Active Learning from Video Streams in a Multi-Camera Scenario

Samaneh Khoshrou
INESC TEC and
Faculdade de Engenharia,
Universidade do Porto
Email:skhoshrou@inescporto.pt

Jaime S. Cardoso
INESC TEC and
Faculdade de Engenharia,
Universidade do Porto
Email:jaime.cardoso@inescporto.pt

Luis F. Teixeira
Dep. Engenharia Informática
Faculdade de Engenharia,
Universidade do Porto
Email:luisft@fe.up.pt

***Abstract*—While video surveillance systems are spreading everywhere, extracting meaningful information from what they are recording is still prohibitively expensive. There is a major effort under way in order to make this process economical by including an intelligent software that eases the burden of the system. In this paper, we introduce an incremental learning framework to classify parallel data streams generated in a multi-camera surveillance scenario. The framework exploits active learning strategies in order to interact wisely with operators to address various problems that exist in such non-stationary environments, such as concept drift and concept evolution. If we look at the problem as mining parallel streams, the framework can address learning from uneven parallel streams applying a class-based ensemble, a problem that has not been addressed before. Favourable results indicate the success of the framework.**

I. Introduction

Networks of surveillance cameras have gained more importance over the last decades. Recording for hours, days, and possibly years provides massive amounts of information coming from an evolving environment in where traditional learning methods fail to reflect the evolution taking place. In such environments, the underlying distribution of data changes over time – often referred to as *concept drift* – either due to intrinsic changes (pose change, movement, etc.), or extrinsic changes (lighting conditions, dynamic background, complex object background, changes in camera angle, etc.). Thus, models need to be continually updated to represent the latest concepts. The problem is further aggravated when new objects enter the scene – referred to as *class evolution* in machine learning literature – as new models need to be trained for the novel classes.

Figure 1 demonstrates a typical surveillance scenario. When entering the scene, the object will enter the coverage area of at least one of the cameras. In such environments where objects move around and cross the FOV of multiple cameras, it is more than likely to have multiple streams, potentially overlapping in time, recorded at different starting points with various lengths, for the same individual object. An effective and appropriate algorithm to fit in our scenario is required to: a) learn from multiple streams; b) mine streams with various length and starting points (uneven streams); c) handle the concept drift; d) accommodate new classes; e) deal with partially labelled or unlabelled data; f) be of limited complexity; g) handle multi-dimensional data.

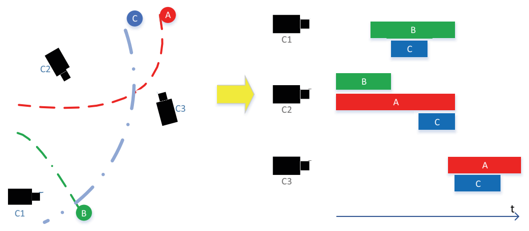


Fig. 1: Typical surveillance scenario

We put forward a framework to learn continuously from parallel video streams with partially labelled data, allowing us to learn novel knowledge and reinforce existing knowledge that is still relevant. The framework focuses on the classification of multiple video objects being tracked by a video tracking system. The framework receives directly the tracked sequences outputted by the tracking system and maintains a global object identity common to all the cameras in the system. Thus, a suitable outcome of the framework is a timeline graph assigning each stream in each camera to an identity for the indicated presence period.

This work extends our prior framework, Never Ending Visual Information Learning (NEVIL) [1]. NEVIL was instantiated with *discriminative approaches* in order to actively classify parallel video streams. While having the benefit of a robust classifier design, this strategy is not without difficulties. Firstly, if a new class appears, it will be difficult to detect the novelty. Secondly, the framework becomes biased towards the majority class in the case of severe class imbalance. To address these problems, the NEVIL framework is extended in this work with *generative models*, allowing a double threshold strategy to detect novel classes. As before, if the reliability of a decision is below a user-defined threshold (either due to a class evolution or abrupt concept drift), NEVIL will query the oracle for labelling. Additionally, if the confidence that a new class is present in batch is high enough, the framework will immediately mark it as such without querying the oracle. Besides, applying *generative* *models* avoids the pitfall of classifying every batch as the majority class.

