Incremental Learning for Bootstrapping Object Classifier Models

Cem Karaoguz[1,2] and Alexander Gepperth[1,2]

***Abstract*— Many state of the art object classification applications require many data samples, whose collection is usually a very costly process. Performing initial model training with synthetic samples (from virtual reality tools) has been proposed as a possible solution, although the resulting classification models need to be adapted (fine-tuned) to real-world data afterwards. For this *bootstrapping* process, we propose to use an incremental learning algorithm from the cognitive robotics domain which is particularly suited for perceptual problems. We apply it to a pedestrian detection problem where a synthetic dataset is used for initial training, and two different real-world datasets for fine-tuning and evaluation. The proposed scheme greatly reduces the number of real-world samples required while maintaining high classification accuracy. We additionally demonstrate several innovative incremental learning schemes for object detection, the basic idea being that usually only very few background samples are actually similar to pedestrian samples. By suitable arrangement of incremental learning steps, we can keep classification models simple by representing only such "hard" background samples.**

Fig. 1: Temporal sequence of training steps during the bootstrapping process from a source dataset (usually synthetic) to a target dataset (usually real-world, here: KITTI). The first step of positive bootstrapping includes training on positive samples from the source dataset, then training on positive samples from the target dataset. The subsequent negative bootstrapping is conducted in an analogous fashion except that it excludes "easy" negative samples.

I. INTRODUCTION

Applications in domains of driver assistance systems and autonomous driving demand visual recognition of traffic participants invariant to changes like illumination, viewpoint etc. To cope with this, modern object classification systems based on statistical learning require datasets with large numbers of annotated samples recorded under different conditions. Building such large datasets is usually a tedious process and comes with high costs in resource and time. Furthermore, changes in various aspects of the application (e.g., hardware) might require construction of modified datasets and retraining of models from scratch. This retraining "from scratch" is necessary since none of the methods usually used for classification have what is termed incremental learning capacity (see [1] for a discussion of the term) that would allow to update models with new samples without complete retraining, and without "damaging" already learned knowledge.

In this context, a very popular setting for activities like domain adaptation [2] is to consider model training on a *source dataset* with easy-to-obtain synthetic samples, e.g., from a virtual reality tool, and then to adapt (or fine-tune) the model to a *target dataset* with hard-to-obtain real-world samples. The beauty of this *bootstrapping approach* is that the number of hard-to-obtain samples required for fine-tuning is usually far inferior than the number required for training from scratch, and thus a great increase in efficiency can be achieved.

In this article, we show that dedicated incremental learning algorithms, as proposed in the domain of machine learning and developmental robotics, can be used as a ready-made tool to greatly facilitate the bootstrapping process for perceptual tasks in intelligent vehicles. Incremental learning approach inherently deals with the problem of adaptation to changing data statistics, e.g., learning representations of new object categories online. Domain adaptation can be regarded as a similar problem in the sense that representations of objects learned in a source domain have to be adapted to a target domain where statistics of source and target domains are different. This motivates the application of incremental learning to the domain adaptation problem. In particular, we use the incremental learning approach presented in [3] which is particularly suited for high-dimensional perceptual problems, for implementing the bootstrapping approach between a synthetic and several real-world datasets for pedestrian classification. In each experiment a source and a target dataset are used (usually corresponding to synthetic and real-world data), both of which have positive (object) and negative (background) samples.

Bootstrapping is performed in two phases: with positive samples (from source and target datasets) first, and subsequently with negative samples (also from both datasets), see Fig. 1. Positive bootstrapping aims to form a coarse representation of an object, using the source dataset and subsequently

[1]Cem Karaoguz and Alexander Gepperth are with ENSTA ParisTech, UIIS Lab University of Paris-Saclay, 91762 Palaiseau, France

[2]Cem Karaoguz and Alexander Gepperth are with INRIA FLOWERS Team, 200 Avenue de la Veille Tour, 33405 Talence, France `cem.karaoguz@ensta.fr`, `alexander.gepperth@ensta.fr`

adapting it to the target dataset with only a limited number of examples. Negative bootstrapping aims at filtering out simple negative samples from the training process using the learned positive object representations, so that only "hard" negative samples are represented. This scheme eliminates the necessity of multiple rounds of training employed by many state of the art object classifiers as well as leaves more resources in the model to represent object characteristics.

