Tracking with Deep Neural Networks

Jonghoon Jin
Purdue University
jhjin@purdue.edu

Aysegul Dundar
Purdue University
adundar@purdue.edu

Jordan Bates
Purdue University
jtbates@purdue.edu

Clement Farabet
New York University
cfarabet@nyu.edu

Eugenio Culurciello
Purdue University
euge@purdue.edu

***Abstract*—We present deep neural network models applied to tracking objects of interest. Deep neural networks trained for general-purpose use are introduced to conduct long-term tracking, which requires scale-invariant feature extraction even when the object dramatically changes shape as it moves in the scene. We use two-layer networks trained using either supervised or unsupervised learning techniques. The networks, augmented with a radial basis function classifier, are able to track objects based on a single example. We tested the networks tracking capability on the TLD dataset, one of the most difficult sets of tracking tasks and real-time tracking is achieved in 0.074 seconds per frame for 320×240 pixel image on a 2-core 2.7GHz Intel i7 laptop.**

I. Introduction

Visual tracking is one of the most crucial components for robotic vision system. It is even more difficult when the object changes its scale, pose, or illumination. Many brilliant approaches in computer vision have been proposed to conduct long-term visual tracking [1]–[4], but these fail when occlusion happens slowly. Visual tracking still remains a challenging task in computer vision. It is interesting to note that the human visual system outperforms all existing machines in vision applications. Humans do not need any information or prior knowledge of the object before tracking. They are able to track unknown object with only few images.

In recent years, deep learning methods have advanced and become the dominant artificial vision system for classification and categorization problems. The architecture of deep learning is inspired by the mammalian visual system where a simple algorithm is invoked in the neocortex through a recursive hierarchy [5]–[10]. Most deep architectures follow the multi-stage Hubel-Wiesel architecture, consisting of a hierarchy of layers where each layer consists of filtering, non-linearity, and pooling stages. Recent works have shown how well deep networks can learn features from data in either a supervised [6], [11] or unsupervised manner [12]–[15] and state-of-the-art results have been achieved with optimized learning algorithms and massive datasets [16], [17]. However, most of these works are not applicable to real-time applications that are strictly constrained by the network size, the dataset volume, and the optimization algorithm.

In this paper, we present a tracking model with deep neural networks for real-time robotic vision applications. The network is trained in different ways: no training, supervised training, and unsupervised training. Our goal of this experiment is to evaluate the potential and limitations of our prototype for general purpose tasks rather than specific tasks. We measured performance with TLD dataset [18].

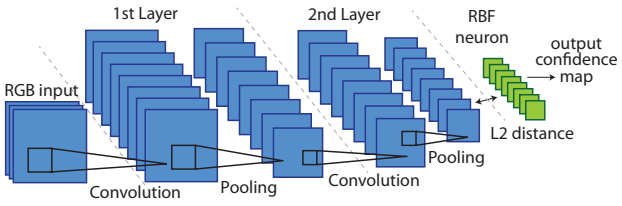


Fig. 1. Structure of our tracking system.

This paper presents as follows: Section II explains the tracking algorithm and preliminaries. Section III describes network properties used in our experiment. Section IV shows properties of dataset and experimental results and section V presents analysis and limitations of our approach.

II. Methods

In this work, we use a Convolutional Neural Network (ConvNet) [19] with filter banks trained by different training methods. Figure 1 summarizes the structure of our approach. Once training is completed, each RGB frame is applied, in the same ways as human retina senses light [20], to deep neural network and transformed into the multi-dimensional feature vectors at the output of ConvNet. Then, a radial basis function network (RBFN) is used to produce confidence map based on the distance between multi-dimensional feature vectors and a reference vector. The algorithm is described as follows:

Algorithm Tracking with convolutional neural network
Input Video frames $(I^{(1)}, ..., I^{(t)})$
Rectangle $r^{(1)}$ of object in the first frame
Output Rectangle $(r^{(2)}, ..., r^{(t)})$ of object

Initialization (for frame $I^{(1)}$):

1) Normalize frame $I^{(1)}$ by subtracting the mean and dividing the standard deviation.
2) Extract a small patch $X^{(1)}$ from rectangle $r^{(1)}$.
3) Feed the patch $X^{(1)}$ to the network and compute the single output vector $Y^{(1)} = f_{ConvNet}\left(X^{(1)}\right)$ where $Y^{(1)} \in \Re^{k \times 1}$.
4) Generate the first neuron, centered at $Y^{(1)}$, in RBFN to save the feature vector $Y^{(1)}$ for positive prototype.

