PAE: Portable Appearance Extension for Multiple Object Detection and Tracking in Traffic Scenes

IBRAHIM SOLIMAN MOHAMED AND LIM KIM CHUAN

Machine Learning and Signal Processing, Centre for Telecommunication Research & Innovation, Faculty of Electronics and Computer Engineering, Universiti Teknikal Malaysia Melaka, Durian Tunggal 76100, Malaysia

Corresponding author: Lim Kim Chuan (kimchuan@utem.edu.my)

This work was supported in part by LED Vision Sdn. Bhd., and in part by the Universiti Teknikal Malaysia Melaka (UTeM).

ABSTRACT Multi-object tracking (MOT) is an important field in computer vision that provides a critical understanding of video analysis in various applications, such as vehicle tracking in intelligent transportation systems (ITS). Several deep learning-based approaches have been introduced to basic motion and IoU trackers by extracting appearance features to assist in challenging situations such as lossy detection and occlusion. This study proposes a portable appearance extension (PAE) for single-stage object detection to jointly detect and extract appearance embeddings using a shared model. Furthermore, a novel training framework with a single image and without re-identification annotations is presented using an augmentation module, saving a tremendous amount of human labeling effort and increasing the real-world adoption rate. Using UA-DETRAC dataset, RetinaNet-PAE and SSD-PAE achieve comparable results with current state-of-the-art models, where RetinaNet-PAE prioritizes detection and tracking performance with a 58.0% HOTA score and 4 FPS. In contrast, SSD-PAE prioritizes latency performance with a 47.3% HOTA score and 40 FPS.

INDEX TERMS Computer vision, object detection and tracking, appearance embedding, vehicle tracking.

I. INTRODUCTION

Multiple Object Tracking (MOT) is the process of predicting the trajectories of multiple objects identified across video frames. With the rise of deep learning and object detection models, several experiments have been conducted to propose and evaluate the contribution of adopting tracking with appearance (embedding) using deep neural network. Recent research works proved that deep-learning-based trackers [2]–[4] help in adopting MOT in many real-world applications, including customer behavior studies, autonomous driving, person and vehicle tracking, and traffic management systems.

Former MOT methods [1], [5] rely on motion and velocity states of the objects to perform the tracking task using Kalman Filter [6]. In addition, intersection over union (IoU) measurement associates objects across frames. Although these two methods score fast inference time and achieve the requirements of real-time inference, they suffer in tracking occluded objects or objects with complex movement patterns. Object appearance features were introduced to tackle these issues and have shown remarkable improvements in current trackers. This concept was first introduced as a separate appearance embedding extractor that runs sequentially on each detected bounding box by an object detector [3].
two-step method has improved tracking accuracy but suffers from slow inference rate owing to the high computation required by running two models (Object Detector, Embedding Extractor) and extracting appearance embeddings for detected objects independently. This problem occurs because of extracting features of detected objects twice, once by the object detection model and once by the embedding model.

With the rise of multi-task learning, end-to-end object detection models have introduced a new architecture that can detect objects and extract embeddings in one-shot using shared feature maps extracted by object detection’s backbone [2], [4], [7], [8]. It started by having a re-identification branch in Mask-RCNN [7] to extract embedding features from proposals. Then, JDE [2] introduced a real-time joint object detection and embedding network for single-stage object detection YOLOv3 [9]. Subsequently, FairMOT [4] proved that CenterNet [10] could achieve remarkable detection and tracking accuracy as an anchor-free single object detection model and proposed a single image training by using a pre-trained model on a large re-identification dataset. Finally, EMOT [8] applied the same idea as the re-identification branch backed by the EfficientDet [11] object detector, which claimed to have the best balance between accuracy and latency. However, these one-shot models require a large dataset with video sequence annotations, which involves a significant amount of human effort in the labeling process, and it can prevent adoption in real-world applications since not all applications can fulfill such requirements. Moreover, most of these large datasets are available only for pedestrian tracking. Furthermore, each network was introduced with a different backbone and object detection architecture, requiring a set of changes and experiments to design the same methodology with different network architectures.

In this study, we present a portable appearance extension (PAE) that fits most current object detection architectures. We can train our proposed model on a single image used in a normal object detection network without video sequence annotations, powered by our proposed augmentation bag, which enables the learning of object embedding from a single frame in a self-supervised manner using only the normal detection labels. With this extension, we can ease and accelerate the process of real-world application adoption of object detection and tracking models. In contrast to other studies and following EMOT [8], a traffic dataset (UA-DETRAC) [12] was chosen to evaluate our proposed method for real-time vehicle tracking. The data originally includes the tracking annotation. Although we do not utilize it during training, it is used during the evaluation. HOTA [13] metric is chosen for evaluation due to its outstanding performance in user research [13] compared to MOTA [14] and PR-MOTA metrics. To boost the performance of our tracker, we adopt a hyperparameter optimization technique for object association, IoU, and Kalman Filter, which should be applied independently for any training dataset.

We can summarize our contributions as follows:

- proposing a portable appearance extension for object detection architectures with a novel single image training capability for extracting objects embedding simultaneously with detected objects with the help of augmentation techniques.
- experimenting with tracker hyperparameter optimization and providing a new baseline evaluation using HOTA metric on a traffic scenario dataset (UA-DETRAC).

II. RELATED WORK

A High-quality object detector model is essential for achieving satisfactory performance in multiple object tracking. This approach of tracking-by-detection achieves the best MOT performance on various benchmarks and real-world applications [2], [4], [7], [8]. First, this section discusses the current state of object detection development and its categories. Then, we review the state of object tracking and its progress over the years, starting from tracking by motion and IOU models [1], [5], [6] until achieving the state-of-the-art performance with joint end-to-end detection and tracking networks [2], [4], [7], [8], appearance features were used to contribute to tracking algorithms.