A. Literature Review

Learning from evolving video streams over time is an area largely unexplored despite the abundance of applications generating this information. Much of the learning literature is concerned with a stationary environment, with a fixed and known number of categories to be recognized and enough available resources (labelled data, memory and computational power) [2, 3]. To get closer to a practical solution, where obtaining labelled instances is an issue, semi-supervised learning (SSL) approaches have been deployed. In [4], the person identification task is posed as a graph-based semi-supervised learning problem, where only a few low quality webcam images are labelled. The framework is able to track various objects in limited drifting environments. The classification of objects that have been segmented and tracked without the use of a class-specific tracker, has been addressed with an SSL algorithm in [5]. Given only three hand-labelled training examples of each class, the algorithm can perform comparably to equivalent fully-supervised methods, but it requires full-length tracks (it is therefore an off-line process) generated by a perfect tracker (each stream represents a single object), which would be challenging for real applications, where multiple streams are available simultaneously.

In our scenario, we look at the problem as learning from multiple data streams (herein, visual data) in wild environments, that views the whole or segments of a stream as a unique element to classify. Parallel stream mining methods in the literature [6, 7, 8] require equal-length streams coming from a fixed number of sources. Thus, they would fail to leverage information from time-varying video tracks. Learning from time-changing data in complex environments has mostly appeared in a data mining context and various approaches have been proposed. Ensemble-based approaches constitute a widely popular group of these algorithms to handle concept drift [9, 10] and in some recent works class evolution [11], as well. Ensembles provide a natural mechanism to update a knowledge by adding, removing or updating the knowledge [12]. Most approaches use a fixed size ensemble in where a classifier is trained from the most recent knowledge. This classifier replaces the oldest [13] or the least contributing member [10]. Although discarding the classifier controls the complexity, it brings up some issues, because by discarding a classifier, we simply throw away the knowledge of all the classes available in the chunk. Thus, if a class appears after a long time, it may be misclassified as most of the classifiers may include the information of the most frequent and recent classes. This problem may be addressed with approaches where all the classifiers are kept in the ensemble. Learn++.NSE [11] generates a classifier for each batch of training data and keeps all the classifiers. It applies a dynamic weighting strategy to define the share of each ensemble in the overall decision. As the success is heavily dependent on labelled data, this method would not be applicable in the wild. Class-based ensemble is firstly introduced in [14] where a model is trained for each class in a chunk. The ensemble keeps a fixed size ensemble of each class and it has been shown that this approach is more robust than traditional ensembles, though it needs the presence of training partition in each chunk. Although this approach does not fit directly in our scenario, we expect that class-based ensemble can improve our framework.

NEVIL [1] trains a classifier for the batches available in a time slot. The classifiers are kept in an ensemble and participate in the final decision using a weighted sum strategy. If the decision is not reliable enough, the batch will be sent to an oracle to annotate it. Since NEVIL computes the posterior probability (that must sum to 1), it is likely to assign a high enough (reliable) probability to a new class and mislead the system. Thus in order to get higher accuracy we need to spend more human resources. To address this problem, a class-based ensemble is introduced, where models of each class are stored separately. In the next section we discuss our proposed algorithm for parallel stream classification.

II. Learning Framework

In this section we present our framework, that is designed for learning from non-stationary environments where no labelled data is available. The framework is able to interactively query the user to obtain the desired outputs at carefully chosen batches. The algorithm is a one-pass class-based ensemble of classifiers that trains a separate model (h_t^j) for a class (j) at every time slot (t). It also keeps models of each class in a separate ensemble (*Intra-Ensemble*). A time-adjusted weighting strategy combines the probabilities outputted by the models in order to make the final decision. The framework receives multiple visual streams, generated by a typical tracking algorithm, which analyses sequential video frames and outputs the movement of targets between the frames. Environmental challenges such as varying illumination, lack of contrast, bad positioning of acquisition devices, blurring caused by motion as well as occlusion make data often noisy and/or partially missing. We address these challenges by a batch divisive strategy, as learning from a data batch may reduce the noise and fill the gaps caused by miss-tracking.

The algorithm is provided with a series of data batches $\mathcal{D}_t^{m_i}$, where m_i is the index of the i_{th} stream present at time slot t, TS_t, (not all streams are necessarily present). Note that a stream corresponds to a track generated by the tracking system and a single camera can yield multiple streams. A single batch aggregates B frames. The starting time of each stream is potentially different from stream to stream but batches are aligned between streams. Inside each frame the data corresponds to some pre-selected object representation (e.g. bag of words, histogram) and is out of the scope of this paper.

