ESANN 2017 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 26-28 April 2017, i6doc.com publ., ISBN 978-287587039-1. Available from http://www.i6doc.com/en/.

Acceleration of Prototype Based Models with Cascade Computation

Cem Karaoguz[1] and Alexander Gepperth[2]

1- ENSTA ParisTech - UIIS Lab
University of Paris-Saclay, 91762, Palaiseau - France
2- University of Applied Sciences Fulda - Applied Computer Science department
Leipzigerstr. 123, 36037 Fulda - Germany

Abstract. Prototype-based generative description of data space is shown to be effective in incremental learning. However, computation of similarities of input vectors to prototypes may be demanding especially in the face of high input dimensions and high number of prototypes. The main contribution of the paper is the acceleration of the prototype-based model by a cascade computation approach. The evaluation of the presented architecture on a human detection and pose estimation problem shows that the cascade computation results in a significant reduction of computational resource requirements at the expense of minor degradations in the classification performance.

1 Introduction

Incremental learning can be loosely described as the ability of a learning agent to be able to update its internal models to cope with changing data statistics without ravaging the already learned knowledge. For an object classification system, being able to learn new object classes after an initial training with a set of classes is a typical example. Real world applications of incremental learning often have to deal with high dimensional data with large statistical variations. A resource efficient, prototype based incremental learning architecture for applied perceptual problems was proposed in [1]. However, the computational demands increase as dimensions of the input vector increases as well more prototypes are included in the model for better representational capability. In this work, we extend this architecture with a cascade computation framework similar to [2] and [3] that reduces the amount of computational operations without having a major impact on the classification performance. Particularly, we focus on reducing the amount of computational operations during the evaluation of the generative model without having a major impact on the classification performance via a cascade computation framework. This is achieved by a pruning scheme based on early identification of the prototypes that are out of the local region of the input vector. The proposed architecture is able to perform multi-pose, multi-class object recognition with a state of the art performance and a considerable reduction in computational cost. We show the feasibility of our system on human detection and pose classification tasks.

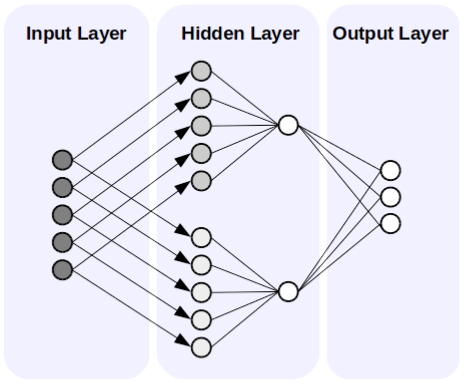

Fig. 1: Overview of the proposed architecture. See text for the explanation.

1.1 Related work

Cascade computation framework is basically a pruning scheme which is applied to numerous discriminative models. The strategy of this framework is based on coarse to fine evaluation to reject negative examples as early as possible. An early example of cascade computation principle is demonstrated in [2] with boosting models and later extended in [4] and [3]. Additionally, [5] demonstrates the application of cascade computation to SVM for rapid object detection.

2 Methods

The proposed architecture is a three layer neural network shown in Fig. 1. The hidden layer comprises prototype vectors (only two prototypes are shown in the figure) which are compared to the input vector received from the input layer using a similarity measure. The proposed cascade framework interrupts this computation early on for sufficiently dissimilar prototypes. The resulting prototype activities are fed to the all-to-all connected output layer where the prediction of class memberships is done via regression. Details of the architecture is outlined in [1]. We utilize superscripts ^{I}, ^{H} and ^{O} for entities related to input, hidden and output layers, respectively.

2.1 Network Model

As the input vector, any vectorized representation of image can be used with our model. We utilized aggregated channel features proposed in [6] due to its relevance to the related work.

The input vector is projected onto a topologically organized prototype space that is represented as weight vectors w^H_m where $w^H \in \mathbb{R}^{\mathrm{N}\times\mathrm{M}}$. The prototype layer acts similar to the self-organizing map (SOM) algorithm [7] and the weights are learned following the SOM learning rule. The projection of the input starts with computing the distance between the input vector and all prototypes:

$$\bar{z}^H(m) = ||w_m^H - z^I|| \tag{1}$$

where $z^I = f$ is the input vector, $||\cdot||$ is the Euclidean norm. The prototype with the smallest distance is called the best matching unit and indicated as m^*. Next, we calculate the (graded) activations of prototypes:

$$\tilde{z}^H = g_\kappa\left(\bar{z}^H\right) \tag{2}$$

where the activation function g_κ is Gaussian with standard deviation κ. The activation function converts the distance measures into similarity and keep them in the $[0, 1]$ interval. A transfer function is further applied to sparsify these activations:

$$z^H = \mathrm{TF}_{\theta,p}\left(\tilde{z}^H\right) \tag{3}$$

where z^H is the final prototype activation map and $\mathrm{TF}(\cdot)$ represents a monotonous non-linear transfer function, $\mathrm{TF} : [0, 1] \rightarrow [0, 1]$ which maintains the best matching unit value unchanged while non-linearly suppressing smaller values.

