Feature Selection using Deep Neural Networks

Debaditya Roy *, K. Sri Rama Murty † and C. Krishna Mohan *
*Visual Learning and Intelligence Group (VIGIL), Department of Computer Science and Engineering
†Department of Electrical Engineering
Indian Institute of Technology Hyderabad, India
{cs13p1001, ksrm, ckm}@iith.ac.in

***Abstract*—Feature descriptors involved in video processing are generally high dimensional in nature. Even though the extracted features are high dimensional, many a times the task at hand depends only on a small subset of these features. For example, if two actions like running and walking have to be identified, extracting features related to the leg movement of the person is enough. Since, this subset is not known apriori, we tend to use all the features, irrespective of the complexity of the task at hand. Selecting task-aware features may not only improve the efficiency but also the accuracy of the system. In this work, we propose a supervised approach for task-aware selection of features using Deep Neural Networks (DNN) in the context of action recognition. The activation potentials contributed by each of the individual input dimensions at the first hidden layer are used for selecting the most appropriate features. The selected features are found to give better classification performance than the original high-dimensional features. It is also shown that the classification performance of the proposed feature selection technique is superior to the low-dimensional representation obtained by principal component analysis (PCA).**

Keywords. *Supervised Feature Selection, Deep Neural Networks, Action Recognition*

I. Introduction

Understanding human actions in videos is challenging because of high variability in temporal scale, complexity of articulated motion and high degree of freedom in movements. Even for identifying a small number of actions, it is rather difficult to accurately represent action information with a small set of features. Features are either extracted from individual frames [1] or from entire videos [2]. Among them, one of the most successful [3] feature descriptors is improved dense trajectory (IDT) [4] which is a concatenation of well known features like Histogram of Oriented Gradients (HOG) [5], Histogram of Oriented Optical Flow (HOOF) [1] etc. However, high dimensional representations like IDT (426 D) could contain redundant dimensions which may not be necessary for classification of all kinds of actions. Therefore, based on the nature of the task at hand (actions classes), dimensionality of such representations can be reduced without affecting classification performance. For experimental verification of the same, a DNN with high dimensional input was built for action classification. A smaller network formed with these selected features can be used in portable devices with modest processing and memory requirements. Further, the entire network can be stored in-place in the memory leading to speedup [6].

A. Related Work

Majority of the methods for reducing network complexity rely on pruning non-essential weights [7]–[9] in intermediate layers or reducing input dimensionality [10]–[13]. The basic objective of network pruning is to obtain a sparse network which can be trained with low computational complexity. Weight pruning is carried out either by setting those weights to 0, which when perturbed, lead to no change in error rate [13] or by penalizing the cost function during back-propagation [14], [15], [8], [16], [17] so that the unnecessary weights are driven to 0. However, these network pruning methods as described in [10] and [18] require extensive analysis of hidden layers which is difficult in case of DNN with high dimensional inputs.

DNNs have always been hard to train especially when the input dimension is very high. There have been a number of attempts to address this issue. In [19], Denil et al. show that given a few weights for each feature, it is possible to not only predict all the other weights but also eliminate some of the weights. It is shown for multi-layer perceptrons, learning 25% of the parameters achieves the same error as learning all the weights. Sainath et al. [20] reduce the number of parameters in the last layer of a DNN using low-rank matrix factorization. The *softmax* layer generally used at the last layer for classification is more suitable for low-rank factorization but on other layers error rate increases with low-rank structure imposed. Moreover, last layer method was shown to be suitable for networks with large number of output classes (2200, 5999).

One of the most popular methods for training large DNNs effectively is using *maxout* activation function [21] which essentially outputs the maximum of all inputs and is a piecewise linear approximation to an arbitrary convex function. While it generally leads to a lower complexity in networks with reduced parameters compared to rectifier networks, to be truly effective on real datasets pre-processing in the form of cross-channel pooling is necessary to achieve good performance. However, no such preprocessing is required for the proposed feature selection method and still a reduction in the number of parameters can be achieved.

