Sparse and Overcomplete Image Rerpesentations

Katrina Evtimova
kve216@nyu.edu
NYU Center for Data Science

1 Introduction

The volume of images generated on a daily basis is astounding. Approximately 350 million photos are uploaded on Facebook alone every day which amounts to 4000 new photos per second.[1] Large quantities of image data combined with automated systems can be useful in numerous downstream applications, such as filtering sensitive content or providing a textual description of image contents for the visually impaired.

Building algorithms that can perform *image processing* automatically is a long-standing research problem in the field of computer vision. A crucial part of the problem is figuring out what kind of useful insights can and need to be extracted from the input data while keeping in mind the underlying application. For example, if the goal is to assign objects in an image to different categories (known as *image classification*) and one of the categories is zebra, it would be useful to have an algorithm which can detect the striped pattern on zebras' coats. The presence of stripes in an image can be thought of as a *feature* which can help to determine whether the image contains a zebra or not. The goal of image processing is to build feature-extraction algorithms that take a d-dimensional input image[2] $\mathbf{y} \in \mathbb{R}^d$ and produce features $\mathbf{z} \in \mathbb{R}^q$ which are useful in some downstream application. We refer to image features as image *representations* interchangeably in the remaining of the text.

What are different ways to extract useful features $\mathbf{z}$? Figure 1 shows that one can apply a convolution to an image in order to extract its edges. This convolution is an example of a feature extractor hand-crafted to detect a specific feature, and requires domain expertise to design. Typically, a downstream application requires multiple feature extractors so hand-crafting them can also take a lot of effort. Using machine learning, it is possible to build feature extractors from large amounts of image data automatically. One of the most popular machine learning algorithms for feature extraction is called sparse coding. My research focuses on improving this method by addressing some of its limitations.

The rest of this document is organized as follows. Section 2 introduces a particular set of desirable properties for image features $\mathbf{z}$, namely sparsity and overcompleteness. Section 3 discusses a common way to extract sparse and overcomplete image representations using sparse coding. Section 4 gives an overview of my research which proposes a novel extension to the method presented in Section 3. Section 5 discusses future research directions for my method.

[1] Source: https://www.omnicoreagency.com/facebook-statistics/.

[2] Images in computer vision are usually defined as 3D tensors consisting of three color channels, each of which is a pixel grid of size $W \times H$. Here we assume an image is flattened to a 1D array of size $d = 3 \times W \times H$.

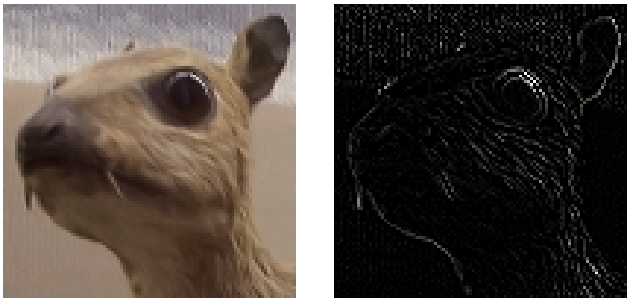

Figure 1: **Left**: image of an animal. **Right**: same image after being convolved with an edge detection convolutional kernel $\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$. Source: Wikipedia.

2 Sparse and Overcomplete Representations

The motivating question in this section is what useful properties image representations can have. Similarly to how the simple smiling emoji icon ☺ can convey the complex notions of a face and happiness, a good image representation can capture more abstract and higher-level concepts than data at the level of the raw pixels.

Figure 2: An image of a mountain range. Source: Rohit Tandon on Unsplash.

Speaking more formally, image data contains a lot of redundant information. Consider the picture of a mountain ridge in Figure 2. Pixels that belong to the sky are highly correlated with their neighboring pixels. The same is true for pixels depicting the mountain slopes. This leads to the idea that an image representation can contain some high-level factors (such as the presence of a sky or a mountain) that explain its contents in a more concise way than its raw pixels. This idea is generalized in the notion of sparsity. An image representation is *sparse* if only a small number of its components are non-zero for a given input. Intuitively, sparsity introduces the assumption that only a small number of factors are relevant for a

given image. This makes sparsity a desirable property for image representations as it can make them more interpretable [1; 2].