Tracking (for each frame $I^{(t)}$, $t > 1$)

978-1-4673-5239-0/13/$31.00 ©2013 IEEE

1) Normalize frame $I^{(t)}$ by subtracting the mean and dividing the standard deviation.
2) Slice the whole frame $I^{(t)}$ into small patches $X_{ij}^{(t)}$ (i, j are indexes for row and column).
3) Feed the small patches $X_{ij}^{(t)}$ to the network and compute the output vectors $Y_{ij}^{(t)} = f_{ConvNet}\left(X_{ij}^{(t)}\right)$ where $Y_{ij}^{(t)} \in \Re^{k\times 1}$.
4) Produce confidence map based on the distance between inputs $Y_{ij}^{(t)}$ and the neuron $Y^{(1)}$ in RBFN.
5) Draw a rectangle $r^{(t)}$ where the peak of the map is larger than threshold τ.

We used the Torch7 software for all our training and testing experiments [21], which incredibly shortens the processing time compared to Matlab.

A. Convolutional Neural Network

Our approach to tracking tasks mostly relies on the power of the ConvNet. [6] The ConvNet uses reduced parameters in the network compared to multilayer perceptrons (MLP). Considering the fact that the size of datasets for training is determined by the number of parameters, we can greatly reduce the cost of datasets with the help of shared weights. Each layer consists of three sequential operations as follows:

- Spatial convolutions with kernels
- $tanh$ non-linearity
- Pooling

Note that the window size of last L2-pooling layer varies depending on the size of the object that we are interested in since convolution does not give fixed output size for any size of input.

1) Spatial Convolution: Each layer contains k feature maps h_{ijk} and the equation for the feature map is described as follows:

$$h_{ijk}^{(t)} = tanh\left(W_k * X_{ij}^{(t)} + b_k\right) \tag{1}$$

Note that b_k is a bias, $W \in \Re^{n\times n}$ is a kernel and $X \in \Re^{n\times n}$ is an image patch where n is the size of the window. The shared weights W^k are independent of time index i and spatial index j.

2) Pooling: Pooling is applied in order to create spatial invariance while passing the distinct features to the next layer. Spatial L2-pooling is frequently used and defined as:

$$y_k^{(t)} = \sqrt{\frac{1}{mn}\sum_{i=1}^{m}\sum_{j=1}^{n}\left\|h_{ijk}^{(t)}\right\|_2^2} \tag{2}$$

where $m \times n$ is the size of the window and $y_k^{(t)}$ is the k-dimensional vector.

B. Radial Basis Function Network

The output feature vector from the network is passed to the RBFN. It computes the distance from the reference vectors defined as centroids of neurons and finds the closest match.

$$z\left(Y\right) = \sum_{i=1}^{N} w_i \phi\left(\|Y - Y_i\|\right) \tag{3}$$

where X_i is a multi-dimensional feature vector of the object and N defines the number of prototypes of the object. w_i is a weight and we used a 32-dimensional Gaussian function for radial basis function $\phi\left(x\right)$.

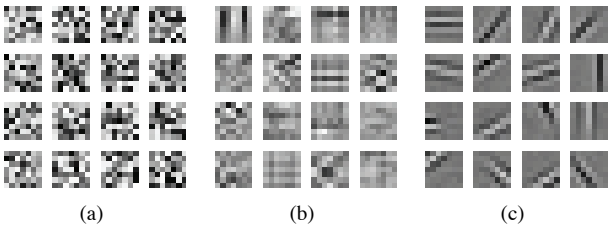


Fig. 2. Kernels of 1st layer from each network: (a) kernels randomly initialized, (b) kernels trained with back-propagation in a supervised method, (c) kernels trained with K-means clustering in an unsupervised method.