When performing experiments using synthetic data from [4] and real-world data from the KITTI and Daimler datasets [5], [6], we show that only a few annotated real samples are enough to sufficiently adapt models to the (slightly different) statistics of real samples. Hence, the proposed framework significantly reduces the number of real images and corresponding annotations required by the model training process and renders models reusable across different datasets, eliminating the necessity of model re-training from scratch.

A. Related Work

The related work can be addressed in two main groups: incremental learning and domain adaptation. A common strategy for incremental learning is to partition the input space and use local models for each partition. This avoids common problems of machine learning like catastrophic forgetting [7] or concept drift [8] since learning is always localized in the input space, in the sense that a change of statistics in one part of that space will not affect learning in other, distant parts. The manner of performing this partitioning is very diverse, ranging from kd-trees [9] to genetic algorithms [10], adaptive Gaussian receptive fields [11]. Equally, the choice of local models varies between linear models [11], Gaussian mixture regression [9] or Gaussian Processes [12]. The choice has to be made regarding the constraints on the computational complexity imposed by the application.

Bootstrapping object classifiers is closely related to domain adaptation problem where an object classifier trained on a source dataset needs to be operated reliably on a different target dataset. Two major approaches exist to face this problem: feature transformation and model adaptation. Feature transformation relies on projecting feature vectors to a space compatible with the classifier of the source domain (e.g., [13], [14]). On the other hand, model adaptation is based on adjusting the parameters of an already learned model or learning complementary models to cope with changing data statistics. This is also the approach taken in the current work. An incremental domain adaptation framework is presented in [15] where two separate classifiers are used. A linear combination of domain and target classifiers gives the final classification result and the weight of each classifier is determined by their recorded performance. In [16] a Gaussian process regression model is constructed from confident outputs of a classifier and the scores of data instances with low prediction values are modified by this model. A-SVM is introduced in [17] which enables domain adaptation for SVM based classifiers by learning a perturbation function between source and target classifiers. This idea is extended to cope with multiple target domains by hierarchically organizing these target domains in [18]. The majority of these works are based on discriminative models where direct adaptation of the model to changing data statistics is problematic. Hence, these methods often train new models (target models) on top of the existing ones (source models) or learn a residual function, i.e., statistical difference between datasets. In contrast, in the presented work models are updated directly and continuously in the presence of new data. Hence, the approach is generic and works without prior knowledge about datasets.

II. METHODS

The proposed architecture is a three-layer neural network that maps a given image representation (input vector) to categories (output vector). The architecture is illustrated in Fig. 2. Adopting the common notation of neural networks, we utilize superscripts ^{I}, ^{H} and ^{O} for entities related to input, hidden and output layers, respectively. The input layer of the network is composed of a feature vector which is generated from the input data. The hidden layer of the network projects the input vector onto the prototype space based on a distance metric. Sub-spaces of the input space are coarsely approximated by hyperspheres whose centers are defined by the prototypes in the hidden layer. The output layer is composed of all-to-all connected neurons that map local input space regions (i.e., sets of prototypes) to class memberships using simple linear regression learning.

A. Projection

The hidden layer of the network is composed of topologically organized prototypes represented as weight vectors $w^H_{\vec{m}}$ where $w^H \in \mathbb{R}^{\mathrm{N}\times\mathrm{M}}$. Prototypes are distributed in a two dimensional grid (see Fig. 2), hence prototype locations are indicated as vectors $\vec{m}$. However, we drop the vector notation for brevity and simply use m which can also be interpreted as prototype ID. The hidden layer acts similar to the well-known self-organizing map (SOM) algorithm [19]: the projection of the input onto the hidden layer starts with computing the distance between the input vector and all prototypes:

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

where z^I is the input vector, $||\cdot||$ is the Euclidean norm. The prototype m^* with the smallest distance is called the best matching unit. In our model, the hidden layer re-encodes the input in a way that enables incremental learning while preserving information. Therefore, instead of reducing the output of the hidden layer to the best-matching unit (as it is usually done for SOMs), we calculate the (graded) activations of *all* hidden layer units:

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

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:

$$z^H = \mathrm{TF}_p\left(\tilde{z}^H\right) \tag{3}$$

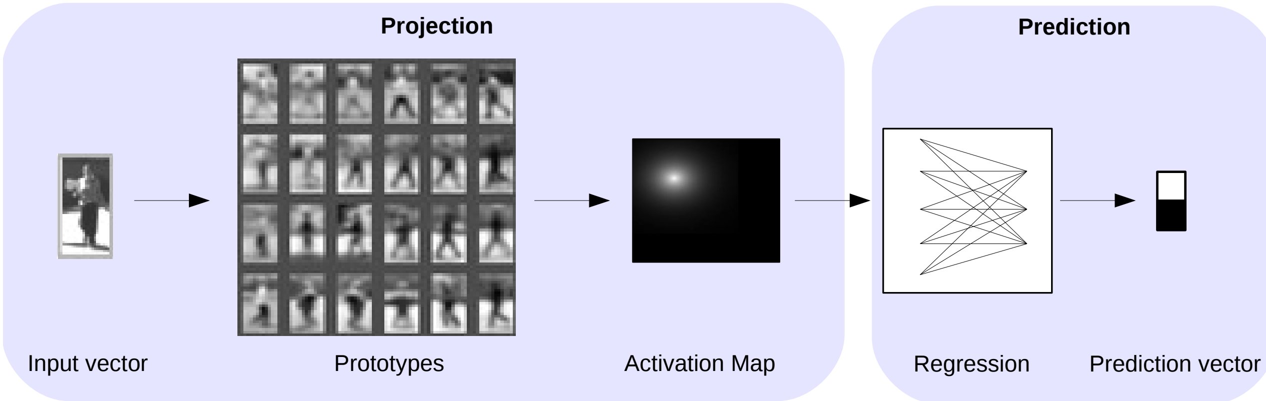


Fig. 2: The overview of the proposed architecture applied to pedestrian detection task. The feature vector is composed of image intensity values for the sake of visualization whereas HoG features are used in the actual implementation. The input vector is projected on the topologically organized prototype space that involves comparison with all prototypes. This renders an activation map of prototypes where prototypes similar to the input vector give higher values. At the prediction step the resulting activation map is mapped to a class-membership vector via regression.

where TF$(\cdot)$ represents a monotonous non-linear transfer function, TF $: [0,1] \rightarrow [0,1]$ which maintains the best matching unit value unchanged while non-linearly suppressing smaller values:

$$\mathrm{TF}_p(\tilde{z}^H) = \frac{\left(\tilde{z}^H\right)^p}{\left(\tilde{z}^H(m^*)\right)^{p-1}} \tag{4}$$

B. Prediction

Hidden layer is connected to the output layer in all-to-all fashion with weights $w^P \in \mathrm{I\!R}^{\mathrm{M}\times\mathrm{C}}$. Generation of output layer activities is performed by a simple linear transformation of hidden layer activities z^H:

$$z^O(m) = w^O_m \cdot z^H \tag{5}$$

The class associated with the unit that has the strongest activity in the output layer becomes simply the predicted class if the activity exceeds a threshold.

C. Learning model parameters

Prototype adaptation is performed online using the conventional SOM update step except that it takes into account a control signal λ coming from the output level of the hierarchy:

$$w^H_m \leftarrow w^H_m + \lambda \epsilon^H g_\sigma(||m - m^*||)(z^I - w^H_m) \tag{6}$$

where $g_\sigma(x)$ is a zero-mean Gaussian function with standard deviation σ. The control signal λ is a binary value that is set to 1 only if the current estimate of class membership, i.e., the output layer activities z^P is either uncertain or wrong. The uncertainty is measured from the bounded difference between first and second maximum of the output layer activities. If the difference is below a threshold θ_m the control signal is set to 1. In accordance with standard SOM training practices, the SOM learning rate and radius, ϵ^H and σ, are maintained at ϵ_0, σ_0 for $t < T_1$ iterations and are exponentially decreased afterwards in order to attain their long-term values $\epsilon_\infty, \sigma_\infty$ at $t = T_{\text{conv}}$.