The ensemble obtained by all *Intra-Ensemble (IE)* generated up to the current time slot TS_t is named the composite hypothesis H_{t-1}. With the arrival of the current data batches $\mathcal{D}_t^{m_i}, \quad i = 1 \cdots M$, the algorithm tries to predict the most probable class label for each of the

Algorithm 1 NEVIL.g

```
Input: $\mathcal{D}_t^{m_i}, i = 1, ..., M$
$W_0 \leftarrow \frac{1}{k}$
$H_0 \leftarrow W_0$
**while** $\mathcal{D}_t$ is $True$ **do**
    **Batch label prediction (Section II-1)**
    $p(C_k|\mathcal{D}_t^{m_i}) \leftarrow (\mathcal{D}_t^{m_i}, H_{t-1})$
    **Batch Confidence Level Estimation (Section II-2)**
    $BCL \leftarrow p(C_k|\mathcal{D}_t^{m_i}, H_{t-1})$
    **Model design (Section II-3)**
    $h_t^j \leftarrow \mathcal{D}_t^j,\ j = 1, ..., k$
    **Composite model structure and update (Section II-4)**
    $IE_t^j \leftarrow h_t^j,\ j = 1, ..., k$
    $H_t \leftarrow (IE_t^1, ..., IE_t^k,\ H_{t-1},\ W_t)$
**end while**
```

batches in current TS_t based on the probability estimate $p(C_k|\mathcal{D}_t^{m_i}, H_{t-1})$, where C_k runs over all the class labels observed so far.

This kind of on-line learning approach addressed in this work can suffer if labelling errors accumulate, which is inevitable. Unrelated objects will sooner or later be assigned the same label or different labels will be assigned to different views of the same object. To help mitigate this issue, we allow the system to interact with a human, to help it stay on track.

Algorithm 1 outlines our approach. Initially, the composite model is initialized to yield the same probability to every possible class (uniform prior). When the batches $\mathcal{D}_1^{m_t}$ in time slot t become available, the framework starts computing the probabilities $p(C_k|\mathcal{D}_t^{m_i}, H_{t-1})$ for each batch $\mathcal{D}_t^{m_i}$ in the time slot. Once $p(C_k|\mathcal{D}_t^{m_i}, H_{t-1})$ is obtained, a batch confidence label (BCL) is estimated; if BCL is high enough (above a predefined threshold), the predicted label

$$\arg\max_{C_k} p(C_k|\mathcal{D}_t^{m_i}, H_{t-1})$$

is accepted as correct, otherwise the user is requested to label the data batch. The labelled batches (either automatically or manually) are used to generate new separate models h_t^k (k runs over all the classes available in t) that are kept in intra ensembles IE_k, integrating in the composite, yielding H_t.

Four tasks need now to be detailed: a) the batch label prediction (by the composite model); b) the BCL estimation; c) the model design in current time slot; d) the composite model structure and update.

1) Batch Label Prediction: A batch $\mathcal{D}_t^{m_t}$ is a temporal sequence of frames $\mathcal{D}_{t,f}^{m_t}$, where f runs over 1 to the batch size B on top. The composite model, H_{t-1}, can be used to predict directly $p(C_k|\mathcal{D}_{t,f}^{m_i}, H_{t-1})$ but not $p(C_k|\mathcal{D}_t^{m_i}, H_{t-1})$. The batch (multiframe) Bayesian inference requires conditional independence

$$\begin{aligned}&p(\mathcal{D}_t^{m_i}|C_k, H_{t-1}) = \\ &p(\mathcal{D}_{t,1}^{m_i}, \cdots, \mathcal{D}_{t,B}^{m_i}|C_k, H_{t-1}) = \\ &p(\mathcal{D}_{t,1}^{m_i}|C_k, H_{t-1}) \cdots p(\mathcal{D}_{t,B}^{m_i}|C_k, H_{t-1}) = \\ &\textstyle\prod_{j=1}^{B} p(\mathcal{D}_{t,j}^{m_i}|C_k, H_{t-1})\end{aligned}$$