In the proposed method we select important features by analyzing the first layer activation potentials of a DNN classifier. The method is completely supervised with emphasis on the discrimination potential of features. This is a departure from unsupervised dimensionality reduction methods like PCA where emphasis is on best representation and reconstruction.

The rest of the paper is organized as follows. Section 2 describes reveals the baseline system for evaluating the performance of DNNs on action recognition. Section 3 describes the methodology behind the feature selection method. In section 4 the results and subsequent implications are shown and finally in section 5, the conclusions on the proposed method are presented.

978-1-4799-1959-8/15/$31.00

II. Baseline system for action recognition

Action recognition in videos is gradually becoming prominent in the field of computer vision and machine learning, with important applications like surveillance, human behaviour understanding etc. Finding a task-aware representation of videos for reliable action recognition is quite challenging and this paper aims to address this concern.

A. KTH Dataset

The KTH dataset [22] is a controlled dataset consisting of six human action classes, namely, walking, jogging, running, boxing, hand waving, and hand clapping. Each action is performed by 25 subjects in four different scenarios: outdoors, outdoors with scale variation, outdoors with different clothes and indoors. Each video clip is roughly 4 seconds in duration shot with a frame rate of 25 fps. With 100 videos each for an action, KTH is commonly used to benchmark any algorithm on action recognition as it has sufficient examples for training data-intensive models like DNNs with the added simplicity of no occlusion and minimum background noise. Since, no official splits are provided, a split of $70 - 10 - 20$ for train-validation-test was used in the experiments.

B. Dense Trajectory features

Improved Dense Trajectory Features (IDT) describe human actions in a video by tracking the movement of particles in a neighborhood. It was shown to be the explicit feature ensemble for action recognition and is easily scalable to large number of classes. In a nutshell, each feature vector defines the path of a particle, in this case a pixel, in a restricted neighborhood for a finite number of frames. The dimension of the feature vector is 426 for each tracked point with heavy overlap to ensure full coverage of the motion. It completely conveys the information about the absolute movement, position, relative movement of the particle.

- **Trajectory:** The first 30 dimensions depict the change in position of the particle measured over 15 frames and are known as the dense trajectory points. They are extracted over a neighborhood of $2 \times 2 \times 3$ where the first two dimensions denote spatial proximity and the third denotes temporal proximity to the pixel being tracked. An example of the trajectory points for running is shown in figure 1.
- **HOG:** The next 108 dimensions depict HOG features which localize the location of the particle in relation to the video frame. HOG have been shown to be excellent human detectors [5] and describe a particle as a measure of the dominant gradients in its neighborhood. In figure 2, the HOG features detected for a person in a frame while clapping and walking are shown. In case of a particle, the HOG feature describe the dominant shape of the particle like the shoulder or hands.
- **HOOF:** HOOF features [1] form the next 96 dimensions. HOOF is generally a description of movement of a particle in subsequent frames where a small neighborhood (3×3 or 5×5) is considered, to determine the direction and movement of the particle. Also, due to the change in scale of the actions a pyramidal approach is considered. For instance, in figure 3 the optical flow or HOOF features are shown for running and hand waving actions. Note that in case the background is static, as in this case, HOOF portrays the motion signature as in the hand motion for hand waving and the entire body motion for running. The difference between HOOF and dense trajectory stems from the fact that HOOF tracks the movement of all pixel neighborhoods whereas dense trajectory only tracks the movement of particles which remain in their neighborhood during the entire duration of 15 frames.
- **MBH:** The last two features considered are Motion Boundary Histograms (MBH) in both horizontal and vertical direction denoted as MBHx and MBHy (96 dimensions each) which quantify the relative motion between two particles in both the vertical and horizontal direction. This feature is mainly used to reduce the contribution of camera motion to optical flow features and stabilizes the optical flow features.

