Journal of the Brazilian Computer Society, 2026, 32:1, doi: 10.5753/jbcs.2026.4636

OneTrack-M: A Multitask Approach for Transformer-Based MOT Models

Luiz Carlos Silva de Araujo [**Federal University of Amazonas** | ***luizcls@icomp.ufam.edu.br***]
Carlos Mauricio Seródio Figueiredo [**State University of Amazonas** | ***cfigueiredo@uea.edu.br***]

Institute of Computing, Universidade Federal do Amazonas, Av. General Rodrigo Octavio Jordão Ramos, 1200 - Coroado I, Manaus - AM, 69067-005, Brazil.

Received: 25 June 2025 • **Accepted:** 05 December 2025 • **Published:** 27 March 2026

Abstract Multi-Object Tracking (MOT) is a critical problem in computer vision, essential for understanding how objects move and interact in videos. This field faces significant challenges such as occlusions and complex environmental dynamics, impacting model accuracy and efficiency. While traditional approaches have relied on Convolutional Neural Networks (CNNs), the introduction of transformers has brought substantial advancements. This work introduces OneTrack-M, a transformer-based MOT model that enhances tracking computational efficiency and accuracy. Our approach introduces the transformer encoder as the model backbone, significantly reducing processing time and increasing inference speed. Additionally, we employ innovative data preprocessing and multitask training techniques to address occlusion and diverse objective challenges within a single set of weights. Experimental results demonstrate that OneTrack-M achieves at least 25% faster inference times compared to state-of-the-art models in the literature while maintaining or improving tracking accuracy metrics. These improvements highlight the potential of the proposed solution for real-time applications such as autonomous vehicles, surveillance systems, and robotics, where rapid responses are crucial for system effectiveness.

Keywords: MOT, Multiple Object Tracking, Transformers, Fast Tracking, End2End, Unified Model

1 Introduction

Computer vision simulates the complex nature of human visual perception, enabling machines to interpret visual data. This field continuously strives to replicate the human ability to process and understand images, leading to the development of advanced technologies. Significant progress has been driven by algorithmic enhancements and powerful hardware, such as advanced CPUs and GPUs, which facilitate the processing of large data volumes and the execution of complex machine learning (ML) and deep learning (DL) models.

Among computer vision applications, Multi-Object Tracking (MOT) [Bashar *et al*., 2022] stands out as crucial for understanding how objects move and interact in videos. This domain faces challenges like occlusions and complex environmental dynamics, traditionally addressed by a progression of models from feature-based methods to advanced ML algorithms.

In computer vision, Convolutional Neural Networks (CNNs) have been foundational for object detection and classification, tasks essential for MOT [Wojke *et al*., 2017]. CNNs efficiently capture visual features, from simple edges to complex patterns. However, the introduction of transformers has marked significant advancements, outperforming CNNs in tasks like object classification, such as the results in Dosovitskiy *et al*. [2021], and video processing, as in Liu *et al*. [2021b]. This is due to the transformers' ability to model long-range relationships in data, making them ideal for understanding contexts and temporal relationships between objects.

Originally developed for natural language processing, transformers also have shown substantial potential in MOT applications, offering new possibilities for analyzing and interpreting complex image sequences. Despite their promise, transformer models still face challenges related to efficiency, both computationally and in tracking accuracy, prompting the search for innovative solutions.

Computational efficiency remains a common challenge in MOT, encouraging the exploration of advanced ML/DL techniques combined with hardware optimizations to enhance tracking accuracy while reducing computational resource requirements. Some of the main works that use transformers in MOT have major speed issues, such as Zeng *et al*. [2022], Sun *et al*. [2021], and Meinhardt *et al*. [2022]. But not only them, as research in this field tends to favor accuracy over inference speed, meaning most models will find issues when applied to real-world scenarios where skipping frames can become an issue.

The need for reducing inference times in MOT systems relies on applications that depend on real-time processing, such as autonomous vehicles, which rely on rapid responses to avoid obstacles and prevent collisions, and surveillance systems, which require quick processing to detect anomalies and track individuals effectively. Swift model responses are crucial in robotics and other fields requiring real-time environmental interpretation by intelligent agents. This work introduces an approach to enhancing efficiency with transformers in MOT systems by leveraging transformer encoders as the model backbone for temporal data interpretation. Such an architecture can maintain system accuracy while improving inference speeds when compared to literature MOT models.

Additionally, we address a common issue in end-to-end models—very different objectives with a single set of weights being trained—by incorporating multitasking training techniques and concepts.

The contributions of this work are as follows:

- OneTrack-M represents a method to build and train such MOT models, now emphasizing achieving faster inference times.
- The training technique used is innovative, as most works will try to train the model by adjusting the weights of losses to form a composite loss.
- The channel-wise encoding method used was developed to reflect how our input data was modeled.

2 Related Works

This chapter covers the main works that influenced the decisions made in the OneTrack-M design.

2.1 Classical ML Approach to MOT

First, it is important to clarify that ”classical” refers to machine learning techniques that are not considered deep, such as Support Vector Machines (SVMs), K-nearest neighbors, K-means, and others. Although these techniques lack the generalization capabilities of deep methods, they still have value for solving simpler problems with less data and requiring shorter inference times.