An image can contain more than one object and there are many possible objects (trees, rocks, and snow can appear on a mountain ridge; birds, clouds, and airplanes can appear in the sky; etc.). This is a motivation to use image representations whose dimensionality is greater than the dimensionality of the original data. Such representations are called *overcomplete* and they are more robust to corruptions in the input image (e.g. missing pixels) than smaller-dimensional representations [7]. Further support for extracting image representations which are both sparse and overcomplete comes from biology - the visual cortex in humans uses such representations [5].

3 Sparse Coding

How can one obtain sparse and overcomplete image representations in practice? In the discussion below we assume that we are given d-dimensional input images $\mathbf{Y} \in \mathbb{R}^{d \times N}$ where N is the number of images and we would like to find corresponding sparse and overcomplete representations $\mathbf{Z} \in \mathbb{R}^{q \times N}$ for each image in the input data (note that $q \gg d$).

A popular machine learning approach for solving this problem uses an auxiliary linear operator $\mathcal{D} \in \mathbb{R}^{d \times q}$ which is parameterized by its elements and is trained to map representations $\mathbf{Z}$ to the original input images $\mathbf{Y}$. In other words, the parameters of $\mathcal{D}$ are adjusted to make the product $\mathcal{D}\mathbf{Z} \in \mathbb{R}^{d \times N}$ close to the original images $\mathbf{Y}$. The motivation for introducing the operator $\mathcal{D}$ is that if such an operator exists, then the information contained in the sparse and overcomplete $\mathbf{Z}$ is sufficient to reconstruct the input images $\mathbf{Y}$.

More formally, finding both $\mathbf{Z}$ and $\mathcal{D}$ happens by solving the following optimization problem:

$$\underset{\mathcal{D},\mathbf{Z}}{\operatorname{argmin}} \|\mathbf{Y} - \mathcal{D}\mathbf{Z}\|_2^2 + \alpha\|\mathbf{Z}\|_1, \tag{1}$$

where $\alpha \in \mathbb{R}^+$ is a regularization parameter. The first term in equation 1 measures how good the reconstructions $\mathcal{D}\mathbf{Z}$ are. We refer to this term as the *reconstruction error*. The second term forces representations in $\mathbf{Z}$ to be sparse by penalizing their l_1 norm. The operator $\mathcal{D}$ is usually referred to as a *dictionary* since it can "decode" the information in $\mathbf{Z}$ back to the space of images. Each column of $\mathcal{D}$ can be thought of as a d-dimensional building block or a "word". Each column $\mathbf{Z}_{[:,i]} \in \mathbb{R}^q$ corresponds to the representation or "code" of the i-th image in $\mathbf{Y}$. The non-zero components in $\mathbf{Z}_{[:,i]}$ select columns of $\mathcal{D}$ relevant for the reconstruction of the i-th input image $\mathbf{Y}_{[:,i]}$.

Solving the optimization problem in equation 1 requires a 2-step iterative procedure. The first step assumes that the dictionary $\mathcal{D}$ is fixed and computes the corresponding codes for this dictionary:

$$\hat{\mathbf{Z}} = \underset{\mathbf{Z}}{\operatorname{argmin}} \|\mathbf{Y} - \mathcal{D}\mathbf{Z}\|_2^2 + \alpha\|\mathbf{Z}\|_1, \tag{2}$$

This is called the *inference* step. The second step for solving the optimization problem in equation 1 uses the codes $\hat{\mathbf{Z}}$ computed in step 2 to update the parameters of the linear dictionary:

$$\hat{\mathcal{D}} = \underset{\mathcal{D}}{\operatorname{argmin}} \|\mathbf{Y} - \mathcal{D}\hat{\mathbf{Z}}\|_2^2. \tag{3}$$

