework [20,24–27], provided a large training set is available. If only a small number of training samples is available, unsupervised pre-training methods, such as *restricted Boltzmann machines* (RBM) [28] or autoencoders [29], have proven highly effective.

An AE is a neural network which sets the target values (of the output layer) to be equal to the input values, using hidden layers of smaller and smaller size, which comprise a bottleneck. Thus, an AE can be trained in an unsupervised manner, forcing the network to learn a higher-level

representation of the input. One or more hidden layers of neurons are used between the input and output layers, where each layer is usually set to have fewer neurons than those in the input and output layers, thus creating a bottleneck structure. In this form, each input is first mapped to a hidden layer (smaller than the input layer), with the objective of reconstructing the original input at the output layer. Note that for a single hidden layer, the set of weights w between the input layer and the hidden layer, called the "encoder layer", and the set of weights w' between the hidden layer and the output layer, called the "decoder layer", can be tied (by setting $w' = w^T$). Similarly, an AE with more than one hidden layer can maintain tied weights (A typical equation for a feedforward neural network is $f(x) = \sigma_2(b_2 + w_2\sigma_1(b_1 + w_1x))$, while for autoencoders with tied weights, the equation used is $f(x) = \sigma_2(b_2 + w_1^T\sigma_1(b_1 + w_1x))$. Setting $w_2 = w_1^T$ eliminates many degrees of freedom, which has its important advantages.) between the corresponding encoder and decoder layers. In any case, AEs are typically trained using backpropagation with stochastic gradient descent to reduce the reconstruction error [30].

An improved approach, which outperforms basic autoencoders in many tasks, is due to *denoising autoencoders* (DAEs) [29,31]. These are built as regular AEs, where each input is corrupted by added noise, or by setting to zero some portion of the values. Although the input sample is corrupted, the network's objective is to produce the original (uncorrupted) values in the output layer (see Figure 6). Forcing the network to recreate the uncorrupted values results in reduced network overfitting (also due to the fact that the network rarely receives the same input twice), and in extraction of more high-level features. Note that the noise is added only during training, while in prediction time the network is given the uncorrupted input.
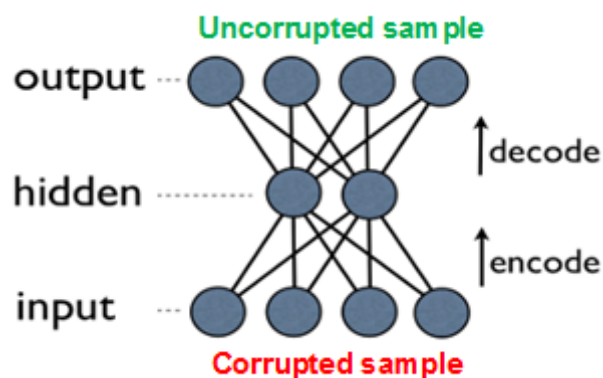

Figure 6. Denoising autoencoder with one layer; during training, the input sample is corrupted, while the expected output is the original, uncorrupted input.

For any autoencoder-based approach, once training is complete, the decoder layer(s) are removed, such that a given input is transferred through the network and yields a high-level representation of the data. In most implementations (such as ours), these representations can then be used for supervised classification.

In the next section, we present our convolutional autoencoder approach, which operates solely on raw pixel data. This supports our main goal, i.e., learning representations of given brain images to extract useful information more easily, when building classifiers or other predictors. The representations obtained are vectors, which can be used to solve a variety of problems, e.g., that of GO classification. We will consider a representation as promising, if it proves helpful for building accurate supervised classifiers for the biological categories in question.

2.5. Convolutional Autoencoders (CAEs)

CNNs and AEs can be combined to produce a CAE, which, as mentioned in [32], is one of the most preferred architectures in deep learning. This may explain why several approaches involving the

combination of these methods have been explored previously. One problem with AEs and DAEs is that both do not account for the local 2D image structure and thus may not be translational-invariant. Furthermore, when using fully-connected AEs, realistic input sizes may introduce redundancy in the parameters to be learned and increase their number significantly. However, CNNs discover repetitive localized features all over the input, such as in successful models like [33,34]. As with CNNs, the CAE weights are shared among all locations in the input, preserving spatial locality and reducing the number of parameters. A common convention for combining CNNs with AEs (or DAEs), which maintains the input size, is to have for each encoder layer a corresponding decoder layer, i.e., each convolutional layer would have a *deconvolutional* layer, and each max-pooling layer would have an *unpooling* layer that restores the original values lost by the max-pooling subsampling.

Deconvolutional layers are essentially the same as convolutional layers, and, similarly to standard autoencoders, they can either be learned or be set equal to (the transpose of) the original convolutional layers, as with tied weights in autoencoders.

There are a number of variants to the *unpooling* operation (see, e.g., [35–37]). In our CAE, unpooling simply duplicates the same maximal value in the corresponding 2×2 window of the next layer (Figure 7). This means that there is no need to record the locations of the maxima within each pooling region.
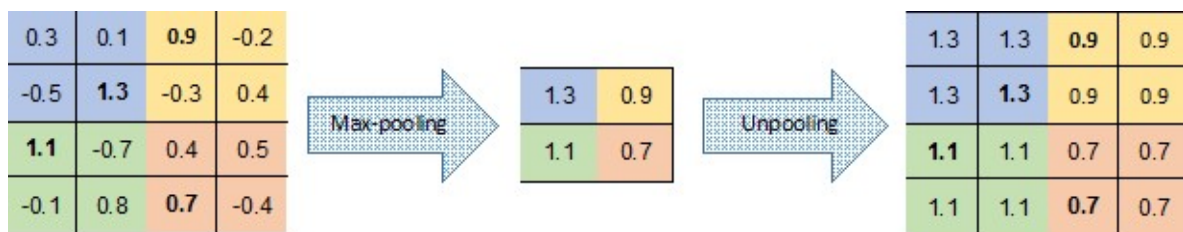

Figure 7. Pooling and unpooling layers; for each pooling layer, the max value is kept, and then duplicated in the unpooling layer.

