Instance-level Object Recognition using Deep Temporal Coherence [⋆]

Miguel Lagunes-Fortiz[1][0000−0002−1502−7818], Dima Damen[2][0000−0001−8804−6238], and Walterio Mayol-Cuevas[1,2]

[1] Bristol Robotics Laboratory, University of Bristol, Bristol, UK
[2] Computer Science Department, University of Bristol, Bristol, UK
{mike.lagunesfortiz,Dima.Damen,Walterio.Mayol-Cuevas}@bristol.ac.uk

Abstract. In this paper we design and evaluate methods for exploiting *temporal coherence* present in video data for the task of instance object recognition. First, we evaluate the performance and generalisation capabilities of a Convolutional Neural Network for learning individual objects from multiple viewpoints coming from a video sequence. Then, we exploit the assumption that on video data the same object remains present over a number of consecutive frames. A-priori knowing such number of consecutive frames is a difficult task however, specially for mobile agents interacting with objects in front of them. Thus, we evaluate the use of temporal filters such as Cumulative Moving Average and a machine learning approach using Recurrent Neural Networks for this task. We also show that by exploiting *temporal coherence*, models trained with a few data points perform comparably to when the whole dataset is available.

Keywords: Object Recognition · Temporal Modeling · Deep Learning

1 INTRODUCTION

State-of-the-art object recognition using Convolutional Neural Networks (CNNs) is commonly achieved trough class-level learning [12].

However, the approach of using large databases is somewhat unsuitable to the widely encountered situation for intelligent agents performing tasks with *specific* and individual objects in front of them. This calls for methods capable of using only a few viewpoints of the objects of interest and being able to detect and re-detect these objects on subsequent unseen and possibly noisy scenarios.

This motivates our work for a model trained with few data (*i.e.* low hundreds of training examples per instance ,as available in a few seconds of video) that can achieve the same level of performance as one trained with an order of magnitude more of training data. To achieve this, we leverage the end-to-end nature of very deep CNNs for learning and extracting features and using temporal filters for exploiting the *temporal coherence* present on video data to make the predictions

[⋆] This work was funded in part by the Mexican scientific agency Consejo Nacional de Ciencia y Tecnologia (CONACyT)

more robust against the commonly encountered changes on perspective, scale, illumination, object's pose and adversarial noise.

First, we explore how the way of collecting data (*i.e.* different distributions of the training data) influences the performance of the CNN. This is, if acquiring images by following a vertical-slices trajectory leads to the same level of performance of a more methodical way, such as using a sphere with all Point-of-Views (POV). We evaluate too if frame-to-frame training leads to better results compared to skipping-frames in seeks to use data more efficiently and speed up the training.

Then, we exploit the *temporal coherence* trough filters applied across frame-to-frame CNN predictions. The first filter is a simple algorithm used for predicting the next element on a sequence and consists on averaging the predictions over a number of frames. This method already offers improvement and does not require a training stage but does requires a careful selection of the number of frames to be fused in order to avoid fusing a large number of predictions containing different objects. The second method is a recurrent neural network trained to produce a sequence of predictions with temporal coherence, the architecture is depicted in Fig. 1.

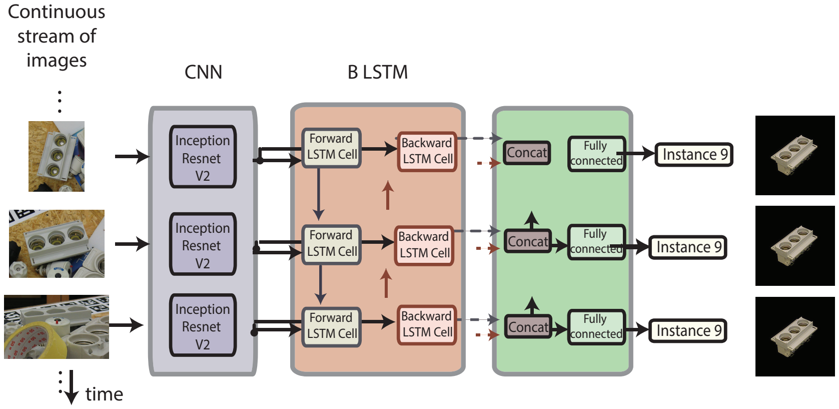


Fig. 1. A continuous stream of images goes trough a CNN and BLSTM for producing predictions with temporal coherence.

