Utilizing Transfer Learning, Graph Matching, and Spatial Attention with CARLA Pre-trained Models

Vasanth Iyer[1(✉)] and Igor Ternovskiy[2]

[1] Grambling State University, Grambling, LA 71245, USA
iyerv@gram.edu.com
[2] Air Force Research Labs, Wright-Patterson AFB, OH 45433, USA
igor.ternovskiy@us.af.mil

Abstract. This research explores practical applications of Transfer Learning and Spatial Attention mechanisms using pre-trained models from an open-source simulator, CARLA (Car Learning to Act). The study focuses on vehicle tracking using aerial images, utilizing transformers and graph algorithms for keypoint detection. The proposed detector training process optimizes model parameters without heavy reliance on manually set hyperparameters. The loss function considers both class distribution and position localization of ground truth data. The study utilizes a three-stage methodology: pre-trained model selection, fine-tuning with a custom synthetic dataset, and evaluation using real-world aerial datasets. The results demonstrate the effectiveness of our synthetic transformer-based transfer learning technique in enhancing object detection accuracy and localization. When tested with real-world images, our approach achieved an 88% detection, compared to only 30% when using YOLOv8. The findings underscore the advantages of incorporating graph-based loss functions in transfer learning and position-encoding techniques, demonstrating their effectiveness in realistic machine learning applications with unbalanced classes.

Keywords: Vehicle tracking · Aerial images · Transformers · Graph algorithms for key point detection · CNN · CARLA · Synthetic dataset

1 Research Innovation and Objective(s)

In Machine Learning, a model M with parameters and hyper-parameters would be described as $Y \approx M_H(\Phi|D)$, where Φ are parameters and H are hyper-parameters. D represents the training data and Y represents the output (class labels for vehicles). M_H represents hyper-parameters which are manually set that includes learning rate, number of layers, anchors, etc. Hyper-parameters help optimize to optimize the model to achieve higher mean-average-precision (MoU). N is the number of training samples which is assumed to be large in deep learning. The synthetic detector's objective during training is to find an

K. Arai (Ed.): FTC 2024, LNNS 1156, pp. 76–92, 2024.
https://doi.org/10.1007/978-3-031-73125-9_6

estimate of parameter $\hat{\Phi}$ that optimizes a loss function $\mathscr{L}$ are based on H, then consequently the estimated training parameters are also dependent on hyper-parameters. The experimental hyper-parameters such as Non-Maximum Suppression (NMS), are not learnt during training but fixed. In our detector training process; loss function balances model performance without relying on non-learnable hyperparameters, ensuring that the model performs well on real data despite its reliance on synthetic data during training. To increase probability of prediction the training process should not overly rely of hyper-parameters as synthetic data pre-training does not easily adapt to a real-world domain. As a matter of distinction, the results, such as label prediction, are based on model parameters $Y_{pred}\,(\Phi)$ or some initialized value during training in YOLO, while our proposed transformer based synthetic detector, the graph is matched to a constant Ground Truth (GT) using relative input Positional Encoding (K) and geometric-attention based Object Queries (Q) without any hyper-parameter tuning assuring one-to-one match during training unlike current popular models which have many human assumptions and are manually optimized.

2 Introduction

A most popular high performance object detection algorithm designed to detect and classify objects in images detector, YOLOv8 (You Only Look Once version 8) [4] has over 3.2 million parameters to learn during training. Furthermore, the choice of optimizer, loss function and size of the input images also could impact the training process. To address domain adaptation which in this case is image GT positions in synthetic environment [9] to real-world occurrences the loss function needs to learn spatial placement of the GT not only localization. Otherwise, the number of detected objects [1] will be accurate but not their positions. To better understand the loss function structure [1,3] we will analyse the mathematical definitions:

$$L(p_i, t_i) = \underbrace{\frac{1}{N_{cls}} \sum_i L_{cls}(p_i, p_i^*)}_{features} + \underbrace{\lambda \frac{1}{N_{reg}} \sum_i L_{reg}(t_i^*, t_i^*)}_{localization} \tag{1}$$

The dataset image features are captured in the first term using class distribution, which is a mixed Gaussian. The second term for positions localization of the GT are learnt using regression. The first term can be learnt using enough samples, but the second term may not converge even with large datasets due to 2D regression multi-object accuracy limitations. The simulation to real learning overview of the dataset is presented in Fig. 1.

The major task is to find a loss function which resolves the deficiencies of the 2nd regression term due to class imbalance by increasing the accuracy of the model, it could be done by decreasing false positives [1]. Equation (1) shows how two-dimensional bounding box used for localization in YOLOv8 [4] compared to learned object position using the transformer based attention encoder. The

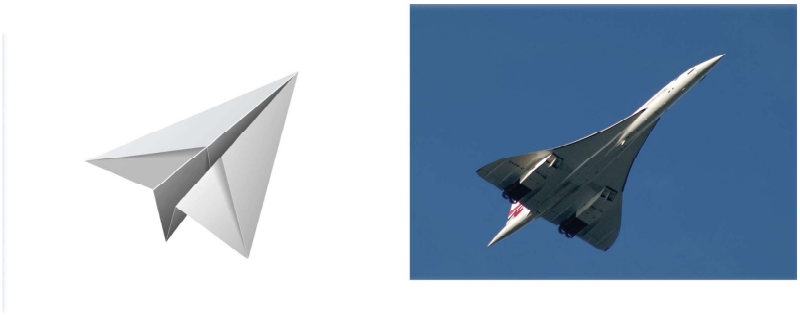

Fig. 1. Synthetic dataset generations for learning relative object position

transformers have introduced $\sin(t)$ and $\cos(t)$ functions to learn horizontal and vertical spatial information by encoding the values at different angles. Given a 2D image patch as shown in Eq. (6) two sinusoidal functions can be used for encoding vertical and horizontal angles that determines bounding box position. The frequency and phase of the sinusoidal functions vary for different positions over time t.