Since the output layer performs linear regression, the weights are modified via online gradient descent, optimizing the mapping of hidden layer activities z^H to the target representation z^T containing the "true" class of a sample:

$$w^O_m \leftarrow w^O_m + 2\epsilon^O z^H \left(z^O(m) - z^T(m)\right) \tag{7}$$

In contrast to the hidden layer learning rate, the learning rate of linear regression, ϵ^P remains constant at all times.

D. Incremental Learning for Bootstrapping

We employ incremental learning for bootstrapping of models in two phases: Positive bootstrapping aims at learning the characteristics of positive samples from source and target datasets (i.e., synthetic and real-world data). In order to realize this, the system is exposed to positive samples from the source dataset for T_{bsp+} iterations. After this, fine-tuning is performed by exposing positive samples from the target dataset to the system. The incremental learning scheme outlined in Sec.II-C ensures that weights are adapted only if the system cannot correctly classify a given sample. Hence, prototypes which sufficiently describe a real-world sample are not touched.

Negative bootstrapping is performed subsequently and follows a scheme similar to the positive bootstrapping with additional heuristics. The prototype-based representation previously trained on positive samples enables early rejection of negative samples, even before actual classification, due to the generative nature of the model: Eqn. (2) computes the activation of an input vector, and negative samples yielding low activation values in this process are dissimilar to positive prototypes (i.e., objects). Setting a threshold θ_{bsp} can identify such samples in order to exclude them from further computations, i.e., classification and learning. This has two benefits: first, the model only represents negative samples which are very similar to objects. With minimum allocation of the model's resources to negative samples, more representational resources (prototypes) can thus be allocated for representing the object class. Secondly, the system can use its own output directly during tests with annotated images without having to crop and prepare negative samples beforehand.

The training schedule employed in experiments (see Fig. 1) starts with positive bootstrapping for T_{bsp+} iterations, followed by fine tuning with positive samples for T_{ft+} iterations and finalized by negative bootstrapping for T_{bsp-} iterations.

III. EXPERIMENTS

A. Setup and Parameters

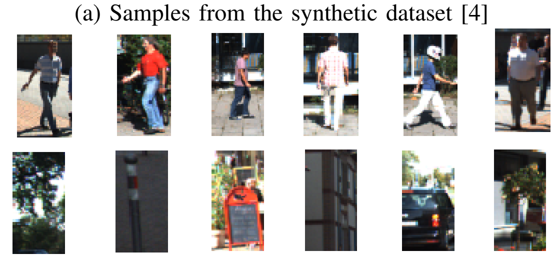
Three different datasets are used in the experiments: as a synthetic (source) dataset, the Virtual Pedestrian dataset presented in [4] is used, whereas as real-world (target) ones the KITTI Vision Benchmark Suite [5] and the Daimler Mono Pedestrian Detection Benchmark Dataset [6] are used. Additionally, sub-sets of the real-world datasets with smaller number of positive samples are generated (samples are drawn randomly from training sets) for the fine tuning process. Negative samples are kept as they are since the negative bootstrapping process seeps through these samples automatically. Sample images from the datasets are shown in Fig. 3, and the number of samples used in the experiments are shown in Tab. I for each dataset. Experiments are conducted in the following settings:

- Baseline: the model is trained and tested on the source dataset.
- Baseline-Small: the model is trained and tested on a smaller sub-set of the source dataset.
- Cross-Dataset: the model is trained on the source and tested on a target dataset without the fine tuning process.
- Bootstrapping: the model is trained on the source dataset and fine tuning is applied with the smaller sub-set of the target dataset (see Sec. II-D). The test is done on the target dataset.

