2012 11th International Conference on Machine Learning and Applications

Convolutional Neural Support Vector Machines: Hybrid Visual Pattern Classifiers for Multi-robot Systems

Jawad Nagi[*], Gianni A. Di Caro[*], Alessandro Giusti[*], Farrukh Nagi[#], Luca M. Gambardella[*]
[*]*Dalle Molle Institute for Artificial Intelligence (IDSIA), Lugano, Switzerland*
[#]*University Tenaga Nasional, Putrajaya, Malaysia*
{jawad,gianni,alessandrog,luca}@idsia.ch, farrukh@uniten.edu.my

***Abstract*—We introduce Convolutional Neural Support Vector Machines (CNSVMs), a combination of two heterogeneous supervised classification techniques, Convolutional Neural Networks (CNNs) and Support Vector Machines (SVMs). CNSVMs are trained using a *Stochastic Gradient Descent* approach, that provides the computational capability of online incremental learning and is robust for typical learning scenarios in which training samples arrive in mini-batches. This is the case for visual learning and recognition in multi-robot systems, where each robot acquires a different image of the same sample. The experimental results indicate that the CNSVM can be successfully applied to visual learning and recognition of hand gestures as well as to measure learning progress.**

I. Introduction

When faced with visual recognition tasks, multi-robot systems can take advantage from multiple mobile sensors distributed in different locations in an environment. However, especially in swarm robotic systems, each robot typically has limited computational capabilities and low-quality cameras: this motivates the development of powerful but computationally inexpensive classifiers. We present a new classifier with such properties, building on our previous works in the domain of visual pattern recognition for mobile swarm robotic systems [1], [2], [3].

The classifier is derived from the combination of well-known classifiers for visual recognition: Support Vector Machines (SVMs) [4] and Convolutional Neural Networks (CNNs) [5], a variant of Multi-Layer Perceptrons (MLPs). In recent research, such an approach has been adopted frequently, with MLPs and CNNs used for supervised learning of features directly from image intensities, and SVMs [6] for the actual classification of such features. Examples of applications include handwriting recognition [7], [8] [9], [10], network intrusion detection [11], face detection and recognition [12], texture recognition [13], pedestrian detection [14] and general object recognition [15]. CNNs alone also demonstrated exceptional results for tasks such as document recognition [5], [16] and general object classification [16].

We propose *Convolutional Neural Support Vector Machines* (CNSVMs), a heterogeneous combination of supervised CNNs and SVMs for visual pattern learning and recognition tasks. Instead of training each component separately, our system features a single training approach and allows for computationally light incremental learning based on small chunks of training samples. This is particularly suitable in the context of swarm/multi-robot systems, where for each training sample, multiple images, from different points of view are gathered in parallel by the different robots.

The rest of this paper is organized as follows. Section II discusses the related work in different domains. Section III provides the background details of CNNs, SVMs and our proposed CNSVM. Section IV illustrates our human-swarm interaction use case, and Section V presents the results and findings of our system using emulated tests. Section VI presents concluding remarks.

II. Related Work

Combining supervised MLPs and SVMs was initially proposed in [17], where the SVM method was used to optimize the MLP weights, output weight vector, and interconnection matrix. This requires to solve two correlated quadratic programming problems, resulting in a computationally complex system. A hybrid MLP-SVM was later proposed in [7] for handwritten digit recognition. In this case, the two models are trained separately with disjoint sets of samples. Similarly, in [12], [14] and [10], building on the approach of [7] and based on interdependent supervised classifiers, a hybrid system was proposed comprising of two separate models.

In [18] and [11], supervised classifiers such as CNNs, MLPs, SVMs and k-Nearest Neighbours are combined in a "mixture of experts" approach: the output of parallel classifiers is used to produce the final result. In [8], the use of MLPs with SVMs is proposed for improving handwriting recognition. However the approach only employs the SVM in cases where the handwritten characters cannot be recognized with a high confidence by the MLP.

Our approach differs from these previous ones as we build a single model instead of using disjoint classifiers trained separately. In this respect, more relevant previous work include [13], where a hybrid CNN-SVM approach is presented: the CNN is trained using the back-propagation algorithm and the SVM is trained using a non-linear regression approach. In [15] the authors have verified the claim in [7] that a kernel can be learned by cascading a feature extractor (i.e., CNN) with a fixed kernel function (i.e., SVM). More recently, in [19] the authors proposed a hybrid MLP-SVM

978-0-7695-4913-2/12 $26.00 © 2012 IEEE
DOI 10.1109/ICMLA.2012.14

architecture employing MLPs in a cascaded configuration for feature learning, where the output from the MLPs is implicitly fused using a SVM with a voting strategy.

