The Journal of Open Source Software

Lightning-UDA-Detect: Easily run unsupervised domain adaptation object detection

Eoghan Mulcahy[1¶], **John Nelson**[2], **and Pepijn Van de Ven**[1,2]

1 Electronic & Computer Engineering, University of Limerick, Limerick, V94T9PX, Ireland **2** Health Research Institute, University of Limerick, Limerick, V94T9PX, Ireland ¶ Corresponding author

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor: Gabriela Alessio Robles

Reviewers:

- @AnnikaStein
- @samirmartins
- @ns-rse

Submitted: 23 May 2023
Published: unpublished

Summary

Here, we present the *Lightning-UDA-Detect* library, designed to easily run unsupervised domain adaptation (UDA) based object detection. *Lightning-UDA-Detect* is structured for straightforward installation and includes automated provenance through configuration files and built-in experiment tracking. UDA can transfer knowledge from a source domain with annotated data to a target domain with unlabeled data only. UDA is particularly useful in real-world application-based settings, as annotation tasks are arduous and time-consuming. Several algorithms have been developed to utilize UDA specifically for object detection in recent years. However, implementations are difficult to install and lack user-friendliness. *Lightning-UDA-Detect* allows researchers to interact easily with and extend several popular UDA architectures specifically for object detection tasks. We present evidence that our library is proven to achieve the officially reported performance of several UDA architectures under the mAP@50 metric, the standard metric for measuring object detector performance.

Statement of need

The seminal paper for the task of unsupervised domain adaptive object detection “Domain Adaptive Faster R-CNN for Object Detection in the Wild” (DA) (Chen et al., 2018) extended Faster-RCNN (Ren et al., 2015) to use multi-level domain classifiers which enforce the learning of domain invariant features between a labeled source domain and an unlabeled target domain. The DA model was further extended by (Chen et al., 2021) to incorporate object scaling into the domain classifiers in “Scale-Aware Domain Adaptive Faster R-CNN” (SADA) which improved performance by 2.7 mAP@50. Finally, SADA was extended by (Hoyer et al., 2023) to create “Masked Image Consistency for Context-Enhanced Domain Adaptation” (MIC), which achieves current state of the art performance by extending the previous methods and adding an extra module to learn spatial context relations of the target domain. MIC improves on SADA by 3.6 mAP@50. Each of these implementations is important to the community of researchers who work with unsupervised domain adaptation for object detection.

The official implementations for DA (Wang, 2019), SADA (Chen, 2020) and MIC (Hoyer, 2022) are all built on maskrcnn-benchmark (Massa & Girshick, 2018) which has been deprecated and poses many compilation issues to the average user. Hence, installing and running these implementations is difficult, which impedes research and further advancements in UDA.

Lightning-UDA-Detect integrates with the stable and popular package torchvision (Meier & Vryniotis, 2016), part of the Pytorch (Paszke et al., 2019) project. Integrating with torchvision reduces the dependencies of *Lightning-UDA-Detect* and removes the need to compile from source. *Lightning-UDA-Detect* unifies three important architectures of unsupervised domain adaptation detection in an easy-to-run, installable and performance tested package. *Lightning-UDA-Detect* reduces the barrier to entry for working with UDA-based detection models.

Features & Functionality

- *Lightning-UDA-Detect* uses Hydra (Yadan, 2019) configuration files. Hydra allows for straightforward modification of experiment variables through hierarchical configuration by composition. This approach ensures provenance by creating a human-readable record of the experiment parameters which are separate and independent of the source code.
- *Lightning-UDA-Detect* has built-in online logging with *Weights & Biases* (Biewald, 2020). When running experiments, *Lightning-UDA-Detect* writes configuration values to an immutable run-file. Throughout runs, all relevant metrics e.g. training loss, validation accuracy and GPU power usage, are stored in the online logger. This run history provides proof and ease of reproducibility of results.
- *Lightning-UDA-Detect* integrates with Pytorch-Lightning (Falcon, 2022) a deep learning framework that facilitates automated usage of best practices. It also makes code more readable and understandable by abstracting model engineering code into standard functions e.g. `training_steps`, `validation_steps` and `process_data` .

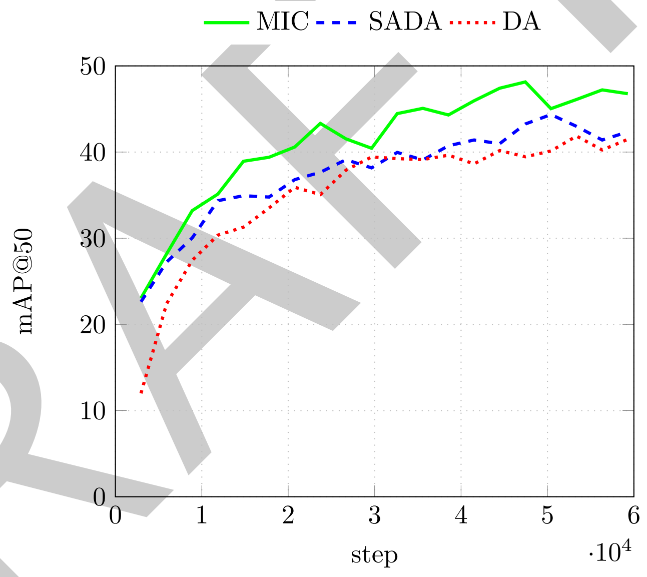

Figure 1: Mean Average Precision @50.