2 RELATED WORK

Instance-level Object Recognition (*i.e.* learning specific objects) was broadly studied during the past decades trough the process of extracting and matching visual feature descriptors as in [16], [1], [5]. This is not necessarily a flawed approach *per se*, but one that can lead to flaws when integrating visual components together. The effectiveness of end-to-end methods via Convolutional

Neural Networks, specifically showcased on class-level object recognition, aims to address such an integration problem.

CNNs tend on proposing deeper architectures for learning features at various levels of abstraction and achieve higher generalization capabilities. Such is case for the Inception-Resnet-V2 architecture [18]. While deeper architectures achieve lower error rates on large-scale image classification datasets, are still prone to recognition errors on real-world applications given a limited invariance to rotation [3], occlusions [17] and adversarial noise [7].

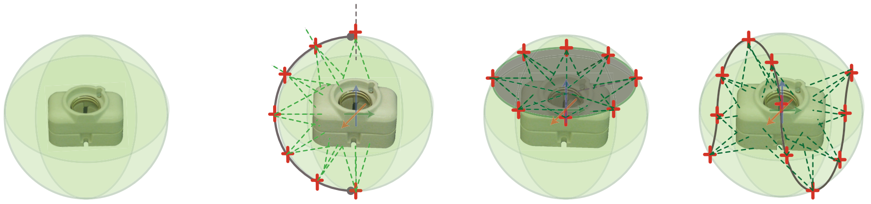
On the other hand, the use of recurrent architectures on top of CNNs has shown to be a useful approach for exploiting temporal information on video data for tasks such as Object Recognition as in *CortexNet* [2], 6-D camera relocalisation [4] and Object Tracking [19]. Within Recurrent Neural Networks, BLSTMs [9] posits as the state-of-the-art for visual sequence learning [8].

Our recognition framework operates in a less restrictive domain compared to existing work on data association like *Object Tracking*, where localization and identity of objects is known in the first frame and the tracker finds their localization on the subsequent ones. In contrast, no such *prior* information is required and we do not assume strong saliency of objects. Finally, it's different from *Video Classification* since we do not consider motion occurring between frames and there are multiple instances across the video data. Our model learns from RGB images only and can operate with video data of any length.

3 METHODOLOGY

One of the aims of this work is to find efficient ways of collecting data for multi-view Object Recognition in seeks for short training time of the model. Considering the ultimate, yet less likely in real situations, availability of an object's full view sphere Fig 2(a), we evaluate the following exploration trajectories: A vertical slice from the sphere as depicted in Fig. 2(b), a circular slice situated at 45° from the horizon Fig. 2(c) and a sinusoidal trajectory that travels around the sphere Fig. 2(d). Each of the trajectories generates a dataset containing **10% total** images compared to the sphere. Similarly, on the CORe-50 dataset we normal-random sampled the video clip sequences to form datasets with **10%** and **50% total** images to evaluate how different distributions of the training data (e.g. frame-to-frame vs. normal-sampled images) affects on the recognition performance. We then perform the next steps for each training set generated.