In the case of MOT, which is a complex problem, the classical techniques that show good results are mainly those based on the Kalman filter, such as Bewley *et al*. [2016]. In this case, the Kalman filter provides temporal context for the system, working sequentially with any object detection model. The idea is to predict the future position from previous detections in a window of frames. From these values, the current detections are associated with the previous ones. More generally, particle filters are also an option to infer the trajectory of objects over time, as done in Jaward *et al*. [2006].

Another approach to this is optical flow to model the movement of each object over time, as in Su *et al*. [2019]. Optical flow techniques calculate the motion of objects by analyzing the differences in intensity between consecutive frames. This method helps in tracking by providing motion vectors, which can be used to predict the next positions of the objects, aiding in associating detections across frames. Optical flow is particularly useful in scenarios where objects move continuously and smoothly, making it a valuable tool in the MOT toolkit.

2.2 Deep Learning in MOT

Recent advances in deep learning techniques have revolutionized the field of Multi-Object Tracking (MOT), overcoming the limitations of traditional and classical approaches, particularly in challenging scenarios. Models like DeepSORT [Wojke *et al*., 2017], which combines the Kalman filter with features extracted by deep networks, exemplify the state-of-the-art in MOT, showcasing the superior generalization capabilities of these models.

Introducing deep learning into MOT broke down previous barriers, enabling various applications and innovations. In this context, Convolutional Neural Networks (CNNs) stand out for their ability to extract local information from images, widely adopted in deep learning models as shown in works such as Du *et al*. [2023], Zhang *et al*. [2022], and Meinhardt *et al*. [2022]. In these works, the convolutional part of the architecture is responsible for extracting image features that are processed in a subsequent stage.

In addition to spatial features, processing temporal sequences is essential for tracking objects in videos. Recurrent Neural Networks (RNNs) are used to maintain relevant information over time, while transformers, with their attention layers, are emerging as a robust approach to relate image sequences, as illustrated in Zeng *et al*. [2022] and Meinhardt *et al*. [2022].

Beyond temporal features, re-identification-based methods use objects' visual aspects to associate identities, as seen in Aharon *et al*. [2022] and Mostafa *et al*. [2022]. Although efficient, these methods can lose crucial temporal information, as they rely solely on the similarity of objects over different frames, rather than modeling for their movement or even both simultaneously.

CenterTrack [Zhou *et al*., 2020] introduced a new way of solving MOT. It innovates by rethinking how data is modeled, making the association process between frames a matter of tracking each centroid displacement across frames. This approach makes inference more direct and changes how objects are detected by using heatmaps. Thus, the model can learn all these parts in a unified way.

Recent work, such as Hybrid-SORT [Yang *et al*., 2024] and CuesTrack [Zheng, 2024] still applies object tracking techniques over centroid trajectories after an object detection phase. Hybrid-SORT combines the well established method of using a Kalman filter over the object centroid for trajectory prediction with a momentum calculation for its direction and velocity. CuesTrack achieved superior performance by applying a LSTM model to predicts the object's next position and movement direction.

Finally, it is essential to highlight that the progress of MOT is closely linked to the advances in object detection brought by deep learning. This synergy between detection and tracking improves existing models and paves the way for developing new techniques and approaches in MOT.

2.3 Image Models with Transformers

Transformers improved natural language processing, and now they play a crucial role in computer vision by providing innovative approaches to complex tasks. The Vision Transformer (ViT) [Dosovitskiy *et al*., 2021] adopts a novel strategy by splitting images into fixed-size patches, treating them as sequences of tokens, and applying attention layers to capture global dependencies. This approach contrasts with the local convolutions of CNNs, enabling ViT, when trained with large data and parameters, to outperform CNN models in image classification.

However, new transformer variants, such as the Swin Transformer [Liu *et al*., 2021a], advance even further in computer vision. The Swin Transformer excels in its ability to effi-

ciently model image hierarchies, adapting to different scales and contexts, making it particularly suitable for object detection and segmentation tasks. This model even serves as the basis for another state-of-the-art model in video classification: the Video Swin Transformer [Liu *et al*., 2021b].

In parallel, DETR (Detection Transformer) [Carion *et al*., 2020] represents another significant innovation, simplifying the object detection process by eliminating complex post-processing steps like Non-Maximum Suppression (NMS). DETR models object detection as a set prediction problem, using a combination of transformer encoder and decoder to generate direct detections without predefined anchors.

These innovations in transformer-based models demonstrate a significant technical advance and expand the possibilities for developing more efficient and accurate MOT systems. Using transformers in MOT promises a new wave of approaches that can better handle challenges such as occlusions and variations in object size, enhancing tracking system robustness and effectiveness.

Thus, transformers bring new techniques and methodologies to MOT while establishing a new paradigm in processing and interpreting visual data, paving the way for continuous innovations in multi-object tracking.

2.4 Transformers in Multi-Object Tracking (MOT)

Incorporating transformer-based models into Multi-Object Tracking (MOT) represents a paradigmatic shift, offering innovative and sophisticated approaches to traditional challenges in this field. One pioneering example is the TrackFormer model described in Meinhardt *et al*. [2022]. This model employs an encoder-decoder transformer architecture, similar to those used in Natural Language Processing (NLP) tasks, to track objects in video sequences. The distinctive aspect of TrackFormer lies in its tracking methodology: it continuously generates and updates tracking queries to maintain object identification across frames. This approach demonstrates transformers' ability to manage complex spatial and temporal information, maintaining tracking consistency without needing a dedicated object detection model. One key aspect of the model is that it does not forego the use of CNNs, as they are still used here for feature extraction before feeding this data into the transformer component of the model.