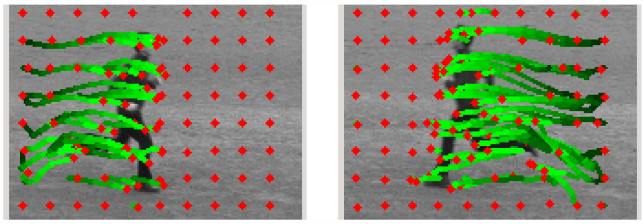
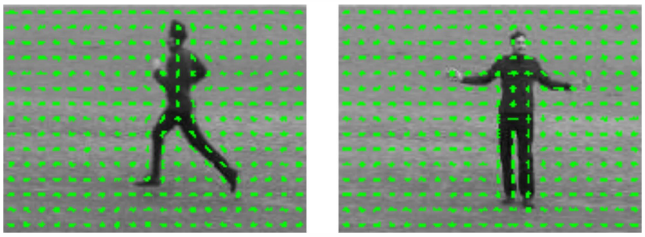
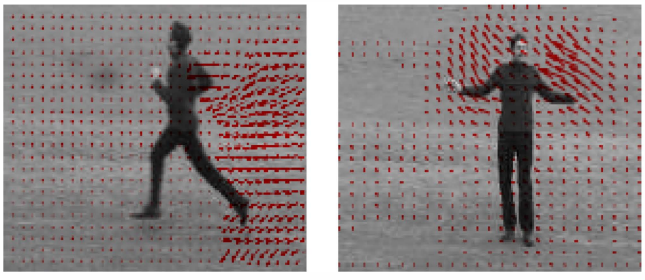


Fig. 1. Dense Trajectory features for two instances of running



(a) (b)

Fig. 2. HOG features calculated on (a) running and (b) clapping. Notice that the gradients align along the outline of the body.



(a) (b)

Fig. 3. HOOF features calculated on (a) running and (b) hand waving

TABLE I. PERFORMANCE OF BASELINE NEURAL NETWORK ON KTH DATASET

Feature	Number of hidden layers		
	1	2	3
IDT (426 D)	83.24	86.60	89.57

C. Baseline Neural Network

The architecture for the DNNs used for action recognition consist of 1 ($426L - 1000R - 6S$), 2 ($426L - 1000R - 1000R - 6S$) and 3 ($426L - 1000R - 1000R - 1000R - 6S$) hidden layers trained in pylearn2 library [23]. The entire experiment was carried out by employing discriminative pre-training [24] where each hidden layer was subject to a dropout [25] probability of 0.4. Learning rate was adjusted based on the network performance on the validation data. For the 3 networks reported above, the classification performance is reported in table I. Inquiries were made on whether all the input dimensions participated equally in classification and if so, what was the extent of the participation. The level of contribution would determine the usefulness of that dimension. Further, only a few useful dimensions could replicate the performance of the entire network and maybe even better it. This led to the identification of significant features and the effect they have on both on the original network and their performance, in isolation for a less-complex DNN.

III. FEATURE SELECTION

The main idea behind the proposed approach is to analyze the contribution of each of the input dimensions to identify the features (inputs) important for classification. Typically sensitivity analysis [26] is used to show the importance of individual input dimensions on output by perturbing the weights connected to the input. The upper bound on the sensitivity for any layer of a multi-layer perceptron network determines the optimal number of neurons required in the network. However for ReLU, such analysis is often not required as the neurons which are inactive may not get trained at all [27]. For a DNN, sensitivity analysis does not work well beyond 1 or 2 layers. Hence, to correctly analyze the contribution of an input feature, we study its activation potential (averaged over all training values of the input and hidden neurons) relative to the total activation potential. The higher the activation potential contribution of an input dimension, the more likely is its participation in hidden neuronal activity and consequently, classification.

