IMPROVING OBJECT CLASSIFICATION USING POSE INFORMATION

Hugo Penedones Ronan Collobert
Francois Fleuret David Grangier

Idiap-RR-30-2012

NOVEMBER 2012

Improving Object Classification using Pose Information

Hugo Penedones[1], Ronan Collobert[1], François Fleuret[1], and David Grangier[2]

[1]Idiap Research Institute, Martigny, Switzerland
[2]NEC Laboratories, Princeton, NJ

April, 2011

Abstract

We propose a method that exploits pose information in order to improve object classification. A lot of research has focused in other strategies, such as engineering feature extractors, trying different classifiers and even using transfer learning. Here, we use neural network architectures in a multi-task setup, whose outputs predict both the class and the camera azimuth. We investigate both Multi-layer Perceptrons and Convolutional Neural Network architectures, and achieve state-of-the-art results in the challenging NORB dataset.

1 Introduction

Object classification is the problem of assigning the correct label to an image of an object. In opposition to the problem of object recognition, which deals with single instances, object classification deals with classes, or categories, of objects. That is, the same label will be assigned to a set of different objects that are considered to be semantically related. For example, we might be interested in classifying an image as a *car* or *bike*, independently of the specific instance of car or bike we are given. In this scenario, our algorithms should be immune to the *intra-class* variability, but exploit the *inter-class* variability to precisely define decision boundaries.

This has been a fundamental problem since the beginning of computer vision, and great progress has been achieved since then.

Many directions of research have been explored, and one can argue that the major milestones in the last decades were achieved using a combination of:

1. generic machine learning algorithms: this includes Support Vector Machines, Boosting, and different kinds of Neural Networks. [1, 8, 12]

2. hand-designed feature extractors, such as: SIFT, HoG, or Haar filters, to name just a few. [14, 23, 22]

3. greater quantities of data and computational power.

Despite of the great progress, computer vision algorithms are far from having the same performance as humans in the problem of object classification.

In our work, we are interested in investigating the effects of having pose information available. For that reason, we decided to work with the small-NORB dataset [13], which is a collection of 24300 images, designed to test object classification algorithms. It was created by taking pictures of 50 different objects, belonging to 5 categories. The images cover a wide range of variations across pose (azimuth, elevation) and lighting conditions. Collections like Caltech-256 [9], ImageNet [6] or LabelMe [19] have more realistic pictures, but this comes with advantages and disadvantages. Such datasets are good to test if an algorithm can classify typical web pictures into common categories such as: cars, bicycles, faces, chairs, etc. However, using this kind of data, algorithms can exploit texture, color and background scene statistics. This is perfectly fine for many useful applications, but in this research work we are interested in testing the limits of methods that have to rely mostly on shape. As the pose of an object changes, the 2D shape in the image space will also change. The NORB dataset is composed of small gray-scale images of texture-less objects and it is therefore ideal four our studies. Note that even though it is a very challenging dataset, a human would be capable of solving the NORB object classification problem with virtually no mistakes. [1]

1.1 Related work

1.1.1 Multi-task learning

In [3], the authors define multi-task learning as ”*an approach to inductive transfer learning that improves learning for one task by using the information contained in the training signals of other related tasks*”. Indeed, the final goal of a learning algorithm is to minimize the expected risk in testing conditions. However, the learning happens by looking at training data only. Within the hypothesis set there are normally many hypothesis that have zero training error, but a choice among those must be made. Therefore, there is a need for an *inductive bias*. There are many alternatives: one can chose to prefer simpler models (Occam’s razor); encode domain prior-knowledge by restricting the hypothesis set or designing features; assume i.i.d data and use cross-validation; or transfer learning from a related task. Multi-task learning follows the latest approach. In particular, the model predicts two or more properties about the same input data, and each task will work as an inductive-bias for the other(s). In a work with similarities with ours [5], authors use multi-task learning for a Natural Language problem: the model is trained jointly to predict part-of-speech tags, chunks, and other related tasks.

