Portland State University

PDXScholar

Computer Science Faculty Publications and Presentations

Computer Science

5-2017

Sparse Coding on Stereo Video for Object Detection

Sheng Y. Lundquist
Portland State University

Melanie Mitchell
Portland State University, mm@pdx.edu

Garrett T. Kenyon
Los Alamos National Laboratory

Follow this and additional works at: https://pdxscholar.library.pdx.edu/compsci_fac

 Part of the Computer Engineering Commons

Let us know how access to this document benefits you.

Citation Details

Lundquist, S. Y., Mitchell, M., & Kenyon, G. T. (2017). Sparse Coding on Stereo Video for Object Detection. arXiv preprint arXiv:1705.07144.

This Pre-Print is brought to you for free and open access. It has been accepted for inclusion in Computer Science Faculty Publications and Presentations by an authorized administrator of PDXScholar. Please contact us if we can make this document more accessible: pdxscholar@pdx.edu.

arXiv:1705.07144v1 [cs.CV] 19 May 2017

Sparse Coding on Stereo Video for Object Detection

Sheng Y. Lundquist
Portland State University
New Mexico Consortium
shenglundquist@gmail.com

Melanie Mitchell
Portland State University
Santa Fe Institute

Garrett T. Kenyon
Los Alamos National Laboratory
New Mexico Consortium

Abstract

Deep Convolutional Neural Networks (DCNN) require millions of labeled training examples for image classification and object detection tasks, which restrict these models to domains where such a dataset is available. We explore the use of unsupervised sparse coding applied to stereo-video data to help alleviate the need for large amounts of labeled data. In this paper, we show that unsupervised sparse coding is able to learn disparity and motion sensitive basis functions when exposed to unlabeled stereo-video data. Additionally, we show that a DCNN that incorporates unsupervised learning exhibits better performance than fully supervised networks. Furthermore, finding a sparse representation in the first layer, which infers a sparse set of activations, allows for consistent performance over varying initializations and ordering of training examples when compared to representations computed from a single convolution. Finally, we compare activations between the unsupervised sparse-coding layer and the supervised layer when applied to stereo-video data, and show that sparse coding exhibits an encoding that is depth selective, whereas encodings from a single convolution do not. These result indicates promise for using unsupervised sparse-coding approaches in real-world computer vision tasks in domains with limited labeled training data.

1 Introduction

Over the last decade, Deep Convolutional Neural Networks (DCNNs) trained with supervised learning have emerged as the dominant paradigm for computer vision. These networks have shown impressive results on computer vision tasks such as image labeling [23] and object detection [20]. However, one drawback of DCNNs is that they rely on large collections of training data that have been annotated by humans, e.g., ImageNet [3]. Obtaining a sufficient amount of labeled data can be difficult. For example, collecting labeled training data for detecting brain tumors from Magnetic Resonance Imaging (MRI) data can be problematic, as some tumors are relatively rare in occurrence. Human annotations for labeling can be expensive in this domain as well, as it requires human specialists to provide such labels. Furthermore, human annotations can vary widely from person to person, which adds noise to labels. Finally, there can be serious privacy concerns that impede the creation of a large database. As such, DCNNs are restricted to domains for which there exists large datasets of labeled examples. In this paper, we explore the use of unsupervised sparse coding within a DCNN as an alternative to layers typically used by such networks that compute encodings directly (i.e., encodings computed with a single convolution).

Inspired by theories of efficient coding in neural computation [1], sparse coding [17] aims to infer efficient, non-redundant encodings of a given input (e.g., photographs and videos). Specifically, sparse coding learns to represent an input (e.g., an image) as a linear combination of basis vectors drawn from a provided set (referred to as a dictionary). Each basis vector is weighted by a scalar coefficient (referred to as an activation), and the set of activations are taken to be the encoding of the input. Sparse coding constrains the activations to be sparse (i.e., to have few nonzero activations), such that resulting activations are non-redundant. Here, inferring the sparse set of activations for

representation is an optimization problem, unlike finding encodings in layers typically used by DCNNs (which we refer to in this paper as direct layers), where activations are computed with a single convolution followed by a nonlinearity.