A. OBJECT DETECTION

With the rise of deep learning in recent years, we progressively enable computer vision applications to learn and extract useful features without feature engineering; two types of object detection have been raised in this field: single-stage detectors and two-stage detectors.

Single-stage detectors are designed as an end-to-end network in which the module of region-of-interest proposals is eliminated compared to two-stage detectors, and predefined anchors or anchor-free architectures replace it. Anchors [15] are predefined training samples in different proportions to facilitate various object sizes and scales during training. Alternatively, the anchor-free design [16] eliminates the hassle of anchor hyperparameters and IoU calculation by adopting a pixel-to-pixel single feature point as an object inspired by segmentation architectures. As a result, single-stage detectors are often simpler to train, more computationally efficient, and suitable for edge computing. The pioneers of single-stage detectors, YOLO [9], [17]–[19] and SSD [15], were the first evolved rapidly over versions. Version 1 [17] was designed with 24 Convolutional layers followed by two fully connected layers, inspired by GoogleLeNet [20] with a 7 × 7 grid size, two predictors (anchors), and non-max-suppression (NMS) to suppress overlapped objects. Version 2 [18] adopted the idea of anchors from [15] and introduced k-means clustering to dynamically identify the most suitable anchors according to each dataset. Furthermore, it increased the grid size to 13 × 13, introduced Darknet19 as a feature extractor (backbone), replaced the fully connected layers with fully convolutional layers, and applied a batch normalization layer
between weights and activation function. YOLOv3 [9] introduced a deeper backbone (Darknet53) and adopted a feature pyramid network (FPN) [21] that proposed a top-down pathway to combine multiscale features to increase the accuracy of small object detection. YOLOv4 [19] adopted CSPNet [22] with ResNext50 [23] or DarkNet53 and Spatial Pyramid Pooling Layer (SPP) [24], which reduces the pooling layer and improves the receptive field that helps distinguish significant context features. YOLOv4 [19] introduced two types of modifications: Bag of Freebies and Bag of Specials. Bag of Freebies are modifications that improve the network’s performance without adding inference time in production like cut-mix and mosaic data augmentation, drop-block regularization, cosine annealing scheduler, and self-adversarial training. Bag of specials are modifications that slightly impact the inference time and positively impact the overall accuracy of the model, such as Mish activation, DIoU-NMS, and modified SPP-block. Finally, SSD [15] was the first to introduce anchor boxes, applied multiscale prediction with different grid sizes, and replaced the idea of fully connected layers with a fully convolutional dense predictor head. It initially adopted VGG16 [25] as a feature extractor; however, with the rise of MobileNet [26] and the necessity of having a lightweight object detection network, SSD-MobileNet has become the best choice until today as one of the most computationally efficient models for CPU and embedded edge devices.

Two-stage detectors tend to have better accuracy but are more computationally expensive for training, and they are not suitable for edge CPU or embedded device deployment. In addition, because of the single-stage detectors’ efficiency in training and inference, innovation in this branch has stopped since the release of Faster RCNN [27] and its cascaded version [28].

B. OBJECT TRACKING

Object tracking can be classified into several categories. First, Single Object Tracking (SOT) and Multi-Object Tracking (MOT) are computer vision algorithms, where the first algorithm is used to track a single object in a video scene, and the latest specializes in tracking multiple objects in a video scene. MOT is required to track multiple vehicles across video scenes for traffic light scenes. Multi-object tracking can be divided into online [5], [29], [30] and offline tracking methods [31]–[33] based on whether tracking relies on future frames. Online tracking uses only previous and current frames. In comparison, offline tracking requires a sequence of previous, current, and future frames. Therefore, we focus on reviewing online multi-object tracking algorithms.

It starts with non-deep-learning algorithms, which assume that object detections are available and shift their focus on objects’ associations across frames. Object detection models are considered as black boxes in these methods, even the tracker performance is strongly reliant on detector performance. Kalman Filter [6] and IoU are often used in non-deep-learning trackers (SORT [34], IoU-Tracker [29]). These algorithms are computationally fast and straightforward for real-time tracking; however, they can not sustain an acceptable tracking accuracy in challenging situations such as occlusion or unsystematic object movement.

With the rise of deep learning, the appearance feature (embedding) has started to shine as a new contributor to the tracker’s performance and data association. Researchers have considered object detection and tracking (re-identification) as two separate tasks in DeepSORT [5] and SiameseCNN [35]. Single or two-stage object detectors such as YOLOv3 [9] or Faster RCNN [27] are applied to localize objects from an input image, followed by another CNN to extract the identification feature (embedding) of each cropped object. Embeddings are used to link objects across frames in addition to the traditional computing of IoU and the use of a Kalman Filter [6] as a motion model and Hungarian algorithm to accomplish the association task. The main advantage of separating the detection and re-identification networks is developing and optimizing each task separately. However, this approach leads to slow training time due to separate training for each task. Furthermore, this approach suffers from slow inference and can not achieve the real-time inference required in many applications due to its sequential execution.