The MOTR model introduced by Zeng *et al*. [2022] represents another significant advancement. MOTR expands the concept of DETR to the MOT context, adopting an encoder-decoder alongside feature extraction with CNNs (Resnet50 in this case) to detect and track objects simultaneously. It uses tracking queries that update over time, similar to TrackFormer, but with a distinct implementation. This technique allows MOTR to handle object detection as a set-prediction problem, improving tracking accuracy, but at the cost of computing time. This innovation marks an important milestone in simplifying the MOT process and improving its effectiveness.

Furthermore, the development of TransTrack, as reported by Sun *et al*. [2021], also contributed significantly to MOT. TransTrack adopts a hybrid approach, integrating convolutions for feature extraction and a transformer for detection association. This hybrid model illustrates the potential of combining CNNs and transformers in MOT, effectively addressing tracking challenges in complex contexts and suggesting a promising direction for future research.

In the field of transformers-based solutions, a novel approach is presented by DiffusionTrack [Luo *et al*., 2024]. In this case, progressive denoising diffusion process elevates the tracker's effectiveness, enabling it to discriminate between various objects. Its idea is to enable the model to learn detection and tracking simultaneously from noise and ground-truth detection boxes.

The aforementioned models indicate substantial progress in MOT methods, demonstrating how transformers can reshape tracking strategies, leading to more accurate, efficient, and robust systems. Successful integration of transformers into MOT signals a considerable advance in computer vision, pointing to new possibilities for research and development. However, two common issues exists in all these models: the use of a complete transformer architecture (i.e., encoder and decoder) and very high inference times, making real-time usage impractical.

2.5 Limitations of Current Solutions

The main limitation found in most recent works, which have been a focus of development in the field, is the inference time. Works such as Mostafa *et al*. [2022] and Zhang *et al*. [2021] have simplified the system into a single model that performs both detection and tracking end-to-end. Both employ a CNN to extract features from the images and then perform MOT. However, they do not excel in metrics evaluation compared to other slower methods.

As mentioned above, these faster approaches use CNNs as the backbone of their solution. These methods present issues in both fields, metrics and speed, that are addressed by the solution proposed in OneTrack-M. This improvement is largely made possible by an observation made by Zhang *et al*. [2021], who describe an unfairness with respect to the training step of such end-to-end models. It is also noted by the lack of temporal awareness in the regular CNN structure, which in turn can be easily learned by attention layer-based models.

2.6 Multitask Training (MTL)

Regarding the effectiveness of multitask training, in Caruana [1997], the authors show that neural networks trained on multiple related tasks can achieve better performance in each task compared to isolated training, proving that the method can lead to better knowledge transfer and optimal weight selection to solve a multifaceted problem.

Some studies suggest that MTL can be effective in data-scarce scenarios since joint learning enables the model to use information from all tasks to improve its generalization capability. In Zhang and Yang [2017], the authors explore approaches and discuss how the method can overcome data scarcity.

This approach closely resembles how authors develop and train their MOT models, where an end-to-end model is commonly trained with all tasks simultaneously. However, Zhang *et al*. [2021] argues about the dissonance between learning

these tasks, which harms the overall learning of the model, such that training one task hinders the learning of another. The authors' solution was to use the joint optimization approach to solve this issue, with specific loss functions for each task. In our case, we propose a novel approach, using multi-step training, where the model is trained more than once, each time with one target task (first detection and then tracking), but all in the same model. More details about this approach are given in Section 3.6.1.

3 OneTrack-M Model

The development of OneTrack-M is significantly based on studies by Zhou *et al*. [2020] and Zeng *et al*. [2022]. The former establishes a reference model that uses object centers for tracking, while the latter serves as a benchmark for tracking approaches that integrate transformers. Both works provide a theoretical and methodological foundation for this project.

The OneTrack-M architecture incorporates the pre-trained encoder from the model in Dosovitskiy *et al*. [2021], which is used to extract features from image sequences and generate attention maps. This methodological choice leverages the Vision Transformer's effectiveness in understanding visual nuances present in image sequences to extract relevant features, which are then fed into multiple CNNs for learning the fine-grained tasks required for MOT. These specific layers are added on top of this model for key multi-object tracking (MOT) functions, detailed below:

- **Heatmap Head**: Identifies object centers in each image.
- **Dimension Head**: Estimates the dimensions (height and width) of detected objects.
- **Center Displacement Head:** Calculates the variations in object center positions between frames.

Figure 1 illustrates the complete architecture, showing the interconnection of these components. The following sections better describe each element, clarifying their functions and the role they play in the OneTrack-M model's overall context.