The idea of using efficient codes for detection stems from biology; biological vision systems are the most accurate and robust system for object detection. Sparse coding has been shown to exhibit similar properties to biological neurons in early stages of mammalian visual processing [17]. It follows that investigating biologically-inspired algorithms could provide novel insights into computer vision tasks, such as object detection. Additionally, the efficiency in representation exhibited by sparse coding can be advantageous in specialized, non-Von Neumann hardware architectures. Indeed, there has been work in implementations of sparse coding on low-power neuromorphic [10] and quantum [15] hardware.

One domain in which non-redundant encoding would be useful is in multi-view sensing. For example, two cameras offset horizontally that capture the same light post will result in the post being shifted by an amount inversely proportional to the distance of the post from the cameras [18]. To represent the post, an algorithm trying to detect the post in both cameras can either use a monocular post detector with two activations (one for each camera), or use a single activation that is able to detect the post with a given shift. Sparse coding would favor the latter, as it allows for a more efficient representation of the stereo inputs. Similarly, efficient representation of a post from consecutive frames within a video should allow for motion-aware activations that correspond to the distance from the camera [5]. It follows that a sparse-coding model that aims for efficient representation of stereo-video input should have some notion of depth. Indeed, Lundquist et al. [13] have shown that depth-selective activations emerge by applying sparse coding to stereo data, and we show in Section 4.3 that this result generalizes to stereo-video data.

Our contributions in this paper are as follows: (Section 4.1) Exposing sparse coding to stereo-video frames result in binocular temporal basis vectors that are disparity and motion sensitive. (Section 4.2) A DCNN that incorporates unsupervised learning is able to achieve better performance on a vehicle detection task with a minimal amount of training labels. Additionally, replacing the direct layer with sparse coding as the first layer in the DCNN allows for consistent performance that is robust to random conditions. (Section 4.3) Output activations from a sparse-coding layer are depth selective, which may provide an explanation for the difference in performance we observe in this study.

1.1 Related Work

Lundquist et al. [13] demonstrated that representations of stereo images obtained through sparse coding allow for an encoding that achieves better performance than a direct encoding layer in the task of pixel-wise depth estimation. The authors show that the sparse encoding is inherently depth selective, whereas the direct encoding is not. In this paper, we extend this work to encode stereo-video clips and compare encodings on a vehicle-detection task.

A closely related study by Coates et al. [2] compared unsupervised and supervised methods with two-layer networks on an image classification task. The authors demonstrated that an unsupervised layer does not outperform a comparable layer with a direct encoding. While our experimental results agree with this finding in the case of two layers, we find that an additional supervised layer on top of an unsupervised sparse-coding layer allows the network to outperform the supervised network.

Recent work by Lotter et al. [12] shares the motivation of utilizing unsupervised learning to alleviate the need for labeled training data. Specifically, the authors use unsupervised learning to predict future frames of a video. They additionally show that their network achieves better performance than standard DCNNs when each is trained on only a limited amount of training data. In contrast to future frame prediction, our work aims to achieve image representation through sparse coding. Additionally, Lotter et al. uses a recurrent neural network [8] as the backbone of their network, whereas we use sparse coding as the unsupervised learning algorithm.

Other work [4, 19] explores the use of unsupervised learning techniques within a supervised network. However, most work in this area does not explore natural scenes (instead, focusing on datasets such as MNIST for handwritten digit recognition). Here, we extend this work to the domain of stereo video captured "in the wild". Additionally, we explicitly compare performance between the use of unsupervised learning versus supervised learning within a DCNN.

There has been other work in unsupervised learning of multi-view data [9, 14, 16]. In contrast, our work aims to explicitly compare unsupervised learning to supervised learning for the task of vehicle detection in multi-view data.