The number of positive samples for the settings Baseline-Small and Bootstrapping is set to 50. We use the following fixed parameters for our system: the number of prototypes in the hidden layer: $M = 20 \times 20 = 400$, $\epsilon^O = 0.001$, $\sigma_0 = 6$, $\epsilon_\infty = 0.001$, $\sigma_\infty = 1$, $T_1 = 50000$, $T_{\text{conv}} = 150000$, $p = 20$ and $\tau = 0.001$. Both SOM and LR weight matrices are initialized to random uniform values between -0.001 and 0.001. Training examples are always randomly and uniformly drawn from the current training set. The Histograms of Oriented Gradients features extracted from samples following [20] to be used as input vectors. This method is chosen due to the availability of the feature extraction framework however, any vectorized representation of images can be used with the system. The number of iterations is set to $T_{bsp+} = 500000$, $T_{ft+} = 300000$ and $T_{bsp-} = 700000$.

For comparison to the state of the art, a discriminative approach is also implemented and evaluated to assess how close training on the source (synthetic) dataset can get to training on target (real-world) datasets, and whether it is feasible to be used as a basis for bootstrapping. We chose a state-of-the-art boosting algorithm for the purpose, which is widely used in discriminative object detection (see [21] for a survey). Our implementation follows [22] with the number of weak classifiers set to 1000 which are selected from a pool generated using the aggregated channel features explained in the paper.

(a) Samples from the synthetic dataset [4]



(b) Samples from KITTI dataset [5]

(c) Samples from Daimler dataset [6]

Fig. 3: Representative samples from datasets used in experiments

TABLE I: Number of samples used in experiments

	Training		Test	
	Positive	Negative	Positive	Negative
Synthetic	1700	1000	716	2000
KITTI	1000	10000	532	4968
Daimler	5000	5000	10660	8488
KITTI-small	50	10000	N/A	N/A
Daimler-small	50	5000	N/A	N/A

B. Cross-Database Performance of Discriminative Models

Fig. 4 shows the performance of Adaboost classifiers trained and tested on various datasets. Boosting-RonR is the classifier trained and tested on KITTI dataset hence it serves as a baseline for the real-world dataset. Similarly, Boosting-SonS is trained and tested on source dataset and serves as a baseline for this dataset. Boosting-SonR is the classifier trained on the source dataset and tested on KITTI as a target dataset. The results suggest that synthetic data has indeed

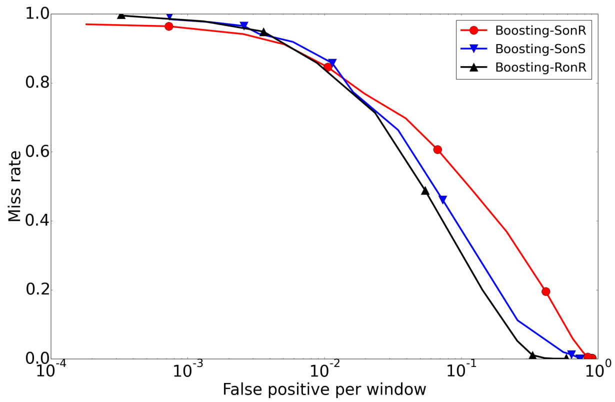


Fig. 4: Results of experiments with boosting. The legend indicates training and test datasets in the form of Training-on-Test. S and R refer to source and target (in this case KITTI) datasets.

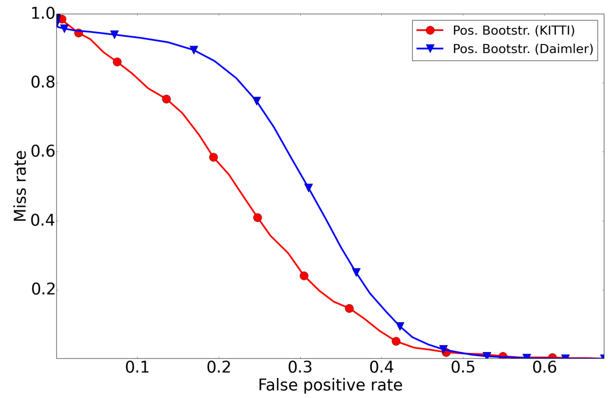


Fig. 5: Performance of the systems trained with only positive bootstrapping (source and target) on KITTI and Daimler datasets.

statistical drift with respect to the real-world one. However, the extent of this drift is still below a margin that would allow us to use the source dataset for positive bootstrapping purposes. The results are also in accordance with [23] where a similar study is done for a comparable synthetic dataset and Daimler dataset using SVM classifiers.