3.1 Preprocessing

The initial stage of our model involves specific data handling to present it appropriately to the neural network. The main challenge here is to provide temporal context between selected frames. This is done by stacking all images from a temporal window along the channel dimension. For a standard image with dimensions $[3, Width, Height]$, considering a window size W, the input format becomes $[3 * W, Width, Height]$. From these images, patches are generated and later converted into tokens via a linear layer. This methodology allows the neural network to fully absorb the temporal context, aligning with the preprocessing strategies adopted in previous works like Dosovitskiy *et al*. [2021], but adapting it for video analysis in this project. The components of this model section are illustrated in Figure 2. It is worth noting that no input is needed for the network besides the images, normalized and resized into a square matrix of size 224, which is the same as that used in the reference ViT model.

3.1.1 Image Stack and Patch Generation

According to the procedure described by the Vision Transformer authors, this stage consists of, given a square size (16x16 or 32x32), equally dividing the image into patches. This work maintained the 224x224 image size, again compatible with the experimental procedure of the previously mentioned base work. Thus, for 16x16 size patches, 196 small images are obtained $(\frac{224}{16})^2$. Each one passes through a linear layer to generate a vector projection of each patch, with a size of 768.

This stage becomes efficient because it uses a convolutional layer with an output of 768 channels, a kernel, and a stride matching the patch size. Thus, each convolution step processes a patch, either 16x16 or 32x32, without overlap, and then projects the layers to the desired size, meaning these square patches become a 768-sized vector. In Figure 1 these steps are illustrated in the Stacked Images, Stack Patch, and Linear Projection blocks.

3.1.2 Normalizations

In the context of tracking, which relies on analyzing object center displacements, and considering a video stream at 60 frames per second, a 5-frame temporal window represents only 83 milliseconds. The displacements observed between consecutive frames tend to show relatively small magnitudes in this short interval. To enhance the model's predictive ability given this characteristic, displacement values were normalized to the range $[-1, 1]$.

The normalization parameters were determined according to Milan *et al*. [2016], which establishes a benchmark for MOT. We have chosen the MOT17 dataset as the reference for displacements, and the specific values used in the normalization are detailed in Table 1, providing a quantitative reference for this crucial preprocessing stage. This practice not only makes data interpretation easier for the model but also ensures more efficient adaptation to subtle movement variations, enhancing tracking accuracy.

We expect that the chosen parameters should work for similar scenarios (as we show for MOT20 dataset), but it is important to note that if the model need to be adapted to very different datasets, it would be necessary to adjust the normalization values considering the new data displacement peculiarities.

Table 1. Displacements observed in the MOT17 training dataset [Milan *et al*., 2016].

Min. X	Max. X	Min. Y	Max. Y
-0.0174	0.0057	-0.0157	0.0166

3.2 Channel Wise Encoding

Despite the effort to incorporate temporal context, the model may face difficulties due to the lack of a clear indication of the sequential order of image patches. Therefore, providing explicit positional context is crucial to optimizing performance in attention-based models.

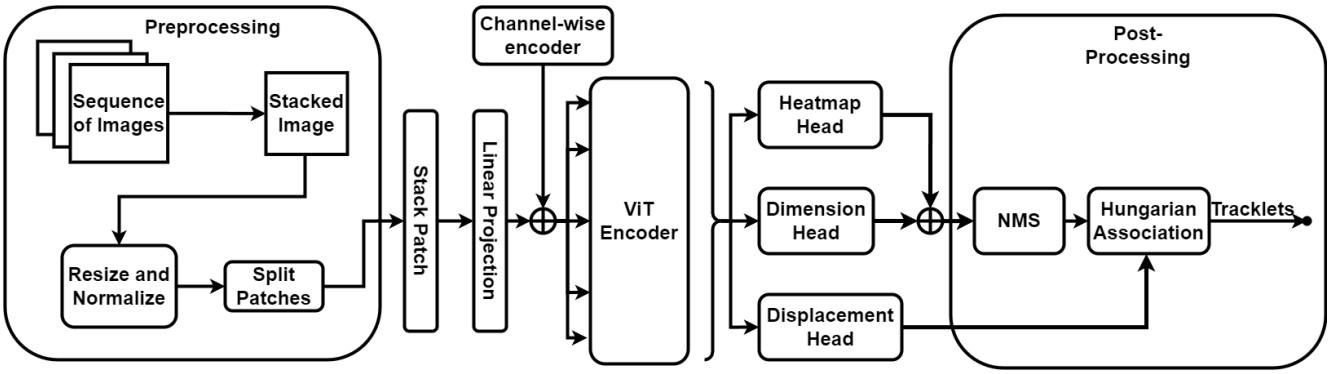

Figure 1. Implemented architecture for the OneTrack-M model.

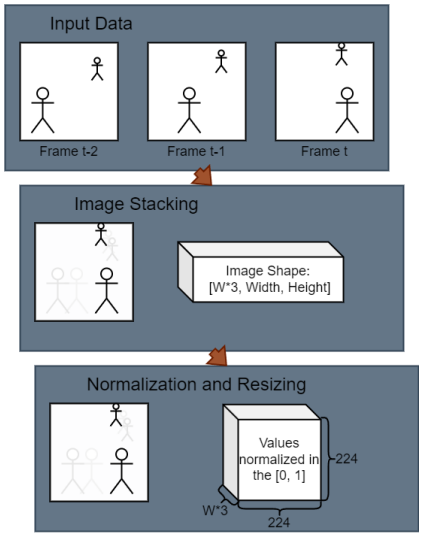

Figure 2. Detailed representation of preprocessing step for the model.