A. Activation Potential analysis

The activation potential of the first layer of the baseline neural network (with 3 hidden layers) were analyzed for selecting the important features. The output of the j^{th} neuron at the 1^{st} layer is:

$$ReLU : f_R(a_j) = max(0, a_j) \tag{1}$$

where a_j is the activation potential of j^{th} hidden neuron, computed as:

$$a_j = \mathbf{w}_j^T \mathbf{x} + b_j \tag{2}$$

where $\mathbf{w}_j$ is the weight vector connecting j^{th} hidden neuron, $\mathbf{x}$ is the input vector and b_j is the bias applied to the neuron.

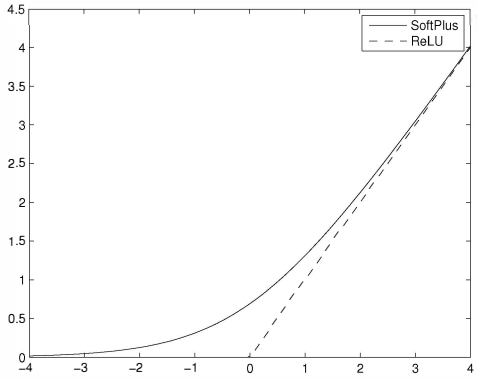


Fig. 4. ReLU and SoftPlus functions

ReLU is typically approximated with a *softplus* function shown in fig. 4.

$$SoftPlus : f_S(a_j) = log(1 + e^{a_j}) \tag{3}$$

For x_i, i.e. the i^{th} dimension of the input example $\mathbf{x}$ connected to j^{th} hidden neuron by w_{ji}, the activation potential is calculated as:

$$a_{ij} = w_{ji} x_i + b_j \tag{4}$$

The average absolute activation potential contributed by the i^{th} dimension of M training examples $x^{(1)}, x^{(2)}, ..., x^{(k)} ..., x^{(M)}$ connected to j^{th} hidden neuron is given by:

$$p_{ij} = \frac{1}{M} \sum_{k=1}^{M} |a_{ij}^{(k)}| \tag{5}$$

where k represents k^{th} training example $x^{(k)}$. The absolute value is taken to penalize large negative weights for adversely contributing to the activation of the neurons.

The relative contribution i^{th} input dimension towards the activation potential of j^{th} hidden neuron is calculated as:

$$c_{ij} = \frac{a_{ij}}{\sum_{i=1}^{N_{inp}} p_{ij}} \tag{6}$$

where N_{inp} is the dimension of input example $\mathbf{x}$. The net positive contribution $\mathbf{c}_i^+$ of an input dimension i over all hidden neurons is given as:

$$\mathbf{c}_i^+ = \sum_{j=1}^{N_{hid}} f_R(c_{ij}) \tag{7}$$

where N_{hid} is the number of neurons in the first hidden layer.

Since, the entire network is built on ReLU units the visible-hidden pair (i, j), negative c_{ij} can be set to 0 to show that i^{th} input dimension does not contribute to the activation of j^{th} hidden neuron. The same fact can also be used in selecting significant features. If we observe each of the input features in isolation and its net positive contribution to the activation of the neurons $\mathbf{c}_i^+$, it can be concluded that the features with highest $\mathbf{c}_i^+$ are more likely to instigate the neurons to participate in classification.

The activation potential was analyzed for the first hidden layer of $426L - 1000R - 1000R - 1000R - 6S$ network since