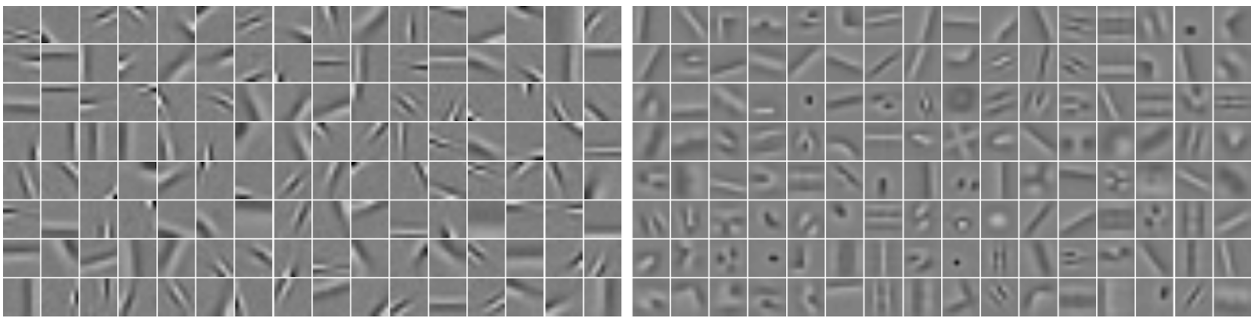

Figure 3: **Left**: Columns of a linear dictionary $\mathcal{D} \in \mathbb{R}^{784\times128}$ trained on grayscale image patches of size 28×28. Each column is a square since it is reshaped to match the input patch size of 28×28. **Right**: Dictionary of 128 convolutional filters learned on grayscale images. Each square is a two-dimensional convolutional filter.

This is called the *learning* step. Note that the l_1 penalty term is missing as the minimization is done with respect to the elements of $\mathcal{D}$, assuming that the representations $\hat{\mathbf{Z}}$ are fixed. The two steps are repeated until some convergence criteria is reached. This iterative procedure is known as *sparse coding*.

4 Extension of Sparse Coding

This section gives an overview of my research on extracting sparse and overcomplete image representations with a novel architecture for the decoder $\mathcal{D}$ which addresses two limitations of the sparse coding setup.

4.1 Limitations of Sparse Coding

The elements of a linear dictionary $\mathcal{D}$ typically trained through sparse coding are highly redundant [3]. This is evident in Figure 3 (Left) which shows the columns of such a dictionary $\mathcal{D}$ learned on greyscale image patches.[3] In particular, there are dictionary columns which are shifted versions of each other. This limitation can be overcome by learning a dictionary $\mathcal{D}$ of convolutional filters [6; 3; 8; 9]. Such filters are presented in Figure 3 (Right). The reader can see that the convolutional filters are more diverse and can detect more complex shapes which in turn means that the corresponding sparse representations can contain more complex information.

Another limitation of the traditional sparse coding setup comes from the lack of compositionality. Let us return to the sample image of a mountain range in Figure 2. As we discussed in section 2, a high-level description of this image can be that it contains two main parts, mountain and sky. If we take a closer look, we can notice more details: there are clouds in the sky; some parts of the mountain are covered in snow and others are covered in trees; and the trees have branches. One can build a hierarchy of the objects in the image and the relationships between them starting with the higher-level ones and moving to the finer-detailed ones (e.g. mountain $\supset$ trees $\supset$ branches $\supset \ldots$). This structure reflects the

[3]The dictionary columns are reshaped to the original 2-dimensional image patch size.

compositionality of the world we live in and cannot be captured with a single representation $\mathbf{z} \in \mathbb{R}^q$. This observation leads to a motivating idea behind my research — build a hierarchy of sparser and overcomplete representations for a given image $\mathbf{y}$, namely $\mathcal{H}(\mathbf{y}) = \{\mathbf{z}_0, \ldots, \mathbf{z}_{K-1}\}$ for some $K > 1$, where each representation $\mathbf{z}_i$ builds on top of and adds finer-grained details to the information contained in the set of higher-level representations $\{\mathbf{z}_j\}_{j=i+1}^{K}$.