III. Network properties

We used three different networks for our experiment: network without pre-training, network with supervised and network with unsupervised learning methods. Figure 2 shows kernels from different networks at a glance. All kernels in the network are 7×7. The first layer and the second layer contain 32 kernels respectively. The connections are established by the random connection matrix in the Torch7 library. L2-Pooling is added to the output of each layer and the size of pooling window is 2×2 with the step size of 2.

A. Parameters with random initialization

We initialize the network with random parameters. Kernels in the first layer are randomly generated from normal distribution with zero mean and a standard deviation of 0.8. For the second layer, kernels follow a normal distribution with zero mean and a standard deviation of 0.4. These parameters are chosen empirically. Using high standard deviation saturates all network outputs to ± 1 since the logistic function $tanh$ is bound by that value.

B. Parameters with supervised learning method

Next, we use a network with supervised training. This network is trained with fully labeled images from the *Barcelona* dataset to learn low-level features [22]. Using back-propagation with stochastic gradient descent (SGD), supervised learning forces kernels to learn common features from the dataset quickly compared to unsupervised learning methods.

C. Parameters with unsupervised learning method

The last network for our experiment is trained with an unsupervised K-means clustering learning [23]. The advantage of K-means learning is that it is fast and more biologically plausible than SGD. We used CIFAR10 dataset [24] to train

our network. No labeled data is required for training and only the parameter is the number of centroids. K-means updates kernels by averaging given inputs and their closest centroids as follows:

$$w_j^{(i)} = \begin{cases} w_j^{(i)} + D^{(j)\top} x^{(i)} & \text{if } j = \arg\max_l \left| D^{(l)\top} x^i \right| \\ w_j^{(i)} & \text{otherwise} \end{cases} \tag{4}$$

where $x^{(i)}$ is a random patch from dataset and D is a dictionary of kernels. $w_j^{(i)}$ is j-th kernel and should be normalized after updates.

IV. Experimental Results

We evaluate the performance of our system on a challenging benchmark, the TLD dataset. It is an appropriate dataset to evaluate performance for tracking task since each video sequence has various challenges such as occlusion and scale/pose/illumination changes as summarized in the table I. Each video contains only one target and its ground-truth. Performances are measured by the F-measure $F = \frac{PR}{P+R}$. Note that Precision P is a rate of correctly predicted results among all predictions labeled as positive and Recall R is a rate of correctly predicted results among actual positives in ground-truth. Detection is classified as positive if the overlapped area between bounding boxes from prediction and ground-truth is greater than 25 percent.

Through our experiment, tracking performance is measured and presented in table II. Our tracker shows the best performance for the car dataset, and this beats the state-of-the-art result of the P-N tracker [3]. However, overall performance fluctuates for each dataset and in fact the records from our network are not as comparable as those from various approaches in computer vision.

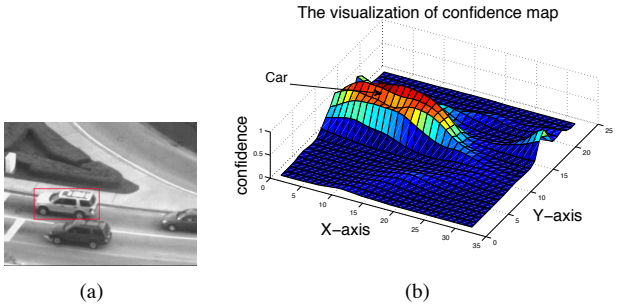
To focus on the performance versus the way the networks are trained, the network with K-means clustering performs the best prediction among others. It is interesting to note that the random network shows the best records for some datasets such as *Pedestrian 1, Motocross* and *Panda*.

For the sake of the parallel structure of our system, it takes about 0.074 seconds to process one frame (13.5 fps) which is 320×240 pixel image on a 2-core 2.7GHz Intel i7 laptop.

V. Analysis and Limitation

The tracking system described in this paper is built with deep neural networks which is a novel approach. This model is based on the idea of neocortical algorithm that human brain processes a simple and universal algorithm over the cortex through the recursive hierarchy. Convolutions with kernels followed by poolings is recursively applied. This is similar to H-MAX [10]. We train filters with different types of learning methods instead of using hard-wired Gabor filters.