1.1.2 Deep-learning

Artificial Neural Networks were some of the first models being developed by A.I. researchers, with the Perceptron being studied by Rosenblatt in the late 50s [18]. How-

[1]Perhaps with the exception of one or two vehicles, that according to subjective judgement could be labeled either as ”car” or ”truck”.

ever, such a simple linear classifier revealed to be insufficient to solve the very complex tasks in vision, speech, robotics, etc. The appearance of Multi-layer Perceptrons, which include transfer layers with non-linearities, opened the doors to tackle more ambitious problems. The trend of using more complex network architectures continued, and nowadays the ”deep-learning” term is used for models with several layers of non-linearities. This is opposed to more ”shallow” models such as Perceptrons, MLPs with one single hidden layer, SVMs or Boosted classifiers (of simple weak-learners). Convolutional Neural Networks [11] and Restricted Boltzmann Machines [20] are examples of deep-architectures that have proven to work in real-world problems. They might differ in terms of approach (discriminative versus generative models) and of learning algorithms, but they have in common the fact that the cost functions to minimize are non-convex. For this reason, some form of gradient-descent algorithm is normally used.

1.1.3 Pose-indexed features

In [7], the authors use pose-labeled data to improve object detection. However, the pose is treated as a latent variable, and there is a search through the pose-space at test time (which can be done in a coarse-to-fine manner). In our case the pose is predicted directly from the input data and is treated in the same way as the class labels.

1.1.4 Pose-embeddings

In [10], authors use the pose annotations during learning by enforcing an embedding to occur. Pictures of the same object at neighbor poses are forced to have a similar representation. Time-coherence in videos has also been exploited to enforce a pose embedding and improve object recognition [15]. In [17], the problem of face detection and head-pose estimation are solved jointly, by enforcing an embedding to occur in a 9-dimensional space where a 3D ”head-manifold” lives. If the embedding of an image in this space is too far from the ”head-manifold” then is labeled as background. On the other hand, if the image embedding lies on the manifold, the sample is considered to represent a face, and the head pose can be estimated (even analytically).

1.1.5 On the NORB dataset

In the original paper [13], the best classification error obtained in the uniform-NORB was 6.6%, using a Convolutional Neural Network architecture. Other methods performed less well, in particular SVMs, k-NN and linear classifiers, with errors of 12.6%, 16.6% and 30.2%, respectively. Restricted Boltzmann machines [16] have also been tested on NORB achieving an error of 5.20%. In [21], a hierarchical network using edges information achieves a classification error of 2.87%. Another recent work [4], reports an error rate of only 2.53%, however the method also relies on hand-crafted features, such as contrast-extractors and edge detectors.

In our work, we intend to investigate another direction. Instead of adding hand-designed feature extractors or engineering more complex network architectures, we

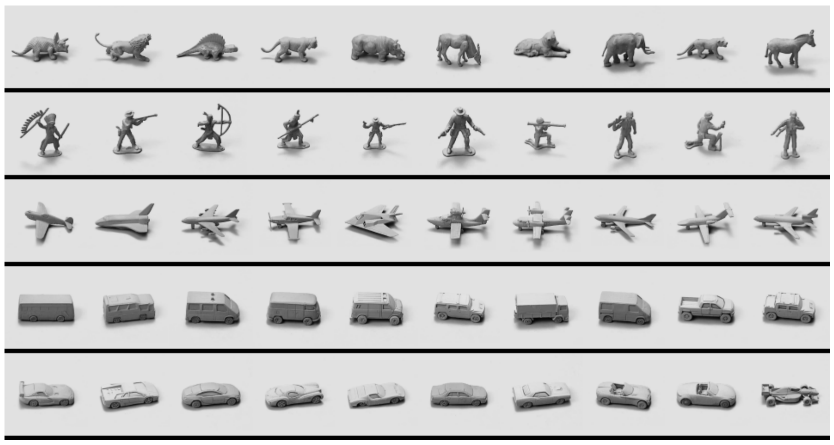

Figure 1: The NORB dataset, with objects within a category aligned at azimuth zero.

want to see the effect of exploiting a richer data labeling. In particular we are interested in using the information about pose of the objects. NORB is well suited for this, because the authors registered the camera azimuth and elevation at which each picture was taken.

2 Learning

We are interested in learning two functions:

a classifier $f : \mathbb{R}^{2\times H\times W} \rightarrow \mathbb{R}^{C}$, and

a pose-estimator $g : \mathbb{R}^{2\times H\times W} \rightarrow \mathbb{R}^{C\times 2}$, where:

H and W are height and with of each image (stereo), and C is the number of classes.