Similarly to an AE, after training a CAE, the unpooling and deconvolutional layers are removed. At this point, a neural network, composed from convolution and pooling layers, can be used to find a functional representation, as in our case, or initialize a supervised CNN. An example of a complete structure of a convolutional autoencoder is presented in Figure 8.

Figure 8. Illustration of convolutional autoencoder; the CAE consists of two convolutional layers and their two corresponding deconvolutional layers, and two max-pooling layers and their corresponding unpooling layers.

Similarly to a DAE, a CAE with input corrupted by added noise is called a *convolution denoising autoencoders* (CDAE).

2.6. Challenges of Applying End-to-End Deep Learning for Annotating Gene Expression Patterns

In this section we discuss challenges present in applying end-to-end deep learning for functional representation and methods for overcoming them.

In the traditional approach to problems of this nature, raw data (e.g., pixels) are not fed directly into the training module, and, instead, a feature extraction phase is first performed, where raw data are converted into a list of features. When applying end-to-end deep learning, however, raw data are directly fed into the input layer of a deep neural network. The idea is that the neural network would perform *implicit feature learning*, and learn more complex and meaningful nonlinear features that would be superior to the handcrafted features by human experts.

While the results obtained due to deep learning represent major improvements over the traditional approach in nearly every domain, deep learning requires substantially larger amounts of training data. While in the traditional approach the model already receives useful features, and needs only to learn how to use those features for successful prediction, in deep learning, the features themselves need to be learned as well.

In the case of gene annotation of expression patterns, we only have thousands of images for training, which are not sufficient, by themselves, for obtaining optimal results. One successful approach in such a case is to use *transfer learning*, i.e., first train the neural network on a different dataset (for which ample data are available), and then use that pretrained neural network for training on the task at hand. The idea is that the neural network would learn to perform feature extraction by training it on the larger dataset, and then fine-tune the feature set for the desired task. This would work well assuming that the larger dataset and the target dataset have a similar sample distribution. (i.e., concepts learned from the larger dataset are transferable to the other).

Zeng et al. [38] present such a transfer learning approach; in order to obtain a CNN for annotating, eventually, gene expression patterns, they first train a model from *OverFeat* [39] on natural images (during the pre-training phase), and then employ this pre-trained network for extracting features of ISH images. This rationale stems from recent studies of using *ImageNet* data [25] (an image dataset with thousands of categories and millions of labeled natural images) in training a CNN model, followed by feature extraction (by the model) from other datasets for obtaining overall promising performance.

Using the OverFeat model, Zeng et al. first resize all images to 231×231 pixels, as required by the model. Each ISH image is then processed by the CNN, extracting feature vectors only from the last layer in each stage. These vectors are flattened and concatenated into a single feature vector for each layer. Finally, to build the representation vector, features are extracted for each section separately and an element-wide maximum across feature vectors, from all sections of the same experiment of a gene expression, is computed. This global feature vector of size 10,521 represents the gene expression of a given ISH image. The representation is then used to perform gene expression pattern annotation, similarly to [12], i.e., by training on L_2-regularized logistic regression classifiers. The overall average AUC obtained was 0.894 ± 0.014, compared with 0.820 ± 0.046, due to a similar bag-of-words approach presented in [12].

Even though the results of Zeng et al. show some improvement obtained by transfer learning, these gains are limited, since the ImageNet dataset they have used for pretraining contains natural images, which are vastly different from images intended for gene annotation. In other words, pretraining a neural network on real-world natural images might not be entirely beneficial for its subsequent training for the gene attribution task in question.

In the next section, we present our novel method, which uses pretraining on the same sample distribution (as in [12]), via our proposed unsupervised learning. This pretraining feature learning will prove more effective for the desired task, in terms of the eventual accuracy gained.

3. Proposed Approach

We now discuss our approach for generating a functional representation of ISH images. Deep learning methods have proven recently very successful in finding compact feature representations for many tasks. These networks are trained extensively, in an attempt to capture object representations that are invariant to changes in location, and to some extent also to scale, orientation, lighting, etc. CNNs and AEs, as described in the Background, achieve small errors when used in supervised

and unsupervised learning modes, respectively. Challenging tasks tackled in a supervised learning mode are, e.g., ImageNet [25] and large-scale image recognition [40]; in an unsupervised mode, such tasks include, e.g., mimicking faithfully biological measurements of the visual areas [41], building class-specific feature detectors from unlabeled images [42], or creating short binary codes for images based on their content, using deep autoencoders according to Krizhevsky and Hinton [20].