The following sections of this paper provide a comprehensive exploration of our proposed methodology, experimental setup, results, and discussions. Section 2 delves into the related work on object tracking frameworks, highlighting the distinctions between online algorithms and deep-learning model-based algorithms. In Sect. 3, we present our methodology, detailing the process of selecting pre-trained models, fine-tuning with a custom synthetic dataset, and evaluating [24] with real-world aerial datasets [10]. Section 4 discusses the implementation of transformers and the integration of graph algorithms for keypoint detection. Section 5 showcases the experimental results, comparing our approach with state-of-the-art models like YOLOv8. Finally, Sect. 6 concludes with a discussion on the implications of our findings and potential future research directions.

3 Related Work

This section reviews the related work on object tracking frameworks, categorizing the tracking algorithms into two primary groups due to real-time constraints: (a) online algorithms and (b) deep-learning model-based algorithms.

In the first category, trackers such as correlation filters [11,13,14,17] update the features in real-time [12]. However, they can be inefficient, and the online feature space is limited to data available during tracking. Our work belongs to the second category [1,15,16,18–21], where tracking algorithms match the target in the feature space. Since the training of these models is performed offline, they can be much more efficient and capable of performing inference at higher frames per second (fps).

The state-of-the-art performance in object detection is based on YOLOv8 [6,25,26] shown in Table 1, which has evolved over the years by continually

Table 1. Comparison of hyperparameters across YOLO versions

YOLO Version	Learning Rate	Batch Size	Input Image Size (pixels)	Epochs	Anchor Boxes	NMS Threshold
YOLOv1	0.001	64	448	135	Not Used	0.4
YOLOv2	0.001	64	416	160	5	0.4
YOLOv3	0.001	64	416	273	9 (3 scales)	0.5
YOLOv4	0.001	64	512	500	9 (3 scales)	0.5
YOLOv5s	0.01	16	640	300	9 (3 scales)	0.5
YOLOv5m	0.01	16	640	300	9 (3 scales)	0.5
YOLOv5l	0.01	16	640	300	9 (3 scales)	0.5
YOLOv5x	0.01	16	640	300	9 (3 scales)	0.5
YOLOv7	0.01	16	640	500	12 (3 scales)	0.5

improving its benchmarks. However, this evolution has also resulted in an increase in the number of parameters in the model representation. A disadvantage of this approach is that these parameters are predetermined before experiments such as anchor boxes, and Non-Maximal Suppression (NMS) thresholds as shown in Table 1. The set of hyper-parameters to reduce false positives from Table 1 is further tuned for higher accuracy, assuming that the inputs are real and the target objects are in similar settings.

Current research addresses the gap between the public availability of training data and data privacy and restriction concerns. In some domains, this gap necessitates feature engineering using synthetic data. Our work addresses this issue by using pre-trained data that employs 100% synthetic, perfectly labeled data. Furthermore, we demonstrate for the first time that our proposed transformer model [8] is invariant to any training assumptions as it uses GT positions instead which leads to higher accuracy when tested on realistic settings using CARLA pre-trained models. This review includes previous work by the same authors and other significant studies in the field.

4 Method

The transfer learning task is divided into three stages. First, we focus on pre-trained model selection. There are several large pre-trained models available, such as COCO, ImageNet, and VGG16, which serve as a solid starting point for object detection tasks like ours that involve synthetic data for object tracking [27]. In our project, we utilize the COCO pre-trained models, which encompass 1000 classes, as a baseline for our specific vehicle classes.

In the next phase, we fine-tune the pre-trained models by utilizing a custom synthetic dataset generated through the CARLA autonomous car simulator. This dataset includes vehicles with diverse backgrounds, which helps improve the model's adaptability.

The localization is performed in this stage using bounding boxes. The end-to-end pipeline integrates the COCO classification stage with bounding box localization. Within this stage, the bounding box localization prediction is combined

with the classification results at each grid location, as explained later in this section.

Finally, we proceed to the last stage, which entails testing the synthetic model using real-world aerial datasets [10] containing similar backgrounds. This allows us to assess the model's performance in more realistic scenarios.

Overall, our approach involves selecting a pre-trained model, fine-tuning it with a custom synthetic dataset comprised of vehicles generated from CARLA, Fig. 2, and ultimately evaluating its effectiveness using real-world aerial datasets with comparable backgrounds. The detection dataset includes small objects at various scales and may also have missing classes.