In this scenario, we opted for an innovative approach through the implementation of a technique called Channel Wise Encoding. This method does not limit itself to simple, fixed, or learned positional encodings, but seeks a more comprehensive representation of the data spatial and temporal structure. Specifically, it adds unique embeddings for each image patch channel, reflecting the temporal window in video processing.

Channel-wise encoding innovates by assigning a distinct embedding vector for each frame within the temporal window, distinguishing not only the spatial features but also the temporal position of each patch within the sequence. This approach is vital for the model to understand object movement and order, as the sequential image sequence provides crucial tracking information.

Initially, these encodings are a trainable parameter tensor of dimensions $[W, n_d]$, where W is the temporal window size and n_d is the embedding dimension. This vector is then added to the vector derived from the image patches. Consequently, this strategy not only facilitates differentiation between frames but also improves the model's ability to capture temporal dynamics. It represents an effective solution for integrating temporal information into transformer-based models designed for multi-object tracking, with the additional benefit of minimizing inference time impact.

These vectors are added to the linear projections of each patch obtained in the previous preprocessing stage.

3.3 The ViT Encoder

The strategy adopted by the model involves using the Vision Transformer encoder pre-trained on ImageNet [Murad and Pyun, 2017] to extract attention maps, converting each image patch into an input sequence element. Unlike the conventional application of ViT for classification, the classification token is omitted in this context. This modification adapts ViT to function as a feature extractor focused on understanding spatial and temporal context while retaining the original settings of the pre-trained network.

This approach is significantly different from traditional methods described in literature, which often employ an encoder-decoder architecture, as evidenced in works like Sun *et al*. [2021], Meinhardt *et al*. [2022], Zeng *et al*. [2022], and Zhang *et al*. [2023]. Although robust, these solutions face the challenge of long inference periods. By simplifying to just the encoder and transferring the responsibility of the final output to a more efficient component, OneTrack-M has managed to significantly reduce processing time.

The choice of a pre-trained network is justified by the large data volume required for effective training of transformer networks. The limitation of the available data in the benchmark dataset used, combined with concerns that including external data could compromise comparability between different models, reinforces the relevance of this decision. Thus, using the pre-trained ViT emerges as a pragmatic solution, allowing prior learnings to be leveraged without requiring extensive training sets, aligning with OneTrack-M's goals of efficiency and effectiveness.

3.4 Output Heads

OneTrack-M's output structure consists of three main components: heatmaps, dimensions, and displacements. As shown in Figure 3, each of these heads comprises a sequence of two fully connected layers followed by two convolutional layers, specifically activated for each output type. For heatmaps and object dimensions, sigmoid activation is used, while hyperbolic tangent activation is necessary for displacements to represent the $[-1, 1]$ value range possible for this output.

The heatmap processing stage that adjusts its format to that expected by the output heads is illustrated in Figure 4. For this stage, we will process the attention maps into a lower dimension using a series of fully connected layers, resulting

in a tensor of shape [batch, 196*W, 196], which gets reshaped into [batch, W, 196, 196]. This final tensor is the input used to perform inference in the three implemented heads.

This configuration aims to detect information grids, aligning with the output method proposed by CenterTrack [Zhou *et al*., 2020]. Differentiation lies in the occlusion handling: unlike the baseline, which is limited to two consecutive images, our approach stacks multiple frames for input. This allows the model to preserve the displacement context of objects for a longer period, even in the presence of occlusions.

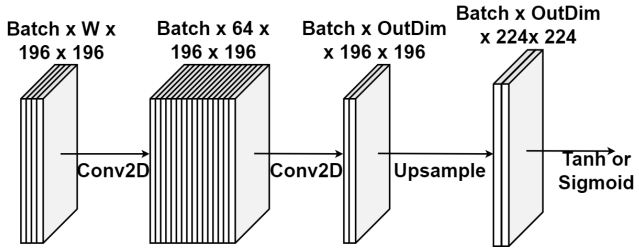

Figure 3. Detailed schematic of the network responsible for the last part of the model's heads. Each head has the same structure, varying only in activation function and output dimension. For heatmaps, $OutDim = 1$, while for the other two, $OutDim = 2$.

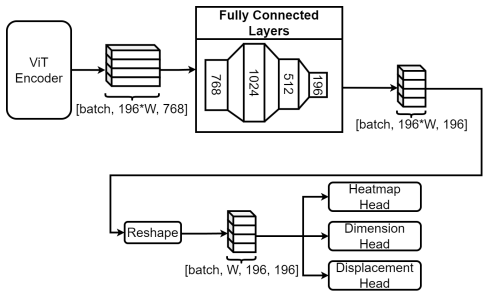

Figure 4. Output stage schematic after data is processed by Vision Transformer. The first tensor block represents attention maps extracted by the model. Each block is a sub-step of this final model stage.

3.4.1 Heatmap Head

The heatmap output identifies object centers, allowing unlimited object tracking—a significant advantage over other MOT approaches with transformers, like Zeng *et al*. [2022] and Zhang *et al*. [2023]. To manage complexity, the image dimension is reduced before inference and then expanded back to its original dimensions. This procedure is consistent across all model heads. The activation for this output is done by a sigmoid function, limiting values between 0 and 1.