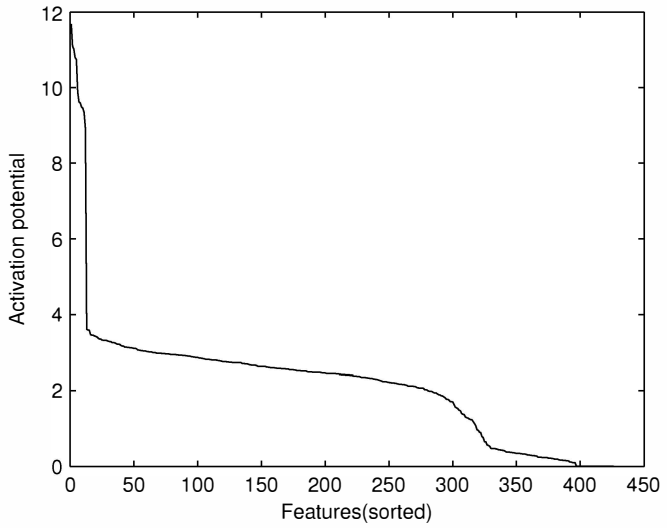


Fig. 5. Net positive contribution due to each of the input dimensions in sorted order.

direct correlation between inputs and pre-activation can be established only in this layer. In figure 5, the input features are sorted in decreasing order of $\mathbf{c}_i^+$ to highlight the big change in activation potential contribution. Upon closer inspection, roughly after 30 features there is a big dip in the contribution. Taken as an ensemble, the top 30 features have contributions $\mathbf{c}_i^+$ which is almost 3 times the activation potential of the ensemble of the next 200-odd features according to the decreasing order of activation potential. This shows that these features contribute heavily to classification in the network and bring about most of the neural interactions. Then there is a pronounced decline in the activation potential and hence, the network with first 100 features does not show any improvement in classification. After about 300 features, there is a gradual gradient and contribution of the last 30 features is almost negligible as compared to the top 30. It was observed that the behaviour of $\mathbf{c}_i^+$ is almost similar for both 1-layer and 3-layer networks, indicating the importance of the selected features.

IV. Results and Discussion

In order to validate our hypothesis about the importance of features with higher action potential, the top 30 features recovered from the average pre-activation dynamics study were then tested in a neural network $30L-100R-6S$ with roughly the same dropout as in the baseline neural network. The classification results are presented in table II. It is interesting to note that classification performance indeed improves on the network with far fewer parameters. In the 1-hidden layer network there is an improvement of about 2% and the gap widens to almost 4% as the network grows to 3-hidden layers. However, it is important that these top features can be obtained after a thorough analysis of only a 1-hidden layer baseline network.

The confusion matrices depicted in figure 6 show the classification results on each class for 1, 2 and 3 hidden layer networks. It can be seen that the misclassification is mainly caused among classes which look alike like running, walking and jogging. At the video level, the assigned label is measured as the majority of the frame labels obtained from the network. Classification performance of 90.75%, 92.43%, 98.31% are reported for 1, 2 and 3 hidden layer networks with top 30 features. The last network improves the classification results presented on KTH in [2] with action bank features.

TABLE II. Performance of DNN with top-30 features on KTH dataset

Feature	Number of hidden layers		
	1	2	3
Top-30 features	85.63	88.9	93.93

TABLE III. Average runtime performance per epoch of the top-30 and original network on KTH dataset

Feature	Criteria	Number of hidden layers		
		1	2	3
Top-30 features	Runtime (in sec.)	78	86	115
	Parameters	3600	13600	23600
IDT (426 D)	Runtime (in sec.)	253	492	1008
	Parameters	432000	1432000	2432000

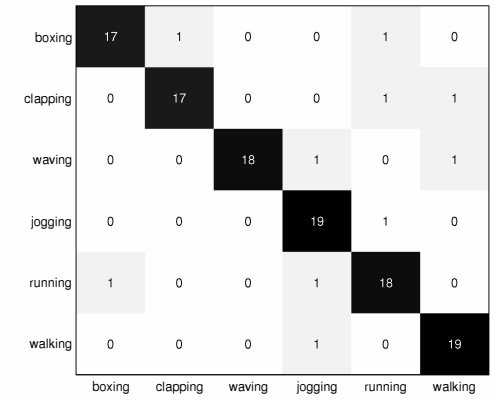
Further comparison between top 30 components chosen by principal component analysis (PCA) and by the proposed feature selection method are shown in table IV. The proposed approach produces a 3-hidden layer $30L-100R-100R-100R-6S$ network with better classification performance. Also, classification performance of top-100 features and the last-30 features according to the proposed scheme are also presented. The top 100 features show no improvement over the original network whereas the last-30 features perform worse. Also, according to the t-sne [28] visualization of feature points presented in fig. 7, it can also be seen that the most confusing classes viz. jogging and running are better separated for the top 30 features as compared to the next 30 or last 30 features.