Our research follows [12], but with a modified approach for learning the gene representation and training the GO classifiers. We use CNNs and CAEs to learn the representation, and then use the same classification method as in [12] to measure the classification accuracy (that obviously depends on our different representation). The scheme representing the work flow of [12] is presented in Figure 9, while our work flow is presented in Figure 10 (see Section 3.1).
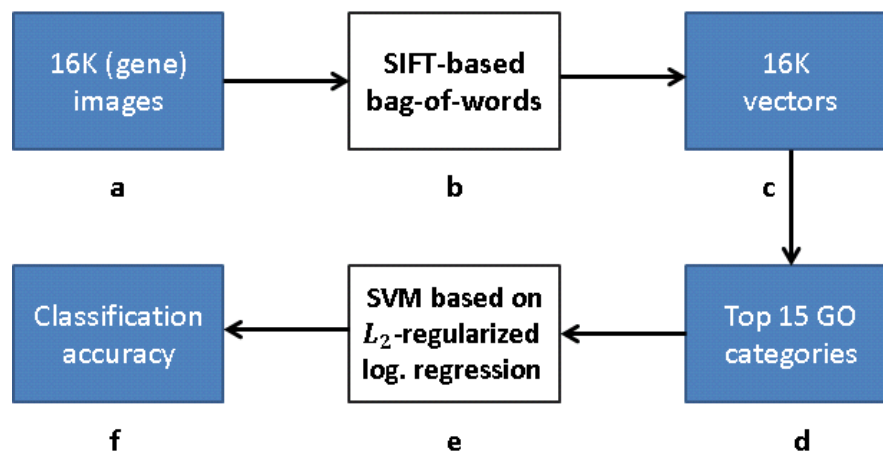

Figure 9. Processing pipeline [12]: (**a**) 16K original images in pixel grayscale indicating level of gene expression; (**b**) "bag-of-words" via clustering clustering of SIFT feature descriptors; (**c**) vector representation due to bag-of-words; (**d**) 16K vectors trained with respect to each of 15 GO categories with best AUC classification in [12]; (**e**) L_2-regularized logistic regression classifiers for top 15 GO categories; and (**f**) classification accuracy obtained.

We applied our method to the genomic set of neural mouse ISH images available from the Allen Brain Atlas. This dataset contains whole-brain, expression-masked images of gene expression measured using ISH. For each gene, a different adult mouse brain was sliced into 25 slices, each 100 mm thick. We used the most medial slice for each series. In this way, the expression was measured for the entire mouse genome, for a total of 15,000 genes. That is, the dataset includes 16,351 images representing 15,612 genes. Each image reveals highly-complex patterns of gene expression varying on multiple scales.

To learn detectors of functional GO categories, using the representation obtained for every gene, we introduce a novel CDAE architecture for the problem in question, to find the compact representation of each ISH image. The CDAE includes pooling and unpooling layers, as well as convolution and deconvolutional layers, as was explained in the Background. The input is corrupted for denoising purposes, and should be normalized, in principle, by subtracting its mean from each pixel value and dividing by its standard deviation. As for any AE, layers are trained in an unsupervised manner, using backpropagation for updating the network's weights; this is a well-known procedure for feature extraction to gain a compact representation. After the CDAE is trained, the deconvolution and unpooling layers are removed, so that the middle (and smallest) hidden layer becomes the output layer. Feeding this network a single ISH image at a time, the resulting output serves as the compact representation of this image, i.e., the representation of the corresponding gene. These representations are vectors which can then be used to solve a variety of problems, including GO classification.

3.1. CDAE for GO Classification

Figure 10 depicts a framework for capturing the representation similar to FuncISH. Recall that a SIFT-based module was used in [12] for feature extraction. Our scheme learns, alternatively, a CDAE-based representation, before carrying out the GO classification.

For the unsupervised training of our CDAE we use the genomic set of neural mouse ISH images available from the Allen Brain Atlas, which includes 16,351 images representing 15,612 genes. These JPEG images have an average resolution of $16,000 \times 8000$ pixels. To obtain a representation vector of size $\sim$2000, the images were resampled initially to 960×480 pixels, and were eventually downsampled (via successive max unpooling operations) to 60×30 pixels. Despite this heavy down-sampling, we believe that local and global patterns of expression are still preserved to a sufficient extent, in the sense that significant differences between ISH images of different genes can still be captured, thus rendering their representation at the above spatial resolution as meaningful. This working assumption is supported by the down-sampling done also in [12], which succeeded in generating adequate representation, as well as by our own results, as detailed below.

Regarding the supervised GO classification, we used the L_2-regularized logistic regression classifier as described in [12]; specifically, a two-phase *5-fold cross validation* was invoked, per each category, for training the classifier and tuning the logistic regression regularization hyper-parameter. Training requires careful consideration, in this case, due to the highly-imbalanced nature of the training sets. While in [12] the full set of 16K gene images was split into five non-overlapping equal sets (without controlling the number of "positives" in each split), we split the "positives" into five non-overlapping equal subsets and the "negatives" into five non-overlapping equal subsets, and combined a different positive subset with each negative subset to obtain five non-overlapping equal-sized subsets, per each category. We then trained the classifiers on four of these subsets, and tested the performance on the fifth. This procedure was repeated five times, each time with a different test subset.
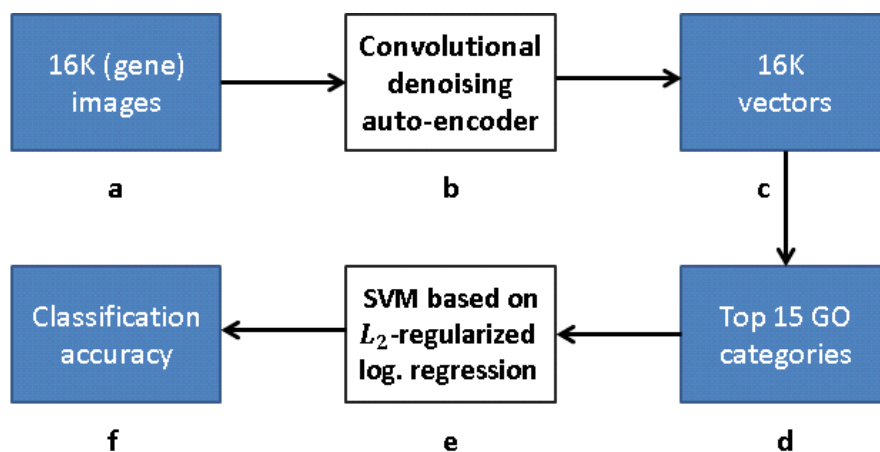

Figure 10. Our work flow: (**a**) 16K grayscale images indicating level of gene expression; (**b**) CDAE-based feature extraction for a compact vector representation of each gene; (**c**) vector representation due to feature extraction; (**d**) 16K vectors trained with respect to each of 15 GO categories with best AUC classification in [12]; (**e**) L_2-regularized logistic regression classifiers for top 15 GO categories, and (**f**) classification accuracy obtained.