2 Sparse Coding

Sparse coding aims to represent an input over a set of hidden units such that the original signal is recoverable with minimal degradation. Sparse coding aims to minimize the cost function

$$J(\boldsymbol{I}, \boldsymbol{a}, \boldsymbol{\Phi}) = \frac{1}{2}(\overbrace{\|\boldsymbol{I} - \boldsymbol{a} \circledast \boldsymbol{\Phi}\|_2}^{\text{Reconstruction error}})^2 + \lambda \overbrace{\|\boldsymbol{a}\|_1}^{\text{Sparsity term}} . \tag{1}$$

More specifically, sparse coding aims to minimize the difference between a given input $\boldsymbol{I}$ and a reconstruction, where the difference is measured by Euclidean distance (i.e., $\|\cdot\|_2$, or the L_2 norm). The reconstruction is calculated via a linear combination of basis vectors drawn from a dictionary $\boldsymbol{\Phi}$, weighted by coefficients $\boldsymbol{a}$ (called "activations"). The sparsity term constrains the activations $\boldsymbol{a}$ to be sparse, by measuring the sum of the absolute value of $\boldsymbol{a}$ (i.e., $\|\cdot\|_1$, or the L_1 norm)[1]. λ is a user-set parameter that controls the trade-off between the reconstruction error and sparsity.

Here, $\circledast$ denotes the transposed convolution operation [25][2] which calculates the reconstruction $\boldsymbol{a} \circledast \boldsymbol{\Phi}$. Each transposed convolution has a mapping to a corresponding convolution, e.g., $\boldsymbol{I} * \boldsymbol{\Phi} = \boldsymbol{u}$ with input $\boldsymbol{I}$, output $\boldsymbol{u}$, and the convolution operation denoted as $*$. In contrast, the transposed convolution is a function of the activations $\boldsymbol{a}$ instead of the input. In this paper, the transposed convolution is three-dimensional: the corresponding convolution is over the time, height, and width axes of the input.

The process of sparse coding involves finding a set of activations $\boldsymbol{a}$ and learning a dictionary $\boldsymbol{\Phi}$ that minimize Equation 1, given a training set. This learning process is unsupervised: labels on the training data are not used. The minimization of Equation 1 is broken into two parts: encoding and dictionary learning.

Encoding involves inferring the sparse activations from a given input. Specifically, sparse coding finds the encoding by minimizing Equation 1 with respect to $\boldsymbol{a}$ while holding $\boldsymbol{\Phi}$ fixed. The final activations $\boldsymbol{a}$ are taken to be the output activations of the input $\boldsymbol{I}$. A sparse coding layer can replace a direct layer in a DCNN by using activations $\boldsymbol{a}$ as the output of the layer, analogous to computing the activations in a direct layer. For encoding, we use the Locally Competitive Algorithm (LCA) [21], a hardware friendly, biologically informed optimization algorithm that minimizes Equation 1 for encoding.

Learning a dictionary for sparse coding is analogous to learning filter weights via backpropogation in a DCNN. However, learning weights in a DCNN requires labeled training data, and weights are updated to improve mappings from input to labels. In contrast, learning a dictionary aims to find weights that are able to represent the input efficiently for the purpose of reconstruction. In the domain of images, weights learned from sparse coding tend to represent oriented edges [17].

In our method, the input was first encoded using LCA. This is followed by one gradient descent step of minimizing the cost function with respect to $\boldsymbol{\Phi}$ while holding $\boldsymbol{a}$ fixed. Weights are normalized after each update in order to avoid a degenerate solution to Equation 1 in which the magnitudes of the weights become infinitely large and the corresponding activations negligible such that the inner product of the two remains constant, which results in a low L_1 norm of activations. Updating $\boldsymbol{\Phi}$ is repeated for multiple input batches until convergence.

[1] The L_1 norm is used as a surrogate to the L_0 norm (i.e., the number of nonzero elements), as Equation 1 is nonconvex with respect to $\boldsymbol{a}$ if the L_1 norm is replaced with an L_0 norm.

[2] Zeiler et al. [25] denote this operation as a "deconvolution", while others claim that "transposed convolution" is a more accurate term, since the transposed convolution is not the inverse operation of a convolution.

Abbreviation	Encoding	Weight Initialization	Weight learning
DirectRand	Direct	Random	None
DirectUnsup	Direct	Unsupervised	None
DirectFinetune	Direct	Unsupervised	Supervised
DirectSup	Direct	Random	Supervised
SparseUnsup	Sparse Inference	Random	Unsupervised

Table 1: Different first layers of networks tested. *Sparse Inference* denotes the process to find the sparse set of activations (i.e., finding $\boldsymbol{a}$ in Equation 1), whereas *Direct* denotes activations calculated from a single convolution. Unsupervised weights were obtained using sparse coding.