A. Runtime analysis

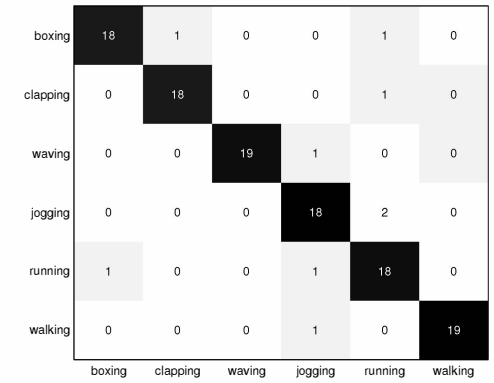
Feature selection on a 3-hidden layer $426L-1000R-1000R-1000R-6S$ original network to a less complex $30L-100R-100R-100R-6S$ network reduces the number of parameters to a mere 1% of the original. The machine of choice was a GPU server with 64 GB memory, 6 NVIDIA Tesla K20Xm GPUs with 6 GB memory each and an Intel Xeon 32-core processor. The runtime characteristics of both networks are recorded in table III. We show that even from a 1-hidden layer network onwards, the network with lower complexity performs better than the original network with a fraction of the training time as the original network. The number of parameters is mainly responsible for this huge decrease in average training time per epoch. It is worth noting that even a 1-hidden layer network with the original 426 input dimensions trains slower than a 3-hidden layer top-30 network

TABLE IV. Classification performance comparison of different networks with input features chosen with feature selection and PCA on KTH dataset

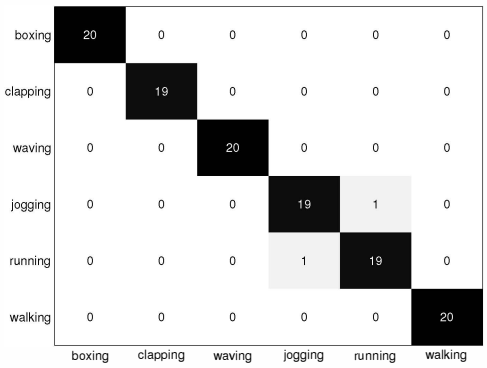
Features	Classification Performance (3 layers)
Feature Select-Top30	93.93
PCA-Top30	79.22
Feature Select-Top100	89.57
Feature Select-Last30	70.41



(a)

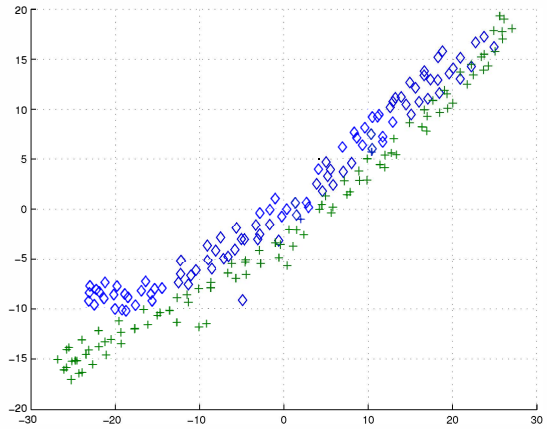


(b)