Following the projection, the prediction step maps the prototype activations to individual classes via linear regression:

$$z^O(m) = w_m^O \cdot z^H \tag{4}$$

where $w^O \in \mathrm{I\!R}^{\mathrm{M}\times\mathrm{C}}$ are weights of the output layer trained by stochastic gradient descent.

2.2 Prototype pruning via cascade

The distance computation step in Eqn. 1 can be decomposed as:

$$\bar{z}^H(m) = \sqrt{\sum_{n=1}^{N}(w_{m,n}^H - z_n^I)^2} \tag{5}$$

where $w_{m,n}^H$ and z_n^I are n^{th} element of prototype vector m and input feature vector, respectively. We define a cascade error term for prototype m as:

$$D_{m,k} = \sum_{n=1}^{k} d_{m,n} \tag{6}$$

where $k \leq N$ and $d_{m,n} = (w_{m,n}^H - z_n^I)^2$. The goal is to abort the distance computation at an earlier stage $k << N$ if the accumulated distance measure $D_{m,k}$ already becomes too high. Fig. 2 illustrates cascade error of one prototype computed for 12000 samples. In addition to positive and negative samples, we introduce the term pseudo-negative for all positive samples, which do not fall into the receptive field of the prototype. It can be observed that for each stage n a rejection threshold θ_n can be found that separates negative and pseudo negative samples from positive samples with respect to an optimization criterion. Hence, at stage $n = k$, if $D_{m,k}$ exceeds the threshold θ_k the distance computation is stopped and the distance value $\bar{z}^H(m)$ is also set to a very large value rendering the activation value zero.

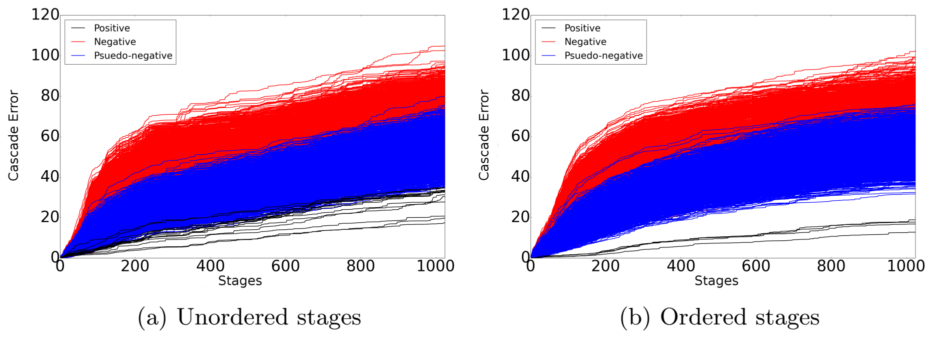

Fig. 2: Cascade error traces of one prototype, best viewed in colour.

2.3 Learning cascade parameters

Within the context of cascade, for each prototype, we have to optimize rejection thresholds $\theta_{m,n}$ such that a desired detection/false positive trade off is achieved. We employ an online learning approach that is consistent with the incremental learning principle employed in the training process of prototype and regression weights. For a given positive sample the following update rules is applied to every stage $n \in \{1...N\}$ of each prototype $m \in \{1...M\}$:

$$\theta_{m,n} \leftarrow max(\theta_{m,n}, D_{m,n}). \tag{7}$$

This update rule ensures that every positive sample for a given prototype stays below the threshold in all stages. This achieves already good results in terms of detection rate/cascade speed trade off since the separation between positive and (pseudo-)negative samples is usually well defined in our application.

Since prototypes are compact representations of input data, prototype variance of a stage describes how much the feature element at that stage varies in the dataset. Hence, ordering the error computation stages with respect to higher to lower prototype variance achieves an ordering of features from specific to general. This leads to the possibility of finding better boundary conditions as done in [3]. This is demonstrated in Fig. 2: compared to the unordered case (Fig. 2a), the ordered case (Fig. 2b) results in a broader margin to separate positive and negative/pseudo-negative samples. The incorporation of the ordering in the learning processes is done by following the ordered indices for a given prototype weight vector and feature vector during the computation of $D_{m,n}$.

3 Experiments

Experiments are conducted on human detection and pose classification task. Daimler pedestrian database [8] is used in the experiments. Approximately 12,000 positive samples and 10,000 negative samples are included in the dataset. All samples are scaled to 32 by 64 pixels. Model parameters are set to the same values presented in [1].