In recent research, a joint detection and tracking network started to rise with the success of multitask learning in deep learning. It started with Tracktor [36], which adapts Faster RCNN [27] as a detector with a re-identification branch in predictors’ heads. Subsequently, CenterTrack [37] has introduced the advantages of using CenterNet [10] as a detector and it tries to estimate inter-frame offset bounding boxes using previous and current frames, it supports static single image training by generating a previous frame using augmentation methods (scaling and translating) on the current frame. Then, JDE [2] extended YOLOv3 [9] with a re-identification branch that jointly extracts objects’ features. Finally, FairMOT [4] enhanced JDE by replacing YOLOv3 with CenterNet object detector, which benefits from anchor-free architecture, and presented a single image training with the condition of pretraining the model on a large re-identification dataset (CrowdHuman [38]) in the same domain as the targeted dataset. Joint detection and tracking model runs on a real-time video inference rate. However, in the training phase, they require labeled video datasets that involve human efforts in labeling. Due to its numerous applications like traffic monitoring, analysis, and control, vehicle tracking is one of the most significant tasks in MOT. Tracking objects of varying sizes and viewpoints in diverse lighting settings and significant occlusions are considered obstacles in the vehicle tracking process. We are proposing a novel lightweight extension with an augmentation bag that extends SSD [15] and RetinaNet [39] with embedding branches that make them capable of learning detection, and tracking from a single frame without the necessity of re-identification labeling which leads to more adoption in real-world applications.
I. S. Mohamed, L. K. Chuan: PAE for Multiple Object Detection and Tracking in Traffic Scenes

II. PROPOSED METHOD

We present the technical implementation of our portable appearance extension, which could be added to several object detection networks to jointly extract re-identification appearance embeddings for detected objects and contribute to the tracker algorithm in occlusion scenarios or complex movement patterns. The main contribution of our proposed method is single image training with the help of the proposed augmentation bag that includes various augmentation techniques, which saves the annotation efforts of video sequences, therefore increasing the visibility of efficient and rapid adoption in real-world applications. Additionally, we apply hyperparameter optimization for tracking association algorithms that could efficiently be applied as a preprocessing on the training dataset.

A. AUGMENTATION BAG

We adopt a set of predefined augmentation techniques which apply randomly during training to help in generating synthetic batches from a single image. This technique simulates the effect of video frames on each batch, so we can train our appearance extension to distinguish and group objects according to their appearance and avoid high human labeling cost of the video sequence annotation; however, the augmentation module is removed during the inference stage.

Horizontal flipping, image padding, brightness adjustment, contrast adjustment, and random cropping are used as our augmentation bag space $\in AUG$ where each technique has its hyperparameters, by feeding a single input image $x^{b,w,h,c}$ to the augmenting module, a synthetic input batch $X^{b,w,h,c}$ where $b, w, h, c$ donate batch size, width, height, number of channels respectively are generated as shown in Fig 2. Each slice of our batch is a result of applying a combination of augmentation techniques $AUG$ randomly to the original training image.

B. OBJECT DETECTION

Our proposed PAE could be integrated with any object detection network with typical prediction heads consisting of classification and detection branches. Moreover, it works with either single-stage or two-stage networks and can be modified to work with anchor-free architectures. In this paper, we adopt two popular object detection networks.

1) SINGLE SHOT DETECTOR - SSD

Original SSD [15] architecture relies on VGG16 [25] as a feature extractor which suffers from high latency and computation power during inference and training. By replacing VGG16 with a more efficient and lightweight MobileNetV2 [26] backbone, real-time inference requirement is achieved with faster fine-tuning and training time. SSD is a single-stage object detector that directly predicts bounding boxes and classes using multiple prediction heads from multiscale feature maps independently, as shown in appendix A. In particular, six prediction heads are used to predict bounding boxes and classes. Each head is implemented by applying a $3 \times 3$ convolution neural network, followed by a $1 \times 1$ convolution layer which generates the final targets (classes scores, bounding boxes, appearance embeddings). In addition, default boundary boxes (Anchors) are proposed on each cell of the feature map grid at different scales, we use a total of 32 anchors divided as follows 4, 4, 6, 6, 6, 6 starting from the largest to smallest dimensional feature maps, respectively. Finally, NMS is applied to the concatenation of...
predictions from multiple scales to filter and limit our final predictions to top-100.

2) RETINANET

By adopting another single-stage object detector (Retinanet) with ResNet-50 as a backbone, we achieve a satisfactory balance between speed and accuracy compared to SSD, which suffers from a low accuracy in detecting small or dense objects. As shown in Appendix A, Retinanet adopts Feature Pyramid Network (FPN) to obtain rich semantics at multiple levels as it combines low-resolution semantically robust features with high-resolution semantically weak features, which increase its accuracy, especially on small objects. Furthermore, Retinanet introduced an enhancement version of Cross-Entropy Loss called Focal Loss, to handle the imbalanced class problem caused by foreground vs. dense background sampling of anchor boxes.

C. APPEARANCE EMBEDDING EXTENSION

Appearance embedding head focuses on learning well-discriminating features between different objects. Theoretically, the distance between similar objects should be smaller than the distance between different objects to help re-identify occluded objects during inference. We propose a lightweight CNN architecture as shown in Table 1 to be applied on extracted feature maps from the network backbone to obtain a 256 embedding vector per predicted anchor box. The raw output shape per scale before applying NMS is $F^{b,A,256}$ where $b$ and $A$ denote batch size and number of anchors, respectively.

| Name              | Parameters                                      |
|-------------------|-------------------------------------------------|
| Depthwise Convolution | 1x1, $s=1$, depth=1, dilation=1                      |
| BatchNorm          | decay=0.97, epsilon=0.001                        |
| Relu               | max=6                                           |
| Dropout            | rate=0.3                                        |
| Convolution        | 1x1, bias=True                                  |
| $l_2$ Normalization| -                                              |

D. EMBEDDING LOSS

$$L_i(r_a, r_p, r_n) = || r_a - r_p ||_2 - || r_a - r_n ||_2 + m$$  \(1\)