Figure 5 compares a dataset sample with a model prediction, where each object is marked by a peak in the heatmap, smoothed by a Gaussian filter as shown by Zhou *et al*. [2020] and Zhou *et al*. [2019].

3.4.2 Dimension Head

This head estimates the dimensions (width and height) of each object using a grid of shape $[2, Width, Height]$. Each matrix element corresponds to an object's dimensions at its central location. Although the outputs are processed simultaneously, fine-tuning is necessary to integrate these results into the final inference. This output is activated by a sigmoid function to keep the range between 0 and 1. This head is similar to the one implemented in CenterTrack [Zhou *et al*., 2020].

Figure 5. Example of an image with heatmap detection and its peaks. A concentration of people is noticeable in the central region, with empty areas in the upper and lower regions, demonstrating good model capability.

3.4.3 Displacement Head

Working similarly to the Dimension Head but with different weights, this part of the model calculates object center displacements in the two image dimensions, considering the previous frame. Even with data from earlier instances, the regression is only made between the current and the immediate previous moment. Considering the displacement can be in any direction, this head is activated by a hyperbolic tangent function, making the range of values -1 and 1.

Figure 6 presents the model's complete output, including object centers, dimensions, and displacements. These displacements are essential for associating objects to specific trajectories, enabling effective tracking over time.

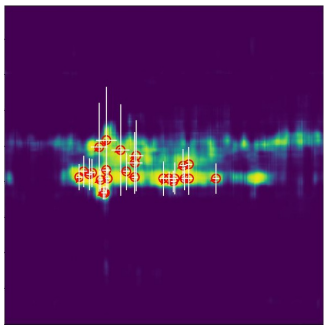

Figure 6. Complete model output image. Here you can see the size of each detected object (white bars) and the movement direction (red arrows) from their centers.

3.5 Post-Processing and Data Association

Each model head represents a piece of the complete MOT output. However, two additional steps are required to fully build the tracking routine. The first involves processing and uniting all model outputs, while the second employs a common approach to associate current inferences with the object's tracking history.

3.5.1 Combining Outputs

To integrate the model outputs, a specific procedure is followed:

1. **Apply a Threshold on the Heatmap:** This step identifies the detected object centers by applying a threshold to the heatmap.
2. **Determine Object Dimensions and Displacements:** Using the identified centers, the object's dimensions and displacements are determined.

In Figure 5 we see the raw outputs of each detection found by the model in a given frame. It is to possible to see that some of these detections are either overlapping or are not precise. This is addressed by applying the Non-Maximum Suppression (NMS) algorithm to mitigate false positives, by suppressing overlapping and low-confidence detections. This method is widely used in object detection systems, including YOLO variants such as Wang *et al*. [2022].

Given a set of bounding boxes B with their respective confidence scores S and an overlap threshold T, the NMS algorithm proceeds as follows:

1. **Select the Box with the Highest Confidence Score** b_{max}: Select bounding box b_{max} with the highest confidence score s_{max} from S.
2. **Compare with Other Boxes** b_i: Compare b_{max} with each box b_i in B, except itself.
3. **Calculate Intersection over Union (IoU):** For each comparison, calculate the Intersection over Union between b_{max} and b_i, denoted by $IoU(b_{max}, b_i)$.
4. **Suppression:** If $IoU(b_{max}, b_i) > T$, box b_i is considered redundant and removed from B.
5. **Repeat:** Repeat steps 1 to 4 until all boxes in B are evaluated or removed.

Figure 7 illustrates predictions before and after NMS, demonstrating the algorithm's effectiveness in reducing false positives. Image (a) shows detected objects immediately after the image is processed by the model, while image (b) shows after the redundancy reduction algorithm.

Figure 7. Image (a) shows the detections over the image before applying NMS step. Image (b) shows how the detections become after applying NMS.

To track detected objects, current detections must be associated with previous ones. The Hungarian algorithm is an optimization method for solving assignment problems. Its goal is to find the perfect match with the minimum cost between elements of two sets, minimizing the total association cost. In the context of multi-object tracking, it is used to associate current detections with existing trajectories based on a distance or similarity metric (in this case, the IoU between bounding boxes).

Formally, the Hungarian algorithm solves the assignment problem as follows: Let C be a cost matrix $n \times m$, where n is the number of detected objects and m the number of trajectories. The algorithm seeks a permutation P of m elements such that the sum of assignment costs $\sum_{i=1}^{n} C_{i,P(i)}$ is minimized, where $P(i)$ is the trajectory assigned to detected object i.

Given a set of detected objects D and a set of existing trajectories T, the algorithm proceeds as follows:

1. **Construct the Cost Matrix:** A cost matrix C is built, where each element $C_{i,j}$ represents the cost of assigning detected object d_i to trajectory t_j. The cost can be defined based on Euclidean distance, inverse IoU, or any other relevant dissimilarity metric.
2. **Apply the Hungarian Algorithm:** The algorithm is applied to the cost matrix to find the minimum-cost assignment. This results in a one-to-one mapping between detected objects and trajectories, where each object is associated with at most one trajectory and vice versa.
3. **Update Trajectories:** Based on the assignments made, trajectories are updated with new detections. Trajectories without corresponding detections may be marked as "lost" or terminated. Similarly, detections without corresponding trajectories are marked as new objects.