4.2 Novel Decoder

We propose a novel decoder $\mathcal{D}$ trained to reconstruct a given image input $\mathbf{y}$ using the hierarchy of sparse overcomplete representations $\mathcal{H}(\mathbf{y})$. The decoder $\mathcal{D}$ is non-linear: it is a composition of convolutional filters and non-linear operations which allows for a non-linear relationship between the sparse representations in $\mathcal{H}(\mathbf{y})$ and the original image $\mathbf{y}$. The exact architecture is outside the scope of this text but, at a high level, the decoder $\mathcal{D}$ can be regarded as a function which takes the hierarchy of features $\mathcal{H}(\mathbf{y}) = \{\mathbf{z}_0, \ldots, \mathbf{z}_{K-1}\}$ as input, applies different compositions of convolutions and non-linear operations to each $\mathbf{z}_i$, and combines the results to produce a reconstruction $\mathcal{D}(\mathbf{z}_0, \ldots, \mathbf{z}_{K-1})$ of the original image $\mathbf{y}$. The structure of the decoder $\mathcal{D}$ allows the information in each representation $\mathbf{z}_i$ to be complementary to and finer-grained than the information contained in the representations $\{\mathbf{z}_j\}_{j=i+1}^{K}$.

It is also important to note that in the case when the dictionary $\mathcal{D}$ contains convolutional filters such as the ones in Figure 3 (Right), the representations $\mathbf{z}_i \in \mathcal{H}(y)$ are no longer q-dimensional vectors but are three-dimensional tensors of size $c_i \times w_i \times h_i$ where $c_i, w_i, h_i \in \mathbb{N}$. This is due to the fact that each of the c_i feature sets of size $w_i \times h_i$ in $\mathbf{z}_i$ is convolved with a corresponding convolutional filter in $\mathcal{D}$. Figure 4 shows the sparse and overcomplete representations $\mathcal{H}(\mathbf{y}) = \{\mathbf{z}_0, \mathbf{z}_1, \mathbf{z}_2\}$ for a sample image $\mathbf{y}$. The reader can see that $\mathbf{z}_0$ has 4 feature sets, $\mathbf{z}_1$ has 14, and $\mathbf{Z}_2$ has 42. Also, the dimensionality of each of the representations in $\mathcal{H}(\mathbf{y})$ is greater than the dimensionality of the original image $\mathbf{y}$ and approximately 70% of the components in each $\mathbf{z}_i$ are equal to 0.

Similarly to the case of sparse dictionary learning from section 3, we compute the hierarchy of sparse overcomplete representations $\mathcal{H}(\mathbf{y}) = \{\mathbf{z}_0, \ldots, \mathbf{z}_{K-1}\}$ and the decoder $\mathcal{D}$ by solving the following minimization problem:

$$\underset{\mathcal{D}, \mathbf{z}_0, \ldots, \mathbf{z}_{K-1}}{\operatorname{argmin}} \quad \|\mathbf{Y} - \mathcal{D}(\mathbf{z}_0, \ldots, \mathbf{z}_{K-1})\|_2^2 + \alpha \sum_{i=0}^{K-1} \|\mathbf{z}_i\|_1, \tag{4}$$

As in the case of sparse coding, the solution requires a procedure which iterates between an inference step and a learning step. The main difference between our setup and the sparse coding one is that during the inference step, we minimize with respect to only one of the sparse representations $\mathbf{z}_i$ and we fix the parameters of the decoder $\mathcal{D}$ and the elements of all the other representations $\{\mathbf{z}_0, \ldots, \mathbf{z}_{i-1}, \mathbf{z}_{i+1}, \ldots, \mathbf{z}_{K-1}\}$. During the learning step, we update the parameters of the decoder to minimize the expression in formula 4 keeping all the representations $\{\mathbf{z}_0, \ldots, \mathbf{z}_{K-1}\}$ fixed. This iterative procedure ends when some convergence criteria is met.