C. Effects of Positive Bootstrapping

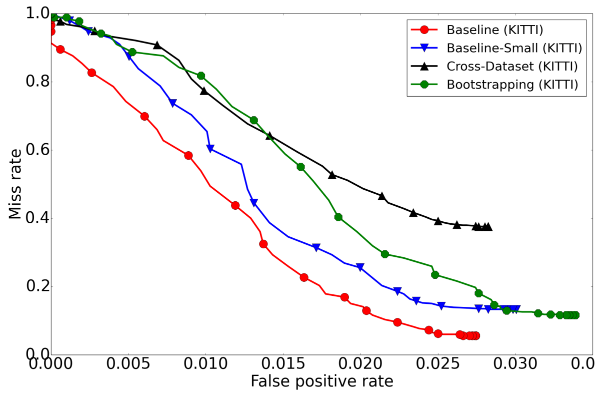
Firstly, we examine the performance of the system after positive bootstrapping only. This corresponds to the state of the models achieved after box 1 in Fig. 1 is processed. Results shown in Fig. 5 indicate that already at this stage, it is possible to filter out half of the negative samples and still achieve around 1% of miss rate on both KITTI and Daimler datasets. Adding the negative bootstrapping on top of this (i.e., after both box 1 and box 2 in Fig. 1 are processed) greatly reduces the amount of false positives as shown in Fig. 6. The variation of false positive performance of the proposed method is much less compared to discriminative models (see Fig. 4) hence the axis is not plotted in the log scale. Tab. II summarizes detection rates achieved after the full bootstrapping process at constant operating points. The model trained on the synthetic dataset yields varying performance on target datasets (indicated as Cross-Database performance). When applied to Daimler dataset, this model can achieve a performance close to the setting Baseline-Small whereas on KITTI, the performance is the lowest among all. Bootstrapping improves the models in both cases: the detection performance is increased from 62% to 82% on KITTI and from 92% to 95% on Daimler. On both datasets, the Bootstrapping setting can achieve better results than Baseline-Small indicating that virtual dataset indeed provides the models with a useful knowledge basis. Compared to the baseline detectors, the bootstrapped models still deliver inferior performance in both datasets. However, the gap is only 2% on Daimler dataset whereas it is 12% on KITTI. We can reduce this gap up to 6% if we move our point of operation from 0.027 FPR to 0.035 FPR.

D. Effects of Negative Bootstrapping

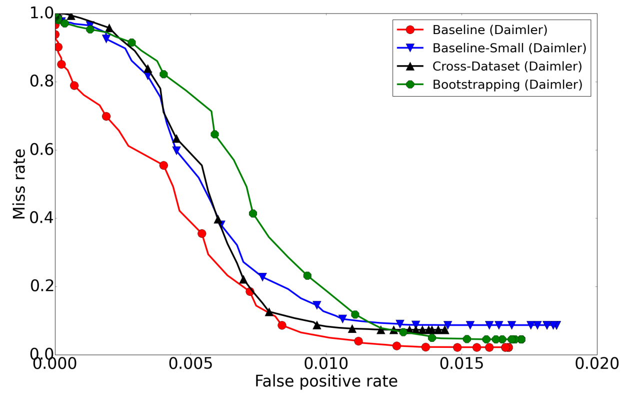
As shown earlier, a generative model trained with only positive samples can filter out ”easy” negative samples: for a sample, if no similar prototype is found, the sample can be regarded as negative. Hard negatives on the other hand, are used in the training. This process is referred to as negative bootstrapping. In order to analyze the effects of negative bootstrapping, we keep track of the negative samples that are rejected during the training process. For both Daimler and KITTI datasets around 22% of the negative samples are filtered out by the models during the whole negative bootstrapping process. This is not to be confused with the performance of the system where models are trained with positive bootstrapping only (Fig. 5). During negative bootstrapping, as hard negative samples are accepted by the system the generative model starts building their explicit representations (i.e., prototypes). As the training process advances these prototypes accept more negative samples, which similar to them, reducing the total number of negative samples filtered out by the end of training. Fig. 7 shows some examples of the accepted and skipped negative samples from Daimler dataset. It can be noticed that the system not only eliminates easy samples such as sky, but also samples with texture. On the other hand, samples with high texture seem to be included in the training process. These may generate feature vectors that can confuse the models depending on texture patterns e.g. specific constellation of tree branches and leaves or shadows cast on the road surface.