Note that the pose-estimator g, whose goal is to predict a single scalar (the azimuth angle), is outputting in a $2 \times C$-dimensional space. The reason there is one output vector per class, is that pose-alignments between object of different classes might be meaningless. How to put an airplane, an animal and a human in the same reference pose? The second choice, that is, making regression of a two-dimensional vector, instead of just of a scalar value, is a due to two technical reasons: 1) whatever the non-zero 2D vector, it always forms an angle with the x-axis (no problem of bounding a scalar in a finite interval); and 2) it's easier to encode the modulo property (0 degrees is the same as 360 degrees [2]).

[2]in fact we predict the azimuth modulo 180, because there is a big symmetry in the data that could lead

For a sample x of class y and ground truth azimuth θ, we can denote the 2-dimensional output vector corresponding to the class y by $f_y(x)$ and $g_y(x)$.

2.1 Pose Loss functions

There is more than one loss function that could be used to make the pose prediction. The first one, using the cosine, does not take into account the norm of the current output. The second one, using L_1 or L_2 norms is more strict and forces the model to approximate a vector of norm 1. Although the first one is semantically more adequate, the second one reveals to be more robust to numerical problems.

2.1.1 Cosine loss

$$l_{regress}(g(x), y, \theta) = 1 - \frac{g_y(x) \cdot r(\theta)}{\|g_y(x)\| \|r(\theta)\|}$$

where $r(\theta) = (cos(\theta), sin(\theta))$

That is, we want the angle between the 2D output of the network for the correct class to be aligned with the ground truth azimuth (which can be done by looking at the cosine of the angle between the two).

2.1.2 L_1 regression loss

Another alternative is to directly minimize, for example, the L_1 norm of the difference between the output and the ground truth:

$$l_{regress}(g(x), y, \theta) = \|r(\theta) - g_y(x)\|_1$$

In the experiments section we report only the results obtained with the L_1 norm.

2.2 Classification Loss function

Discriminative learning in a multi-class setting can be done using a standard criterion, such as the Negative Log-Likelihood:

$$l_{class}(f(x), y) = -\log\left(\frac{\exp(f_y(x))}{\sum_{j=1}^{C} \exp(f_j(x))}\right)$$

Note that when $f_y(x)$ is much greater than all of $f_j(x)$, the penalty will tend to $-\log(1) = 0$. On the other hand, when at least one $f_j(x)$ is much greater than $f_y(x)$ the penalty tends to $-\log(0) = +\infty$.

to contradicting supervision and over-fitting.

2.3 Combined loss

The final loss, as seen from the model's parameters shared among tasks, can be seen as a weighted sum of the two task-specific losses:

$$l = l_{class} + \lambda l_{regress}$$

3 Architectures

To implement the multi-task learning with neural networks, we do something very standard: the weights of the first layers are shared, whereas the last linear layer is task-specific. See Figure 2 for a visual diagram.

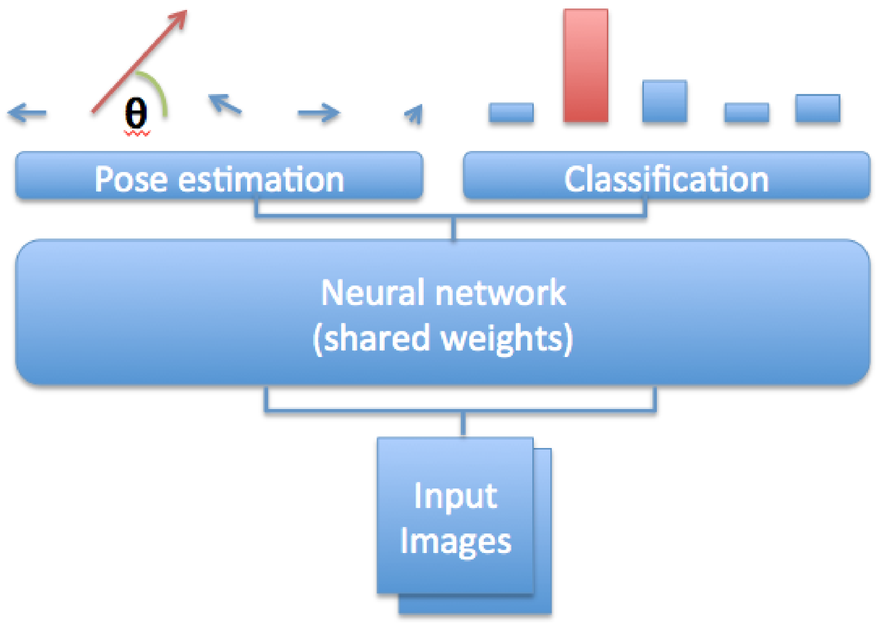

Figure 2: Network diagram and output interpretation.