4. Empirical Results

4.1. Determination of Hyper-Parameters

In this section, we discuss the determination of various hyper-parameters for training, and present a comprehensive set of experimental results. In principle, the hyper-parameters were chosen by running the system, each time, on a large range of possible values for a given hyper-parameter (while keeping the others fixed), and looking for those values which provide the best results. Using insights

gained during this extensive experimentation allowed us to converge faster, in some cases, to optimal hyper-parameter values, without having to run the system on all possible values.

4.1.1. Denoising Rate

As described in Section 2.4, the idea behind denoising autoencoder (DAE), in general, and our convolution denoising autoencoder (CDAE), in particular, is to force the hidden layer to capture more robust features, and prevent it from simply learning the identity. This is achieved by training the CDAE to reconstruct the input from a corrupted version of it.

We use the same stochastic corruption process as in [29], which randomly sets a predefined amount of the image pixels to zero. Hence, the CDAE is forced to predict the missing values from the uncorrupted values, by reconstructing them for randomly selected subsets of missing patterns. Figure 11 presents test results versus the percentage (from 0% to 80%) of pixels randomly set to zero, for input images of different size. The blue plot in the figure corresponds to input images resized to 200×200 pixels, yielding (after two max-pooling layers) a compact, ($50 \times 50 =$) 2500-dimensional vector representation. The green and orange plots both correspond to a compact, 60×30 (vector) representation, but they originate from input images resized to 480×240 pixels and 240×120 pixels, respectively. Note that in order to reach the same vector dimensionality, we use an additional pooling layer (and a corresponding unpooling layer) in the network that inputs images of size 480×240 pixels.
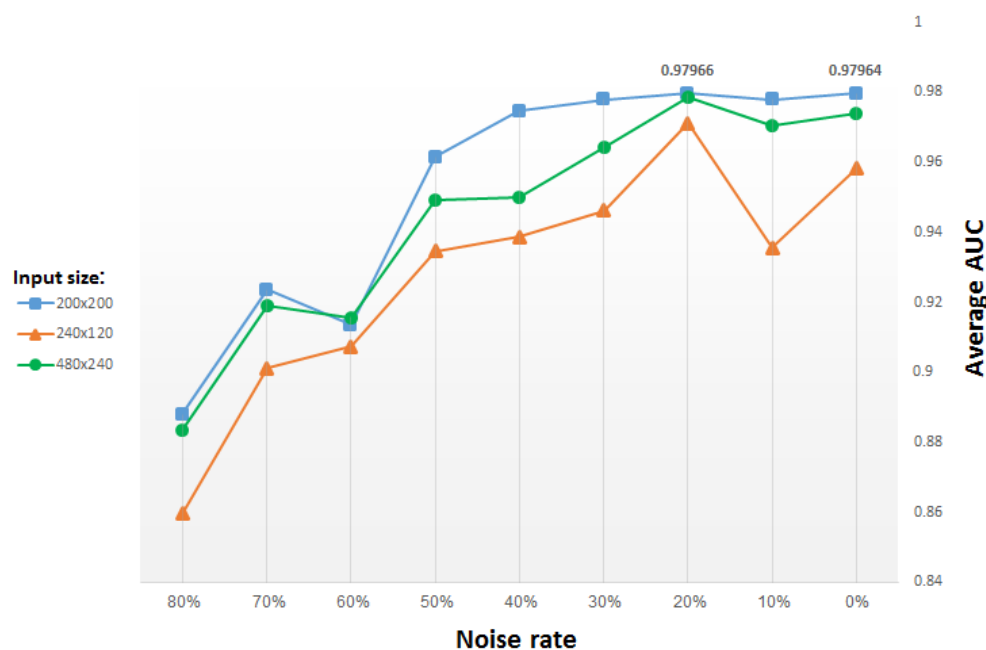

Figure 11. Average AUC for classifiers of top 15 GO categories versus denoising rate (i.e., percentage of pixels randomly set to zero), using CDAE functional representation; each plot corresponds to a different input image size, with other parameters fixed.

For each input size, the average AUC drops when the denoising rate increases above 30% (i.e., when 30% pixels selected randomly are set to zero). For small denoising rates, the average results improve slightly for added noise, reaching a peak for a denoising rate of roughly 20%. Thus, we use for most of our study a denoising rate of 20%, for which the best results are obtained.

4.1.2. Filter Size

As mentioned in Section 2.3, a typical CNN, or a CDAE, is composed of a series of convolution and pooling layers. The units in the convolutional layers are organized in feature maps, where each feature map unit is connected to a local region of units in the (feature maps of the) previous layer,

through a set of weights known as a filter. In this way, the convolutional layer performs its function in the network, which is to detect locally connected features from the previous layer. This is an important role in spatial data such as images, where often local values are highly correlated. For this reason, the number of weights to be learned is not dependent on the number of neurons, but on the number of feature maps and the size of the filter.