Figure 1 outlines the mAP@50 validation graph during training for each model. Mean average precision (mAP) is determined by calculating the average precision (AP), which is defined by how well a predicted bounding box aligns with the ground truth bounding box considering at least a 50% overlap threshold, each of these AP scores is then averaged to calculate the overall mAP@50. We used torchmetrics (Nicki Skafte Detlefsen et al., 2022) to calculate our mAP values. We ran all models on a single NVIDIA GeForce RTX 2080 SUPER GPU. We trained each model for 60,000 steps, using the same seed for every run.

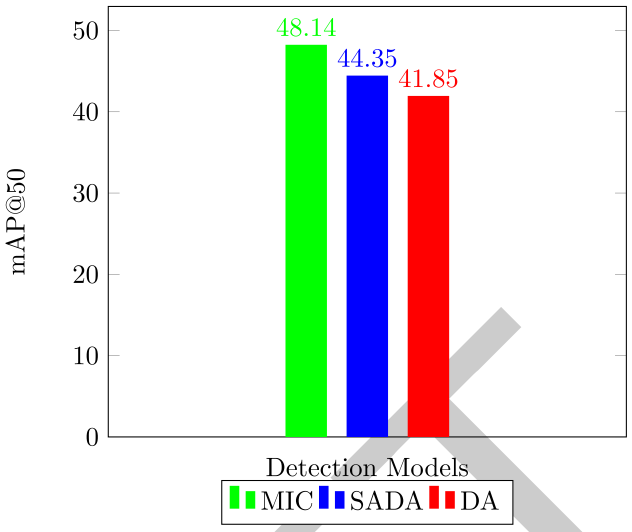

Figure 2: Max Mean Average Precision @50.

Figure 2 shows the max mAP@50 scores for each model . Our DA model achieved a score of 41.85 mAP@50 on the Cityscapes dataset, which is 0.55 higher than the officially reported value of 41.30. Our SADA model scored 44.35 mAP@50, 0.35 higher than the officially reported value of 44.00 mAP@50. Finally, our MIC model achieved a mAP@50 score of 48.14 which is 0.54 higher than the official value of 47.60. Our results show that our re-implementations can be trusted to perform correctly for each model.

Further Development

In future other UDA architectures such as 02net (Gong et al., 2022) and ILLUME (Khindkar et al., 2022), could be added to the package, along with new yet-to-be-released architectures. Contributions to the package are warmly welcome; please open an issue to discuss new features.

References

Biewald, L. (2020). Experiment tracking with weights and biases. In *GithHub repository*. Github. https://github.com/wandb/wandb

Chen, Y. (2020). Scale aware domain adaptive faster r-CNN. In *GitHub repository*. GitHub. https://github.com/yuhuayc/sa-da-faster

Chen, Y., Li, W., Sakaridis, C., Dai, D., & Gool, L. V. (2018). Domain adaptive faster r-CNN for object detection in the wild. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 3339–3348. https://doi.org/10.1109/CVPR.2018.00352

Chen, Y., Wang, H., Li, W., Sakaridis, C., Dai, D., & Gool, L. V. (2021). Scale-aware domain adaptive faster r-CNN. *International Journal of Computer Vision*, *129*, 2223–2243. https://doi.org/10.1007/s11263-021-01447-x

Falcon, W. (2022). *Pytorch lightning*. https://github.com/PytorchLightning/pytorch-lightning

Gong, K., Li, S., Li, S., Zhang, R., Liu, C. H., & Chen, Q. (2022). Improving transferability for domain adaptive detection transformers. *arXiv Preprint arXiv:2204.14195*.

Hoyer, L. (2022). MIC: [CVPR23] official implementation of MIC: Masked image consistency for context-enhanced domain adaptation. In *GitHub repository*. GitHub. https://github.com/lhoyer/MIC

Hoyer, L., Dai, D., Wang, H., & Gool, L. V. (2023). *MIC: Masked image consistency for context-enhanced domain adaptation*. https://arxiv.org/abs/2212.01322

Khindkar, V., Arora, C., Balasubramanian, V. N., Subramanian, A., Saluja, R., & Jawahar, C. (2022). To miss-attend is to misalign! Residual self-attentive feature alignment for adapting object detectors. *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 3632–3642.

Massa, F., & Girshick, R. (2018). maskrnn-benchmark: Fast, modular reference implementation of Instance Segmentation and Object Detection algorithms in PyTorch. In *GitHub repository*. GitHub. https://github.com/facebookresearch/maskrcnn-benchmark

Meier, P., & Vryniotis, V. (2016). TorchVision: PyTorch's computer vision library. In *GitHub repository*. GitHub. https://github.com/pytorch/vision

Nicki Skafte Detlefsen, Jiri Borovec, Justus Schock, Ananya Harsh, Teddy Koker, Luca Di Liello, Daniel Stancl, Changsheng Quan, Maxim Grechkin, & William Falcon. (2022). *TorchMetrics - Measuring Reproducibility in PyTorch*. https://doi.org/10.21105/joss.04101

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., … Chintala, S. (2019). PyTorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems 32*, 8024–8035. http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster r-CNN: Towards real-time object detection with region proposal networks. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems* (Vol. 28). Curran Associates, Inc. https://proceedings.neurips.cc/paper/2015/file/14bfa6bb14875e45bba028a21ed38046-Paper.pdf

Wang, H. (2019). Domain-adaptive-faster-RCNN-PyTorch: Domain adaptive faster r-CNN in PyTorch. In *GitHub repository*. GitHub. https://github.com/krumo/Domain-Adaptive-Faster-RCNN-PyTorch

Yadan, O. (2019). Hydra - a framework for elegantly configuring complex applications. In *GithHub repository*. Github. https://github.com/facebookresearch/hydra