3 Experiments

We compare an unsupervised sparse coding layer with several types of direct layers, and test performance on a vehicle detection task on stereo video when there exists a minimal amount of available training labels.

We used the KITTI object detection dataset [6] for experiments. The dataset contains approximately 7000 training examples, of which 1000 examples were held out for testing. Each training example consists of three stereo frames ordered in time as input, with bounding box annotations for various objects in the left camera's last frame as ground truth.

Stereo-video inputs are normalized to have zero mean and unit standard deviation and downsampled to be 256×64 pixels. Stereo inputs are concatenated together such that the input contains six channels, i.e., RGB inputs from both left and right cameras. Time was kept in a separate dimension for three-dimensional convolutions (for direct layers) or transposed convolution (for sparse coding layers) across the time, height, and width axes of the input.

The ground truth for the task was generated by sliding a 32×16-pixel non-overlapping window across the left camera's last frame. A window is considered to be a positive instance if the window overlaps with any part of a car, van, or truck bounding box provided by the ground truth. The ground truth takes the form of a three dimensional tensor of dimensions $8 \times 4 \times 2$, corresponding to the width, height, and two classes (*vehicle* and *not vehicle*) respectively. A window belonging to the vehicle class is denoted with a value of 1, and 0 otherwise.

The final output of the network is a set of probabilities, one for each window corresponding to the probability of that window containing a vehicle. The cross entropy between the ground truth and estimated probabilities was used as the supervised cost function to train all supervised layers within the network.

Various encoding schemes for an n-layer network were tested. The first layer encodes the stereo-video frames with a variety of different methods, summarized in Table 1. Unsupervised weight initialization used weights obtained from sparse coding, trained on all available training data. All convolutional layers use a three-dimensional convolution that mirrors the transposed convolution done in the sparse-coding layer.

Once the first layer is set to one of the three possible options, the remaining $n - 1$ layers contain direct layers learned via backpropagation of the supervised loss. ReLU was used as a nonlinear activation function. Max pooling is used on activations after each layer. Finally, each layer except the first layer also uses dropout [24] to avoid overfitting the network to the training set.

Every two-layer network was initialized with random weights. The three layer network was initialized with learned weights from the two layer network plus an additional randomly initialized convolutional layer added. Likewise, the four layer network was initialized with the learned weights of the three layer network. All layers except the last layer contain 3072 filters (or basis vectors). Each model was repeated six times with different random initial conditions and random presentations of the training data to get a variance measure of each network. However, the unsupervised weights and activations obtained from sparse coding were not repeated due to computational time constraints. We discuss the implications of this in Section 5.

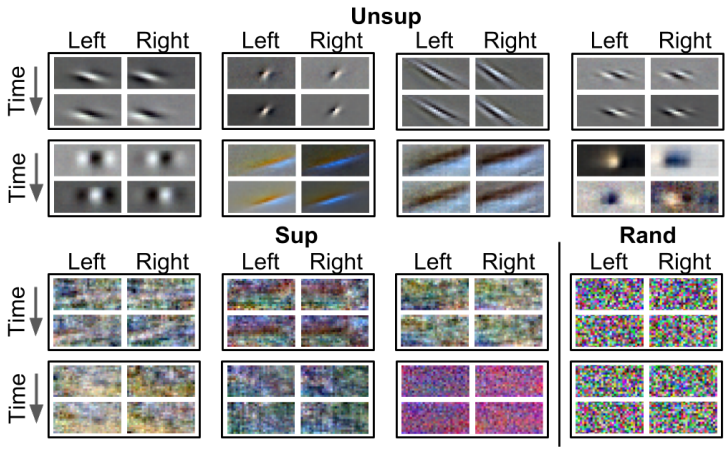


Figure 1: Representative weights for unsupervised weights (**Unsup**), fully supervised weights from random initial conditions (**Sup**), and random weights (**Rand**). **Unsup** weights tend to be edge detectors, such that the detectors are sensitive to the angle and frequency of the edge. Additionally, the edge detectors typically exist in pairs with some offset, which correspond to motion and disparity sensitivity. **Sup** weights from tend to be selective for color. **Rand** weights are random.