There are certain criteria for selecting an optimal filter size. It is important to note the effects of a small filter size versus a large one. If a network is required to detect/recognize an object which consists of a large number of pixels, a relatively large filter (e.g., 11×11 or 9×9) may be used. Otherwise, a smaller filter (e.g., 3×3 or 5×5) should be used. In addition to feature capturing, the number of weights to be learned also depends on the filter size; larger filters can increase the training time of the network, and even prevent it from converging, due to too many parameters to be learned. Regarding the intended gene expression based on processing the available ISH brain images, we can think of features distinguishing between genes as relatively small.

Figure 12 compares the best results obtained as a function of filter size, using a CDAE for images (and representation vectors) of different sizes. (All CDAEs have a fixed number of eight feature maps for every convolutional layer.) As suspected, a 3×3 filter gives the best results for the problem in question.
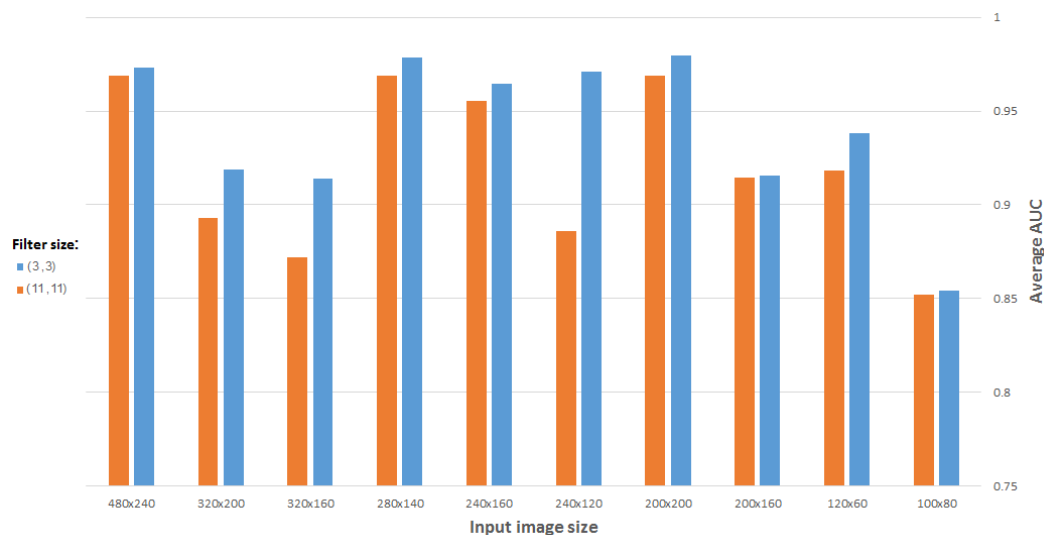

Figure 12. Average AUC versus filter size, using CDAE for different image sizes.

4.1.3. Representation Vector Size

As discussed in Section 2.2, Ref. [12] presents FuncISH, a learning method of functional representations of ISH images, using a histogram of local descriptors on several scales. Their method captures compact representation vectors of size 2004 out of the neural ISH images. Our aim is to find as small a representation as possible that would still adequately represent the gene related to the ISH image, and classify it correctly to the GO categories.

Figure 13 presents the maximum (over all runs) of the average AUC results for different representation vector sizes, using our CDAE. The plots depicted on the right- and left-hand side of the figure correspond to two and three max-pooling layers, respectively. (We used here merely two and three max-pooling layers on smaller input images, in order to save considerable run-time during the experimentation. The performance of the system is reported eventually for the resampled images of size 960×480 pixels using four max-pooling layers.) The filter size was fixed to 3×3, while other network hyper-parameters (e.g., learning rate, number of convolutional layers, etc.) were experimented with to find the best results. The following general observation can be made: the larger the representation vector, the better the performance. The combined objective should thus be to work

with the largest possible images and with as compact representation vectors as possible, while still meeting a required level of classification accuracy.
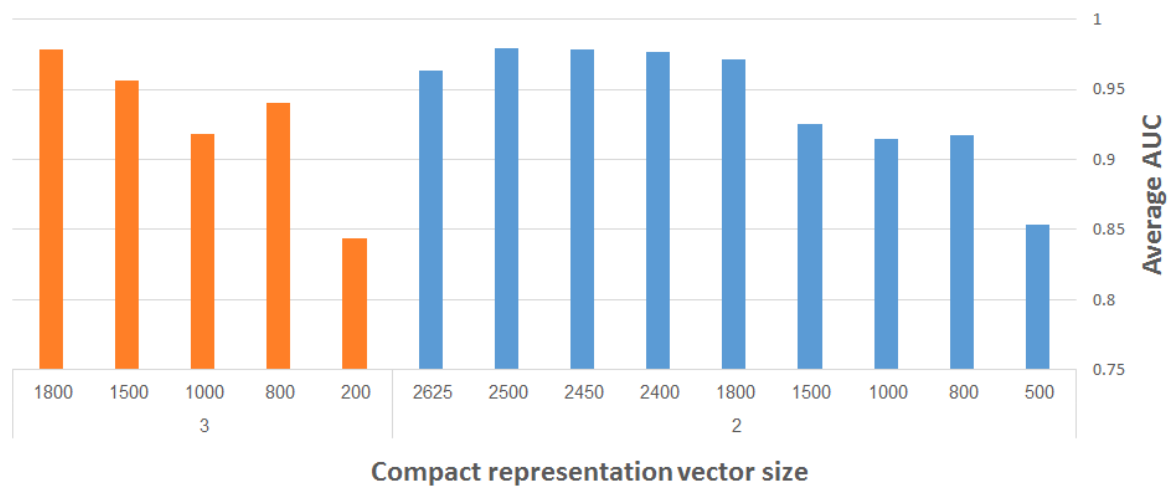

Figure 13. Maximum average AUC versus representation vector size, using CDAE with fixed-size (3×3) filters; results on the right- and left-hand side correspond to use of two and three max-pooling layers, respectively.