Following DeepSORT, we train our embedding head with triplet loss [40] to learn distributed features by the notion of similarity and dissimilarity of objects in the same batch. However, due to our methodology of using a single image for training, we learn the embedding through a distance optimization task compared to a classification task that requires a unique ID for each object in the dataset, as stated in DeepSORT. In Fig. 2, each batch $X^{b,w,h,c}$ contains a positive anchors $r_p$ which are represented by same object ID across multiple augmentation scenarios with the searched anchor $r_a$. Additionally, the negative anchors $r_n$ are other objects with different IDs compared to $r_a$. Adopted from the original triplet loss paper, the triplet loss shown in (1) demands a difference of the distance between $r_a$ and $r_n$ to be larger than a predefined margin $m \in \mathbb{R}$. This triplet loss (1) is added to the overall network loss.

E. TRAINING

Although the proposed network could be trained end-to-end, both SSD-PAE and RetinaNet-PAE are trained in two stages in our experiments. End-to-end training requires many hyper-parameters tuning during training to weigh each task’s loss $L_{loc}$, $L_{cls}$, $L_{emb}$ separately according to training iterations. Since the embedding loss is not useful during the beginning of training, particularly before the converging of $L_{loc}$, $L_{cls}$ since all predicted boxes will have random distance scores; therefore, we first train the object detector model, and secondly, we appended our embedding head with the first stage’s weights for fine-tuning the model and training of embedding head. Instead, we could freeze the object detector weights and train the embedding head only in certain cases.

F. ONLINE INFERENCE AND ASSOCIATION

The augmentation module is removed during inference, and NMS is applied to prediction heads with a 0.6 IOU threshold and maximum detection of 100 objects. The model outputs a classification, confidence, detection box and embedding feature predictions with shapes of $(b, 100)$, $(b, 100)$, $(b, 100, 4)$ and $(b, 100, 256)$ respectively where $b$ donates the batch size. The tracker is updated with the detection outputs for each frame in real-time. Following DeepSORT, the cosine distance between embedding vectors is measured to associate activated tracklets and objects, subsequently using the Hungarian algorithm to accomplish the association task. Moreover, Kalman Filter is used to predict tracklet locations in the upcoming frame and compute the squared Mahalanobis distance between detections using its center position and state distributions, to maintain the confusion of matching objects with similar appearance embedding features.

G. TRACKER HYPERPARAMETERS OPTIMIZATION

Tracker algorithm with its associations used in DeepSORT has various numbers of hyperparameters as shown in Table 2, which impact the overall tracking performance. By optimizing these hyperparameters on a sample sequence of the training dataset, we believe we could achieve a better tracking accuracy using the same detection model. This tuning requires a labeled video sequence, and it could be applied once to every new dataset or task.

IV. EXPERIMENTS

A. DATASET

University at Albany DEtection and TRACking (UA-DETRAC) [12] dataset is used in training and evaluating our MOT models. UA-DETRAC is a challenging real-world multi-object detection and tracking benchmarking dataset. It contains 100 real-world traffic scene videos captured using Canon EOS 550D camera. The data were collected from...
TABLE 2. Tracker hyperparameters with default values chosen from DeepSORT and proposed optimization range. Hyperparameter description is stated in Appendix B.

| Hyperparameter              | Default | Optimization Range   |
|-----------------------------|---------|----------------------|
| Distance metric             | cosine  | euclidean, cosine    |
| Matching Threshold          | 0.3     | 0.1-0.9              |
| Matching Budget             | None    | 0-10                 |
| Max IOU Distance            | 0.7     | 0.3-0.9              |
| Tracker Max Age             | 30      | 1-50                 |
| Tracker N Init              | 3       | 1-10                 |
| Position Weight             | 20      | 1-200                |
| Velocity Weight             | 160     | 1-400                |

TABLE 3. UA-DETRAC training and testing dataset details.

|                | Frames | Bounding Boxes | Unique IDs |
|----------------|--------|----------------|------------|
| Training       | 83,791 | 598,281        | 5,983      |
| Testing        | 56,340 | 675,774        | 2,353      |
| Total          | 140,131| 1,274,055      | 8,336      |

24 different locations in China to represent a variety of traffic conditions, including highways, cross junctions, and normal urban roads, besides the extensive variations of vehicle size (scale), camera view, illumination (cloudy, night, sunny, and rainy) and occlusion of vehicles. Low-resolution JPEG images of 960×540 pixels are used to capture 25 frames per second (FPS). The total number of frames exceeds 140,000, and 8250 unique vehicles are humanly annotated. Notably, this dataset contains 1.12 million labeled bounding boxes with IDs and four types of vehicles (car, bus, van, and others).

Training and testing splits of 60 and 40 sequences are made by the author of the dataset as shown in Table 3. UA-DETRAC annotations have ignored regions in each sequence that do not contain any annotations; these regions are colored with black boxes during training only. The testing dataset is divided by the author [12] into three subsets of easy (10 sequences), medium (20 sequences), and hard (10 sequences), which indicates the difficulties of detection and tracking in the recorded videos (sequences), a sample of each subset is shown in Fig. 3.