Our approach starts with a pre-trained Network and replacing the last layer corresponding to the Fully Connected (FC) layer that produces the output predictions (*logits*); the replacement involves adjusting the dimensionality corresponding to the number of desired objects to learn. Then we keep all Convolutional layers fixed and only the last fully connected layer is first trained, with its weights and biases initialized with a normal-random distribution. This process is commonly referred as using the *CNN as a Feature Extractor* [14] and it will be useful for choosing hyper-parameters from the temporal filters later on. Once



(a) Full Sphere (b) Vertical Slices (c) Circular Slices (d) Sinusoidal

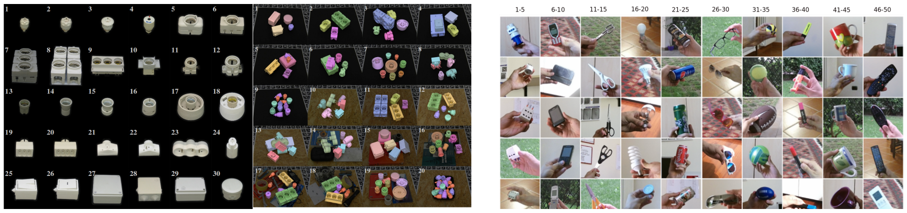
Fig. 2. Proposed Trajectories for extracting images, each forming a training set with a total of 90% less images compared to a full sphere (left) of Points of View.

the Feature Extraction model has converged, then a fine-tuning process takes place, retraining all the variables on the network until reaching convergence.

For both training phases we used the cross entropy as loss function H on its discrete form that reduces the error between the constructed probability distribution q by the CNN, and the distribution p from the ground truth, as denoted on Eq 1:

$$H(p, q) = -\sum_{x} p(x) \log q(x) \tag{1}$$

Then, our approach for achieving temporal coherence is by considering both **current** visual information and predictions from **previous** frames. This relies on the assumption: *On the stream of images, a given object is likely to persist between adjacent frames* and thus, predicted *logits* from the CNN must be consistent over these frames. We only take into account the final predictions from the CNN, in contrast with methodologies used on video and action classification where information from middle and early layers is used by 3D convolutions [13] or recurrent connections [10] for modeling *motion* information which is not required for object recognition applications.



(a) T-LESS Train set (b) T-LESS Test set (c) CORe50 dataset

Fig. 3. T-LESS and CORe50 datasets images borrowed from [11] and [15] respectively, we recommend using the digital version of this document for a closer inspection of the instances and their ID number.

As Temporal Filters we use two techniques: ***Cumulative Moving Average***, a baseline and simple sum-rule in which an average calculation is performed over the *logits* vector $\boldsymbol{x}$ from the CNN by a window size of n frames as denoted on Eq 2. The running average is applied across a continuous stream of images, consisting of the training sequences concatenated for empirically searching for n with the best trade-off between producing coherent predictions and avoiding fusing predictions with different objects. For finding n we use the Feature Extraction model for producing predictions using the training set and during evaluation, we use the fine-tuned model with the testing sequences concatenated, emulating an stream of images that an agent might receive during deployment.

$$x_{t+1} = \sum_{i=t}^{t-n} \frac{\boldsymbol{x}_i}{n} \tag{2}$$

The second technique is to use a ***Bidirectional Long-Short Term Memory (BLSTM)*** Network that takes a sequence of n predictions from the CNN and filters them to have temporal coherence. To train the BLSTM we used the Feature Extraction model, previously obtained, for producing sequences using the training set and with the ground truth labels as targets to the BLSTM. We used the same training data used for training the backbone CNN and similarly to [2] all training video clips are concatenated and presented to the network until convergence. We used the Feature Extraction model since has a lower performance compared to the fine-tuned one (mainly because the majority of its weights comes from a different dataset) thus, it will produce erroneous predictions on the training set which allows the BLSTM to learn how to correct such incoherent predictions. During evaluation we used the same testing set concatenated as for the Moving Average.