Overall, when compared to a linear dictionary, our convolutional decoder should be able to capture more complex relationships between the sparse representations in $\mathcal{H}(\mathbf{y})$ and images $\mathbf{y}$ and also provide $\mathcal{H}(\mathbf{y})$ with a hierarchical structure.

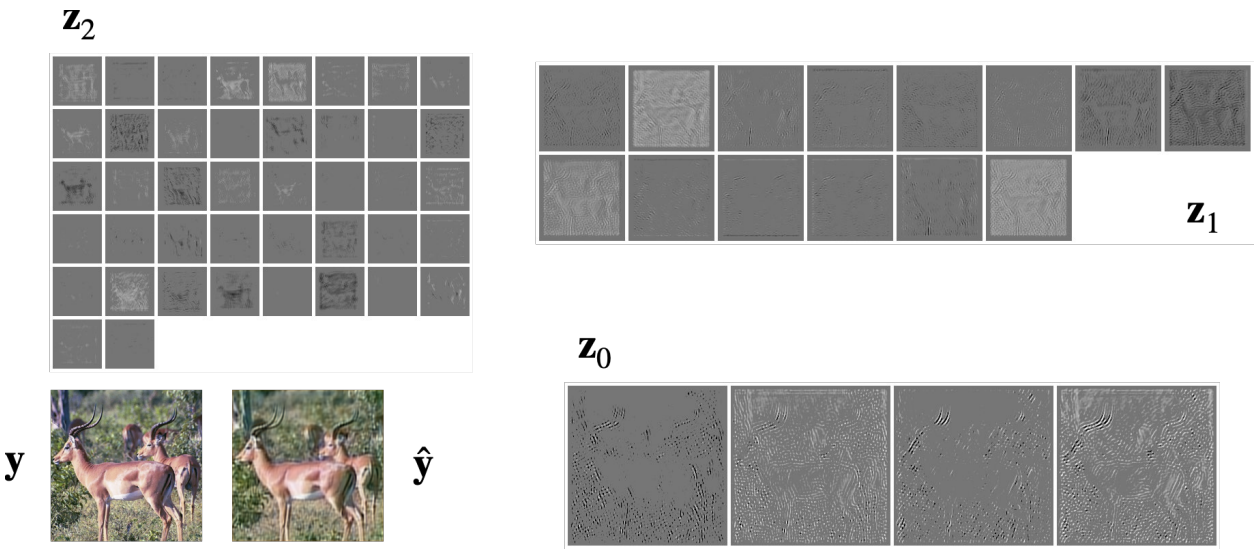

Figure 4: Input image $\mathbf{y}$ along with its hierarchy of sparse overcomplete representations $\mathcal{H}(\mathbf{y}) = \{\mathbf{z}_0, \mathbf{z}_1, \mathbf{z}_2\}$ and reconstruction $\hat{\mathbf{y}}$. In this example, there are 4 feature sets in $\mathbf{z}_0$, 14 in $\mathbf{z}_1$, and 42 in $\mathbf{z}_2$. Approximately 70% of the components in each sparse representation $\mathbf{z}_i$ are equal to 0.