B. EVALUATION METRIC

Our models are evaluated with HOTA (Higher Order Tracking Accuracy) [13], [41] metric in the testing dataset. HOTA solved the problem of overemphasizing the importance of either detection or association that previous MOT metrics suffer from. In particular, MOTA overemphasizes the impact of accurate detection over an accurate association. Although HOTA provides a single evaluation score for the tracker, which eases the final evaluation process and provides the balance between detection and association, it still could be decomposed into sub-metrics scores which measure the performance of different tracker components. HOTA combines detection and association IoU scores.

\[
\text{DetA}_\alpha = \frac{|TP|}{|TP| + |FN| + |FP|} \tag{2}
\]

\[
\text{AssA}_\alpha = \frac{1}{|TP|} \sum_{c \in TP} A(c) \tag{3}
\]

\[
A(c) = \frac{|TPA(c)|}{|TPA(c)| + |FNA(c)| + |FPA(c)|} \tag{4}
\]

\[
\text{HOTA}_\alpha = \sqrt{\text{DetA}_\alpha \cdot \text{AssA}_\alpha} \tag{5}
\]

HOTA score is the average of the \( \text{HOTA}_\alpha \) scores calculated over different localization thresholds \( \alpha \). Both detection accuracy \( \text{DetA} \) and association accuracy \( \text{AssA} \) could be decomposed further to detection and association recall and precision \( \text{DetRe}, \text{DetPr}, \text{AssRe} \) and \( \text{AssPr} \) respectively. Besides HOTA, we benchmark our models based on detection and tracking time with overall FPS using CPU and GPU, and we report the memory allocation required during deployment.

C. IMPLEMENTATION DETAILS

Our experiments are conducted in an environment setup with the following hardware capabilities: Intel Xeon CPU E5-2620, 47GB of RAM, and Single NVIDIA GTX 1080 Ti. Our environment is powered by Ubuntu 16.04 LTS OS. Tensorflow v2.7 with CUDA v11.3 and cuDNN v8.2.1 is used for our training process, and ONNXRuntime v1.10 is used during inference and speed benchmarking.

Our models are trained using a customized version of Tensorflow Object Detection API [42]. Each training starts with a pretrained model on MS COCO dataset and uses a momentum optimizer with a cosine decay of 0.0025 base learning rate, 1000 warm-up steps with a learning rate of 0.0005, and momentum of 0.9. First, we train a base model without applying any technique from our augmentation bag for 250,000 steps. Second, we train another model by applying our augmentation bag with the parameters listed in Table 4 for same...
number of steps. Finally, we fine-tune the second model with our proposed appearance embedding extension for 100,000 steps with the frozen parameters of the full model’s backbone and prediction heads, except for the embedding extension. Our embedding for each anchor is one-dimensional tensor with a shape of 256, and the margin $m$ used in calculating the triplet loss is 0.7.

**TABLE 4. Augmentation bag parameters during training.**

| Technique            | Parameters |
|----------------------|------------|
| Horizontal flipping  |            |
| Image padding        | $max = 200$|
| Brightness adjustment| $max_d = 0.1$|
| Contrast adjustment  | $min_d = 0.9, max_d = 1.1$|
| Random cropping      | $min_a = 0.75, max_a = 1.0$|

SSD-PAE is trained with a batch size of 32, and the input image is resized to 300 × 300, while RetinaNet-PAE is trained with a batch size of 8 due to GPU memory limitation and larger input image of 640 × 640. The total training time of 11 and 28 is taken for SSD-PAE and RetinaNet-PAE, respectively. For deployment and speed benchmarking, TF saved models are converted to ONNX format with opset v12.

OPTUNA [43] is used with a single sequence of training dataset for tracker’s hyperparameter optimization with the ranges stated in table 2. The overall objective of this optimization process is to increase the average HOTA score of the testing dataset. MVI-20011 training sequence is used with 1000 number of trials.

**TABLE 5. Main ablation study of our proposed method’s components.**

| Method         | $Emb$ | $Aug$ | $Opt$ | HOTA |
|----------------|-------|-------|-------|------|
| SSD-SORT       | ✓     | ✓     |       | 42.33|
| SSD-PAE        | ✓     | ✓     | ✓     | 43.14|
| SSD-PAE        | ✓     | ✓     | ✓     | 47.16|
| SSD-PAE        | ✓     | ✓     | ✓     | 47.36|

In this section, our results are divided into three main parts. First, we present the evaluation of our proposed method compared to current state-of-the-art methods using the full HOTA metric. Then we present the run-time analysis of each method on CPU and GPU, including detection, embedding, and tracking time. Finally, by using UA-DETRAC-test subsets (Easy, Medium, and Hard), we evaluate each subset independently to investigate the downsides of each method.

Table 6 shows an improvement in performance between SSD-SORT and SSD-DeepSORT by 2%, which could be explained as appearance embedding’s contribution in the tracker. SSD-PAE outperforms SORT and DeepSORT extensions on the same training dataset and object detection models. This improvement is scored by the combination of utilizing our augmentation bag with an embedding head and postprocessing hyperparameter tuning of the tracker. JDE suffers from a low HOTA score due to miss detection of small objects, and the HOTA score is improved by increasing the JDE input size to 864 × 480. Finally, our state-of-the-art model RetinaNet-PAE surpasses both SORT and DeepSORT evaluation using the same object detection model (RetinaNet) with a 2% higher HOTA score. We can notice from HOTA, DetA, AssA, and other metrics that detection is the main factor in achieving good tracking accuracy. If we have a poor object detection model that misses small objects or loses detection every frame, the object will not be added to the tracker space, and the comparison between SSD and RetinaNet using the same tracking methods is an explicit confirmation.