Overall, only a few works have considered combining CNNs and SVMs both acting as supervised learning approaches, that is: (i) sharing a common (single) training algorithm, and (ii) providing online incremental learning capabilities. Therefore, inspired by the work presented in [20], [21] and [22] for combining MLPs and SVMs, we develop an online incremental visual pattern recognition system with such characteristics that makes it particularly suitable for application in swarm robotic systems.

III. Supervised Learning Algorithms

A. *Convolutional Neural Networks*

CNNs are hierarchical architectures of MLPs where the successive alternating layers are designed to learn progressively higher-level features, and the last layer produces the classification results [5]. The alternating layers in CNNs provide two basic operations, convolution and sub-sampling (e.g., pooling operations), reminiscent of *simple* and *complex cells* in the mammalian visual cortex [23]. In the following we focus on CNNs for *visual recognition tasks*.

- *Convolutional Layer*: This layer performs a 2D filtering between input images $\mathbf{x}$ and a bank of filters $\mathbf{w}$, producing another set of images h. A connection table CT, represents the input-output relationships, where filter responses from the inputs connected to the same output image are linearly combined. The convolutional layer performs the mapping, $h_j = \sum_{i,k \in CT_{i,k,j}} (\mathbf{x}_i * \mathbf{w}_k)$. A non-linear activation function, a *scaled hyperbolic tangent*, is applied to h, similar to the case of MLPs.
- *Sub-sampling Layer*: In this layer we implement the “Max-Pooling operation”, which introduces small invariance to translation and distortion, that leads to faster convergence and improved generalization [16].
- *Fully-connected Layer*: The input for the fully connected layer (i.e., the second last layer) is a set of feature maps from the lower layer. This layer combines the outputs of the previous layer into a 1-dimensional feature vector, which is then passed through a *scaled hyperbolic tangent* activation function.
- *Output Layer*: The output layer (see Figure 1 in [3]) has one output neuron unit per class label, and acts as a linear classifier operating on the 1-dimensional feature vectors computed from the fully-connected layer.

Typical training algorithms of CNNs use gradient descent approaches for minimizing the training error, since all the layers in CNNs are differentiable. In our previous work [3], we employed the back-propagation algorithm [5] to train a MPCNN, where the gradient was used to minimize the *sum of the squared errors*. In general, however, due to the highly non-linear nature of the error surface, *Stochastic Gradient Descent* (SGD) approaches are more appropriate since they usually avoid being stuck in poor local minima [22].

B. *Support Vector Machines*

SVMs aim at finding a hyperplane in the feature space, which separates the data classes while maximizing the margin in such space. Given a supervised *soft-margin* classification problem and a training set of N data points $\{y_i, \mathbf{x}_i\}_{i=1}^{m}$, where $\mathbf{x}_i \in \mathbb{R}^n$ is the ith input pattern and $y_i \in \mathbb{R}$ is the ith output pattern, the SVM method aims at constructing a classifier of the form [24]:

$$y(\mathbf{x}) = \text{sign}\left(\sum_{i=1}^{m} \alpha_i y_i K(\mathbf{x_i}, \mathbf{x}) + b\right) \qquad (1)$$

where α_i are positive real constants, b is a real constant and $K(.,.)$ represents a non-linear *Kernel function* that maps the input space into a higher dimensional space. To maximize the hyperplane margin for a *soft-margin* SVM, the following cost function (i.e., primal SVM problem) needs to be minimized:

$$\text{minimize}_{\mathbf{w},\xi,b} \left\{ \frac{\|\mathbf{w}\|^2}{2} + C\sum_{i=1}^{m} \xi_i \right\} \qquad (2)$$

subject to the two constraints $y_i(\mathbf{w} \cdot \mathbf{x}_i - b) \geq 1 - \xi_i$ and $\xi_i \geq 0$, for $i = \{1, 2, ..., m\}$. ξ_i is a *slack variable* that measures the degree of misclassification of the data $\mathbf{x}_i$ and the parameter C controls the trade off between errors on training data and margin maximization ($C = \infty$ leads to a hard-margin SVM).