3.6 Training Configurations

3.6.1 Part-Based Multitask Training

As discussed in section 2, Multi-Object Tracking (MOT) models are often trained end-to-end using a multitask learning strategy. However, the results presented in this study, consistent with observations from FairMOT [Zhang *et al*., 2021], indicate that simply assigning a specific loss function to each task does not guarantee optimal model performance. In response to this limitation, a training methodology was adopted that alternates between phases of individualized task learning and joint training phases, known as Part-Based Multitask Training (TMP).

The training routine follows this process, called Part-Based Multitask Training (TMP):

1. **Heatmap Head Training Phase:** Initially, the heatmap generation task is isolated, freezing the weights of the dimension and displacement heads and nullifying their respective weights in the weighted combination of loss functions. This stage focuses exclusively on learning object centers.
2. **Training Period:** This phase lasts for 50 epochs or continues until no reduction in the loss function is observed over 10 epochs.
3. **Individual Training of Other Heads:** The process is repeated for the dimension and displacement heads, training each individually with the same criteria for weight freezing and duration.
4. **Final Joint Training:** After completing 150 focused training epochs, a final joint training phase of 50 epochs is conducted, where all layers are activated and contribute their weights to the weighted average of the loss functions.

This sequential approach allowed the model to learn the specifics of each task in isolation before integrating that knowledge into joint learning. This process facilitated the model's inter-task understanding, resulting in superior performance at the end of the training.

3.6.2 Loss Functions

During the training of the Multi-Object Tracking (MOT) model, various specific loss functions were applied to im-

prove the model's accuracy in detecting and tracking objects. These loss functions are essential to guide the model in accurately detecting object centers as well as their dimensions and displacements. For clarity, each implemented loss function is detailed below and expressed through mathematical equations.

Heatmap Loss The heatmap loss function is designed to optimize the detection of object centers in the heatmap. This function consists of two main parts: **Center Loss** and **Focal Loss**.

Center Loss is calculated as follows:

$$C_{loss} = -\frac{1}{N}\sum_{i=1}^{N}\Big[P_i \cdot \log(\hat{P}_i) + (1-P_i)\cdot\log(1-\hat{P}_i)\Big]\cdot W_i \tag{1}$$

where P_i is the ground truth probability of a pixel being an object center, $\hat{P}_i$ is the predicted probability, N is the total number of pixels, and W_i represents weights determined by the importance of each pixel. This loss function is the same as that demonstrated by CenterTrack [Zhou *et al*., 2020].

Focal loss is defined as:

$$\text{FocalLoss} = -\frac{1}{N}\sum_{i=1}^{N}(1-\hat{P}_i)^{\gamma}\cdot P_i\cdot\log(\hat{P}_i) \tag{2}$$

where γ is the parameter that modulates the loss for well-classified examples, focusing learning on misclassified examples. In this model, γ is set to 4.

Grid Loss This function optimizes the grids representing object dimensions and displacements. It is based on the L1 loss between predictions and ground truth values, applied only where objects are present (mask):

$$\text{GridLoss} = \frac{1}{M}\sum_{i=1}^{M}|G_i - \hat{G}_i| \tag{3}$$

where G_i are the actual grid values, $\hat{G}_i$ are the values predicted by the network, and M is the number of elements selected by the mask.

These loss functions allow the model to accurately learn object location, dimensions, and displacements, facilitating effective tracking in image sequences. They are combined using a weighted average. In the equation below, we see how the final loss value is calculated for each image batch:

$$\begin{aligned} Loss_f = {} & \frac{(CenterLoss + FocalLoss)*w1}{w1+w2+w3} + \\ & \frac{(GridLoss_{dim})*w2}{w1+w2+w3} + \\ & \frac{(GridLoss_{dlc})*w3}{w1+w2+w3} \end{aligned} \tag{4}$$

where $w1 = w2 = w3 = 1.0$ for the final training. As described in TMP, these weights vary to 0 depending on the training stage. $GridLoss_{dim}$ refers to GridLoss applied to the dimension output, and $GridLoss_{dlc}$ to GridLoss applied to the displacement output.

4 Results

This section details the experiments conducted and the results obtained, providing insights into the performance of the OneTrack-M model.

4.1 Experimental Setup

The experiments were conducted on a computer equipped with an Intel Core i7-11700 processor at 3.60GHz and 32GB of RAM, running the Ubuntu 20.04.6 LTS 64-bit operating system. For the training and inference of the model, a GeForce RTX 2060 graphics card with CUDA 11.0 support was used. The choice of the graphics card was based on the compute capability score provided by the manufacturer, ensuring a computing capacity comparable to the Nvidia V100, commonly used in related work. This approximation ensures the validity of the comparison of inference times with previously published results.

4.2 Dataset

The MOT17 [Milan *et al*., 2016] and MOT20 [Dendorfer *et al*., 2020] benchmarks were the datasets used to validate the results of this work. They were chosen because they are widely adopted in MOT research to maintain a consistent comparative baseline. Both MOT17 and MOT20 include predefined training and test datasets. The images cover various scenarios, varying in object proximity and camera movement, focusing on tracking people in moderately dense environments. Particularly, MOT20 differentiates from MOT17 by including very crowded challenging scenes.