The variables from the CNN are frozen and only the gates i (input), f (forget) and o (output) gates are trained using the activation function $tanh$ for the h (states) as shown in Eq 3. The weights W and biased terms b are shared across all the cells. What makes the BLSTM unique to other recurrent approaches, is that the model processes the data sequence in both forward and backward ordering as shown in Fig 1.

$$\begin{gathered} i_t = \sigma(W_{xi}^T \boldsymbol{x}_t + W_{hi}^T \boldsymbol{h}_{t-1} + b_i) \\ f_t = \sigma(W_{xf}^T \boldsymbol{x}_t + W_{hf}^T \boldsymbol{h}_{t-1} + b_f) \\ o_t = \sigma(W_{xo}^T \boldsymbol{x}_t + W_{ho}^T \boldsymbol{h}_{t-1} + b_o) \\ \boldsymbol{h}_t = \boldsymbol{o}_t \otimes tanh(\boldsymbol{c}_t) \end{gathered} \tag{3}$$

We then concatenated the states from the forward $\overrightarrow{\boldsymbol{h}}_t$ and backwards $\overleftarrow{\boldsymbol{h}}_t$ states and train a single layer fully connected network with no activation function as shown on Eq 4, this is for predicting the same number of instances as in the *logits* vector $\boldsymbol{x}$:

$$\begin{aligned}
\overrightarrow{\boldsymbol{h}}_t &= \sigma\big(W^T_{\overrightarrow{x}\overrightarrow{o}}\boldsymbol{x}_t + W^T_{\overrightarrow{h}\overrightarrow{o}}\overrightarrow{\boldsymbol{h}}_{t-1} + b_{\overrightarrow{o}}\big) \otimes tanh(\overrightarrow{\boldsymbol{c}}_t)\\
\overleftarrow{\boldsymbol{h}}_t &= \sigma\big(W^T_{\overleftarrow{x}\overleftarrow{o}}\boldsymbol{x}_t + W^T_{\overleftarrow{h}\overleftarrow{o}}\overleftarrow{\boldsymbol{h}}_{t-1} + b_{\overleftarrow{o}}\big) \otimes tanh(\overleftarrow{\boldsymbol{c}}_t)\\
\boldsymbol{h}_t &= \Big[\overrightarrow{\boldsymbol{h}}_t, \overleftarrow{\boldsymbol{h}}_t\Big]\\
\boldsymbol{y}_t &= W^T_y\boldsymbol{h}_t + b_y
\end{aligned} \tag{4}$$

Since the BLSTM produces predictions for every image, we used the same Loss function H from Eq 1 for training. The architecture is depicted in Fig 1.

4 RESULTS

The first dataset used is T-LESS [11] (Fig 3(a) and 3(b)), which contains thirty industry-relevant objects with no significant texture and no discriminative colour, presenting symmetries and mutual similarities in shape and/or size and some of them are sub parts of others. T-LESS allowed us to evaluate the performance of a CNN for multi-POV instance-level recognition when training data comes with a nicely isolated object on a black background and testing data with increasing complexity.

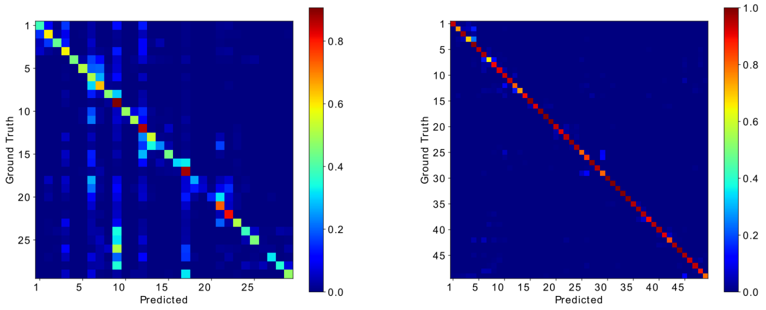
The second dataset is CORe50 [15] and contains a collection of 50 domestic objects belonging to 10 categories: plug adapters, mobile phones, scissors, light bulbs, cans, glasses, balls, markers, cups and remote controls as shown in Fig 3(c). CORe50 allowed us to evaluate the performance of the CNN on the presence of occlusions produced by a hand, alternating backgrounds and lighting conditions, which are well-suited for simulating a number of robotic applications.