Typically SVMs are trained by introducing Lagrange multipliers and solving a dual quadratic programming problem [24], which however is not applicable to CNNs. Therefore, to train a CNN and SVM using a single algorithm with the capability of online incremental updates, we direct our attention towards “primal gradient-based methods”, namely *Stochastic Gradient Descent* (SGD) based approaches, as discussed in Section III-C.

C. *Convolutional Neural Support Vector Machines*

CNNs are efficient at learning *invariant features* from images, but do not always produce optimal classification results. Conversely, SVMs with their fixed kernel function cannot learn complicated invariances, but produce good decision surfaces by maximizing margins using *soft-margin* approaches [15]. In this context, we focus our attention towards investigating a hybrid system, in which the CNN is trained to learn features that are relatively invariant to irrelevant variations of the input. In this way, an SVM with a non-linear kernel can provide an optimal solution for separating the classes in the learned feature space.

Following this approach, we propose *Convolutional Neural Support Vector Machines* (CNSVMs), a heterogeneous combination of the CNN and SVM, in which the output layer of the CNN is replaced by an SVM (i.e., the fully-connected layer of the CNN acts as an input to the SVM). Our CNSVM is trained using a stochastic gradient descent based approach, with the capability of online incremental updates, as discussed later in this section.

Observing the similarities in among CNNs, MLPs and SVMs, the decision function $f(x)$ in MLPs (including CNNs) and SVMs can be written in its general form as $f(\mathbf{x}) = (\mathbf{w} \cdot \phi(\mathbf{x}) + b)$, where $\mathbf{w}$ represents the vector of weights, b is a bias, and all parameters are included in ϕ. For ϕ-machines and SVMs, ϕ is an arbitrary function [25]. The objective of the proposed CNSVM is to find a hyperplane $f(\mathbf{x})$ in the Reproducing Kernel Hilbert Space (RKHS), which separates the data classes while maximizing the margin between the hyperplane and classes. Formally, given a training set $S = \{(\mathbf{x}_i, y_i)\}_{i=1}^{m}$, where $\mathbf{x}_i \in \mathbb{R}^n$ and $y_i \in \{+1, -1\}$ for a binary classification problem (that can be extended to a multi-class problems), the CNSVM minimizes the following cost function:

$$\text{minimize}_{\mathbf{w}} \left\{ \frac{\lambda}{2} \|\mathbf{w}\|^2 + \frac{1}{m} \sum_{i=1}^{m} \ell(\mathbf{x}_i, y_i; \mathbf{w}) \right\} \tag{3}$$

where $\ell(\mathbf{x}_i, y_i; \mathbf{w}) = \max\{0, 1 - y_i f(\mathbf{x}_i)\}$. In Eq. (3), $\lambda \geq 0$ is a regularization parameter (introduced to avoid over-fitting) that scales $\|\mathbf{w}\|^2$. The term $\max\{0, 1 - y_i f(\mathbf{x}_i)\}$ represents the *hinge-loss*, and $f(\mathbf{x}_i)$ is a RKHS with associated kernel $K(.,.)$. The term $\frac{\lambda}{2}\|\mathbf{w}\|^2$ is used to maximize the margin using regularization, whereas the term $\frac{1}{m}\sum_{i=1}^{m} \ell(\mathbf{x}_\mathbf{i}, y_i; \mathbf{w})$ minimizes the training error. The misclassification parameter C, that scales the empirical loss term, is related to the regularization parameter: $\lambda = \frac{1}{mC}$.

Since SVMs are trained by minimizing the primal problem in Eq. (2) subject to constraints, the optimal solution $\mathbf{w}$ (or weight vector) can be represented as a linear combination of the training samples in a high-dimensional space, such that $\mathbf{w} = \frac{1}{\lambda}\sum_{i=1}^{N_{SV}} y_i \alpha_i \phi(\mathbf{x}_i)$, where ϕ is the high-dimensional mapping implicitly defined by $K(.,.)$ and only a subset of samples satisfying the condition, $\alpha_i \neq 0$ are the *Support Vectors* (SVs) in $\mathbf{w}$, represented by N_{SV}. This indicates that the output weight vector $\mathbf{w}$ is a sparse combination of the learned features $\phi(\mathbf{x}_i)$.