From there, and assuming equal prior probabilities, it is trivial to conclude that

$$p(C_k|\mathcal{D}_t^{m_i}, H_{t-1}) = Z \prod_{j=1}^{B} p(C_k|\mathcal{D}_{t,j}^{m_i}, H_{t-1}), \quad (1)$$

where Z is a normalization constant. In practice, as streams have different starting points and lengths, the number of frames may vary ($<= B$) for different batches in a given timeslot. Thus, the products of different number of small probability can make the process of defining an optimal confidence level threshold (see II-2) challenging. In order to make a prediction independent of the number of frames, we estimate the probability of a given batch by:

$$p(C_k|\mathcal{D}_t^{m_i}, H_{t-1}) = \mathcal{M}edian(p(C_k|\mathcal{D}_{t,j}^{m_i}, H_{t-1})) \quad (2)$$

Then, the framework will assign each batch to the class that maximizes $p(C_k|\mathcal{D}_t^{m_i}, H_{t-1})$.

2) The Batch Confidence Level Estimation: Having predicted a class label for a data batch, one needs to decide if the automatic prediction is reliable and accepted or rather a manual labelling be requested.

Various criteria have been introduced as uncertainty measures in literature for a probabilistic framework [15]. Perhaps the simplest and most commonly used criterion relies on the probability of the most confident class, defining the confidence level as

$$\max_{C_k} p(C_k|\mathcal{D}_t^{m_i}, H_{t-1}). \quad (3)$$

3) Model Design: At time slot t, we obtain a new set of batches that are automatically or manually labelled. A model (h_t^j) has been trained by positive instances ($\mathcal{D}_t^j$) of classes at t. Hence, we require models that can carry deep information about the distribution of an individual class. Having in mind our needs, *generative models* seem a natural choice. It is worth mentioning that we assume that all the frames belonging to a batch are from the same object.

4) The Composite Model Structure and Update: The composite model H_t is an ensemble of Intra-ensembles ($IE_t^j, j = 1, ..., k$). Each IE_t^j includes models that are incrementally trained (with no access to previous data) on incoming batches of j^{th} class (h_t^j). The approximation outputted by individual models are combined using a weighted majority voting, where the weights are dynamically updated with respect to the classifiers' time of design.

The prediction outputted by the composite model IE_t^j for a given frame $\mathcal{D}_{t,f}^{m_i}$ is

$$p(C_k|\mathcal{D}_{t,f}^{m_i}, IE_t^j) = \sum_{\ell=1}^{t} W_\ell^t h_\ell^j(C_K|\mathcal{D}_{t,f}^{m_i}),$$

where $h_\ell^j(.)$ is the model trained from batches of j_{th} at TS ℓ, W_ℓ^t is the weight assigned to model ℓ, adjusted for time t. The weights are updated and normalised at each time slot and chosen to give more credit to more recent knowledge. They are chosen from a geometric series $(1, ..., (1-\alpha)^\ell)$. After combing the decisions of the models inside every IE, the ensemble will assign a batch with the label of the IE with the highest probability.

III. Experimental Methodology

A. Datasets

In order to explore the properties of the proposed framework, we evaluated it on multiple datasets covering various possible scenarios in a multi-camera surveillance system. We conducted our experiments on synthetic as well as real datasets.

The synthetic dataset is generated in the form of (X, y), where X is a 2-dimensional feature vector, drawn from a Gaussian distribution N(μ_X, δ_X), and y is the class label. Since in real applications visual data may suffer from both gradual and abrupt drift, we tried to simulate both situations in our streams by changing μ_X and δ_X in the parametric equations. In this experiment, we generated 7 classes $(C1, C2, ..., C7)$; for some $(C5, C6)$ data changes gradually while others also experience one $(C1, C4, C7)$, or three $(C2, C3)$ dramatic drifts. This process is similar to the one used in [11]. The dataset was organized in 4 different scenarios with different levels of complexity, including streams with gradual drift, sudden drift, re-appearance of objects and non-stationary environments where we have abrupt class and concept drift [1]. Each scenario includes:

- *Scenario* I: gradually drifting streams of 5 classes.
- *Scenario* II: streams with abrupt drifts of 5 classes.
- *Scenario* III : re-appearance of objects.
- *Scenario* IV: a non-stationary environment with class evolution as well as concept drift.