All models and experiments are implemented in Python using TensorFlow[3]. All code is available online[4].

4 Results

4.1 Visualization of Learned Weights

Figure 1 shows representative weights at the first layer of various networks. Here, unsupervised weights tend to be localized in that weights are concentrated to form edge detectors. These edge detectors are sensitive to the angle and frequency of the edge, similar to those found in the seminal work of Olshausen et al. on sparse coding [17]. Additionally, these edge detectors have similar weights with some translational offset in the alternate stereo camera as well as in time, which correspond to disparity and motion sensitivity respectively.

In contrast to weights obtained from sparse coding, the first layer of supervised weights look vastly different, showing less structure spatially but more selectivity for color. This is in stark contrast to other work that shows first-layer weights that resemble oriented edge-detectors [11] when trained on monocular images. It's possible that the limited amount of labeled training data is responsible for the observed difference. Weights at the first layer in the random case are, of course, random.

4.2 Vehicle Detection

Figure 2 shows the area under precision versus recall curve of all models trained on all available training data, each tested with two, three, and four layers[5]. Every network was trained on the supervised task six times to obtain error bars for performance. The center of each error bar denotes the median score, and the ends of the error bars denotes the maximum and minimum scores. Here, we find that **SparseUnsup** performs worse than **DirectSup** and **DirectFinetune** with two layers, which agrees with the findings of Coates et al. [2]. However, **SparseUnsup** is able to outperform **DirectSup** with three layers or more. This difference in performance due to the number of layers is

[3] `https://www.tensorflow.org/`

[4] `https://github.com/slundqui/TFSparseCode/`

[5] Prior to submission, we discovered 3 of the 6 runs in direct runs were erroneous which we discarded for Figure 2 and Figure 3. While we plan on fixing this issue for the final submission, we believe that the final conclusion we arrive at in this paper is not affected.

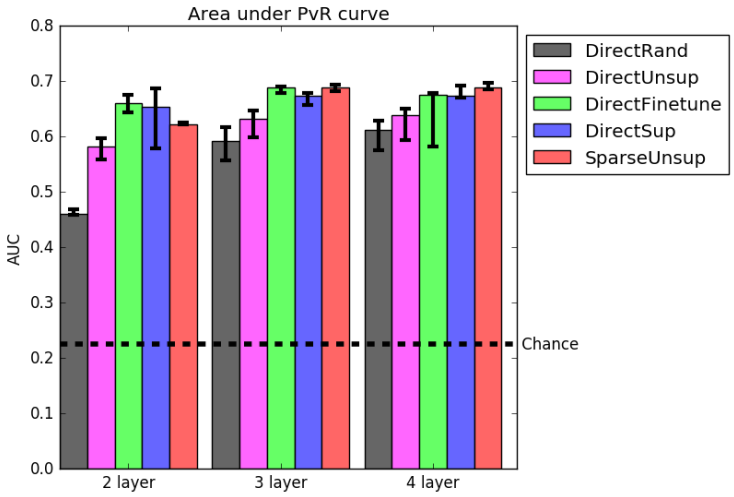


Figure 2: Area under precision versus recall curve for all networks tested with varying depths, trained on all available training data. Each network varies the first layer to be one of five choices (see Table 1 for details on each network). The dotted black line corresponds to randomly picking a class (the class distribution is biased towards *no vehicle*). Every network was trained six times with different random initial conditions and input presentation order to get error bars. The center of each error bar denotes the median score, and the ends of the error bars denote the minimum and maximum performance for experiments. **SparseUnsup** and **DirectFinetune**, which both incorporate unsupervised weights learned from sparse coding, achieve better performance than other models in three layer networks. Additionally, **SparseUnsup** exhibits higher consistency (i.e., smaller error bars) than direct layers.

likely because of the non-linearity required to map the sparse encoding to object class, since a single supervised layer might not have enough capacity to map to object detection accurately.