Typically, the training of CNNs and SVMs is achieved by minimizing a given criterion $R(f(\mathbf{x}_i), y_i)$ over the training set, such that the term $\frac{1}{m}\sum_{i=1}^{m} R(f(\mathbf{x}_i), y_i)$ is minimized. It can be observed that the CNSVM cost function in of Eq. (3) can be minimized using *Stochastic Gradient Descent* (SGD) approaches [22] which are appropriate for online incremental learning. In this context, we propose a *Stochastic Sub-Gradient Projection* (SSGP) approach (see Algorithm 1) for the online incremental training of our CNSVM. The SSGP is well suited towards the scenario of our problem for vision-based learning tasks associated to swarm robotic systems, since it uses mini-batches (or small chunks) of training samples from the swarm (multiple samples of the same image as gathered in parallel by multiple robots) to incrementally update the CNSVM.

Algorithm 1: Stochastic Sub-Gradient Projection

```
Initialize: $\mathbf{w}$ such that, $\|\mathbf{w}\| \leq (1/\sqrt{\lambda})$
for $t = \{1, 2, ...T\}$ do
    Swarm acquires $m$ samples: $\{A_t(i)|\ i = 1, 2, ..., m\}$
    $A_t^+ = \{(\mathbf{x}, y) \in A_t : 1 - y(\mathbf{w}_t \cdot \mathbf{x}) < 1\}$ and $\mathbf{w}_{t+1}$
    $\eta_t = \frac{1}{\lambda t}$ $\mathbf{w}_{t+\frac{1}{2}} = (1 - \eta_t \lambda)\mathbf{w}_t + \frac{\eta_t}{m}\sum_{(\mathbf{x},y)\in A_t^+} y\mathbf{x}$
    $\mathbf{w}_{t+1} = \text{minimize}\left\{1, \frac{1/\sqrt{\lambda}}{\left\|\mathbf{w}_{t+\frac{1}{2}}\right\|}\right\}\mathbf{w}_{t+\frac{1}{2}}$
end
Output: $\mathbf{w}_{t+1}$
```

Algorithm 1 requires two user-defined parameters: (i) parameter m as given in Eq. (3), which defines the number of training samples to use in each mini-batch of training data for computing the sub-gradients, and (ii) T for $t = \{1, 2, ..., T\}$, which represents the number of iterations to perform. In our SSGP implementation, initially the weight vector $\mathbf{w}$ is set to any vector whose norm is almost $\|\mathbf{w}\| \leq \frac{1}{\sqrt{\lambda}}$. At each iteration t of the algorithm, a set of $\{A_t(i)|i = 1, 2, ..., m\}$ samples is acquired and processed by the swarm, with Eq. (3) that takes the form:

$$f(\mathbf{w}; A_t) = \frac{\lambda}{2}\|\mathbf{w}\|^2 + \frac{1}{m}\sum_{(\mathbf{x},y)\in A_t}^{m} \ell(\mathbf{x}_i, y_i; \mathbf{w}) \tag{4}$$

The learning rate then initialized and decayed by a factor $\eta = \frac{1}{\lambda t}$, and A_t^+ is defined as the set of samples in A_t for which $\mathbf{w}$ suffers a non-zero loss. Next, a two step update is performed: (i) $\mathbf{w}_t$ is scaled by $(1 - \eta_t\lambda)$ for all samples $(\mathbf{x}, y) \in A_t^+$, and (ii) the vector $\frac{y\eta_t}{m}\mathbf{x}$ is added to $\mathbf{w}$, where the resulting weight vector is denoted by $\mathbf{w}_{t+\frac{1}{2}} = \mathbf{w}_t - \eta_t \nabla_t$. Therefore, the sub-gradient projection is expressed by:

$$\nabla_t = \lambda \mathbf{w}_t - \frac{1}{|A_t|}\sum_{(\mathbf{x},y)\in A_t^+} y\mathbf{x} \tag{5}$$

where $\mathbf{w}_{t+\frac{1}{2}}$ is updated by the sub-gradient of the objective function evaluated with A_t. As the definition of the hinge-loss implies that ∇_t is a sub-gradient of $f(\mathbf{w}; A_t)$, in this case, $\mathbf{w}_{t+1}$ is set to be the projection on $\mathbf{w}_{t+\frac{1}{2}}$ (i.e., onto a ball of radius $\frac{1}{\sqrt{\lambda}}$) for the set, $B = \{\mathbf{w} : \|\mathbf{w}\| \leq (1/\sqrt{\lambda})\}$, which produces a sharp decrease in the learning rate. Next, $\mathbf{w}_{t+1}$ is obtained by minimizing $\mathbf{w}_{t+\frac{1}{2}}$ (as given in the second last step of Algorithm 1). The optimal solution of the CNSVM lies in the set B and the output of the SSGP is the updated weight vector $\mathbf{w}_{t+1}$. The regularization parameter (or equivalently, the loss factor C in SVMs) needs to be tuned, where $\frac{1}{\lambda}$ accounts for the difficulty of the problem.