3.1 Multi-layer perceptron

This is a simple and standard architecture, and the only hyper-parameter to specify is the number of hidden-units.

3.2 Convolutional Neural Network

This is deep-network architecture, whose layers are:

1. Spatial Convolution: 20 kernels of size 2x5x5 (stereo images), unitary shifts
2. Spatial Sub-sampling: 2x2 kernel, non-overlapping; followed by non-linearity
3. Spatial Convolution: 20 20x5x5 kernels, unitary shifts
4. Spatial Sub-sampling: 2x2 kernel, non-overlapping; followed by non-linearity
5. Spatial Convolution: 20 20x6x6 kernels, unitary shifts
6. Spatial Sub-sampling: 2x2 kernel, non-overlapping; followed by non-linearity
7. Fully-connected linear layer

One attractive property of these architectures is that the output can be visualized and naturally interpreted. We can render and image with a two-dimensional vector per class. The angle that the vector corresponding to the true label forms with the x-axis should be an approximation of the true camera azimuth at which the picture was taken.

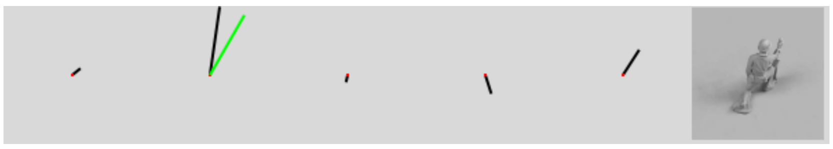

Figure 3: Illustration of network outputs for a NORB sample.

4 Training

The training of the neural networks is done using standard methods [12], however it is worthwhile to mention some technical details:

1. *Data normalization*: we compute the mean and standard deviation of the all training images, and transform the data to have zero mean and standard deviation of 1. In the case of MLP architectures, the normalization is done per pixel, whereas in the Convolutional Network the mean and variance are only scalars.
2. *Network weights initialization*: weights are initialized from a uniform distribution with zero mean and standard deviation $\sigma = \frac{1}{\sqrt{m}}$, where m is the fan-in (the number of connections feeding into the node).
3. *Learning rate correction*: the learning rate used to take a gradient descent step in back-propagation, is divided by the number of neurons in that layer.

Classifier	Input	Test error
Linear [13]	raw 2x96x96	30.2 %
1-NN [13]	PCA 95	16.6 %
SVM Gauss[13]	PCA 95	13.3%
MLP (NLL loss) + pose	**raw 2x96x96**	**$\sim$ 12.0 %**
Conv Net 80 [13]	raw 2x96x96	6.6%
Conv Net (NLL loss) + pose	**raw 2x96x96**	**$\sim$ 5.4 %**

Table 1: Comparison with the results of different classifiers in the small-NORB dataset, as reported in [13].

To train the network, we use the standard *back-propagation* algorithm with *stochastic gradient descent*[2]. The learning rate is chosen as a compromise between training time and smoothness of the training error.

5 Experiments

Training neural networks is a stochastic process. Depending on the seed of the pseudo-random number generator, we can get models with different performances, because:

1. we are optimizing a non-convex functions: the starting point matters.
2. we use stochastic gradient descent, which approximates the true gradient with only one sample
3. network weights are initialized randomly (zero mean Gaussian)
4. dataset is shuffled

In order to properly compare models we do the following: data shuffling and weights initialization is *exactly* the same for the baseline and multi-task version; we run the experiments with different seeds and compute mean and standard deviation of test errors. Unfortunately, due to computing power limitations, we could not do this careful analysis using the full training dataset and the heaviest models. Instead, we did it on a scaled-down version of the experiment, in which we use an MLP with 100 hidden units training on 10% of the dataset, for 5 different seeds. Both for baseline and multi-task version. Has we can observe from the error bars plot, the multi-task version is clearer better and the error bars (of plus or minus one standard deviation) barely overlap at any training stage.

In table 5, we report full scale results of our models and other references reported in the original NORB paper.