For evaluating, **we use a single video containing all the testing scene concatenated**, which means there are different objects across the video data, where the models has to adapt for exploiting the temporal coherence correctly. As evaluation metric we used mean Average Precision (mAP) for T-LESS, consisting of first averaging precisions per class and then globally, due to the data unbalanced on the testing set (e.g. there are 8000 images of object 1 while only 1000 images of object 30) and Precision (P) for CORe50, which calculates the precision across all test set.

We used Inception-Resnet-V2 [18] as the backbone CNN architecture for extracting and learning features, since our temporal model uses the predictions only, smaller models can be used for fitting hardware requirements. The CNN was originally trained on the ILSVRC-2012-CLS dataset, for both re-training phases we used the Cross entropy as Loss. For the first phase of training the FC layer we used a batch size of 128 images and RMSProp (Root Mean Square Propagation algorithm) for solving the optimization problem, with the hyper-parameters: weight decay $w_d = 0.0004$, learning rate from $l_r = 0.001$ to $lr_=0.00001$, decay $\rho = 0.9$, momentum $m = 0.9$ and $\epsilon = 1^{-10}$, with decay occurring every 10 epochs as performed on [18]. For the *fine-tuning* phase we selected the same optimizer but with smaller learning rates, starting at $l_r = 0.0001$ to $l_r = 0.000001$.

4.1 Data Augmentation

In T-LESS, we deal with texture-minimal objects and varying lighting conditions. We initially performed the recommended data augmentation procedure in [6] regarding random cropping and modifications to colour and illumination. Initial results showed however, that the recommended augmentation for textured objects seems to produce inferior results when tested on texture-minimal objects. We thus do not use these data augmentation approaches for the remainder of the experiments. Applying random rotations *on-the-fly* resulted more useful on T-LESS to slightly boosting the performance. On CORe-50 we did not apply any data augmentation technique since video data shows objects with different pose, illumination and background conditions.



(a) T-LESS

(b) CORe50

Fig. 4. Normalized confusion matrices with frame-by-frame evaluation using 100% of training data, the instances ID numbers are consistent with Fig 3.

In Fig 4(a) and 4(b) we present the Normalized Confusion Matrices after training the CNN models using 100% of the data from each dataset. The numbering used on the matrices is consistent with the instances ID number presented in Fig 3.

T-LESS (Fig 4(a)) resulted a more challenging task for the CNN than CORe50 (Fig 4(b)), we believe this is explained in part by the texture-minimal characteristic on the objects which makes them easier to be confused on cluttered, partial and fully occluded conditions, *e.g.* objects like the texture-less box #27 are easily misclassified when stronger features from other objects such as the holes from object #9 appear on the image. Additionally, the available training data doesn't contain occlusions or different background conditions which makes generalization more difficult to achieve.

On CORe50 (Fig 4(b)) we notice that misclassification occurs mostly between objects from the same classes. For example, for the case of glasses (objects labelled from 26-30) present the same geometry and visual information when they are showed from the top view, then only the temples and top-bar are visible and thus, the model can get easily confused. This is indeed a challenging situation that was partially solved with the temporal filters.

4.2 Sampling methods

T-LESS allowed us to test how training with views from different sampling methods affect recognition performance. The images were obtained following the sampling methods described above. The size of the training data from all trajectories is 10% relative to the initial full-sphere set. For CORe50 we tested frame-to-frame images versus normal-sampling from each training video clip, with a total of images of 10% and 50% relative to the total amount of training data available. We run each training session three times for 150 epochs.

Table 1. Sampling methods and amount of training data

T-LESS	
Trajectory	mAP
Vertical	0.430
Circular	0.311
Sinusoidal	**0.432**
baseline 100% training data	0.468

CORe50	
Sampling Method	Precision
10% continuous frames	0.871
10% normal-random sampled	0.906
50% continuous frames	0.921
50% normal-random sampled	**0.941**
baseline 100% training data	0.943