The presented CNSVM can be extended for implementation with non-linear kernels such as the RBF (*Gaussian*) kernel. In this case, the indices of the SVs (i.e. the α_i in $\mathbf{w}$ satisfying, $\alpha_i \neq 0$) can be used to represent $\mathbf{w}$ implicitly, using J and the corresponding α_i values, since the dot-product, $K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_j)$ can be expressed by, $\mathbf{w} \cdot \phi(\mathbf{x}_i) = \frac{1}{\lambda} \sum_{i \in J} y_i \alpha_i K(\mathbf{x}_i, \mathbf{x}_j)$.

Besides returning classifications, SVMs also compute the posterior probabilities for multi-class problems. Therefore, for a k classes problem, our goal is to estimate the probability for each class using the $\mathbf{x}$ data such that $\mathbf{p}_i = \mathbf{p}(y = i|\mathbf{x})$ for $i = \{1, 2, ..., k\}$. Given that r_{ij} is an estimate for the probability of the output of a pairwise classifiers between class i and class j, $\mathbf{p}_i$ the probability of the ith class can be obtained by minimizing the following optimization problem:

$$\text{minimize}_{\mathbf{p}} \left\{ \frac{1}{2} \sum_{i=1}^{k} \sum_{j;j=1} \left(r_{ji}\mathbf{p}_i - r_{ij}\mathbf{p}_j \right)^2 \right\} \qquad (6)$$

which is subject to the constraints $\sum_{i=1}^{k} \mathbf{p}_i = 1$ and $\mathbf{p}_i \geq 0$. Therefore, r_{ij} can be defined as: $r_{ij} \approx \mathbf{p}(y = \{i, j\}, \mathbf{x})$, where, $r_{ij} + rji = 1$.

IV. Application: Human-Swarm Interaction

We consider a *human-swarm interaction* problem (as discussed in our previous works [1], [2]), where a human presents hand gestures to a swarm of mobile robots using colored gloves, and the swarm of robots, after detecting the glove, position themselves uniformly around the glove. Each robot in the swarm captures an image of the given hand gesture based on its point of view from the gesture and then performs color-based segmentation to retrieve the shape of the hand silhouette (contour) in the form of binary images (as illustrated in our previous work in [3]). The segmented binary images are resized to 28×28 pixels and padded with 4 black pixels on each side, resulting in a final image size of 32×32 pixels [3], which is the input for the CNSVM.

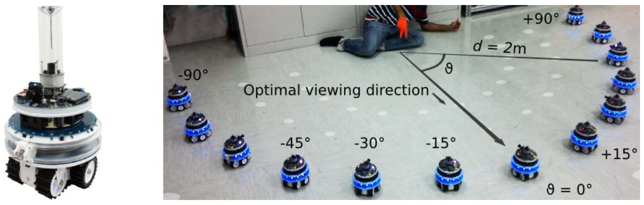

Figure 1: Setup for hand gesture dataset acquisition.

For robot r and given gesture i, the posterior probability estimates $\mathbf{p}_i^r$ from our CNSVM represent the *opinion* of r about the predicted hand gesture, and the element in the $\mathbf{p}_i^r$ with the largest value is the predicted outcome (i.e., the recognized gesture class). In order to obtain a swarm-level prediction of the hand gesture which accounts for all the different existing opinions, a *distributed consensus* is implemented. We adopt a quite straightforward approach to implement this: For each given gesture i, at each robot we fuse the $\mathbf{p}_i^r$ obtained from all robots $r = \{1, 2, \ldots, N_{robots}\}$ from their different viewpoints using a simple averaging approach $\delta_i = \sum_{r=1}^{N_{robots}} \mathbf{p}_i^r$, where δ_i represents the mean of the posterior probability vectors (i.e., the outcome of the distributed consensus), which is the overall decision of the swarm for predicting the given gesture (a more sophisticated consensus approach was described in [1]).