Besides synthetic datasets, we run our experiments on a number of CAVIAR video clips [16] including: OneLeave ShopReenter1, Enter ExitCrossingPaths1, OneShopOneWait1, OneStopEnter2, OneStopMoveEnter1, and WalkByShop1front. Due to the presence of different perspectives of the same person, streams are drifting in time. These sequences present challenging situations with cluttered scenes, high rates of occlusion, different illumination conditions as well as different scales of the person being captured. We employ an automatic tracking approach [17] to track objects in the scene and generate streams of bounding boxes, which define the tracked objects' positions. As the method may fail to perfectly track the targets, a stream often includes frames of distinct objects. A hierarchical bag-of-visterms method is applied to represent the tracked objects, resulting in a descriptor vector of size 11110 for each frame (refer to [18] for additional details). In order to avoid the curse of dimensionality that system may suffer from, Principle Component Analysis (PCA) is applied to the full set of descriptor features as a pre-processing step. Hence, the number of features in each stream is reduced to 85 dimensions.

B. Random Strategy (Baseline)

The Random strategy [19] labels the incoming batches randomly instead of wisely deciding which batches is more informative. Constrained by budget, batches are sent for annotation.

C. Instantiation of Classifiers

In Section II-3, we noted that designing a generative model when we only have positive sample is quite straightforward. However, a challenge may arise since these models output likelihood whereas we have made all the decisions based on posterior probability in the theoretical discussion. Nevertheless, assuming equal priors and employing the Bayes' theorem, the likelihood and posterior are congruent. We chose *Gaussian Mixture Models (GMM)* as the *generative approach*.

D. Evaluation Criteria

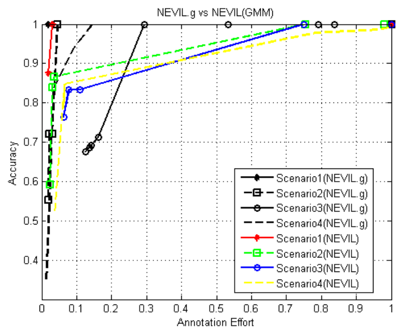
Active learning aims to achieve high accuracy using as little annotation effort as possible. Thus, a trade-off between accuracy and proportion of labelled data can be considered as one of the most informative measures. Let N denote the total number of batches, MCB refer to misclassified batches, then the accuracy of the system in a given time slot is formulated as $Accuracy = \frac{N - \#MCB}{N}$. The total accuracy of a system over a period of time is derived from the mean accuracy of all the time slots.

Assume MLB and TB denote the manually labelled batches and all the batches available during a period (includes one or more time slots), respectively. The *Annotation Effort* is formulated as $Annotation\ effort = \frac{\#MLB}{\#TB}$. One expects that the accuracy increases with the increase of annotation effort.

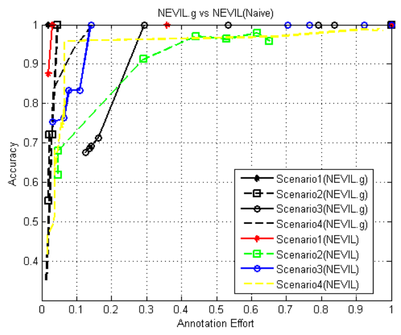
IV. Results

Firstly, multiple tests were run to determine the optimal batch size for each dataset to be explored. The batch size was varied between 1% to 50% of the size of the longest stream available in each dataset. Experiments were repeated for 50 equally spaced values in that range. The optimal batch size varies and is influenced by the characteristics of the streams present in each dataset. Optimal batch sizes have been observed to range between 30 and 35 for real video streams and between 200 and 300 for synthetic sequences.

Figure 2 shows the results of NEVIL.g on multiple synthetic scenarios. To the best of our knowledge, there

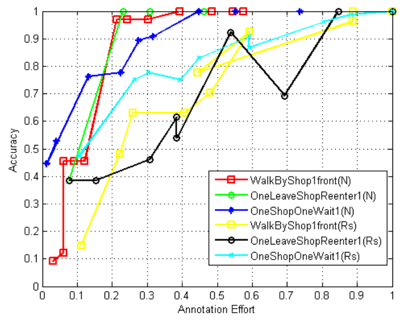


(a)

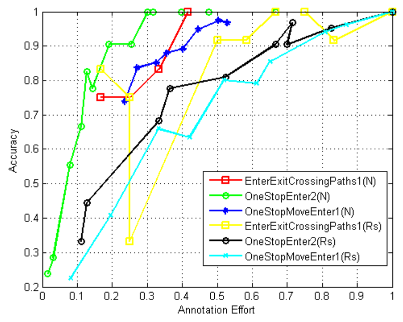


(b)