We find that **DirectFinetune**, which was initialized with unsupervised weights, outperforms other direct models with three layers. Additionally, we find that **SparseUnsup** performs similarly to **DirectFinetune** with three layers. This result is surprising, in that an unsupervised sparse coding layer trained with no supervision of the vehicle detection task achieves similar performance to one that was explicitly trained for vehicle detection, suggesting that activations found via sparse coding allows for a better representation of relevant information than the encoding of a direct convolution. All models outperform the control **DirectRand** and random chance.

SparseUnsup is more consistent in performance, as shown in the error bars, whereas direct models' performance varies depending on initial conditions and random ordering of training data presentations. Interestingly, **SparseUnsup** is more consistent in performance than **DirectUnsup**, where both models use the unsupervised weights learned via sparse coding and only differ in encoding scheme. This suggests that inferring activations in sparse coding is likely the reason for the additional consistency in performance.

Figure 3 shows the performance of **SparseUnsup**, **DirectSup**, and **DirectFinetune** while varying the number of labeled training examplesthat each network was trained on. Here, the unsupervised weights were trained on all available data without training labels. All networks were trained on a subset of available labeled data.

Similar to Figure 2, we find that the range in performance for networks incorporating direct layers to be larger than that of **SparseUnsup** for all numbers of labeled training examples tested. We find that, in the three layer case, both **DirectFinetune** and **SparseUnsup** are more robust to minimal data than **DirectSup**.

In all, these results suggest that the use of unsupervised learning within a DCNN allows for better performance than networks that do not. Additionally, unsupervised sparse coding allows for neural networks to perform more consistently than direct layers. Furthermore, unsupervised sparse coding, which was trained for efficient input representation, has similar performance to a supervised direct layer initialized with unsupervised weights that was explicitly trained for object detection.

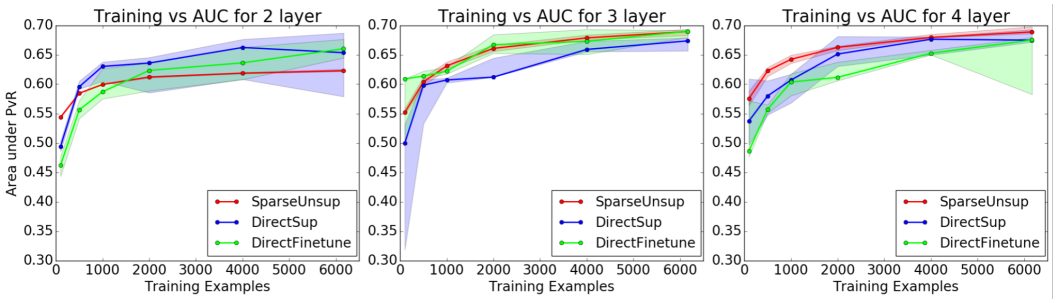


Figure 3: Number of training examples available versus performance for **SparseUnsup** (red), **DirectFinetune** (green), and **DirectSup** (blue) for two (left), three (middle), and four (right) layer networks. Each point is the median score over three independent runs, with the area between the maximum and minimum score filled in. Here, **SparseUnsup** and **DirectFinetune** have similar performance for three layers. However, **SparseUnsup** performs more consistently when compared to the other models. Best viewed in color.