After the swarm performs a prediction and obtains δ_i for a given gesture, we perform *k-means clustering* by using the posterior probability estimates $\mathbf{p}_i^r$ that were used to obtain δ_i. Since we cluster a classifier's posterior probabilities, the *Kullback-Leibler* (KL) *divergence* is employed as our "distance measure" in k-means, as it is more appropriate to measure relations between probabilistic distributions using measures of entropy. Using the k-means with the KL-divergence measure for the set of probability estimates in δ_i, a cluster center cc is determined. Next, the *Euclidean distance* between cc and each $\mathbf{p}_i^r$ in δ_i, for $r = \{1, 2, \ldots, N_{robots}\}$ is computed, which results in a vector of distances d, with a size of $1 \times N_{robots}$ elements. The mean of d results in a quantity defined by $\mu = \frac{d}{N_{robots}}$, which we employ in Section V for measuring the overall learning progress of the swarm (i.e., the level of difficulty for the swarm in classifying given gestures).

V. Experimental Results

We employed a swarm of 13 foot-bots (see Figure 1) to gather $74,000$ images from 65 different view points with known *ground truth*. We considered $k = 6$ classes representing *finger-count* gestures, as illustrated in Figure 2. The results presented in this section are from emulation experiments based on this hand gesture dataset.

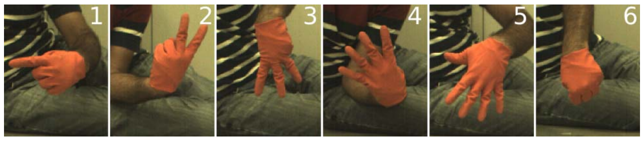

Figure 2: Finger counts from 0 through 5 represent the $k = 6$ hand gesture classes.

In the experiments the robots in the swarm are initially deployed in random positions sampled from the positions from where the real images in the dataset were acquired. Robots exchange messages through a low-bandwidth (100 bytes/sec) infra-red (IR) based line-of-sight wireless device. Initially, each robot trains its individual CNSVM using G_{initial} samples from each one of the $k = 6$ gesture classes, such that $L = G_{\text{initial}} \times k$ is the total number of samples used for the *initial training phase* of the swarm. This phase is used to start-up the learning process. We consider $G_{\text{initial}} = 5$ samples in the following (i.e., the robots are given a very limited initial training). After the initial training phase, the swarm and the human instructor go through a series of *interaction rounds*. At the beginning of each round, the robots move (e.g., to simulate the fact that the swarm performs an assigned task): each robot is randomly assigned a new position among the 65 different viewpoints in the dataset. Within each interaction round, robots do not move, and $G_{\text{interact}}\}$ random gestures are shown to the swarm.

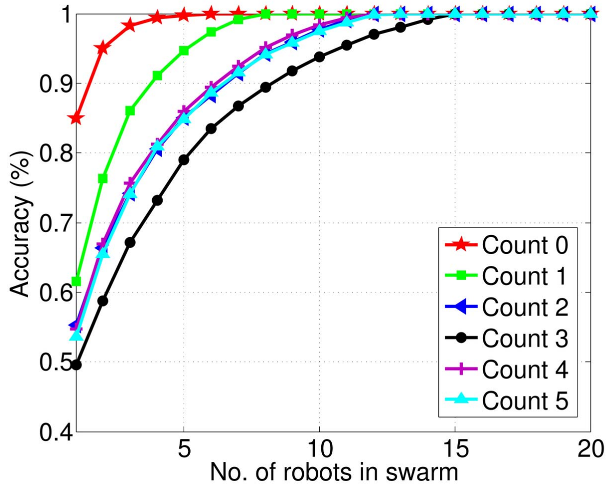

Figure 3: Accuracy of the CNSVM for classifying the $k = 6$ hand gesture classes.

For each of such gestures $i = \{1 \cdots G_{\text{interact}}\}$, the swarm performs a distributed consensus process whose outcome we denote as $\delta_i = 1$ if correct, or $\delta_i = 0$ otherwise. For a given interaction round, this results in a vector of G_{interact} δ_i values, whose average is considered as the swarm accuracy for the interaction round. Then the robots move, and the next interaction round begins.

Figure 3 presents the classification accuracy of the CNSVM with respect to the varying number of robots in the swarm. This clearly indicates that simple shapes, such as finger counts 0 and 1, are easier to classify, and require a smaller swarm, whereas finger counts 3, 4 and 5 result in lower accuracies. We also investigate the effects of different architectures for the convolutional part of our CNSVM, which result in different amounts of features for classification. Figure 4 indicates the classification accuracy of the swarm using different numbers of features, with respect to the varying number of robots: as expected, we observe that the larger the number of features, the better is the overall recognition rate of the swarm. However, in our case the recognition rate does not significantly increase when using more than 200 features.