We now show how to improve the results of FuncISH [12], in conjunction with reducing the size of the representation vector below 2004.

4.2. Best AUC Results

The CDAE architecture for finding a compact representation for these down-sampled images is as follows (see Figure 14):

(1) **Input layer**: Consists of the raw image, resampled to 960×480 pixels, and corrupted by setting to zero 20% of the pixels;
(2) Two sequential convolutional layers with four 3×3 filters each;
(3) Max-pooling layer of size 2×2;
(4) Two sequential convolutional layers with four 3×3 filters each;
(5) Max-pooling layer of size 2×2;
(6) Two sequential convolutional layers with four 3×3 filters each;
(7) Max-pooling layer of size 2×2;
(8) Two sequential convolutional layers with four 3×3 filters each;
(9) Max-pooling layer of size 2×2;
(10) One convolutional layers with four 3×3 filters each;
(11) One convolutional layers with a single 3×3 filters each;
(12) Unpooling layer of size 2×2;
(13) Two sequential deconvolutional layers with four 3×3 filters each;
(14) Unpooling layer of size 2×2;
(15) Two sequential deconvolutional layers with four 3×3 filters each;
(16) Unpooling layer of size 2×2;
(17) Two sequential deconvolutional layers with four 3×3 filters each;
(18) Unpooling layer of size 2×2;
(19) Two sequential deconvolutional layers with four 3×3 filters each;
(20) Deconvolutional layer with a single 3×3 filter, and
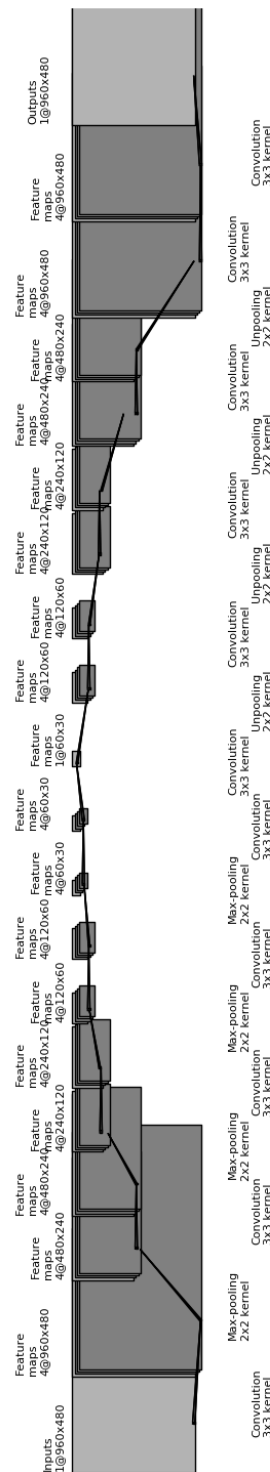(21) **Output layer** with the uncorrupted resampled image.

Figure 14. Illustration of our convolutional autoencoder, used to find a compact representation for the resampled images.

The network was initialized at random. In addition to the above parameters, we picked stride = 1, and used the ReLU(.) activation function for all convolution and deconvolutional layers, except for the last deconvolutional layer, which uses tanh(.). The learning rate starts from 0.05 and is multiplied by 0.9 after each epoch, and the denoising effect is obtained by randomly removing 20% of the pixels of every input image (i.e., 20% of the neurons in the input layer), as was pointed out in Section 4.1.1.

After training the CDAE, all layers past item 12 are removed, so that item 11 (the convolutional layer of size 1×1800) becomes the output layer. Therefore, each image is mapped to a vector of 1800 functional features. Given a set of predefined GO annotations for each gene (where each GO category consists of 15–500 genes), we train a separate classifier for each biological category, similarly to [12]. Specifically, we train an L_2-regularized logistic regression, using 5-fold cross-validation, as explained previously (see Section 3.1).

Our network yields remarkable AUC results for every category among the top 15 GO categories in [12], i.e., the categories for which the best AUC scores were obtained by their FuncISH method, based on SIFT descriptor representations. Using the compact representation extracted from our proposed CDAE architecture, coupled with the trained category classifiers, an exemplary AUC score of 1.0 was achieved for 13 of the 15 categories. The AUC values for the other two categories were significantly better than those in [12] (0.999 vs. 0.87 and 0.96 vs. 0.86). While the average AUC score (of the top 15 categories) in [12] is **0.92**, the average AUC for the same 15 GO categories using our CDAE is **0.997**, i.e., a **96%** reduction in error.

In addition, using this compact representation to classify all genes into the 2081 GO categories yields an extraordinary average AUC result of **0.988**. Compared to that of [38], with an average AUC of 0.894 ± 0.014, we obtain an error reduction of **87%**.

What is also remarkable is that we were able not only to improve the accuracy of the classification, but to reduce the size of the compact representative vector by more than **10%** relatively to [12], and by more than **80%** with respect to the previous deep learning approach in [38], i.e., to obtain a vector length of 1800 from vector lengths of 2004 and 10,521, respectively.

Thus, we succeeded in meeting the two main goals: (1) improving the classification of GO categories using this representation, and (2) reducing the size of the representative vector.