4.1 Pre-trained Models

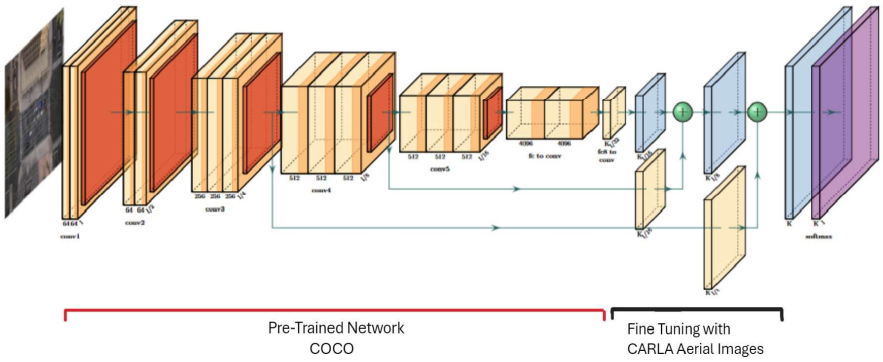

Fig. 2. Fine-Tuning the COCO backbone with synthetic CARLA images

4.2 Transformers

Provided a good backbone like CNN a transformer consists of an encoder and a decoder. Figure 3 illustrates the pipeline for the encoder and decoder with attention. This pipeline further processes the features learned from pre-trained CNN networks for the detection task. We will describe how the self-attention mechanism using the input image matrix. The architecture in Fig. 3 uses six heads to achieve the same generalization of a multi-layer CNN is shown below

$$SelfAttention(\mathrm{X})_t := softmax(\mathrm{A}_t,:)\mathrm{X}\mathrm{W}_{val} \tag{2}$$

As we will focus on the attention matrix A which is represented by $T \times T$ matrix. The weights of W_{key}^{T} represent the input pixel position and $\mathrm{X}\mathrm{W}_{qry}$ consists of the learnt weights of the global regions in which particular classes occur in the image.

$$\mathrm{A} := \mathrm{X}\mathrm{W}_{qry}\mathrm{W}_{key}^{T}\mathrm{X}^{T} \tag{3}$$

As the above matrix is invariant to bounding box position we add position encoding which captures the (x, y) cosine similarity in spatial variations for each location as shown below.

$$\mathrm{A} := (\mathrm{X}+\mathrm{P})\mathrm{W}_{qry}\mathrm{W}_{key}^{T}(\mathrm{X}+\mathrm{P})^{T} \tag{4}$$

Our original CNN loss function shown in Eq. (1) can now be rewritten with additional transformer attention terms as.

$$L(p_i, t_i) = \underbrace{\frac{1}{N_{cls}}\sum_i L_{cls}(p_i, p_i^*)}_{Class-precision} + \underbrace{\lambda\frac{1}{N_{reg}}\sum_i L_{reg}(t_i^*, t_i^*)}_{GIoU-accuracy} \quad + \\ \underbrace{SpatialPosEncoding}_{K-Attention} + \underbrace{OutputEncoding}_{Q-ObjectQueries} \tag{5}$$

4.3 Regression

Using the modified loss function we will explain using the architecture (3) on (K, Q and V) and the earlier Intersection of the Union (GIoU) with GT accuracy in this section. CNN use GIoU regression and in the case of transfer learning the spatial locations are finite during training causing lower accuracy. Due to the location error many pre-trained models with CNN architecture have many false positives when localizing the objects in the image even when the number of objects has been identified correctly. The transformer further encodes position encoding along with input features which is denoted in the third term (k). The input image is segmented into tiles as shown in Fig. 3 . Each tile's features are extracted and encoded with its relative position to other tiles to learn timeline embedding with varying positions. Using multi-head attention approach we would vary learning global features from the image tiles separately for each head. At Fig. 3 the encoder shows multi-scale feature map which handles detection differently-sized objects.

4.4 Attention-Based Object Queries

As explained in the previous section the encoder can handle multi-scale feature encoding for various object sizes (K). Further, we will describe how transformers avoid hand-crafted hyperparameter values such as anchors used in CNN. Our goal is to implement Transfer Learning from synthetic pre-trained model to real word domains. It is well known [7] that the experimental hyperparameters cannot be used in Transfer Learning. Transformers avoid this costly mistake by using an attention-based concept called object queries (Q). After the positional encoding (K) is added to the encoder's input, at the encoder further in the process at the encoder where attention is given according to what it has learned so far from previous inputs. This translates to where the most frequently occurring classes are localized, thus attention at the decoder learns to further learn a relative embedding instead of a linear regression (GIoU) avoiding any over-fitting

during training and prediction bounding boxes as shown in Fig. 3. Typically the value of Q determines the maximum number of objects expected to be in the input images.

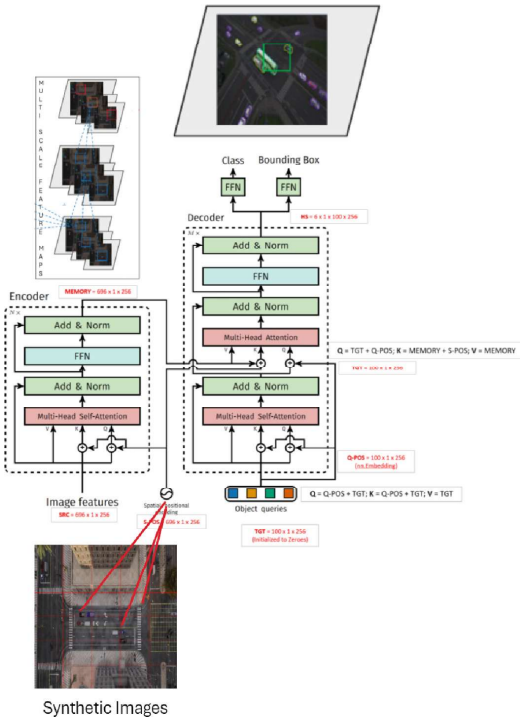

Fig. 3. Regression of bounding boxes using graph algorithms from synthetic dataset to real world datasets