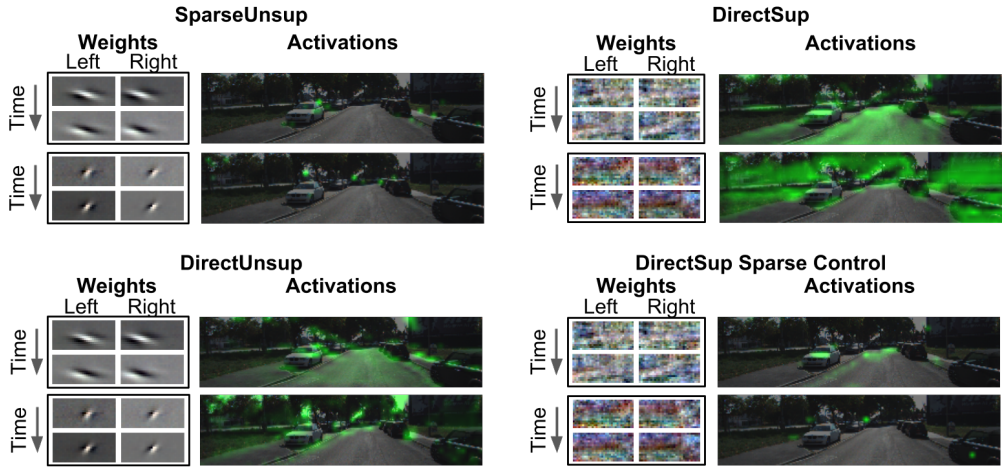


Figure 4: Nonzero activations of select weights overlaid on the input image. Magnitude of pixel values in green correspond to magnitude of activations. See Table 1 for model abbreviations. **SparseUnsup**: Activations for near tuned (top) and far tuned (bottom) weights for the sparse-coding layer. **DirectSup Sparse Control**: Activations from **DirectSup** with a threshold applied such that the number of activations matched that of sparse coding across the dataset. **DirectUnsup** did not have any nonzero activations when controlled for sparsity. Best viewed in color.

4.3 Depth Selectivity

In order to explore further differences between a supervised direct layer and a sparse coding layer that may explain the performance gap, we compared activation maps for the first layer of **SparseUnsup**, **DirectUnsup**, **DirectFinetune**, and **DirectSup** in Figure 4.

We find that the sparse-coding activations are selective to certain depths. For example, in Figure 4 for **SparseUnsup**, the top row shows a fast moving edge detector with a large binocular shift that corresponds to image features close to the camera, whereas the bottom row shows a static edge detector with no binocular shift that corresponds to image features far from the camera. In contrast, no direct layers show depth selectivity.

To control for the difference in number of nonzero activations, a threshold was applied to direct activations to match the number of nonzero activations in sparse coding, as shown in Figure 4 **DirectSup Sparse Control**. Sparse controls for **DirectUnsup** did not produce nonzero activations

on this input. Here, we show that sparsity-controlled activations from **DirectSup** do not show depth selectivity, as activations are active on image features at different depths.

5 Discussion and Future Work

We have shown that a neural network that incorporates unsupervised learning is able to outperform a fully supervised network when there exists limited labeled training data.

We show in Section 4.2 that performance of fully supervised networks can vary substantially when compared to networks with a sparse coding layer. It is likely that the cause of high variance is due to the order in which the network is presented with training examples, since order of presentation can affect supervised learning, as shown by Erhan et al. [4]. An alternative source of the variance is due to the random initialization of weights. Future work must be done to localize the source of the variance.

We did not rerun sparse coding layers multiple times to get a measure of the range in performance due to time constraints. Naturally, one hypothesis as to why **SparseUnsup** performs consistently is because the encoding is static. However, we find that **SparseUnsup** has less range than **DirectRand** and **DirectUnsup**, both of which also have static activations and weights when training for vehicle detection, which supports the hypothesis that sparse inference is key to getting consistent performance.

The goal of this paper is to determine the advantages of unsupervised learning when compared to supervised learning, which is independent of the performance in object detection itself. However, the task explored in this paper is a naive object detection task, whereas object detection in literature defines the task to be drawing bounding boxes around objects of interest. Future work must be done to determine if the findings here generalize to other networks, such as Faster R-CNN [20]. In addition, future work must be done to compare performance with additional unsupervised learning models and additional regularization techniques [24, 22, 4] to prevent overfitting.

Sparse coding solves an optimization problem when encoding, whereas a direct encoding is computed directly from a single convolution. In other words, sparse coding is more computationally expensive in finding activations than a direct convolution. Despite this disadvantage, there has been work in building specialized hardware that implement sparse coding [10, 15], which can help alleviate the computational cost. Additionally, implementations of sparse coding can take advantage of the sparsity of the encoding to limit communication of encodings. Direct approximations to sparse coding [7] may prove to be an effective compromise.

In conclusion, we have shown that a neural network with an unsupervised sparse coding layer is able to learn binocular temporal basis functions using sparse coding. The results described in Section 4.2 provide evidence that unsupervised learning techniques can help achieve better performance than supervised direct layers when only a limited number of labeled training examples are available. Additionally, direct layers in general are more susceptible to random conditions than sparse coding. Finally, we compare activations and show in Section 4.3 that depth selective activations emerge from applying sparse coding to stereo-video data. In all, these results show that unsupervised sparse coding can be useful in domains where there exists a limited amount of available labeled training data.