Fig. 2: Comparison of the performance of NEVIL and NEVIL.g on synthetic scenarios



(a)



(b)

Fig. 3: Comparison of the performance of Random Strategy and NEVIL.g on real scenarios

is no approach (except NEVIL) that can classify parallel data streams while interacting with an oracle (there are however clustering approaches which are not applicable in our scenario). Thus we compared this framework with NEVIL in two different settings: GMM-based NEVIL, to be consistent with the results of this version and Naive-based NEVIL, that gave the best results with synthetic data in the previous work. For most of the scenarios, NEVIL.g (black plots) outperforms NEVIL and we can get 90% accuracy by only 10% annotation effort.

Figure 3 shows that NEVIL.g is specially effective when the human collaboration is low. We see that even at big budget Random strategy may fail due to selection of non or less informative batches (see *OneLeave ShopReenter1*). Figure 4 presents the results on multiple CAVIAR datasets, where various lengths and number of streams from different classes are present. The most complex scenario is *OneStopMoveEnter1*, with 42 streams from 14 classes. We have compared the results of the proposed method with NEVIL's most successful setting in real video scenarios (SVM-based NEVIL) as well as GMM-based NEVIL. Results show a clear improvement when comparing to GMM-based NEVIL (See Figs. 4a, 4b). Comparing to SVM-based NEVIL, our method has also improved. For half of the clips (including *OneStopEnter2, WalkBy Shop1front,OneLeave ShopReenter1*), we obtain over 90% accuracy with a manual labelling of 20% of batches. Although the interaction increase for other scenarios, the value is quite acceptable considering the complexity of the data (we need to annotate 25% of batches to gain 85% correct classification for *OneStopMoveEnter1*).

V. Conclusion

We introduced a class-based ensemble framework for the classification of parallel visual data streams. The framework has shown promising performance with a fairly little human collaboration and can be applied in an on-line process.

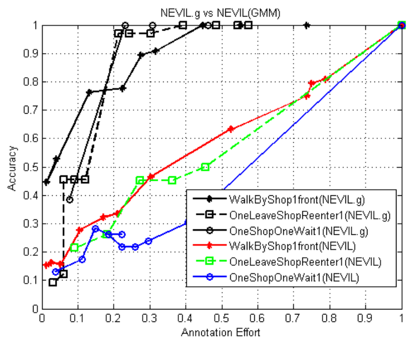
The framework posses most of the characteristics mentioned in Section I for our desired learning algorithm. However, the growing complexity is our main concern as the method does not forget any knowledge. Hence, the main direction for future work would be controlling the complexity of the ensemble in order to make the framework applicable for never-ending scenarios.

Acknowledgment

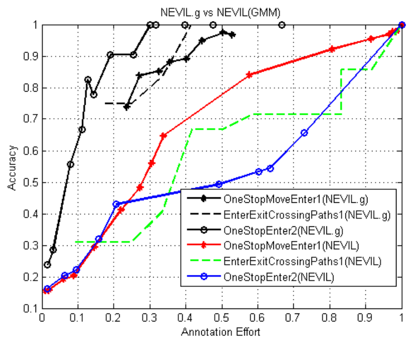
The authors would like to thank Fundação para a Ciência e a Tecnologia (FCT)-Portugal for financing this work through the grant SFRH/BD/80013/2011.

References

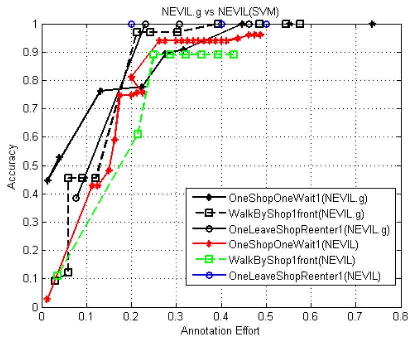
[1] S. Khoshrou, J. S. Cardoso, and L. F. Teixeira, "Active mining of parallel video streams," *Submitted to Machine*



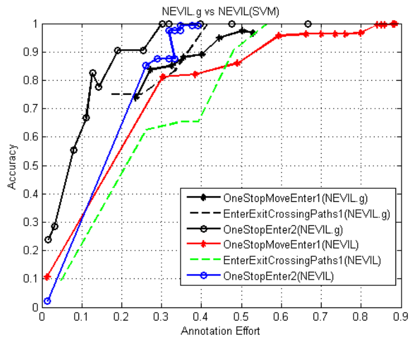
(a)



(b)



(c)



(d)

Fig. 4: Performance evaluation on multiple CAVIAR clips.