4.5 Graph Matching Loss Function

From the previous section, we have learned how to estimate and train k and Q and as these determine the maximum number of objects in a detection. As discussed earlier CNN produces many false positives when objects are densely located an attention-based loss function discussed in this section, adds a new class called the 'no-object' class (also called background class) to mitigate using GT data. We discussed how the training process is different in our loss function used by the synthetic detector [9]. We use an attention-based model that allows

us to learn from the training dataset provided at every training epoch of the underlying structure of the object locations [19]. The variations are learned by using position embedding transformers [3]. Our loss function with bipartite graph matching [30] can be represented below ignoring the position encoding (K, Q) as shown.

$$L(p_i, t_i) = \underbrace{\frac{1}{N_{cls}} \sum_i L_{cls}(p_i, p_i^*)}_{precision} + \underbrace{\lambda \frac{1}{N_{reg}} \sum_i L_{reg}(t_i^*, t_i^*)}_{accuracy} + arg_{min} \sum_i^N \underbrace{\mathcal{L}_{match}(y_i, \hat{y}_{\rho(i)})}_{graphmatching} \tag{6}$$

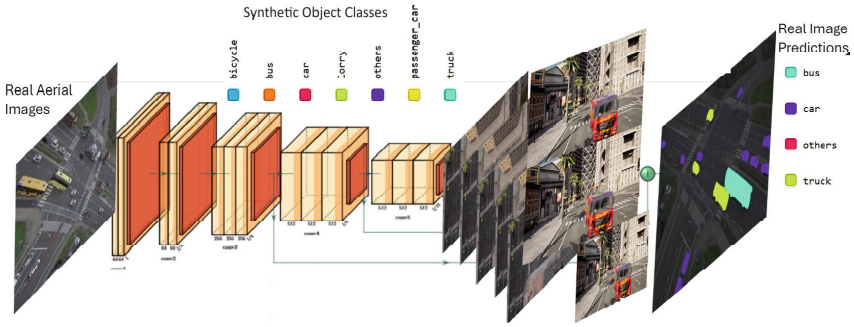

Fig. 4. Real image inference

The pre-trained model utilizes the modified graph loss function at the Transfer Learning stage, its inference architecture [28] is shown in Fig. 4. The encoder stage uses the RGB channels and distills the high-level image information features using CNN filters with four vehicle classes. The inference for real images utilizes seven vehicle classes which are detected using the synthetically pre-trained model based on four vehicle classes. The loss function used here is a bipartite matching graph between predicted classes and bounding box pairs with GT data as shown in Fig. 5. The output of the prediction is a true aerial detection or no-object class as shown in Fig. 5. The loss function used here is a bipartite matching graph between predicted classes and bounding box pairs with GT data as shown in Fig. 5. The output of the prediction is a true aerial detection or null-object class as shown in Fig. 5.

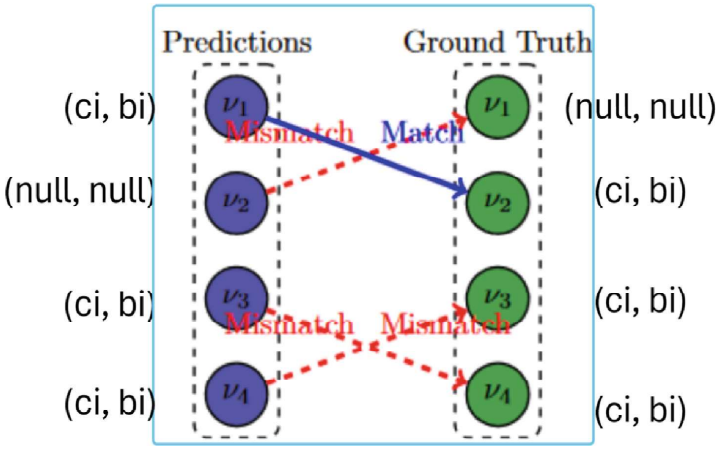

Fig. 5. Bipartite matching during training synthetic images

5 Results and Discussion

The CARLA [9] setup incorporated the following configuration. A static camera [5] was strategically positioned directly above the depicted viewpoint, ensuring a maximum distance of 50 ft. Figure 6 showcases an intersection where vehicles are required to navigate a four-way with traffic light. Leveraging the CARLA API, it becomes possible to capture vehicle labels, RGB images, and positional data during simulation. Throughout the simulation process, the training dataset is continually saved, effectively capturing screenshots of vehicles as they traverse the intersection at each simulation clock tick.