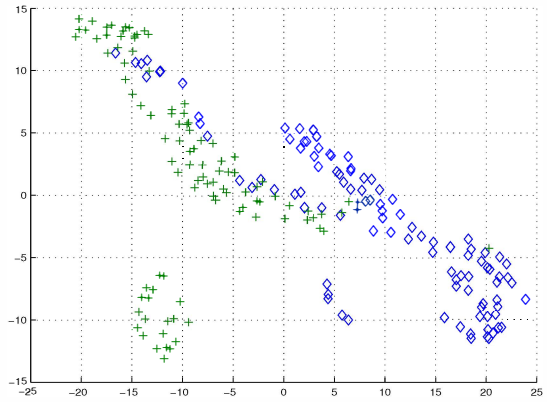


(c)

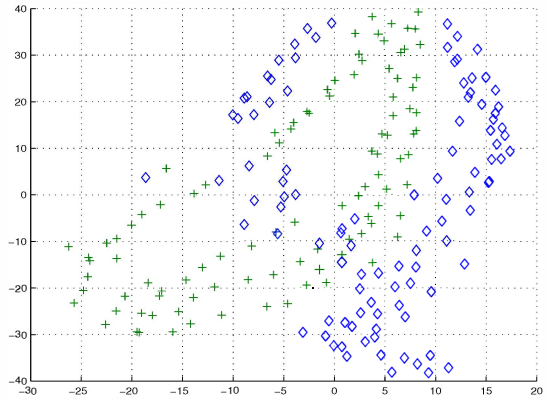
Fig. 6. Confusion matrices for the top-30 dimensions on the KTH dataset (clip-wise) (a) 1 hidden layer (b) 2 hidden layers (c) 3 hidden layers



(a)



(b)



(c)

Fig. 7. t-sne visualization of confusing classes: jogging and running. (a) Feature Select-Top30 (b) Feature Select-31-60 (c) Feature Select-Last30

by a factor of $1/2$ and the new network shows a 10% increase in terms of classification accuracy.

V. Conclusion

Selection of the right features for action classification in videos is extremely challenging. There are lot of feature descriptors available today that produce high dimensional features to describe the activity in the video but to quantify the effect of these features on classification requires extensive analysis. While dimension reduction techniques are available to reduce feature dimensions, their primary focus is good reconstruction and the discriminative information be lost in low dimensional space. Moreover, the aim of these methods is projection rather than selection which is the focus of the proposed method. To this effect, a supervised selection of features is proposed using a neural network to achieve better or comparable classification performance. Runtime gains are also obtained as the result of the reduced number of parameters.

References

[1] R. Chaudhry, A. Ravichandran, G. Hager, and R. Vidal, "Histograms of oriented optical flow and binet-cauchy kernels on nonlinear dynamical systems for the recognition of human actions," in *Computer Vision and*

Pattern Recognition, 2009. CVPR 2009. IEEE Conference on. IEEE, 2009, pp. 1932–1939.

[2] S. Sadanand and J. J. Corso, "Action bank: A high-level representation of activity in video," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2012. [Online]. Available: http://www.cse.buffalo.edu/~jcorso/pubs/jcorso_CVPR2012_actionbank.pdf

[3] X. Peng, L. Wang, Z. Cai, Y. Qiao, and Q. Peng, "Hybrid super vector with improved dense trajectories for action recognition," in *ICCV Workshop on Action Recognition with a Large Number of Classes*, 2013.

[4] H. Wang and C. Schmid, "Action recognition with improved trajectories," in *IEEE International Conference on Computer Vision*, Sydney, Australia, 2013. [Online]. Available: http://hal.inria.fr/hal-00873267

[5] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 1. IEEE, 2005, pp. 886–893.

[6] J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, M. Mao, A. Senior, P. Tucker, K. Yang, Q. V. Le *et al.*, "Large scale distributed deep networks," in *Advances in Neural Information Processing Systems*, 2012, pp. 1223–1231.

[7] R. Setiono and H. Liu, "Neural-network feature selector," *Neural Networks, IEEE Transactions on*, vol. 8, no. 3, pp. 654–662, 1997.

[8] J. M. Steppe and K. W. Bauer, "Improved feature screening in feedforward neural networks," *Neurocomputing*, vol. 13, no. 1, pp. 47–58, 1996.