In Section IV we defined a quantity μ representing the difficulty of the swarm in classifying a given gesture. In Figure 5 we illustrate how μ varies during the learning process, using a swarm size of $N_{robots} = 13$. The lower the value of μ the lower will be the difficulty (i.e., the higher the confidence) for the swarm to predict the given gesture, and vice versa. One may observe that in the initial phases of the learning process, μ decreases as the swarm improves its confidence in predicting gestures. Later on, the value stabilizes and it starts to increase again; we are currently investigating the reasons for this behavior, in an effort to design swarm systems which can self-assess their classification ability and act accordingly.

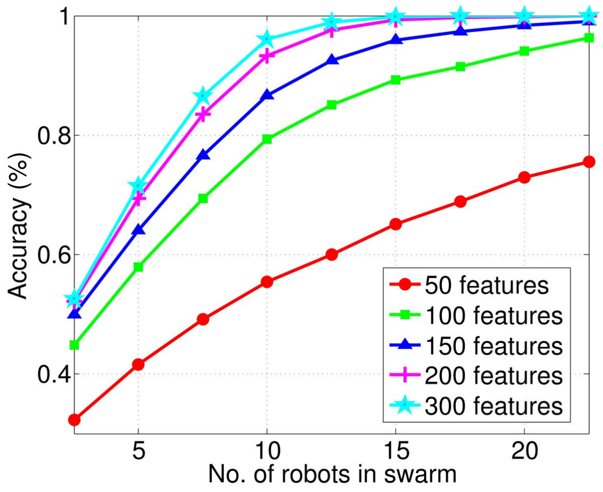

Figure 4: Impact of the number of features and number of robots on the classification accuracy of the CNSVM.

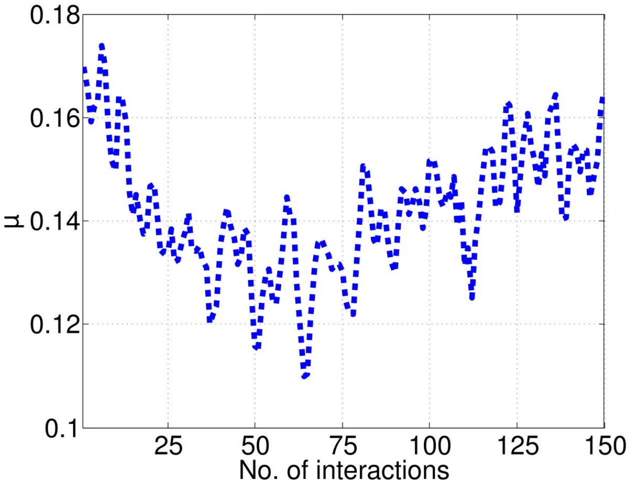

Figure 5: Effect of μ, using different sequences (orders) of gesture classes for training a swarm size of $N_{robots} = 13$.

VI. Conclusion

We introduced Convolutional Neural Support Vector Machines (CNSVMs). These are learning and visual pattern recognition systems that can be trained incrementally with mini-batches of data, thereby providing efficient feature learning ability which is computationally inexpensive. This aspect, together with the ability of robot swarms to gather in parallel batches of data, makes the use of CNSVMs particularly suitable in the context of visual learning and recognition tasks with robot swarms. We have shown a number of emulation experiments using a large dataset of real images, investigating the properties of the approach.

Future work aims to develop techniques for estimating the learning progress of the swarm, which may be used by swarms to autonomously decide when to ask for more training examples, or when to consider the training sufficient.

Acknowledgments

This research was supported by the Swiss National Science Foundation (SNSF) through the National Centre of Competence in Research (NCCR) Robotics. The authors participate to Project 4: *Distributed Robotics*, sub-project 4.5, *Symbiotic human-swarm cooperation*.

References

[1] A. Giusti, J. Nagi, L. Gambardella, and G. A. Di Caro, "Cooperative sensing and recognition by a swarm of mobile robots," in *Proc. of the 25th IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2012, pp. 551–558.

[2] J. Nagi, H. Ngo, A. Giusti, L. M. Gambardella, J. Schmidhuber, and G. A. Di Caro, "Incremental learning using partial feedback for gesture-based human-swarm interaction," in *Proc. of the 21st IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, 2012, pp. 898–905.

[3] J. Nagi, F. Ducatelle, G. A. Di Caro, D. Ciresan, U. Meier, A. Giusti, F. Nagi, *et al.*, "Max-pooling convolutional neural networks for vision-based hand gesture recognition," in *Proc. of the 2nd IEEE International Conference on Signal and Image Processing Applications (ICSIPA)*, 2011, pp. 342–347.