In terms of training time and speed performance, Table 7 presents detection, embedding extraction, and tracking speeds in milliseconds, followed by the overall FPS in CPU and GPU, the memory required for deployment, and finally, the training time in hours for each model. SSD-SORT scores the highest FPS because it does not rely on appearance embedding extraction or distance calculation between embeddings. Comparing DeepSORT and our PAE, we observe double FPS performance improvement on GPU and 4 FPS faster on CPU. On the other hand, RetinaNet-based method has a low FPS but a high HOTA score. JDE suffers in CPU performance but achieves a great FPS on GPU. In conclusion, SSD-based models achieve real-time FPS with slightly less accurate results, while RetinaNet-based models suffer from low FPS but achieve outstanding HOTA scores. In terms of memory usage, it is noticeable that DeepSORT models require higher memory due to the loading of object detection models, either SSD or RetinaNet, and DeepSORT embedding model.

By testing all methods on UA-DETRAC-test subsets (easy, medium, and hard), Table 8 presents the impact of using appearance embedding features in tracking. Furthermore, by studying SORT-based models, suffering on medium and hard subsets is apparent compared to other methods as a consequence of ID switching on tracking caused by occluded objects and losing detection at some frames during the inference of moderate object detection models such as SSD. On the other hand, RetinaNet has an accurate detection capability which may overcomplicate the tracking task by embedding voting in DeepSORT and PAE, where SORT has shown better performance in the easy subset.

**VI. ABLATION STUDY**

In this section, we evaluate the three main components of our proposed method. First, Table 5 presents the results of using SSD with SORT tracking algorithm, which does not use objects’ appearance features in tracking. Then, by adding appearance embeddings with batch size equal to 1 (no augmentation), the HOTA score is increased by 0.81 percent, which could be explained that the embedding branch is learning shallow features since there are few positive and negative samples in every single frame. Following by enabling our
proposed augmentation bag with the augmentation module, the HOTA score has increased with a large margin of 4.02 percent, proving that the embedding branch can learn discriminative features for each object when the number of negative and positive samples is large enough. Finally, by applying hyperparameter optimization for the tracker, we scored 0.2 higher than our previous score.

A. DROPPING FRAMES DURING TRAINING

In this section, we would like to measure the impact of dropping 4 out of each 5 frames during training. Our experiment starts by training both models using a full training set as normal object detector models without embedding head for 250,000 iterations. Then, we attached the embedding extension for both models and freeze their backbone and prediction heads except the embedding branch to start the fine-tuning process for 100,000 iterations. Table 9 shows a slight increase in the HOTA score when training with the full dataset; however, the gain of training with a large training dataset for the embedding is not that significant.

B. DROPPING FRAMES DURING INFERENCE

MOT is an essential component in video analysis, and balancing between speed and accuracy is critical in real-world applications. This section measures the impact of dropping frames during inference using our proposed models (SSD-PAE and RetinaNet-PAE). Streaming or capturing 25 to 60 frames per second raises the following question: Do we need to infer all frames to track objects and understand the video context efficiently? Table 10 shows that doubling our inference speed is possible by dropping 1 out of each 2 frames without yielding much to the HOTA score, approximately 1% only. This approach can lead to a real-time inference for many edge devices; however, we need to consider each application separately to accomplish a trade-off between HOTA and FPS. For example, SSD-PAE can achieve the same
FIGURE 4. Qualitative results of our RetinaNet-PAE model on the UA-DETRAC-test subsets (0.5 object detection confidence threshold is used for all the videos except 0.4 for MVI-39271). The frames are selected with a time interval of 25 frames. Vehicle ID is located at bottom left corner of each bounding box.
TABLE 11. Comparison of DeepSORT embedding models trained on different datasets and evaluated on UA-DETRAC dataset.

| Method       | Training Dataset | HOTA  |
|--------------|------------------|-------|
| SSD-DeepSORT | MARS [44]        | 43.91 |
| SSD-DeepSORT | UA-DETRAC [12]   | 44.20 |

FPS as the SORT algorithm by dropping 2 out of 3 frames while maintaining a 4% higher HOTA score. RetinaNet-PAE using CPU can achieve nearly the same FPS as GPU by dropping 2 out of 3 frames. However, dropping more than two frames may cause a noticeable decrease in the HOTA score, as shown in the 1/5 and 1/9 examples. In this experiment, we filtered the groundtruth of the testing dataset by dropping the annotation of skipped frames to have a fair evaluation.

C. DEEP SORT EMBEDDING MODEL

Although DeepSORT [3] uses a pre-trained embedding model on MARS [44] pedestrians dataset, we retrain its metric feature representation model with UA-DETRAC for a fair comparison with our models. First, UA-DETRAC annotation is converted to MARS format and trained as a classification task for 220,000 steps. Then we freeze the final checkpoint and used it to report DeepSORT performance. As shown in Table 11, the final HOTA score has not significantly impacted when training DeepSORT appearance model on UA-DETRAC dataset. This result indicates that the main purpose of the embedding model is to learn to extract a strong feature representation for each object, which can be achieved using different datasets. Its performance can be slightly improved by training on the same domain or task.

VII. CONCLUSION

Multi-object detection and tracking is a critical problem in the computer vision field. The study in this field is enhancing rapidly to help adopt real-world applications in video analysis and many other applications. However, we require a large dataset with much human effort in labeling to cope with this rapid adoption. In this study, we propose a novel joint object detection and appearance embedding model that learns embedding extraction from a single frame by applying augmentation techniques to help the tracker without requiring re-identification annotation.

Our proposed portable appearance extension (PAE) could be implemented in most object detection models. In our experiment, we implemented it with SSD and RetinaNet models, and we can achieve comparable results with current MOT systems, reducing runtime and memory usage in deployment. In the future, we shall investigate the impact of replacing our augmentation module with a generative adversarial network (GAN) [45], [46] to help mimic an object’s movement to maintain the training with a single image approach.