5.1 Aerial View Dataset

The dataset, generated as described in the experiment section using CARLA [9], was split into the following subsets: 7,300 samples for training, 1,500 samples for validation, and 118 samples for testing. Among these subsets, only the test images were real, while the training sets were generated using the CARLA simulator. The transformer model was pre-trained using a resnet-50 CNN backbone in the architecture. The training set was then converted to COCO JSON format using RoboFlow [29] for training with the Hugging Face trainer. After 26 epochs, the model achieved an impressive mean Average Precision (mAP) of 0.95. To optimize performance, we employed a modified bipartite loss function with a single target class, as depicted in Fig. 10. On average, each epoch took approximately seven minutes to complete.

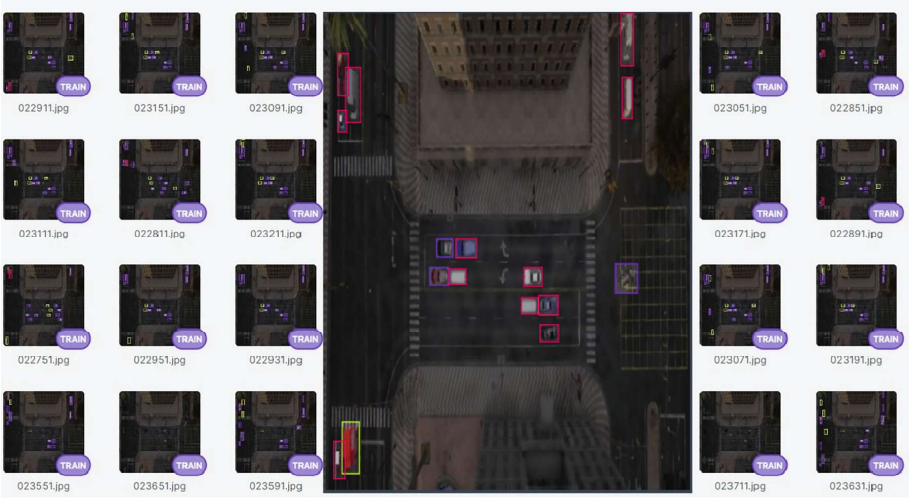

Fig. 6. A SoftMax loss-trained ImageNet Pre-Trained model consisting of 1000 classes

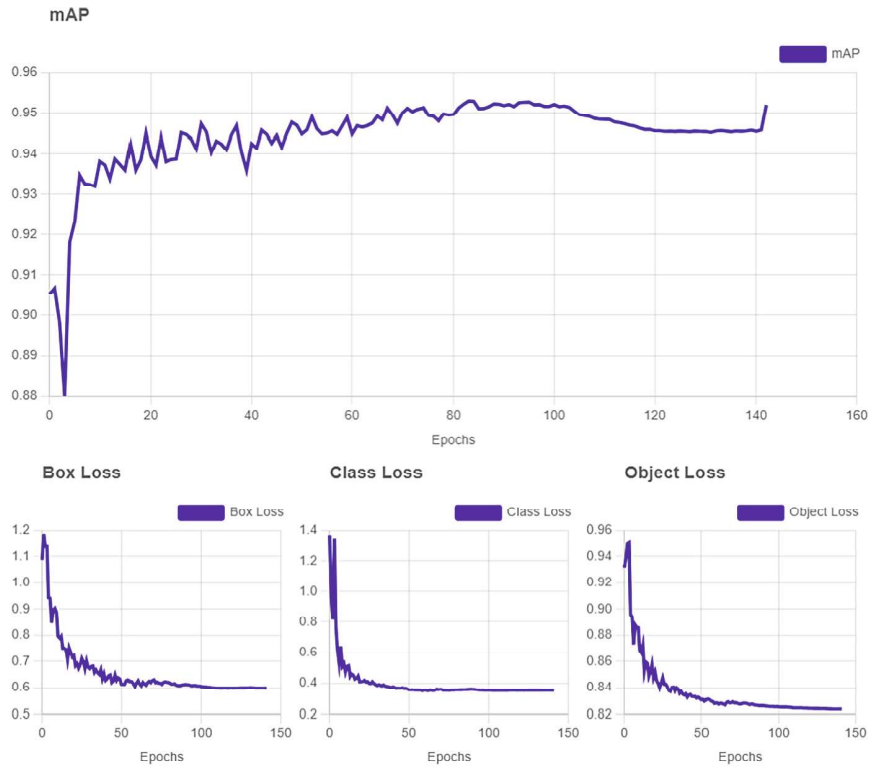

Fig. 7. Aerial view synthetic-model trained with attention transformer for a single target class

6 Results of Testing with Real Images

In our experimental setup, we employed the final trained snapshot of the transformer model that was based on synthetic data, as depicted in Fig. 7 with a single target vehicle class. This allowed us to assess the performance and accuracy of transfer learning in a real-world environment. To evaluate the results, we utilized the COCO evaluator, which calculates the mean average precision (mAP), as demonstrated in Listing x. The utilization of this evaluation metric enabled us to assess the model's efficacy in accurately detecting and localizing objects in real-world scenarios. It served as a means to compare the performance of the transfer-learning model against actual instances encountered in reality, providing valuable insights into the model's practical applicability.