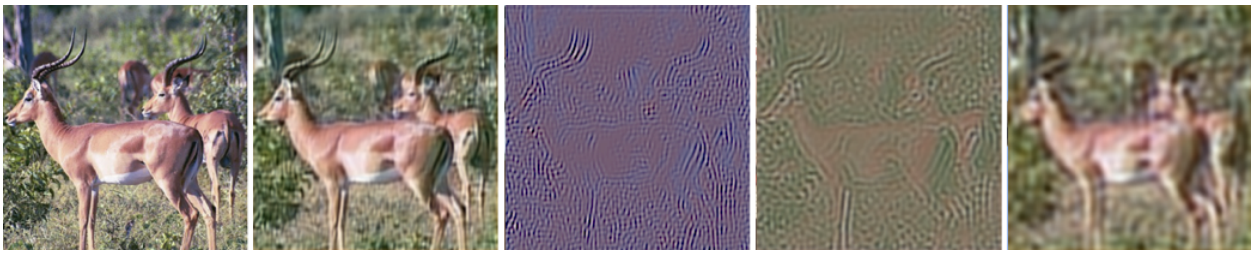

Figure 5: On the left is the original input image. Proceeding to the right are the reconstructions using all of the representations, then using only one of the representations ($\mathbf{z}_0$, $\mathbf{z}_1$, and $\mathbf{z}_2$ respectively). Using all of the representations provides the best reconstruction, while using only a single representation produces blurry or abstract results.

4.3 Information in the Representations and in the Decoder

This section explores what information is contained in the hierarchy of sparse overcomplete representations $\mathcal{H}(\mathbf{y})$ and in the decoder $\mathcal{D}$ proposed in the previous section.

When we use our decoder to obtain sparse representations, we expect that information about an image $\mathbf{y}$ is distributed across every $\mathbf{z}_i$ in $\mathcal{H}(\mathbf{y})$ and that each $\mathbf{z}_i$ contains finer-grained details than any of the representations in the set $\{\mathbf{z}_j\}_{j=i+1}^{K-1}$. To test whether this is the case, we perform the following experiment: we use information only from representation $\mathbf{z}_i$ to reconstruct the original image. In other words, we apply the decoder $\mathcal{D}$ to $\mathbf{z}_i$ only and ignore any information coming from representations $\{\mathbf{z}_0, \ldots, \mathbf{z}_{i-1}, \mathbf{z}_{i+1}, \ldots, \mathbf{z}_{K-1}\}$ for all $i \in \{0, \ldots, K-1\}$. To visualize this, Figure 5 displays an input image of antelopes in the wild along with its reconstructions from all of the representations and each of the representations separately. When we reconstruct using only the information in representation

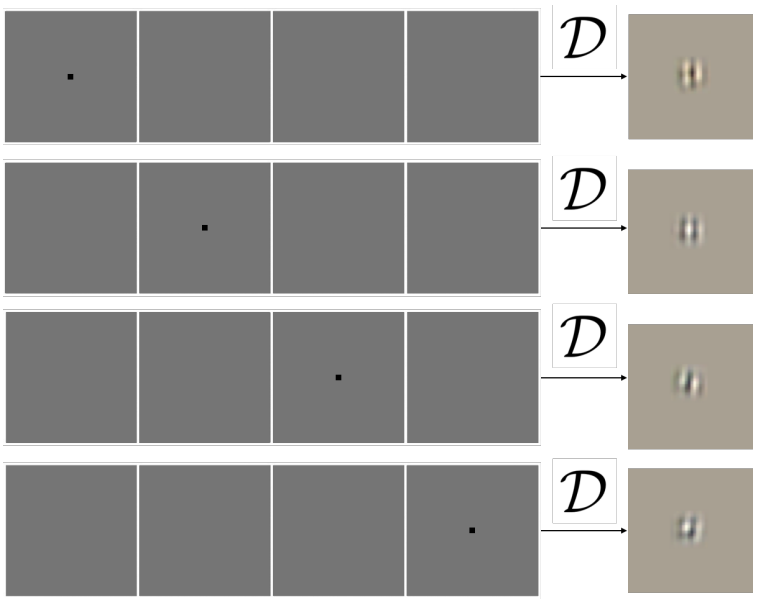

Figure 6: Reconstructions from applying our decoder $\mathcal{D}$ to a modified sparse representation $\mathbf{z}_0$ in which we set the middle component of each of the four feature sets in $\mathbf{z}_0$ to 1 (black square) and all the other components across $\mathbf{z}_0$ to 0 (grey area). The resulting reconstructions contain oriented edges.