5. Conclusions

Many machine learning algorithms have been designed recently to predict GO annotations. For the task of learning meaningful functional representations of mammalian neural images, we employed in this paper deep learning techniques, and found convolutional denoising autoencoders to be very effective. Specifically, the presented scheme for feature learning of functional GO categories improved the previous state-of-the-art classification accuracy from an average AUC of 0.92 to 0.997, i.e., a 96% reduction in error rate. In fact, we managed to classify 86% of the categories with an exemplary AUC of 1.

Furthermore, we demonstrated how to reduce the vector dimensionality by 10% compared to the SIFT vectors used in [12], with very little degradation in accuracy.

Our results attest to the advantages of deep convolutional autoencoders, and especially the novel architecture as was applied here, for extracting meaningful information from very high resolution images and highly complex anatomical structures. Until gene product functions of all species are discovered, the use of CDAEs may well continue to serve in the ongoing design of novel biological experiments.

Author Contributions: Conceptualization, I.C., E.O.D. and N.S.N.; methodology, E.O.D.; software, I.C.; validation, I.C. and E.O.D.; investigation, I.C.; resources, N.S.N.; data curation, I.C.; writing—original draft preparation, I.C.; writing—review and editing, E.O.D. and N.S.N.; supervision, N.S.N.; project administration, N.S.N.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Lein, E.S.; Hawrylycz, M.J.; Ao, N.; Ayres, M.; Bensinger, A.; Bernard, A.; Boe, A.F.; Boguski, M.S.; Brockway, K.S.; Byrnes, E.J.; et al. Genome-wide atlas of gene expression in the adult mouse brain. *Nature* **2007**, *445*, 168–176. [CrossRef] [PubMed]
2. Ng, L.; Bernard, A.; Lau, C.; Overly, C.C.; Dong, H.W.; Kuan, C.; Pathak, S.; Sunkin, S.M.; Dang, C.; Bohland, J.W.; et al. An anatomic gene expression atlas of the adult mouse brain. *Nat. Neurosci.* **2009**, *12*, 356–362. [CrossRef] [PubMed]
3. Henry, A.M.; Hohmann, J.G. High-resolution gene expression atlases for adult and developing mouse brain and spinal cord. *Mamm. Genome* **2012**, *23*, 539–549. [CrossRef] [PubMed]
4. Puniyani, K.; Xing, E.P. GINI: From ISH images to gene interaction networks. *PLoS Comput. Biol.* **2013**, *9*, e1003227. [CrossRef] [PubMed]
5. The Gene Ontology Consortium. The gene ontology project in 2008. *Nucleic Acids Res.* **2008**, *36*, D440–D444. [CrossRef] [PubMed]
6. Ashburner, M.; Ball, C.A.; Blake, J.A.; Botstein, D.; Butler, H.; Cherry, J.M. Gene ontology: Tool for the unification of biology. *Nat. Genet.* **2000**, *25*, 25–29. [CrossRef] [PubMed]
7. Perez, A.J.; Perez-Iratxeta, C.; Bork, P.; Thode, G.; Andrade, M.A. Gene annotation from scientific literature using mappings between keyword systems. *Bioinformatics* **2004**, *20*, 2084–2091. [CrossRef] [PubMed]
8. du Plessis, L.; Skunca, N.; Dessimoz, C. The what, where, how and why of gene ontology—A primer for bioinformaticians. *Brief. Bioinform.* **2011**, *12*, 723–735. [CrossRef] [PubMed]
9. Davis, F.P.; Eddy, S.R. A tool for identification of genes expressed in patterns of interest using the Allen Brain Atlas. *Bioinformatics* **2009**, *25*, 1647–1654. [CrossRef] [PubMed]
10. Hawrylycz, M.; Ng, L.; Page, D.; Morris, J.; Lau C.; Faber, S.; Faber, V.; Sunkin, S.; Menon, V.; Lein, E.; et al. Multi-scale correlation structure of gene expression in the brain. *Neural Netw.* **2011**, *24*, 933–942. [CrossRef] [PubMed]
11. Wolf, U.; Rapoport, M.J.; Schweizer, T.A. Evaluating the affective component of the cerebellar cognitive affective syndrome. *J. Neuropsychol. Clin. Neurosci.* **2009**, *21*, 245–253. [CrossRef] [PubMed]
12. Liscovitch, N.; Shalit, U.; Chechik, G. FuncISH: Learning a functional representation of neural ISH images. *Bioinformatics* **2013**, *29*, i36–i43. [CrossRef] [PubMed]
13. Lowe, D.G. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vis.* **2004**, *60*, 91–110. [CrossRef]
14. Kordmahalleh, M.M.; Homaifar, A.; Dukka, B.K. Hierarchical multi-label gene function prediction using adaptive mutation in crowding niching. In Proceedings of the IEEE International Conference on Bioinformatics and Bioengineering, Chania, Greece, 10–13 November 2013; pp. 1–6.
15. King, O.D.; Foulger, R.E.; Dwight, S.S.; White, J.V.; Roth, F.P. Predicting gene function from patterns of annotation. *Genome Res.* **2013**, *13*, 896–904. [CrossRef] [PubMed]
16. Vembu, S.; Morris, Q. An efficient algorithm to integrate network and attribute data for gene function prediction. In Proceedings of the Pacific Symposium on Biocomputing, Big Island, HI, USA, 3–7 January 2014; pp. 388–399.
17. Zitnik, M.; Zupan, B. Matrix factorization-based data fusion for gene function prediction in baker's yeast and slime mold. In Proceedings of the Pacific Symposium on Biocomputing, Big Island, HI, USA, 3–7 January 2014; pp. 400–411.
18. Pinoli, P.; Chicco, D.; Masseroli, M. Computational algorithms to predict gene ontology annotations. *BMC Bioinform.* **2015**, *16*, S4. [CrossRef] [PubMed]
19. Cohen, I.; David, E.O.; Netanyahu, N.S.; Liscovitch, N.; Chechik, G. DeepBrain: Functional representation of neural in situ hybridization images for gene ontology classification using deep convolutional autoencoders. In Proceedings of the International Conference on Artificial Neural Networks, Alghero, Italy, 11–14 September 2017; pp. 287–296.
20. Krizhevsky, A.; Hinton, G.E. Using very deep autoencoders for content-based image retrieval. In Proceedings of the European Symposium on Artificial Neural Networks, Bruges, Belgium, 27–29 April 2011.
21. Comparative Mammalian Brain Collections. Available online: http://neurosciencelibrary.org/index.html (accessed on 20 Febuary 2019).