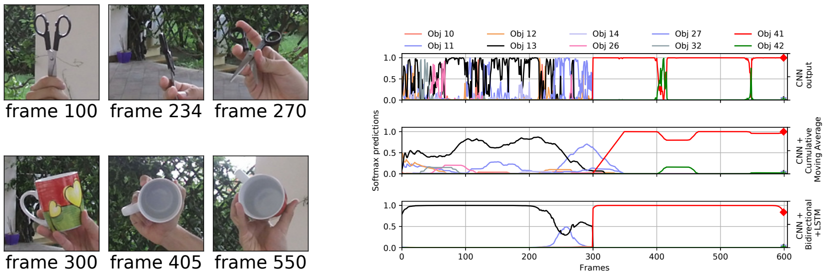
Table 1 contains the results about using trajectories for collecting data vs a full-POV sphere. We run every training session three times, showing only the best run. Results show that using more data leads to the best performance, however models trained with data containing enough variability, such images coming from the vertical or sinusoidal, offered close performance to a model trained with much more data (e.g. 10 times more data) and allows faster convergence times for the CNNs.

Additionally, Table 1 contains the results on CORe50 when different amounts of data are used, comparing frame-to-frame vs normal-random sampling from the video clips. Similarly, model trained with more data leads to the best performance, and normal-sampled slightly outperforms frame-to-frame sampling. However, interestingly the difference between 100 and 10 percent of training is only 3.7%. These results shows that diversity on the training data are key for training CNN models efficiently.

4.3 Temporal component

Table 2 present the mean Average Precision, when the temporal filters are used on the models trained with 100% and 10% of data available.

For T-LESS, the 10% of data available comes from the sinusoidal sampling method since offered the best performance, related to the use of CMA, we varied the averaging window from 1 to 60 frames with steps of 5, we only showed the best performance achieved by a window with size of 25. For the Bidirectional LSTM we varied the number of hidden states (corresponding to the number of frames that the BLSTMs can process at the time) from 100 to 600 cells and we varied the number of neurons on the gates, going from 200 to 1000, we report only the best performance achieved by a length of 400 cells with 600 neurons.



(a) First 600 testing frames

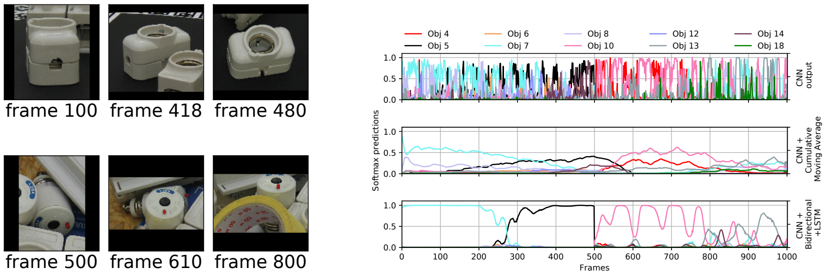
(b) Predictions from each model. The black line represents the black scissors and the red one the red and green mug.

Fig. 5. Fragment of Core50 testing sequence, showing object 13 -black scissors- and 41 -red and green mug-. The temporal filters resulted useful for correcting misclassifications caused by object's pose ambiguity. Please use the electronic version for a closer view and refer to Fig 3(c) for the numbering of the instances.

For CORe50 the 10% dataset comes from the normal-sampled method since outperformed the frame-by-frame one. We performed a similar search for the best window size for the CMA, which resulted in 40 and is consistent with the one reported on [15]. For Bidirectional LSTM the architecture with highest performance was the one with length 500 cells and 400 neurons. We used Adam Optimizer with learning rate $l_r = 1e-4$ during 50 epochs, selecting such a small learning rate was crucial for training the BLSTM for avoiding the well-know *gradient vanishing problem* on Recurrent Neural Networks.

Table 2. Temporal Filters

	T-LESS		CORe50	
Model	100% training data (mAP)	10 % training data (mAP)	100% training data (P)	10 % training data (P)
CNN	0.468	0.432	0.943	0.906
CNN + CMA	0.524	0.479	0.971	0.944
CNN + BLSTM	**0.563**	**0.550**	**0.991**	**0.979**



(a) First 500 testing frames

(b) Predictions from each model. The black line represents the Object 5 and the red one the Object 4

Fig. 6. Fragment of T-LESS testing sequence showing objects 4 and 5. Highly occluded conditions resulted quite challenging for the CNN and this could not be alleviated by the use of Temporal Filters. Please use the electronic version for a closer view and refer to Fig 3(a) for numbering of the instances.