[9] J. M. Steppe, K. W. Bauer, and S. K. Rogers, "Integrated feature architecture selection," *Neural Networks, IEEE Transactions on*, vol. 7, no. 4, pp. 1007–1014, 1996.

[10] R. Reed, "Pruning algorithms-a survey," *Neural Networks, IEEE Transactions on*, vol. 4, no. 5, pp. 740–747, 1993.

[11] L. M. Belue and K. W. B. Jr., "Determining input features for multilayer perceptrons," *Neurocomputing*, vol. 7, no. 2, pp. 111 – 121, 1995. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0925231294E0053T

[12] T. Cibas, F. F. Soulié, P. Gallinari, and S. Raudys, "Variable selection with neural networks," *Neurocomputing*, vol. 12, no. 2, pp. 223–248, 1996.

[13] J. M. Zurada, A. Malinowski, and S. Usui, "Perturbation method for deleting redundant inputs of perceptron networks," *Neurocomputing*, vol. 14, no. 2, pp. 177–193, 1997.

[14] A. Verikas and M. Bacauskiene, "Feature selection with neural networks," *Pattern Recognition Letters*, vol. 23, no. 11, pp. 1323 – 1335, 2002. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167865502000818

[15] J. Basak and S. Mitra, "Feature selection using radial basis function networks," *Neural computing & applications*, vol. 8, no. 4, pp. 297–302, 1999.

[16] Y. LeCun, J. S. Denker, and S. A. Solla, "Optimal brain damage," *Advances in neural information processing systems 2, NIPS 1989*, vol. 2, pp. 598–605, 1990.

[17] B. Hassibi and D. G. Stork, "Second order derivatives for network pruning: Optimal brain surgeon," in *Advances in Neural Information Processing Systems 5,[NIPS Conference]*. Morgan Kaufmann Publishers Inc., 1992, pp. 164–171.

[18] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *The Journal of Machine Learning Research*, vol. 3, pp. 1157–1182, 2003.

[19] M. Denil, B. Shakibi, L. Dinh, N. de Freitas *et al.*, "Predicting parameters in deep learning," in *Advances in Neural Information Processing Systems*, 2013, pp. 2148–2156.

[20] T. N. Sainath, B. Kingsbury, V. Sindhwani, E. Arisoy, and B. Ramabhadran, "Low-rank matrix factorization for deep neural network training with high-dimensional output targets," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE, 2013, pp. 6655–6659.

[21] I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio, "Maxout networks," *arXiv preprint arXiv:1302.4389*, 2013.

[22] I. Laptev and B. Caputo. http://www.nada.kth.se/cvap/actions/.

[23] I. J. Goodfellow, D. Warde-Farley, P. Lamblin, V. Dumoulin, M. Mirza, R. Pascanu, J. Bergstra, F. Bastien, and Y. Bengio, "Pylearn2: a machine learning research library," *arXiv preprint arXiv:1308.4214*, 2013. [Online]. Available: http://arxiv.org/abs/1308.4214

[24] D. Yu, L. Deng, F. Seide, and G. Li, "Discriminative pretraining of deep neural networks," May 30 2013, uS Patent App. 13/304,643. [Online]. Available: http://www.google.com/patents/US20130138436

[25] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[26] X. Zeng and D. Yeung, "Sensitivity analysis of multilayer perceptron to input and weight perturbations," *Neural Networks, IEEE Transactions on*, vol. 12, no. 6, pp. 1358–1366, Nov 2001.

[27] R. K. Srivastava, J. Masci, S. Kazerounian, F. Gomez, and J. Schmidhuber, "Compete to compute," in *Advances in Neural Information Processing Systems*, 2013, pp. 2310–2318.

[28] L. V. D. Maaten and G. Hinton, "Visualizing Data using t-SNE," *Journal of Machine Learning Research*, vol. 9, pp. 2579–2605, 2008.