APPENDICES

A. SSD AND RETINANET ARCHITECTURES

See Fig. 5.

FIGURE 5. (a) Demonstration of SSD convolution architecture which has a prediction head on each scale, however, it doesn’t reuse the multiscale feature maps from different layers compared to RetinaNet (b). (b) Demonstration of RetinaNet contribution of applying FPN architecture to reuse multiscale feature maps from different layers. The number of prediction heads in this figure is for illustration only.

TABLE 12. Tracker hyperparameters description.

| Parameter         | Description                                                                 |
|-------------------|-----------------------------------------------------------------------------|
| Distance metric   | A nearest neighbor distance metric types which return the closest distance between each target and any sample that has been observed in the scene. |
| Matching Threshold| Samples with larger distance than the matching threshold are considered an invalid match. |
| Matching Budget   | If not none, fix samples per class to at most this number. Removes the oldest samples when the budget is reached. If None, we will keep adding to the samples (offline tracker only). |
| Max IOU Distance  | Gating threshold. Associations with cost larger than this value are disregarded. |
| Tracker Max Age   | Maximum number of missed misses before a track is deleted. |
| Tracker N Init    | Number of consecutive detections before the track is confirmed. The track state is set to ‘Deleted’ if a miss occurs within the first \( n_{\text{init}} \) frames. |
| Position Weight   | This weight controls the amount of uncertainty in the model corresponding to the box center position. |
| Velocity Weight   | This weight controls the amount of uncertainty in the model corresponding to the object velocity. |

B. TRACKER HYPERPARAMETERS

See Table 12.

C. HOTA ANALYSIS USING DIFFERENT LOCALIZATION THRESHOLD \( \alpha \)

See Figure 6.
D. JDE TRAINING ON UA-DETRAC

JDE\textsuperscript{576} and JDE\textsuperscript{864} are trained on the UA-DETRAC dataset from scratch for 30 Epochs. Anchor boxes are edited from pedestrian aspect ratio to 1:1 aspect ratio to fit the UA-DETRAC dataset, as stated in one of the GitHub issues at the JDE repository. We have followed the author’s instructions for training as stated in the paper.

FIGURE 6. HOTA analysis using different localization threshold $\alpha$.

TABLE 13. Results referenced from [1] using HOTA and FPS on UA-DETRAC dataset.

| Method         | HOTA | FPS  |
|----------------|------|------|
| RCT [1]        | 39.49| 4.08 |
| KCF [47]       | 55.87| 20.68|
| MEDFLOW        | 53.87| 2.51 |
| DAN [48]       | 39.80| 5.08 |
| AOA [49]       | 31.14| 10.40|
| VIoU [50]      | 42.81| 5.41 |
| JPDA           | 41.47| 17.30|

E. OTHERS’ RESULTS AT FIG. 1

Fig. 1 includes some results marked with “others” name, these results are taken from [1] and its FPS results are not benchmarked on our GPU device and do not include the inference time of the object detection model in calculating FPS.

ACKNOWLEDGMENT

This work was supported in part by LED Vision Sdn. Bhd. and Universiti Teknikal Malaysia Melaka (UTeM).

REFERENCES

[1] T. Mandel, M. Jimenez, E. Risley, T. Namamoto, R. Williams, M. Panoff, M. Ballesteros, and B. Suarez, “Reliable multi-object tracking in the presence of unreliable detections,” 2021, arXiv:2112.08345.
[2] Z. Wang, L. Zheng, Y. Liu, and S. Wang, “Towards real-time multi-object tracking,” in Proc. Eur. Conf. Comput. Vis. (ECCV), 2020, pp. 107–122.
[3] N. Wojke and A. Bewley, “Deep cosine metric learning for person re-identification,” in Proc. IEEE Winter Conf. Appl. Comput. Vis. (WACV), Mar. 2018, pp. 748–756.
[4] Y. Zhang, C. Wang, X. Wang, W. Zeng, and W. Liu, “FairMOT: On the fairness of detection and re-identification in multiple object tracking,” Int. J. Comput. Vis., vol. 129, no. 11, pp. 3069–3087, Nov. 2021.
[5] N. Wojke, A. Bewley, and D. Paulus, “Simple online and realtime tracking with a deep association metric,” in Proc. IEEE Int. Conf. Image Process. (ICIP), Sep. 2017, pp. 3645–3649.
[6] R. E. Kalman, “A new approach to linear filtering and prediction problems,” Trans. ASME, D, J. Basic Eng., vol. 82, no. 1, pp. 35–45, 1960.
[21] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, “Feature pyramid networks for object detection,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., Jul. 2016, pp. 2117–2125.

[22] C.-Y. Wang, H.-Y. M. Liao, I.-H. Yeh, Y.-H. Wu, P.-Y. Chen, and J.-W. Hsieh, “CSPNet: A new backbone that can enhance learning capability of CNN,” in Proc. CVPR, 2019, pp. 390–391.

[23] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, “Aggregated residual transformations for deep neural networks,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), 2017, pp. 1492–1500.

[24] K. He, X. Zhang, S. Ren, and J. Sun, “Spatial pyramid pooling in deep convolutional networks for visual recognition,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 37, no. 9, pp. 1904–1916, Jan. 2014.

[25] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” 2014, arXiv:1409.1556.

[26] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, “MobilenetV2: Inverted residuals and linear bottlenecks,” in Proc. Conf. Comput. Vis. Pattern Recognit., 2018, pp. 4510–4520.

[27] S. Ren, K. He, R. Girshick, and J. Sun, “Faster R-CNN: Towards real-time object detection with region proposal networks,” in Proc. Adv. Neural Inf. Process. Syst., 2015, pp. 1–9.