The results in table II demonstrates that the unsupervised network with Gabor-like filters has the best performance in tracking which also has been proven in many literatures [9], [12], [23], [26]. The combination of deep neural network and radial basis function helps to solve the classification



(a) (b)

Fig. 3. (a) The input frame with the target from *Car* dataset. (b) The confidence map showing a foreground-background separation by deep neural network and radial basis function network.

problems easier by extracting features from inputs space and transforming them into high dimensional vector via $tanh$ nonlinear mapping. The rationale for this is based on *Cover's theorem* that a pattern classification problem cast in a nonlinear high-dimensional space is more likely to be linearly separable than in a low-dimensional space [27]. The confidence map in figure 3 shows a success in the foreground-background separation with the transformation and the object tracking is performed.

Many groups in machine learning focus on huge networks with sophisticated learning techniques [16], [17], but most of them cannot be applied to robotic vision, whereas our aim is to implement a real-time application for robotic tasks. The parallel structure of our model increases the chance of acceleration by multi-processor or custom hardware [28].

However, the deep network as a tracker fails in some datasets because of the limitation in feature representation. The goal of a classification problem is to label and classify data into several categories, but it seems to neglect detail information (i.e., cat's appearance). For robust tracking, the deep neural network should be able to catch not only features but also appearance. We found that the supervised network is not suitable for a general purpose system such as tracking an unknown object. The reason for the failure is that all parameters in the network are adjusted and aligned to increase confidence on the specific categories of the labeled dataset and in the process loses information not relevant to that task. This type of failure is observed in the *Pedestrian 2*, *Pedestrian 3*, and *Carchase* datasets.

The deep neural network with random kernels in our experiment shows comparable performance to the result of supervised network. Similar results have been observed in recent works [29], [30]. These results from randomness support the fact that the random kernels themselves can be considered as a set of basic blocks and they are able to extract features in spite of unknown random pattens. Then, the features are organized by the architecture and pipelined to the next layer. This implies that selection of the network architecture is as important as the training method.

Video Sequence									
Properties	David	Jumping	Pedestrian1	Pedestrian2	Pedestrian3	Car	Motocross	Carchase	Panda
Camera Movement	yes	yes	yes	yes	yes	yes	yes	yes	yes
Partial occlusion	yes	no	no	yes	yes	yes	yes	yes	yes
Full occlusion	no	no	no	yes	yes	yes	yes	yes	yes
Pose change	yes	no	no	no	no	no	yes	yes	yes
Illumination change	yes	no	no	no	no	no	yes	yes	yes
Scale change	yes	no	no	no	no	no	yes	yes	yes
Similar objects	no	no	no	yes	yes	yes	yes	yes	no

TABLE I
PROPERTIES OF THE VIDEO DATASET USED IN OUR EXPERIMENT [25].

Sequence	Frames	ConvNet with random parameters (No pre-learning) Precision/Recall/F-measure	Supervised ConvNet (back-propagation with SGD) Precision/Recall/F-measure	Unsupervised ConvNet (K-means clustering learning) Precision/Recall/F-measure
David	761	0.08 / 0.05 / 0.06	**0.12 / 0.07 / 0.09**	0.08 / 0.05 / 0.06
Jumping	313	0.28 / 0.28 / 0.28	**0.51 / 0.51 / 0.51**	0.22 / 0.22 / 0.22
Pedestrian 1	140	**0.81 / 0.81 / 0.81**	**0.81 / 0.81 / 0.81**	0.64 / 0.64 / 0.64
Pedestrian 2	338	0.36 / 0.46 / 0.41	0.35 / 0.45 / 0.39	**0.61 / 0.64 / 0.63**
Pedestrian 3	184	0.41 / 0.49 / 0.45	**0.48 / 0.57 / 0.52**	0.47 / 0.56 / 0.51
Car	945	0.68 / 0.73 / 0.70	0.37 / 0.40 / 0.39	**0.97 / 0.96 / 0.97**
Motocross	2665	**0.14 / 0.26 / 0.18**	0.12 / 0.23 / 0.16	**0.14 / 0.26 / 0.18**
Carchase	9928	0.20 / 0.21 / 0.21	0.25 / 0.26 / 0.25	**0.38 / 0.43 / 0.40**
Panda	3000	**0.40 / 0.44 / 0.41**	0.36 / 0.40 / 0.38	0.26 / 0.29 / 0.28
Mean	18274	0.26 / 0.29 / 0.27	0.26 / 0.29 / 0.28	**0.35 / 0.39 / 0.37**

TABLE II
PERFORMANCE COMPARISON OF TRACKERS. ALL TRACKERS SHOWN IN THE TABLE IS TRAINED ON THE FIRST FRAME ONLY. WE USED THE FIXED THRESHOLD $\tau = 0.7$ AND ADAPTIVE SPREAD σ FOR RADIAL BASIS FUNCTION.