6 Visualizations

To better understand what is going on inside the neural network, we looked at its output before the last linear layer, using all the images of a single object category. We then

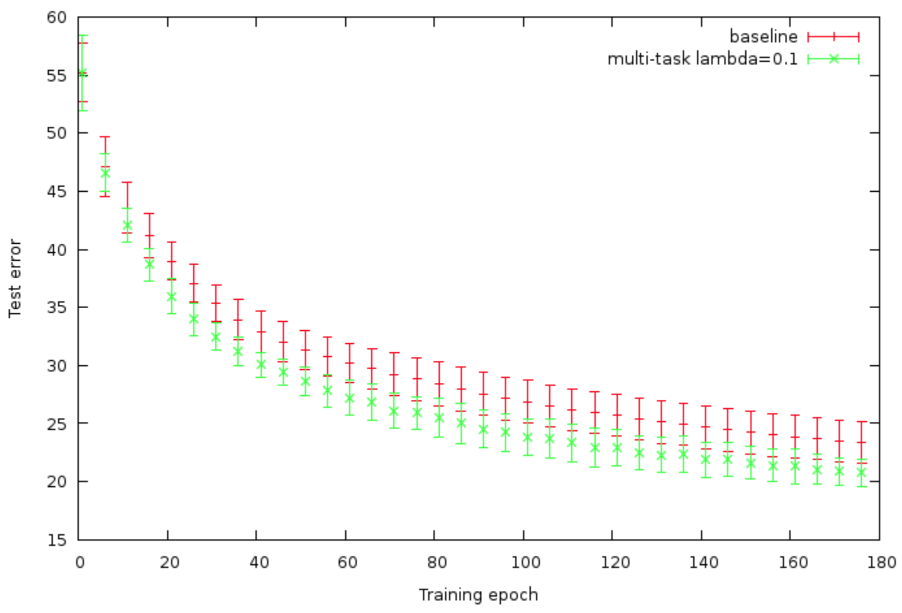

Figure 4: Comparison of two equal MLP architectures, with and without pose component of the loss.

computed the first two principal components and projected the data into it. We rendered each point as a line segment, whose orientation is given by the ground truth azimuth. You can notice in the images that the network exploiting the pose-information was forced to learn and embedding that is much more smooth as the pose varies.

7 Conclusion and future work

In this work we investigated how we could use pose-information as a regularizer to improve generalization of our models. We achieved promising results, specially in the more shallow models. In the future, we plan to explore other forms of transfer learning. For example, one could enrich the collection with synthetic data for which the ground truth pose is known. That would mean one could apply our method even to datasets in which there are no pose labels available.

References

[1] B. Boser, I. Guyon, and V. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152. ACM, 1992. 1

[2] L. Bottou. Stochastic gradient learning in neural networks. *Proceedings of Neuro-Nımes*, 91, 1991. 8

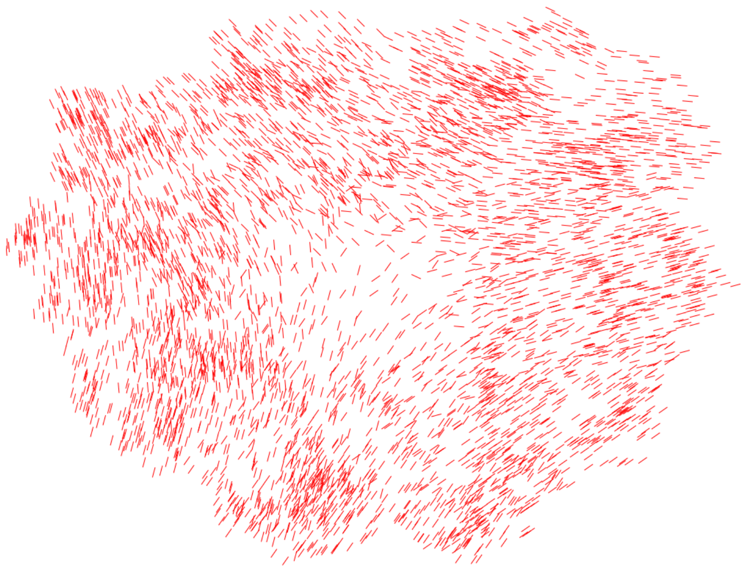

Figure 5: Multi-task network. Two-dimensional PCA projection of the last hidden layer. Line segments are drawn with their ground truth azimuth.