Vision and Applications, 2013. [Online]. Available: arXiv: 1405.3382

[2] S. Ozawa, S. L. Toh, S. Abe, S. Pang, and N. Kasabov, "Incremental learning of feature space and classifier for face recognition," *Neural Networks*, vol. 18, no. 5-6, pp. 575–584, 2005.

[3] J. Wu, "An online-optimized incremental learning framework for video semantic classification," in *12th ACM International Conference on Multimedia*, 2004, pp. 320–323.

[4] M. Balcan, A. Blum, P. P. Choi, J. Lafferty, B. Pantano, M. R. Rwebangira, and X. Zhu, "Person identification in webcam images:an application of semi-supervised learning," in *International Conference on Machine Learning Workshop on Learning from Partially Classified Training Data*, 2005, pp. 1–9.

[5] A. Teichman and S. Thrun, "Tracking-based semi-supervised learning," in *Proceedings of Robotics: Science and Systems*, Los Angeles, CA, USA, June 2011.

[6] J. Beringer and E. Hüllermeier, "Online clustering of parallel data streams," *Data Knowledge Engineering*, vol. 58, no. 2, pp. 180–204, 2006.

[7] P. P. Rodrigues, J. Gama, and J. P. Pedroso, "Hierarchical clustering of time-series data streams," *IEEE Transaction on Knowledge Data Engineering*, vol. 20, no. 5, pp. 615–627, 2008.

[8] L. Chen, L. Zou, and L. Tu, "A clustering algorithm for multiple data streams based on spectral component similarity," *Information Sciences*, vol. 183, no. 1, pp. 35–47, 2012.

[9] M. R. Ackermann, C. Lammersen, M. Märtens, C. Raupach, C. Sohler, and K. Swierkot, "StreamKM++: A clustering algorithms for data streams," in *Journal of Experimental Algorithmics*, 2010, pp. 173–187.

[10] J. Z. Kolter and M. A. Maloof, "Dynamic weighted majority: An ensemble method for drifting concepts," *Journal of Machine Learning Research*, vol. 8, pp. 2755–2790, 2007.

[11] R. Elwell and R. Polikar, "Incremental learning of concept drift in nonstationary environments," *IEEE Transactions on Neural Networks*, vol. 22, no. 10, pp. 1517–1531, 2011.

[12] G. Ditzler and R. Polikar, "Incremental learning of concept drift from streaming imbalanced data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no. 10, pp. 2283–2301, 2013.

[13] A. Bifet, G. Holmes, B. Pfahringer, R. Kirkby, and R. Gavaldà, "New ensemble methods for evolving data streams," in *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

[14] T. M. Al-Khateeb, M. M. Masud, L. Khan, and B. Thuraisingham, "Cloud guided stream classification using class-based ensemble," in *Proceedings of the 2012 IEEE Fifth International Conference on Cloud Computing*, ser. CLOUD '12. Washington, DC, USA: IEEE Computer Society, 2012, pp. 694–701.

[15] B. Settles, "Active learning literature survey," University of Wisconsin–Madison, Tech. Rep. 1648, 2009.

[16] C. project consortium, "Caviar dataset," 2001. [Online]. Available: http://homepages.inf.ed.ac.uk/rbf/CAVIAR/.

[17] L. F. Teixeira, P. Carvalho, J. S. Cardoso, and L. Corte-Real, "Automatic description of object appearances in a wide-area surveillance scenario," in *19th IEEE International Conference on Image Processing*, 2012, pp. 1609–1612.

[18] L. F. Teixeira and L. Corte-Real, "Video object matching across multiple independent views using local descriptors and adaptive learning," *Pattern Recognition Letters*, vol. 30, no. 2, pp. 157–167, 2009.

[19] I. Zliobaite, A. Bifet, B. Pfahringer, and G. Holmes, "Active learning with drifting streaming data," *IEEE Trans. Neural Netw. Learning Syst.*, vol. 25, no. 1, pp. 27–39, 2014.