VI. CONCLUSION

We introduce a deep neural network to solve tracking problems. It extracts distinct features and transforms image patches to high dimensional vectors. Then a radial basis function computes the similarity between prototypes and generates a confidence map. The new position of the object is found at the peak of the map.

We have shown that deep neural networks are able to track the target though performance fluctuates depending on the training method. In addition, our architecture can be processed in parallel and is similar to how the human brain works.

More importantly, training a network in a strongly supervised way does not guarantee high performance for a general purpose system like tracking an unknown object. This is closely connected to the question of how we achieve the best feature representation from a dataset and further study is needed to clarify this issue.

ACKNOWLEDGMENT

Work partly supported by Office of Naval Research (ONR) grant N00014-10-1-0278 and ONR N00014-11-1-0287 - PECASE.

REFERENCES

[1] S. Avidan, "Ensemble tracking," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 29, no. 2, pp. 261 –271, feb. 2007.

[2] D. Comaniciu, V. Ramesh, and P. Meer, "Kernel-based object tracking," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 25, no. 5, pp. 564 – 577, may 2003.

[3] Z. Kalal, J. Matas, and K. Mikolajczyk, "P-n learning: Bootstrapping binary classifiers by structural constraints," *Conference on Computer Vision and Pattern Recognition*, 2010.

[4] A. Dundar, J. Jin, and E. Culurciello, "Visual tracking with similarity matching ratio," *arXiv preprint arXiv:1209.2696*, 2012. [Online]. Available: http://arxiv.org/abs/1209.2696

[5] R. Hadsell, S. Chopra, and Y. LeCun, "Dimensionality reduction by learning an invariant mapping," in *Proc. Computer Vision and Pattern Recognition Conference (CVPR'06)*. IEEE Press, 2006.

[6] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[7] K. Gregor, A. Szlam, and Y. LeCun, "Structured sparse coding via lateral inhibition," in *Advances in Neural Information Processing Systems (NIPS 2011)*, vol. 24, 2011.

[8] M. Riesenhuber and T. Poggio, "Hierarchical models of object recognition in cortex," *Nature neuroscience*, vol. 2, p. 10191025, 1999. [Online]. Available: http://journal.mercubuana.ac.id/data/Hierarchical-models-of-object-recognition-in-cortex.pdf

[9] T. Serre, A. Oliva, and T. Poggio, "A feedforward architecture accounts for rapid categorization," *Proceedings of the National Academy of Sciences*, vol. 104, no. 15, p. 64246429, 2007. [Online]. Available: http://www.pnas.org/content/104/15/6424.short

[10] T. Serre and T. Poggio, "A neuromorphic approach to computer vision," *Communications of the ACM*, vol. 53, no. 10, p. 5461, 2010. [Online]. Available: http://dl.acm.org/citation.cfm?id=1831425

[11] P. Sermanet and Y. LeCun, "Traffic sign recognition with multi-scale convolutional networks," pp. 2809 –2813, 31 2011-aug. 5 2011.

[12] B. A. Olshausen and D. J. Field, "Emergence of simple-cell receptive field properties by learning a sparse code for natural images," *Nature*, vol. 381, no. 6583, p. 607609, 1996. [Online]. Available: http://people.cs.umass.edu/~eshelham/files/papers/olshausen-field-emergence-simple-cells.pdf

[13] A. Hyvärinen and E. Oja, "Independent component analysis: algorithms and applications," *Neural networks*, vol. 13, no. 4, p. 411430, 2000. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0893608000000265

[14] G. E. Hinton, S. Osindero, and Y. W. Teh, "A fast learning algorithm for deep belief nets," *Neural computation*, vol. 18, no. 7, p. 15271554, 2006. [Online]. Available: http://www.mitpressjournals.org/doi/abs/10.1162/neco.2006.18.7.1527

[15] P. Vincent, H. Larochelle, Y. Bengio, and P. A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in

Proceedings of the 25th international conference on Machine learning, 2008, p. 10961103. [Online]. Available: http://dl.acm.org/citation.cfm?id=1390156.1390294

[16] Q. Le, J. Ngiam, A. Coates, A. Lahiri, B. Prochnow, and A. Ng, "On optimization methods for deep learning," in *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, ser. ICML '11, L. Getoor and T. Scheffer, Eds. New York, NY, USA: ACM, June 2011, pp. 265–272.