IV. CONCLUSION

We present an object classification architecture that combines generative and discriminative models facilitating incremental learning. The main contribution of this work is the utilization of the incremental learning approach for bootstrapping of object classification models. Bootstrapping is practised with both positive and negative samples. The



(a) KITTI dataset



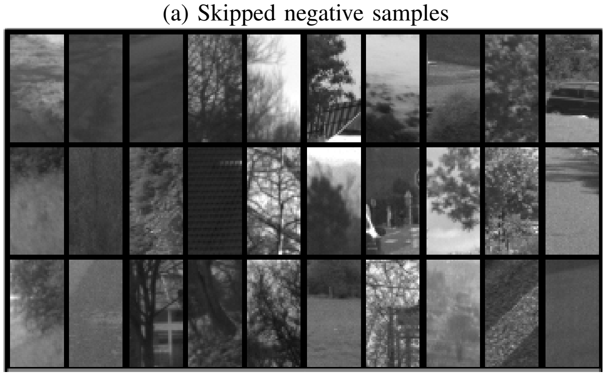
(b) Daimler dataset

Fig. 6: Performance of the systems trained under various settings.

TABLE II: Detection performance obtained in experiments

	Detection Rate	FPR
KITTI Baseline	0.94	0.027
KITTI Baseline-Small	0.87	0.027
Cross-Dataset (KITTI)	0.63	0.027
Bootstrapping (KITTI)	0.82	0.027
Daimler Baseline	0.97	0.015
Daimler Baseline-Small	0.91	0.015
Cross-Dataset (Daimler)	0.92	0.015
Bootstrapping (Daimler)	0.95	0.015

(a) Skipped negative samples



(b) Included negative samples

Fig. 7: Some of the negative samples included and excluded by the system during training on Daimler dataset.

essence of positive bootstrapping is training the generative model with positive samples from one dataset followed by a fine tuning process with positive samples from another dataset. The incremental learning capability of the system allows to make local changes in the model hence, the knowledge acquired from the first dataset is retained if it is useful for the second and statistical properties of the second dataset, which is not covered, is incorporated into the model gracefully. The feasibility of the system is demonstrated on pedestrian detection task. The positive bootstrapping is employed with synthetic data from the synthetic Pedestrian dataset and tested on a real-world dataset. It is possible to further improve the performance to the baseline level via the fine tuning step with only a few labeled examples from the real-world dataset. This is especially useful for applications where the amount of annotated data is limited. Cheaper and more convenient synthetic data can be used to bootstrap the models and state of the art performance can still be achieved.

In addition to positive bootstrapping, we also proposed negative bootstrapping where the system can reject a portion of negative samples using the internal object representations built by positive training. Incremental learning yields the benefit of updating only the parts of the model where the positive/negative discrimination stays weak. In this case, the representation of negative samples in the model is kept at minimum, allowing more resources for positive samples for a better object representation. This also eliminates the necessity of employing conventional bootstrapping methods where the models are initially trained with positive samples, false positives are collected as hard negative samples via the learned models and models are re-trained with positive and hard negative samples. In the proposed approach, the system is trained with positive and negative samples sequentially. Initial training only with positive samples already builds object representations and later at test stage only hard negatives are determined by the system and incorporated in the training. This property, combined with low computational complexity of the models and GPU parallelization renders relatively short training times. For the work presented, the whole training process takes less than 30 minutes.