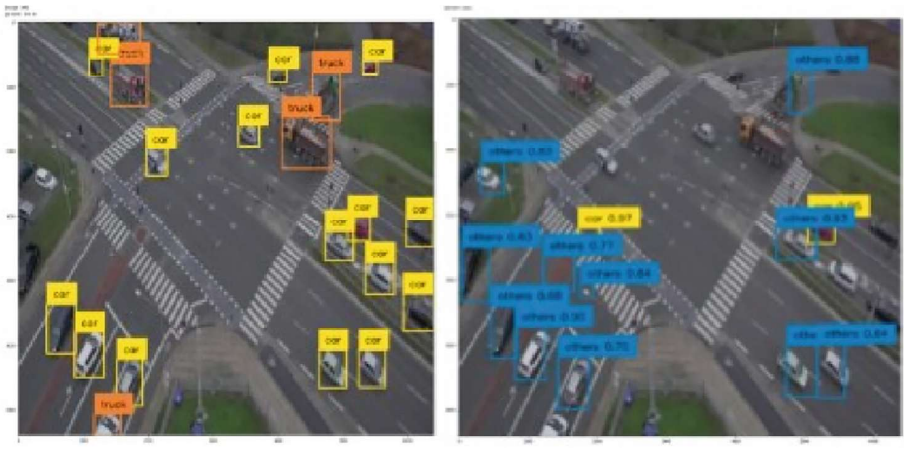

Fig. 8. Transformer model using vehicle classes tested with real traffic intersection

6.1 Effects of Class Imbalance

The 'precision' term in Eq. (6) determines the synthetic pre-trained model's class distribution. By examining the GT for classes such as cars, trucks, buses, and bicycles, we evaluated the synthetic-to-real predictions generated within an actual traffic intersection. The average precision per class was found to be notably low, as illustrated in Fig. 9. Notably, the "others" class exhibited high confidence, while vehicle classes displayed lower confidence levels. This discrepancy can be attributed to class imbalance within the synthetic dataset, leading to numerous cars being misclassified as the "other" class, as demonstrated in Fig. 8.

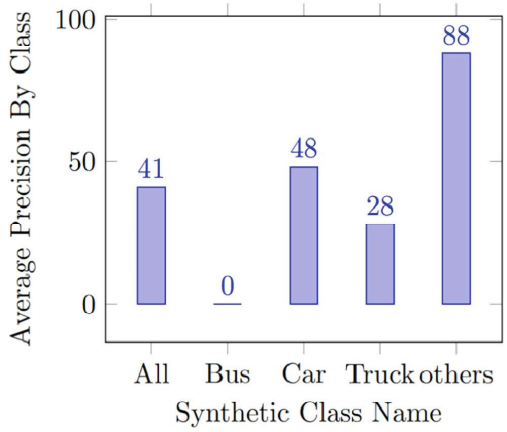

Fig. 9. Average precision of the vehicle model during testing on real images

Transition from Multi-Class to Single-Class Target. To focus is on the graph term 'accuracy' as defined in the Eq. (6), so we are not re-training our model to balance the class distribution. Instead we measure the accuracy without increasing false positives using a single class (called target class) making it invariant to class imbalance. It allows us to evaluate the spatial localization accuracy terms of the loss function.

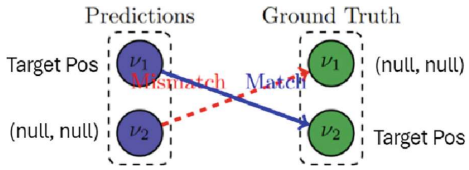

Fig. 10. Loss function uses Sim-To-Real One-Vehicle-Class

The experiments were initially performed on the COCO dataset, which consists of multiple classes. To transition from multi-class tracking [1,2] to single-class tracking, the model was trained using a single target class, specifically the "car" class. The training data included street-view, map-view, and aerial satellite imagery. The performance evaluation on the single-class tracking task indicated that the model achieved improved accuracy by 88% and robustness compared to multi-class tracking. This demonstrates from Fig. 11 the effectiveness of training on a single target class by eliminating any false positives even on unbalanced datasets.

$$Precision = \frac{TP}{TP + FP} \tag{7}$$

$$Recall = \frac{TP}{TP + FN} \tag{8}$$

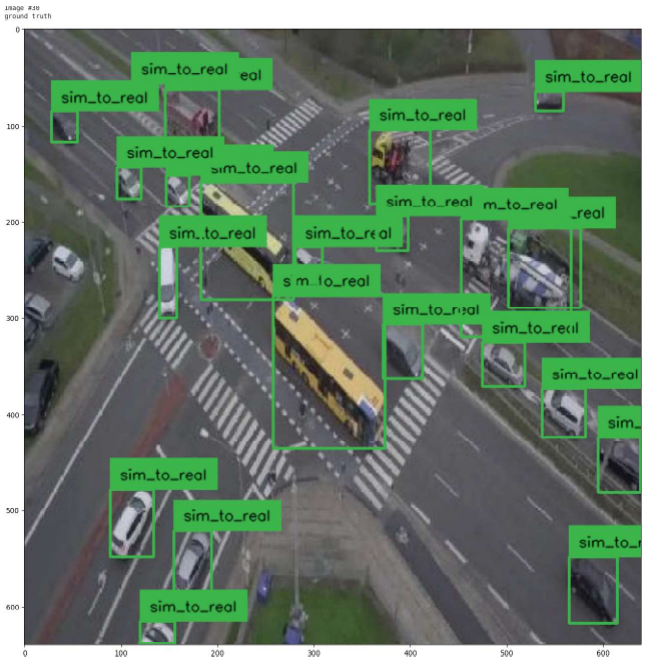

Fig. 11. Effectiveness of training on a single target class

$$Precision_{Graph} = \frac{15}{15+2} = 0.88 \tag{9}$$

$$Recall_{Graph} = \frac{15}{15+5} = 0.75 \tag{10}$$

7 Comparison of Model Architectures

In this section, we will conduct a comparative analysis between the widely used object detector models and our transformer-based model. As we have previously examined the results of transfer learning with transformers in earlier sections, we will now focus on evaluating the precision and recall of the YOLOv8 [4]architecture. Utilizing the same pre-trained MS-COCO model and fine-tuning it with the CARLA generated dataset, we calculate the precision and recall for YOLOv8 using Eq. (2) and Eq. (3). The calculated precision and recall values for YOLOv8 are presented below, as depicted in Fig. 12.