22. Brown, M.; Lowe, D.G. Recognising panoramas. In Proceedings of the 9th International Conference on Computer Vision, Nice, France, 13–16 October 2003; pp. 1218–1227.

23. LeCun, Y.; Boser, B.; Denker, J.S.; Henderson, D.; Howard, R.E.; Hubbard, W.; Jackel, L.D. Backpropagation 550 applied to handwritten zip code recognition. *Neural Comput.* **1989**, *1*, 541–551. [CrossRef]

24. LeCun, Y.; Bottou, L.; Bengio, Y.; Haffner, P. Gradient-based learning applied to document recognition. *Proc. IEEE* **1998**, *86*, 2278–2324. [CrossRef]

25. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. ImageNet classification with deep convolutional neural networks. In Proceedings of the Advances in Neural Information Processing Systems, Lake Tahoe, NV, USA, 3–6 December 2012; pp. 1106–1114.

26. Behnke, S. Hierarchical neural networks for image interpretation. *Lect. Notes Comput. Sci.* **2003**, *2766*, 1–13.

27. Krizhevsky, A. Learning Multiple Layers of Features from Tiny Images. Master's Thesis, Computer Science Department, University of Toronto, Toronto, ON, Canada, 2009.

28. Hinton, G.E.; Osindero, S.; Teh, Y.W. A fast learning algorithm for deep belief nets. *Neural Comput.* **2006**, *18*, 1527–1554. [CrossRef] [PubMed]

29. Vincent, P.; Larochelle, H.; Bengio, Y.; Manzagol, P.A. Extracting and composing robust features with denoising autoencoders. In Proceedings of the 25th International Conference on Machine Learning, Helsinki, Finland, 5–9 July 2008; pp. 1096–1103.

30. Rumelhart, D.E.; Hinton, G.E.; Williams, R.J. Learning representations by back-propagating errors. *Nature* **1986**, *323*, 533–536. [CrossRef]

31. Vincent, P.; Larochelle, H.; Lajoie, I.; Bengio, Y.; Manzagol, P. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *J. Mach. Learn. Res.* **2010**, *11*, 3371–3408.

32. Turchenko, V.; Luczak, A. Creation of a deep convolutional auto-encoder in Caffe. *arXiv* **2015**, arXiv:1512.01596.

33. Lowe, D.; Helmer, S. Object class recognition with many local features. In Proceedings of the Workshop on Generative Model Based Vision, Washington, DC, USA, 27 June–2 July 2004; Volume 12, pp. 187–187.

34. Serre, T.; Wolf, L.; Poggio, T. Object recognition with features inspired by visual cortex. In Proceedings of the Computer Vision and Pattern Recognition Conference, San Diego, CA, USA, 20–25 June 2005; pp. 994–1000.

35. Masci, J.; Meier, U.; Ciresan, D.; Schmidhuber, J. Stacked convolutional auto-encoders for hierarchical feature extraction. In Proceedings of the International Conference on Artificial Neural Networks, Espoo, Finland, 14–17 June 2011; pp. 52–59.

36. Zeiler, M.D.; Taylor, G.W.; Fergus, R. Adaptive deconvolutional networks for mid and high level feature learning. In Proceedings of the International Conference on Computer Vision, Barcelona, Spain, 6–13 November 2011; pp. 2018–2025.

37. Zeiler, M.D.; Fergus, R. Visualizing and understanding convolutional networks. In Proceedings of the European Conference on Computer Vision, Zurich, Switzerland, 6–12 September 2014; pp. 818–833.

38. Zeng, T.; Li, R.; Mukkamala, R.; Ye, J.; Ji, S. Deep convolutional neural networks for annotating gene 544 expression patterns in the mouse brain. *BMC Bioinform.* **2015**, *16*, 147. [CrossRef] [PubMed]

39. Sermanet, P.; Eigen, D.; Zhang, X.; Mathieu, M.; Fergus, R.; LeCun, Y. OverFeat: Integrated recognition, localization and detection using convolutional networks. In Proceedings of the International Conference on Learning Representations, Banff, AB, Canada, 14–16 April 2014.

40. Simonyan, K.; Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv* **2014**, arXiv:1409.1556.

41. Lee, H.; Ekanadham, C.; Ng, A. Sparse deep belief net model for visual area V2. In Proceedings of the Advances in Neural Information Processing Systems, Vancouver, BC, Canada, 3–6 December 2007; Volume 20, pp. 873–880.

42. Le, Q.V.; Monga, R.; Devin, M.; Chen, K.; Corrado, G.S.; Dean, J.; Ng, A. Building high-level features using large scale unsupervised learning. In Proceedings of the International Conference on Machine Learning, Vancouver, BC, Canada, 26–31 May 2013.