Both temporal filters boosted the performance on the Precision (mAP on T-LESS and P on CORe50); related to the BLSTM, the results are consistent on works like in [20], in which the performance gets better using more cells and with the number of neurons being not as relevant as the number of cells.

In Fig 5 and 6 we show an example of the predictions produced by the three models (CNN, CNN + Cumulative Moving Average and CNN + BLSTM) using a fragment of the test set with two objects. On every Figure we present six images on the top, which corresponds to example frames from the testing sequence. Below, we present probabilities *per*-frame about what object is being predicted by each model. In order to maintain the plots readable, we only show the Top-10 objects detected across the presented sequence.

In Fig 5 we show how both filters resulted useful for correcting the erroneous predictions from the CNN, caused by ambiguity at the given object's pose. This is, some views are quite similar among the five mugs (objects label 41-45) such as the top view. Which was alleviated by the using information from previous frames. The first 300 frames corresponds to the black scissors (object 13) and represented by the black line on the predictions plot. The rest 300 frames corresponds to the red and green mug (object 41), represented by a red line.

In Fig 6 we show how the BLSTM performs better than the CMA, in this case by recovering faster after a set of erroneous predictions from the CNN, due to clutter conditions. Additionally, we present how neither of the filters can compensate from a majority of erroneous predictions caused in this case by highly cluttered and occluded objects. This limitation from the CNN on dealing with clutter and occlusions is more acute for minimal-texture objects, especially when objects are parts of other objects. We attribute the big gap on performance between the dataset to this CNNs limitations and how to overcome them remains as an open research question.

All our code was developed using TensorFlow 1.5, we made use of the high level API TF-Slim for using the pre-trained Inception-Resnet-V2 model and the Bidirectional LSTM implementation. All models where trained using a Titan-X GPU, using Cuda 7. A complementary video can be found here (please download the file first, in case can't be run directly in the browser).

5 CONCLUSIONS

In this paper we evaluated multi-view instance-level object recognition trough exploiting the temporal-coherence present on a continuous stream of images. We show how this way of learning can be specially useful when few data points are desired for training the models, accelerating the training process. This is useful for agents exploring the world in front of them and when they need to react and use these objects without delay. The BLSTM resulted more useful for exploiting the temporal coherence, but it does require to be trained while the simpler CMA filter shows itself useful, with the main disadvantage being the dimensionality of the window fusion. Overall our methods show useful improvements on the performance of commonly employed CNNs that do not exploit the temporal element.

References

1. Bay, H., Ess, A., Tuytelaars, T., Van Gool, L.: Speeded-up robust features (surf). Comput. Vis. Image Underst. **110**(3), 346–359 (Jun 2008). https://doi.org/10.1016/j.cviu.2007.09.014, http://dx.doi.org/10.1016/j.cviu.2007.09.014
2. Canziani, A., Culurciello, E.: Cortexnet: a generic network family for robust visual temporal representations. CoRR **abs/1706.02735** (2017), http://arxiv.org/abs/1706.02735
3. Cheng, G., Zhou, P., Han, J.: Learning rotation-invariant convolutional neural networks for object detection in vhr optical remote sensing images. IEEE Transactions on Geoscience and Remote Sensing **54**(12), 7405–7415 (Dec 2016). https://doi.org/10.1109/TGRS.2016.2601622
4. Clark, R., Wang, S., Markham, A., Trigoni, N., Wen, H.: Vidloc: 6-dof video-clip relocalization. CoRR **abs/1702.06521** (2017), http://arxiv.org/abs/1702.06521
5. Damen, D., Bunnun, P., Calway, A., Mayol-Cuevas, W.: Real-time learning and detection of 3d texture-less objects: A scalable approach. In: British Machine Vision Conference. BMVA (September 2012), http://www.cs.bris.ac.uk/Publications/Papers/2001575.pdf
6. Donahue, J., Jia, Y., Vinyals, O., Hoffman, J., Zhang, N., Tzeng, E., Darrell, T.: Decaf: A deep convolutional activation feature for generic visual recognition. CoRR **abs/1310.1531** (2013), http://arxiv.org/abs/1310.1531
7. Fawzi, A., Moosavi-Dezfooli, S., Frossard, P.: Robustness of classifiers: from adversarial to random noise. CoRR **abs/1608.08967** (2016), http://arxiv.org/abs/1608.08967
8. Graves, A., Fernández, S., Schmidhuber, J.: Bidirectional LSTM Networks for Improved Phoneme Classification and Recognition, pp. 799–804. Springer Berlin Heidelberg, Berlin, Heidelberg (2005)