[17] J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, Q. Le, M. Mao, M. Ranzato, A. Senior, P. Tucker, K. Yang, and A. Ng, "Large scale distributed deep networks," in *Advances in Neural Information Processing Systems 25*, P. Bartlett, F. Pereira, C. Burges, L. Bottou, and K. Weinberger, Eds., 2012, pp. 1232–1240. [Online]. Available: http://books.nips.cc/papers/files/nips25/NIPS2012_0598.pdf

[18] Z. Kalal, "TLD dataset," http://info.ee.surrey.ac.uk/Personal/Z.Kalal/TLD/TLD_dataset.pdf, 2010. [Online]. Available: http://info.ee.surrey.ac.uk/Personal/Z.Kalal/TLD/TLD_dataset.pdf

[19] Y. LeCun, K. Kavukvuoglu, and C. Farabet, "Convolutional networks and applications in vision," in *Proc. International Symposium on Circuits and Systems (ISCAS'10)*. IEEE, 2010.

[20] B. A. Wandell, *Foundations of vision*. Sinauer Associates, 1995. [Online]. Available: http://psycnet.apa.org/psycinfo/1995-98050-000

[21] R. Collobert, K. Kavukcuoglu, and C. Farabet, "Torch7: A matlab-like environment for machine learning," in *NIPS Workshop BigLearn*, 2011. [Online]. Available: http://ronan.collobert.com/pub/matos/2011_torch7_nipsw.pdf

[22] C. Farabet, C. Couprie, L. Najman, and Y. LeCun, "Learning hierarchical features for scene labeling," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 99, no. PrePrints, p. 1, 2012.

[23] A. Coates, H. Lee, and A. Y. Ng, "An analysis of single-layer networks in unsupervised feature learning," *AISTATS*, vol. 14, 2011. [Online]. Available: http://www.stanford.edu/~acoates/papers/coatesleeng_aistats_2011.pdf

[24] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," *Master's thesis, Department of Computer Science, University of Toronto*, 2009. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.222.9220&rep=rep1&type=pdf

[25] Z. Kalal, K. Mikolajczyk, and J. Matas, "Tracking-learning-detection," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 34, no. 7, p. 14091422, 2012. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6104061

[26] H. Lee, A. Battle, R. Raina, and A. Y. Ng, "Efficient sparse coding algorithms," in *Advances in Neural Information Processing Systems 19*, B. Schölkopf, J. Platt, and T. Hoffman, Eds. Cambridge, MA: MIT Press, 2007, pp. 801–808.

[27] T. M. Cover, "Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition," *Electronic Computers, IEEE Transactions on*, vol. EC-14, no. 3, pp. 326 –334, june 1965.

[28] C. Farabet, B. Martini, B. Corda, P. Akselrod, E. Culurciello, and Y. LeCun, "Neuflow: A runtime reconfigurable dataflow processor for vision," in *Computer Vision and Pattern Recognition Workshops (CVPRW), 2011 IEEE Computer Society Conference on*, 2011, p. 109116. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5981829

[29] E. Culurciello, J. Bates, A. Dundar, J. Pérez-Carrasco, and C. Farabet, "Clustering learning for robotic vision," *arXiv preprint arXiv:1301.2820*, 2013. [Online]. Available: http://arxiv.org/abs/1301.2820

[30] A. Saxe, P. W. Koh, Z. Chen, M. Bhand, B. Suresh, and A. Ng, "On random weights and unsupervised feature learning," in *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, ser. ICML '11, L. Getoor and T. Scheffer, Eds. New York, NY, USA: ACM, June 2011, pp. 1089–1096.