$$Precision_{Trans} = \frac{6}{6+14} = 0.3 \tag{11}$$

$$Recall_{Trans} = \frac{6}{6+14} = 0.3 \tag{12}$$

Upon comparing the previously obtained precision and recall results from the transformer-based detector, as determined by Eq. (4) and Eq. (5), with those of YOLOv8, it becomes evident that the attention-based transformer model, incorporating position embedding, is well-suited for the task of transfer learning with real-world examples in the context of aerial small-object detection. The proposed approach of using synthetic data generated from a virtual simulator, combined with position embedding transformers and graph algorithms, shows promising results in sim-to-real object tracking. By addressing the limitations of existing detectors and leveraging synthetic data, the developed model achieves improved accuracy and performance in real-world scenarios. Further research could focus on refining the synthetic dataset generation process, addressing class imbalance, and exploring additional techniques to enhance tracking performance.

Fig. 12. Testing the aerial view GT with YOLOv8 model with real images threshold @0.5

8 Contributions

In this section, we explain the challenges and contributions of the present work. We present the novelty and significance of our present work Sim-To-Real Pre-Trained models in Sect. 5. Finally, a brief overview of our transformer

aerial attention idea, "*first ever graph-based loss-function-*", and its variants are given in Sect. 4.5.

8.1 Research Problems Addressed in the Current Paper

The proposed work was envisioned to address the following questions:

- To the best of our knowledge, very few works focused on aerial object detection and tracking using synthetic pre-trained models.
- The gaps in hyper-parameter and human-crafted features that are hard to transfer to other domains such as synthetic to practical applications.
- Generating cheap and publicly available aerial dataset for training object tracking applications in the 'wild'.
- Accuracy comparison to the current state of the art in object detection models available for transfer learning.

8.2 Contributions and Future Work

- Generating error-free dataset and vehicle keypoint-annotation using CARLA simulator for real-time aerial detection.
- Transfer learning of experimental parameters using a novel relative graph embedding process.
- A novel way to map synthetic vehicle class to real target class enhancing tracking accuracy.
- Future work to merge tracking into transformers.

9 Conclusion

In this paper, we presented a novel approach for vehicle tracking using aerial images by leveraging transfer learning and graph-matching spatial attention from CARLA pre-trained models. Our methodology included selecting a pre-trained model, fine-tuning it with a custom synthetic dataset, and evaluating its performance on real-world aerial datasets. The proposed transformer-based synthetic detector demonstrated significant improvements in object detection accuracy and localization, achieving an 88% detection rate compared to only 30% with YOLOv8.

The use of graph-based loss functions [22,23] and position-encoding techniques proved to be effective in addressing the challenges associated with unbalanced classes and domain adaptation. Our results highlight the potential of incorporating synthetic data and advanced machine learning techniques to enhance real-world applications in vehicle tracking.

Future work will focus on refining the synthetic dataset generation process, addressing class imbalance, and exploring additional techniques to further improve tracking performance. The integration of tracking into transformers and the development of more sophisticated graph-based loss functions will also be areas of interest.

Overall, our study underscores the advantages of using transfer learning and graph matching spatial attention in machine learning applications, particularly for tasks involving complex and dynamic environments such as aerial vehicle tracking. The findings from this research contribute to the advancement of object detection technologies and open up new avenues for future exploration in the field.

Acknowledgments. The summer work was supported by the Air Force Summer Faculty Fellowship Program (SFFP) from AFRL. Additionally, support during the semester was provided by the Army Research Office under Grant Number W911NF-21-1-0264 and the National Science Foundation under Grant Number HBCU-EiR-2101181 for developing AI techniques for tracking applications. Some of the work has been used to develop introductory AI courses, thanks to a grant from the Google TensorFlow team.