$\mathbf{z}_0$, we obtain high-frequency information such as edges. Reconstruction from $\mathbf{z}_2$ only contains blurry outlines of the objects in the original image. Additionally, all codes together are needed to accurately reconstruct the input image. These observations support the idea that information in the representations for a given input is complementary. We also note that, as intended, the structure of the representations is hierarchical — representation $\mathbf{z}_2$ contains higher-level information (blurry outlines of objects) while $\mathbf{z}_1$ and $\mathbf{z}_0$ contain finer-grained details such as the structure of the antelopes' horns.

What has our decoder learned? One way to visualize the patterns that can be produced by our decoder $\mathcal{D}$ is to check how each individual element in $\mathbf{z}_i$ affects the decoder's reconstructions. In particular, we consider a setup in which we set the middle component in one of the four feature sets of code $\mathbf{z}_0$ (displayed in Figure 4) to 1 and set all the other components across $\mathbf{z}_0$ to 0. When we give this input to the decoder and ignore any input from $\mathbf{z}_1$ and $\mathbf{z}_2$, we obtain the reconstructions depicted in Figure 6. They contain oriented edges similar to the ones captured by simple cells in the visual cortex of mammals [4]. When we perform the same experiment for feature sets in $\mathbf{z}_1$ and $\mathbf{z}_2$, we observe similar results. It is remarkable that the decoder learns to produce such patterns from data alone.

5 Conclusion

We have introduced an extension to the traditional sparse coding setup. Our decoder is non-linear which means that when compared to a linear dictionary, it can model more complex relationships between the sparse features and the images they represent. Additionally, instead of a single sparse representation, our decoder extracts a hierarchy of sparse and overcomplete

representations which complement each other to reconstruct the original image.

One limitation of our approach is that computing the image representations is slow. In order to reduce the inference time, we are planning to train an encoder which can predict the representations our model produces directly from the input images.

We are also currently experimenting with ways to measure whether the sparse representations our model produces are useful in downstream applications such as image classification. Our hypothesis is that training a logistic regression to predict classes of objects using our representations, rather than the raw pixel images, would produce a higher prediction accuracy with the same number of training samples. In other words, we believe that the information contained in the sparse representations would simplify the task of image classification.

References

[1] Bengio, Y. Learning deep architectures for ai. *Foundations and trends in Machine Learning 2*, 1 (2009), 1–127.

[2] Bengio, Y., Courville, A., and Vincent, P. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence 35*, 8 (2013), 1798–1828.

[3] Kavukcuoglu, K., Sermanet, P., lan Boureau, Y., Gregor, K., Mathieu, M., and Cun, Y. L. Learning convolutional feature hierarchies for visual recognition. In *Advances in Neural Information Processing Systems 23*, J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta, Eds. Curran Associates, Inc., 2010, pp. 1090–1098.

[4] Marĉelja, S. Mathematical description of the responses of simple cortical cells. *JOSA 70*, 11 (1980), 1297–1300.

[5] Olshausen, B. A., and Field, D. J. Sparse coding with an overcomplete basis set: A strategy employed by v1? *Vision research 37*, 23 (1997), 3311–3325.

[6] Ranzato, M., Poultney, C., Chopra, S., and Cun, Y. L. Efficient learning of sparse representations with an energy-based model. In *Advances in neural information processing systems* (2007), pp. 1137–1144.

[7] Teh, Y. W., Welling, M., Osindero, S., and Hinton, G. E. Energy-based models for sparse overcomplete representations. *Journal of Machine Learning Research 4*, Dec (2003), 1235–1260.

[8] Zeiler, M. D., Krishnan, D., Taylor, G. W., and Fergus, R. Deconvolutional networks. In *2010 IEEE Computer Society Conference on computer vision and pattern recognition* (2010), IEEE, pp. 2528–2535.

[9] Zeiler, M. D., Taylor, G. W., and Fergus, R. Adaptive deconvolutional networks for mid and high level feature learning. In *2011 International Conference on Computer Vision* (2011), IEEE, pp. 2018–2025.