9. Graves, A., Jaitly, N., Mohamed, A.: Hybrid speech recognition with deep bidirectional LSTM. In: 2013 IEEE Workshop on Automatic Speech Recognition and Understanding, Olomouc, Czech Republic, December 8-12, 2013. pp. 273–278 (2013). https://doi.org/10.1109/ASRU.2013.6707742
10. Hara, K., Kataoka, H., Satoh, Y.: Learning spatio-temporal features with 3d residual networks for action recognition. CoRR **abs/1708.07632** (2017), http://arxiv.org/abs/1708.07632
11. Hodaň, T., Haluza, P., Obdržálek, Š., Matas, J., Lourakis, M., Zabulis, X.: T-LESS: An RGB-D dataset for 6D pose estimation of texture-less objects. IEEE Winter Conference on Applications of Computer Vision (WACV) (2017)
12. Huang, J., Rathod, V., Sun, C., Zhu, M., Korattikara, A., Fathi, A., Fischer, I., Wojna, Z., Song, Y., Guadarrama, S., Murphy, K.: Speed/accuracy trade-offs for modern convolutional object detectors. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (July 2017)
13. Ji, S., Xu, W., Yang, M., Yu, K.: 3d convolutional neural networks for human action recognition. IEEE Trans. Pattern Anal. Mach. Intell. **35**(1), 221–231 (Jan 2013). https://doi.org/10.1109/TPAMI.2012.59, http://dx.doi.org/10.1109/TPAMI.2012.59
14. Li, Z., Hoiem, D.: Learning without forgetting. CoRR **abs/1606.09282** (2016), http://arxiv.org/abs/1606.09282
15. Lomonaco, V., Maltoni, D.: Core50: a new dataset and benchmark for continuous object recognition. arXiv preprint arXiv:1705.03550 (2017)
16. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. Int. J. Comput. Vision **60**(2), 91–110 (Nov 2004). https://doi.org/10.1023/B:VISI.0000029664.99615.94, https://doi.org/10.1023/B:VISI.0000029664.99615.94
17. Osherov, E., Lindenbaum, M.: Increasing cnn robustness to occlusions by reducing filter support. In: 2017 IEEE International Conference on Computer Vision (ICCV). pp. 550–561 (Oct 2017). https://doi.org/10.1109/ICCV.2017.67
18. Szegedy, C., Ioffe, S., Vanhoucke, V.: Inception-v4, inception-resnet and the impact of residual connections on learning. CoRR **abs/1602.07261** (2016), http://arxiv.org/abs/1602.07261
19. Tripathi, S., Lipton, Z.C., Belongie, S.J., Nguyen, T.Q.: Context matters: Refining object detection in video with recurrent neural networks. CoRR **abs/1607.04648** (2016), http://arxiv.org/abs/1607.04648
20. Zamir, A.R., Wu, T., Sun, L., Shen, W., Malik, J., Savarese, S.: Feedback networks. CoRR **abs/1612.09508** (2016), http://arxiv.org/abs/1612.09508