Without Cascade						With Cascade					
	L	R	F	B	BG		L	R	F	B	BG
L	0.63	0.23	0.02	0.01	0.005	L	0.61	0.21	0.02	0.02	0.003
R	0.24	0.63	0.06	0.02	0.005	R	0.22	0.57	0.05	0.02	0.002
F	0.04	0.05	0.84	0.02	0.005	F	0.02	0.04	0.80	0.03	0.003
B	0.04	0.05	0.07	0.94	0.005	B	0.03	0.05	0.05	0.90	0.004
BG	0.05	0.04	0.01	0.01	0.98	BG	0.12	0.13	0.08	0.03	0.99

Table 1: Classification performance. Rows: predictions, columns: ground truth. L, R, F, B and BG stands for left, right, front, back and background, respectively.

3.1 Detection and classification performance

Initially, the system is trained only with positive samples. In this case the capability of the system to distinguish background samples from humans is limited (20% detection rate at 0.1 FPR) since background samples, which bear similar features to humans, are easily confused by the model. The use of cascade mechanism on this model improves the detection results significantly (85% detection rate at 0.1 FPR) by filtering some of these false positives out already in the projection phase. This is due to the fact that cascade thresholds are trained with both positive and negative samples.

Next, the model is trained with both positive and negative samples. The break down of the performance into individual poses for the case without cascade is shown in Tab. 1, left. A closer inspection reveals that classification errors mostly occur between symmetrical classes: front vs. back and left vs. right. The overall detection rate is measured to be 76% with 0.02 FPR. Compared to the literature, our models outperforms, the best result reported in the benchmark of [9] that is obtained as 74% of detection rate on a different but comparable database. With the cascade extension, detection rate is measured to be 72% with 0.01 FPR. The breakdown of the classification performance into individual classes is given in Tab. 1, right. The performance is comparable to the model learned without the cascade framework and still at the level of state-of-the art performance. The cost of activation map approximation caused by the cascade mechanism can be observed as elevated false classifications.

3.2 Analysis of the computational requirements

In the demonstrated application a 1024 dimensional feature vector must be projected on 900 prototypes which results in 921,600 operations in total. These operations can easily be executed in parallel where a dedicated hardware is available. In our GPU implementation it takes 2.4 milliseconds to process one sample. The cascade framework greatly reduces these computational requirements: when the cascade framework is employed, it only takes 82,697 operations on average to compute an activation map per sample. This corresponds to 9% of all prototypes to be used per sample on average.

4 Conclusion

We present a cascade computing framework for prototype-based neural networks for improving the runtime performance by eliminating the projection of a given input vector on prototypes associated with irrelevant input sub-spaces beforehand. The architecture was tested on human detection and pose classification tasks. Results show that the architecture delivers state of the art performance both for detection and classification. Evaluation of the system with the cascade computation framework confirms 90% reduction in the number of operations to be computed for the evaluation of the generative model. The cascade approach also improves detection performance due to the additional discriminative constraints imposed by the cascade. Even though no prototypes for background samples are learned within the generative model, the cascade mechanism allows to make reliable detections. On the other hand, degradations in the classification performance are observed. This is because the cascade computation delivers an approximation of the prototype activation map to the regression model.

References

[1] A. Gepperth and C. Karaoguz, "A bio-inspired incremental learning architecture for applied perceptual problems," *Cognitive Computation*, pp. 1–11, 2016.

[2] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, vol. 1, pp. I–511, IEEE, 2001.

[3] L. Bourdev and J. Brandt, "Robust object detection via soft cascade," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 2, pp. 236–243 vol. 2, June 2005.

[4] C. Zhang and P. A. Viola, "Multiple-instance pruning for learning efficient cascade detectors," in *Advances in Neural Information Processing Systems*, pp. 1681–1688, 2008.

[5] B. Heisele, T. Serre, S. Prentice, and T. Poggio, "Hierarchical classification and feature reduction for fast face detection with support vector machines," *Pattern Recognition*, vol. 36, no. 9, pp. 2007–2017, 2003.

[6] P. Dollár, R. Appel, S. Belongie, and P. Perona, "Fast feature pyramids for object detection," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 36, no. 8, pp. 1532–1545, 2014.

[7] T. Kohonen, "Self-organized formation of topologically correct feature maps," *Biol. Cybernet.*, vol. 43, pp. 59–69, 1982.

[8] S. Munder and D. M. Gavrila, "An experimental study on pedestrian classification," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 28, no. 11, pp. 1863–1868, 2006.

[9] M. Enzweiler and D. Gavrila, "Integrated pedestrian classification and orientation estimation," in *CVPR*, 2010.