References

1. Iyer, V., Mehmood, A.: Multi-object on-line tracking as an ill-posed problem: ensemble deep learning at the edge for spatial re-identification. In: Arai, K. (ed.) SAI 2022. LNNS, vol. 507, pp. 203–213. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-10464-0_13
2. White, J., Helmstetter, A., Culbertson, J., Ternovskiy, I.: Improving vehicle tracking techniques with a new qualia-based model. SPIE Newsroom (2015)
3. Vaswani, A., et al.: Attention is all you need. arXiv:1706.03762 (2023)
4. Jocher, G., Chaurasia, A., Qiu, J.: Ultralytics YOLO (Version 8.0.0) [Computer software] (2023). https://github.com/ultralytics/ultralytics
5. Iyer, V., Mehmood, A., Babu, B., Reddy, Y.: Spatial Blockchain: Smart Contract Using Multiple Camera Censuses. In: Arai, K. (eds.) FTC 2022 2022. LNNS, vol. 560, pp. 55-66. Springer, Cham (2023). https://doi.org/10.1007/978-3-031-18458-1_4
6. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: unified, real-time object detection. arXiv (2018)
7. Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., Zagoruyko, S.: End-to-end object detection with transformers. arXiv (2020)
8. Sun, Z., Cao, S., Yang, Y., Kitani, K.: Rethinking transformer-based set prediction for object detection. arXiv (2020)
9. Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., Koltun, V.: CARLA: an open urban driving simulator. arXiv (2017)
10. Lam, D., et al.: xView: objects in context in overhead imagery. arXiv (2018)
11. Zulch, P., Distasio, M., Cushman, T., Wilson, B., Hart, B., Blasch, E.: Escape data collection for multi-modal data fusion research. In: 2019 IEEE Aerospace Conference (2019)
12. Grabner, H., Grabner, M., Bischof, H.: Real-time tracking via on-line boosting. In: BMVC (2006)
13. Babenko, B., Yang, M.-H., Belongie, S.: Visual tracking with online multiple instance learning. In: CVPR (2009)
14. Henriques, J.F., Caseiro, R., Martins, P., Batista, J.: High-speed tracking with kernelized correlation filters. IEEE Trans. Pattern Anal. Mach. Intell. (2015)

15. Kalal, Z., Mikolajczyk, K., Matas, J.: Tracking-learning-detection. IEEE Trans. Pattern Anal. Mach. Intell. **34**(7), 1409–1422 (2012). https://doi.org/10.1109/TPAMI.2011.239
16. Held, D., Thrun, S., Savarese, S.: Learning to track at 100 FPS with deep regression networks. CoRR abs/1604.01802 (2016). arXiv:1604.01802
17. Bolme, D.S., Beveridge, J.R., Draper, B.A., Lui, Y.M.: Visual object tracking using adaptive correlation filters. In: 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pp. 2544–2550 (2010). https://doi.org/10.1109/CVPR.2010.5539960.
18. Iyer, V., Shetty, S.: Virtual sensor tracking using byzantine fault tolerance and predictive outlier model for complex tasks recognition. In: Kelmelis, E.J. (ed.) Modeling and Simulation for Defense Systems and Applications X, vol. 9478, pp. 90–96. International Society for Optics and Photonics, SPIE (2015). https://doi.org/10.1117/12.2179406
19. Iyer, V., Aved, A., Howlett, T.B., Carlo, J.T., Abayowa, B.: Autoencoder versus pre-trained CNN networks: deep-features applied to accelerate computationally expensive object detection in real-time video streams. In: Stein, K.U., Schleijpen, R. (eds.) Target and Background Signatures IV, vol. 10794, pp. 304–314. International Society for Optics and Photonics, SPIE (2018). https://doi.org/10.1117/12.2326848
20. Iyer, V., et al.: Fast multi-modal reuse: co-occurrence pre-trained deep learning models. In: Kehtarnavaz, N., Carlsohn, M.F. (eds.) Real-Time Image Processing and Deep Learning 2019, vol. 10996, pp. 35–43. International Society for Optics and Photonics, SPIE (2019). https://doi.org/10.1117/12.2519546
21. Iyer, V., Mehmood, A.: Metadata learning of non-visual features: co-occurrence overlap function for rectangular regions and ground truth data. In: Alam, M.S. (ed.) Pattern Recognition and Tracking XXXI, vol. 11400, pp. 34–42. International Society for Optics and Photonics, SPIE (2020). https://doi.org/10.1117/12.2558829
22. Liu, W., Wen, Y., Yu, Z., Yang, M.: Large-margin softmax loss for convolutional neural networks. arXiv:1612.02295 (2017)
23. Qian, Q., Shang, L., Sun, B., Hu, J., Li, H., Jin, R.: Soft triple loss: deep metric learning without triplet sampling. arXiv:1909.05235 (2020)
24. He, L., Liao, X., Liu, W., Liu, X., Cheng, P., Mei, T.: FastReID: a pytorch toolbox for general instance re-identification. arXiv preprint arXiv:2006.02631 (2023)
25. luxonis. OAK-D: Stereo camera with edge AI (2020). https://luxonis.com/
26. luxonis. DepthAI: Embedded machine learning and computer vision API (2020). https://luxonis.com/
27. Iyer, V.: Ensemble stream model for data-cleaning in sensor networks. Ph.D. thesis, USA (2013)
28. Demidovskij, A., et al.: OpenVINO deep learning workbench: towards analytical platform for neural networks inference optimization. J. Phys.: Conf. Ser. **1828**(1), 012012 (2021). https://doi.org/10.1088/1742-6596/1828/1/012012
29. Dwyer, B., Nelson, J., Solawetz, J., et al.: Roboflow (Version 1.0) [Software] (2022). https://roboflow.com.computervision
30. Iyer, V., et al.: STACK: sparse timing of algorithms using computational knowledge. In: Mukhopadhyay, S.C., Lay-Ekuakille, A., Fuchs, A. (eds.) New Developments and Applications in Sensing Technology. Lecture Notes in Electrical Engineering, vol. 83, pp. 305–320. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-17943-3_16