Acknowledgments

This research is funded by the DARPA Cooperative Agreement Award HR0011-13-2-0015. We would like to thank Dylan M. Paiton for valuable conversations pertaining to this manuscript and Wesley Chavez for proofreading.

References

[1] Horace B Barlow. Unsupervised learning. *Neural Computation*, 1(3):295–311, 1989.

[2] Adam Coates, Andrew Y Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 215–223, 2011.

[3] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR).*, pages 248–255. IEEE, 2009.

[4] Dumitru Erhan, Yoshua Bengio, Aaron Courville, Pierre-Antoine Manzagol, Pascal Vincent, and Samy Bengio. Why does unsupervised pre-training help deep learning? *Journal of Machine Learning Research*, 11(Feb):625–660, 2010.

[5] Steven H Ferris. Motion parallax and absolute distance. *Journal of experimental psychology*, 95(2):258, 1972.

[6] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? The KITTI vision benchmark suite. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[7] Karol Gregor and Yann LeCun. Learning fast approximations of sparse coding. In *International Conference on Machine Learning (ICML)*, pages 399–406, 2010.

[8] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[9] Patrik O Hoyer and Aapo Hyvärinen. Independent component analysis applied to feature extraction from colour and stereo images. *Network: computation in neural systems*, 11(3):191–210, 2000.

[10] Phil Knag, Jung Kuk Kim, Thomas Chen, and Zhengya Zhang. A sparse coding neural network asic with on-chip learning for feature extraction and encoding. *IEEE Journal of Solid-State Circuits*, 50(4):1070–1079, 2015.

[11] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1097–1105, 2012.

[12] William Lotter, Gabriel Kreiman, and David Cox. Deep predictive coding networks for video prediction and unsupervised learning. In *International Conference on Learning Representations (ICLR)*, 2017.

[13] Sheng Y Lundquist, Dylan M Paiton, Peter F Schultz, and Garrett T Kenyon. Sparse encoding of binocular images for depth inference. In *IEEE Southwest Symposium on Image Analysis and Interpretation (SSIAI)*, pages 121–124. IEEE, 2016.

[14] Roland Memisevic and Christian Conrad. Stereopsis via deep learning. In *Advances in Neural Information Processing Systems (NIPS) Workshop on Deep Learning*, volume 1, page 2, 2011.

[15] N.T.T. Nguyen and G.T. Kenyon. Solving sparse representations for object classification using quantum d-wave 2x machine. In *International workshop on Post-Moore's Era Supercomputing (MPES)*, 2016.

[16] Bruno A Olshausen. Sparse coding of time-varying natural images. In *International Conference on Independent Component Analysis and Blind Source Separation*, pages 603–608. Citeseer, 2000.

[17] Bruno A Olshausen and David J Field. Sparse coding with an overcomplete basis set: A strategy employed by V1? *Vision Research*, 37(23):3311–3325, 1997.

[18] Ning Qian. Binocular disparity and the perception of depth. *Neuron*, 18(3):359–368, 1997.

[19] Rajat Raina, Alexis Battle, Honglak Lee, Benjamin Packer, and Andrew Y Ng. Self-taught learning: transfer learning from unlabeled data. In *International conference on Machine learning (ICML)*, pages 759–766. ACM, 2007.

[20] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 91–99, 2015.

[21] Christopher Rozell, Don Johnson, Richard Baraniuk, and Bruno Olshausen. Locally competitive algorithms for sparse approximation. In *IEEE International Conference on Image Processing (ICIP)*, volume 4, pages IV–169. IEEE, 2007.

[22] Patrice Y Simard, David Steinkraus, John C Platt, et al. Best practices for convolutional neural networks applied to visual document analysis. In *International Conference on Document Analysis and Recognition (ICDAR)*, volume 3, pages 958–962. Citeseer, 2003.

[23] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations (ICLR)*, 2015.

[24] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[25] Matthew D Zeiler, Dilip Krishnan, Graham W Taylor, and Rob Fergus. Deconvolutional networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2528–2535. IEEE, 2010.