[28] Z. Cai and N. Vasconcelos, “Cascade R-CNN: Delving into high quality object detection,” in Proc. Conf. Comput. Vis. Pattern Recognit., 2017, pp. 6154–6162.

[29] E. Bochinski, V. Eiselein, and T. Sikora, “High-speed tracking-by-detection without using image information,” in Proc. 14th IEEE Int. Conf. Adv. Video Signal Based Surveillance. (AVSS), 2017, pp. 1–6.

[30] Y. Xiang, A. Alahi, and S. Savarese, “Learning to track: Online multi-object tracking by decision making,” in Proc. Int. Conf. Comput. Vis. (ICCV), 2015, pp. 4705–4713.

[31] A. Milan, S. Roth, and K. Schindler, “Continuous energy minimization for multitarget tracking,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 36, no. 1, pp. 58–72, Jan. 2014.

[32] L. Wen, W. Li, J. Yan, Z. Lei, D. Yi, and S. Z. Li, “Multiple target tracking based on undirected hierarchical relation hypergraph,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2014, pp. 1282–1289.

[33] W. Choi, “Near-online multi-target tracking with aggregated local flow descriptor,” in Proc. Int. Conf. Comput. Vis. (ICCV), 2015, pp. 3029–3037.

[34] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft, “Simple online and realtime tracking,” in Proc. Int. Conf. Image Process., 2016, pp. 3464–3468.

[35] L. Leal-Taixé, C. C. Ferrer, and K. Schindler, “Learning by tracking: Siamese CNN for robust target association,” in Proc. Conf. Comput. Vis. Pattern Recognit. Workshops, 2016, pp. 33–40.

[36] P. Bergmann, T. Meinhardt, and L. Leal-Taixé, “Tracking without bells and whistles,” in Proc. Int. Conf. Comput. Vis., 2019, pp. 941–951.

[37] X. Zhou, V. Koltun, and P. Krähenbühl, “Tracking objects as points,” in Proc. Eur. Conf. Comput. Vis., 2020, pp. 474–490.

[38] S. Shao, Z. Zhao, B. Li, T. Xiao, G. Yu, X. Zhang, and J. Sun, “CrowdHuman: A benchmark for detecting human in a crowd,” 2018, arXiv:1805.00123.

[39] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, “Focal loss for dense object detection,” in Proc. Int. Conf. Comput. Vis., 2017, pp. 2980–2988.

[40] E. Hoffer and N. Ailon, “Deep metric learning using triplet network,” Tech. Rep., 2014.

[41] A. H. J. Luiten. (2020). Trackeval. [Online]. Available: https://github.com/JonathonLuiten/TrackEval

[42] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama, and K. Murphy, “Speed/accuracy trade-offs for modern convolutional object detectors,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., Jul. 2016, pp. 7310–7311.

[43] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, “Optuna: A next-generation hyperparameter optimization framework,” in Proc. 25rd Int. Conf. Knowl. Discovery Data Mining, 2019, pp. 2623–2631.

[44] MARS: A Video Benchmark for Large-Scale Person Re-identification. Springer, 2016.

[45] A. Antoniou, A. Storkey, and H. Edwards, “Data augmentation generative adversarial networks,” 2017, arXiv:1711.04340.

[46] S. D. Wickramaratne and M. Mahmood, “Conditional-GAN based data augmentation for deep learning task classifier improvement using FNIRS data,” Frontiers Big Data, vol. 4, Jul. 2021, Art. no. 659146.

[47] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista, “High-speed tracking with kinetized correlation filters,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 37, no. 3, pp. 583–596, Mar. 2015, doi: 10.1109/TPAMI.2014.2345390.

[48] S. Sun, N. Akhtar, H. Song, A. Mian, and M. Shah, “Deep affinity network for multiple object tracking,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 43, no. 1, pp. 104–119, Jul. 2018.

[49] M. A. G. Al-Sadoon, R. Asif, Y. I. A. Al-Yasir, R. A. Abd-Alhameed, and P. S. Excell, “AOA localization for vehicle-tracking systems using a dual-band sensor array,” IEEE Trans. Antennas Propag., vol. 68, no. 8, pp. 6330–6345, Aug. 2020.

[50] E. Bochinski, T. Senst, and T. Sikora, “Extending IOU based multi-object tracking by visual information,” in Proc. 15th IEEE Int. Conf. Adv. Video Signal Based Surveillance. (AVSS), Nov. 2018, pp. 1–6.

IBRAHIM SOLIMAN MOHAMED was born in Cairo, Egypt, in 1997. He received the B.S. degree (Honors) in computer engineering from the University Teknikal Malaysia Melaka, in 2019, where he is currently pursuing the M.S. degree.

From 2017 to 2019, he has engaged in various artificial intelligent projects at the Machine Learning Laboratory, Faculty of Electronics and Computer Engineering, such as vehicle tracking, license plate detection and recognition, and traffic controller design. Since 2019, he has been working as a Research and Development Engineer at Vitrox, Malaysia, in deep learning researches and models’ optimization and deployment. His research interests include supervised vehicle detection and tracking for adaptive traffic light controller based on reinforcement learning.

LIM KIM CHUAN received the B.Eng. degree in electronic systems engineering, the M.Sc. degree in electronics & IT, and the Ph.D. degree in machine vision and image processing from Sheffield Hallam University, U.K., in 2003, 2005, and 2009, respectively. He is currently an Associate Professor with the Department of Computer Engineering, Faculty of Electronics and Computer Engineering, Universiti Teknikal Malaysia Melaka.