[3] R. Caruana. Multitask learning. *Machine Learning*, 28(1):41–75, 1997. 2

[4] D. Cireşan, U. Meier, J. Masci, L. Gambardella, and J. Schmidhuber. High-Performance Neural Networks for Visual Object Classification. *Arxiv preprint arXiv:1102.0183*, 2011. 3

[5] R. Collobert and J. Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM, 2008. 2

[6] J. Deng, W. Dong, R. Socher, L. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. 2009. 2

[7] F. Fleuret and D. Geman. Stationary features and cat detection. *Journal of Machine Learning Research (JMLR)*, 9:2549–2578, 2008. 3

[8] Y. Freund and R. Schapire. A Short Introduction to Boosting. *Journal of Japanese Society for Artificial Intelligence*, 14(5):771–780, 1999. 1

[9] G. Griffin, A. Holub, and P. Perona. Caltech-256 object category dataset. 2007. 2

[10] R. Hadsell, S. Chopra, and Y. LeCun. Dimensionality reduction by learning an invariant mapping. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 2, pages 1735–1742. IEEE, 2006. 3

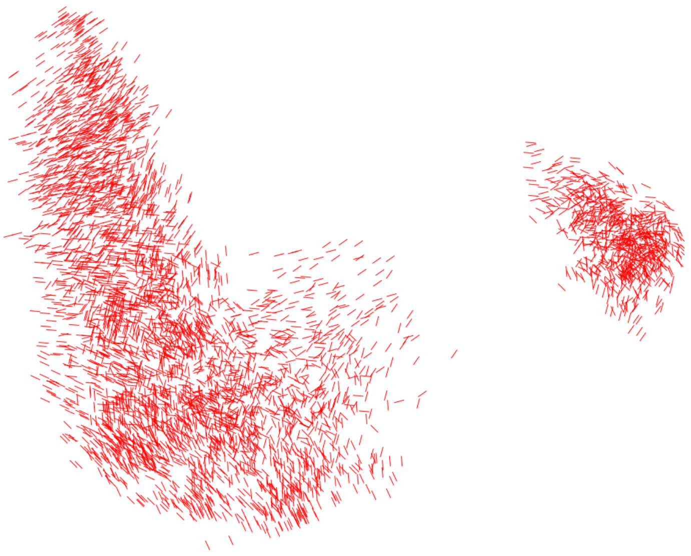

Figure 6: Without pose-information (baseline).

[11] Y. LeCun and Y. Bengio. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, pages 255–258, 1995. 3

[12] Y. LeCun, L. Bottou, G. Orr, and K. M
"uller. Efficient backprop. *Neural networks: Tricks of the trade*, pages 546–546, 1998. 1, 7

[13] Y. LeCun, F. Huang, and L. Bottou. Learning methods for generic object recognition with invariance to pose and lighting. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 2. IEEE, 2004. 2, 3, 8

[14] D. Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004. 1

[15] H. Mobahi, R. Collobert, and J. Weston. Deep learning from temporal coherence in video. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 737–744. ACM, 2009. 3

[16] V. Nair and G. Hinton. 3-d object recognition with deep belief nets. *Advances in Neural Information Processing Systems*, 22:1339–1347, 2009. 3

[17] M. Osadchy, Y. Cun, and M. Miller. Synergistic face detection and pose estimation with energy-based models. *The Journal of Machine Learning Research*, 8:1197–1215, 2007. 3

[18] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386–408, 1958. 2

[19] B. Russell, A. Torralba, K. Murphy, and W. Freeman. LabelMe: a database and web-based tool for image annotation. *International Journal of Computer Vision*, 77(1):157–173, 2008. 2

[20] R. Salakhutdinov, A. Mnih, and G. Hinton. Restricted Boltzmann machines for collaborative filtering. In *Proceedings of the 24th international conference on Machine learning*, pages 791–798. ACM, 2007. 3

[21] R. Uetz and S. Behnke. Large-scale object recognition with CUDA-accelerated hierarchical neural networks. In *Intelligent Computing and Intelligent Systems, 2009. ICIS 2009. IEEE International Conference on*, volume 1, pages 536–541. IEEE, 2009. 3

[22] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1. Citeseer, 2001. 1

[23] Q. Zhu, M. Yeh, K. Cheng, and S. Avidan. Fast human detection using a cascade of histograms of oriented gradients. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 2, pages 1491–1498. IEEE, 2006. 1