[4] T.-N. Do and J.-D. Fekete, "Large scale classification with support vector machine algorithms," in *Proc. of the 6th ICMLA*, 2007, pp. 7–12.

[5] Y. LeCun, L. Bottou, Y. Bengio, *et al.*, "Gradient-based learning applied to document recognition," *Proc. of the IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[6] K. Boukharouba, L. Bako, *et al.*, "Incremental and decremental multi-category classification by support vector machines," in *Proc. of 8th ICMLA*, 2009, pp. 294–300.

[7] A. Bellili *et al.*, "An hybrid mlp-svm handwritten digit recognizer," in *Proc. of the Intl. Conf. on Document Analysis and Recognition (ICDAR)*, 2001, pp. 28–32.

[8] W. W. Azevedo and C. Zanchettin, "A MLP-SVM hybrid model for cursive handwriting recognition," in *Proc. of the IJCNN*, 2011, pp. 843–850.

[9] F. Lauer, C. Y. Suen, and G. Bloch, "A trainable feature extractor for handwritten digit recognition," *Pattern Recognition*, vol. 40, no. 6, pp. 1816–1824, 2007.

[10] X.-X. Niu and C. Y. Ching, "A novel hybrid cnn-svm classifier for recognizing handwritten digits," *Pattern Recognition*, vol. 45, no. 4, pp. 1318–1325, Apr. 2012.

[11] A. Borji, "Combining heterogeneous classifiers for network intrusion detection," in *Proc. of the 12th Asian Computing Science Conf. (ASIAN)*, 2007, pp. 254–260.

[12] K. Mori, M. Matsugu, and T. Suzuki, "Face recognition using svm fed with intermediate output of cnn for face detection," in *Proc. of the IAPR Conf. on Machine Vision Applications (MVA)*, 2005, pp. 410–413.

[13] H. Long and W. K. Leow, "A hybrid model for invariant and perceptual texture mapping," in *Proc. of the Intl. Conf. on Pattern Recog. (ICPR)*, 2002, pp. 135–138.

[14] M. Szarvas *et al.*, "Pedestrian detection with convolutional neural networks," in *Proc. of the IEEE Symp. on Intell. Vehicles (IV)*, 2005, pp. 224–229.

[15] F.-J. Huang *et al.*, "Large-scale learning with svm and convolutional nets for generic object categorization," in *Proc. of the IEEE Intl. Conf. on CVPR*, 2006.

[16] D. Cireşan, U. Meier, J. Masci, L. M. Gambardella, and J. Schmidhuber, "Flexible, high performance convolutional neural networks for image classification," in *Proc. of 22nd IJCAI*, 2011, pp. 1237–1242.

[17] J. A. K. Suykens and J. Vandewalle, "Training multilayer perceptron classifiers based on a modified support vector method," *IEEE Transactions on Neural Networks*, vol. 10, no. 4, pp. 907–911, Jul. 1999.

[18] S. Abdelazeem, "A greedy approach for building classification cascades," in *Proc. of the 7th ICMLA*, 2008, pp. 115–120.

[19] P. Ghanty, S. Paul, and N. R. Pal, "NEUROSVM: An architecture to reduce the effect of the choice of kernel on the performance of svm," *J. of Machine Learning Research*, vol. 10, pp. 591–622, Jun. 2009.

[20] L. Zhang, W. Zhou, and L. Jiao, "Hidden space support vector machines," *IEEE Transactions on Neural Networks*, vol. 15, no. 6, pp. 1424–1434, Nov. 2004.

[21] R. Collobert and S. Bengio, "Links between perceptrons, mlps and svms," in *Proc. of the 21st ICML*, 2004.

[22] R. Wijnhoven and P. de With, "Fast training of object detection using stochastic gradient descent," in *Proc. of the ICPR*, Aug. 2010, pp. 424–427.

[23] D. H. Wiesel *et al.*, "Receptive fields, binocular interaction and functional architecture in the cat's visual cortex," *J. of Physio.*, vol. 160, pp. 106–154, Jan. 1962.

[24] V. Vapnik, *The nature of statistical learning theory*. New York, NY, USA: Springer-Verlag, 1995.

[25] T. Zhang, "Solving large scale linear prediction problems using stochastic gradient descent algorithms," in *Proc. of the 21st ICML*, 2004.