The presented architecture is generic and can be applied to the detection of any other dynamic object such as vehicles, traffic signs and traffic lights. The capability of the algorithm to perform multi-class object classification is already presented in [3]. Bootstrapping multiple object representations in a single model is one of the next steps to extend the architecture.

ACKNOWLEDGMENT

We gratefully acknowledge the support of NVIDIA Corporation with GPU donation for this research.

REFERENCES

[1] A. Gepperth and B. Hammer, "Incremental learning algorithms and applications," in *European Sympoisum on Artificial Neural Networks (ESANN)*, 2016.

[2] S. Ben-David, J. Blitzer, K. Crammer, A. Kulesza, F. Pereira, and J. W. Vaughan, "A theory of learning from different domains," *Machine learning*, vol. 79, no. 1-2, pp. 151–175, 2010.

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

[4] D. Vazquez, A. M. Lopez, J. Marin, D. Ponsa, and D. Geronimo, "Virtual and real world adaptation for pedestrian detection," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 36, no. 4, pp. 797–809, 2014.

[5] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *International Journal of Robotics Research (IJRR)*, 2013.

[6] M. Enzweiler and D. Gavrila, "Monocular pedestrian detection: Survey and experiments," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 31, no. 12, pp. 2179–2195, 2009.

[7] I. J. Goodfellow, M. Mirza, X. Da, A. Courville, and Y. Bengio, "An empirical investigation of catastrophic forgeting in gradient-based neural networks," *arXiv preprint arXiv:1312.6211*, 2013.

[8] P. Kulkarni and R. Ade, "Incremental learning from unbalanced data with concept class, concept drift and missing features: a review," *International Journal of Data Mining and Knowledge Management Process*, vol. 4, no. 6, 2014.

[9] T. Cederborg, M. Li, A. Baranes, and P.-Y. Oudeyer, "Incremental local online gaussian mixture regression for imitation learning of multiple tasks," 2010.

[10] M. Butz, D. Goldberg, and P. Lanzi, "Computational complexity of the xcs classifier system," *Foundations of Learning Classifier Systems*, vol. 51, 2005.

[11] S. Vijayakumar and S. Schaal, "Locally weighted projection regression: An o(n) algorithm for incremental real time learning in high-dimensional spaces," in *International Conference on Machine Learning*, 2000.

[12] D. Nguyen-Tuong and J. Peters, "Local gaussian processes regression for real-time model-based robot control," in *IEEE/RSJ International Conference on Intelligent Robot Systems*, 2008.

[13] J. Hoffman, T. Darrell, and K. Saenko, "Continuous manifold based adaptation for evolving visual domains," in *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pp. 867–874, IEEE, 2014.

[14] B. Kulis, K. Saenko, and T. Darrell, "What you saw is not what you get: Domain adaptation using asymmetric kernel transforms," in *Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '11, (Washington, DC, USA), pp. 1785–1792, IEEE Computer Society, 2011.

[15] J. Xu, S. Ramos, D. Vázquez, and A. M. López, "Incremental domain adaptation of deformable part-based models.," in *BMVC*, 2014.

[16] V. Jain and E. Learned-Miller, "Online domain adaptation of a pre-trained cascade of classifiers," in *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pp. 577–584, IEEE, 2011.

[17] J. Yang, R. Yan, and A. G. Hauptmann, "Cross-domain video concept detection using adaptive svms," in *Proceedings of the 15th international conference on Multimedia*, pp. 188–197, ACM, 2007.

[18] J. Xu, S. Ramos, D. Vázquez, and A. M. López, "Hierarchical adaptive structural svm for domain adaptation," *arXiv preprint arXiv:1408.5400*, 2014.

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

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

[21] P. Dollar, C. Wojek, B. Schiele, and P. Perona, "Pedestrian detection: An evaluation of the state of the art," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 34, no. 4, pp. 743–761, 2012.

[22] 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.

[23] J. Marin, D. V.zquez, D. Gerónimo, and A. López, "Learning appearance in virtual scenarios for pedestrian detection," in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pp. 137–144, IEEE, 2010.