MOT17 contains a total of 18 image sequences with annotations of bounding box coordinates and IDs within the sequence. The annotations follow a format of top, left, height, and width for each object, as well as its prediction confidence and the respective ID (new or not). Figure 8 shows an example of a sample from this dataset, containing a sequence of 3 examples separated by 60 frames. MOT20 includes 8 new sequences to verify object detection capability on handling extremely crowded scenarios.

4.3 Evaluation Metrics

In literature, MOTA (Multiple Object Tracking Accuracy) and HOTA (Higher Order Tracking Accuracy) are well established standards for MOT performance evaluation. MOTA is detailed in Bernardin and Stiefelhagen [2008], it approximates the tracking challenge to concepts such as accuracy. HOTA, introduced in Luiten *et al*. [2020], focuses on specific aspects of MOT, such as consistent trajectory maintenance. The TrackEval library, by Jonathon Luiten [2020], was used to evaluate the model outputs according to these metrics. These metrics are better defined as follows:

- **MOTA**: An aggregate measure that considers false positives, false negatives, and identification errors to calculate overall tracking accuracy. It is expressed as:

$$\text{MOTA} = 1 - \frac{\sum_t (FN_t + FP_t + IDSW_t)}{\sum_t GT_t} \tag{5}$$

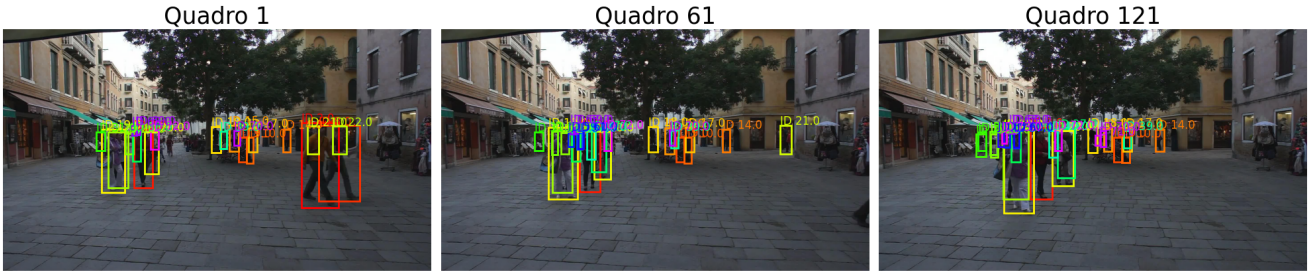

Figure 8. Image sequence and example annotations taken from the MOT17 dataset.

where FN_t, FP_t, and $IDSW_t$ represent the number of false negatives, false positives, and identification switches at each time step t, respectively, and GT_t is the total number of objects.

- **HOTA**: Combines detection and association capabilities into a single metric, balancing performance in both dimensions.

$$\text{HOTA} = \frac{1}{T}\sum_{t=1}^{T} \frac{\sum_{(i,j)\in\mathcal{M}_t} \text{IOU}(i,j)}{|\mathcal{M}_t|} \cdot \frac{|\mathcal{M}_t|}{|\mathcal{G}_t| + |\mathcal{P}_t| - |\mathcal{M}_t|}) \tag{6}$$

In this equation, $\mathcal{M}_t$ represents the set of matched ground truth-prediction pairs at time t, $\mathcal{G}_t$ is the set of ground truth objects, $\mathcal{P}_t$ is the set of predicted objects, and $\text{IOU}(i,j)$ is the Intersection over Union between the ground truth object i and the predicted object j. T is the total number of time steps.

In this work, we also adopted IDS (ID switches), IDF1 (Identity F1 Score) and FPS (Frames per Second) as complementarily evaluation metrics. The first two are indicators of the consistency in maintaining object identities during object tracking, and they are introduced in Ristani *et al*. [2016]. The last is an important metric for considering real time applications. They are described as follows:

- **IDS**: It simply quantifies the number of times an object's identification changes over its trajectory.
- **IDF1**: It compares predicted vs. ground truth detection ID pairs over the entire sequence and computes the F1-Score. Given the IDTP (Identity True Positives), IDFP (Identity False Positives), and IDFN (Identity False Negatives), IDF1 is calculated as:

$$\text{IDF1} = \frac{2 \times \text{IDTP}}{2 \times \text{IDTP} + \text{IDFP} + \text{IDFN}} \tag{7}$$

- **FPS**: This metric relates to a term used to quantify maximum video capture speed. This metric defines how many frames per second the model can infer, including all necessary preprocessing and model tasks. It is given by the inverse of the model's average inference time for a given sequence:

$$\text{FPS} = \frac{1}{I_t/N} \tag{8}$$

where I_t is the total inference time of a sequence in seconds and N is the number of frames in this sequence.

4.4 Benchmark Test Results

Table 2 shows the results obtained by OneTrack-M compared to other established models in the literature for both MOT17 and MOT20 datasets. Some of the literature models did not perform experiments on all metrics for both datasets, but they were kept as empty entries for the sake of coherence.

The proposed model is set with the following configurations: it was trained with the TMP method, it uses the Vision Transformer version Base-16, and uses a temporal window of 5 images at a time as input. This proved to be the most stable configuration, as it balances processing time, qualitative performance, and the ability to correlate relevant information in the time window for each object. Specific evaluations for these parameters are presented in the next section.

It is clear to see that OneTrack-M presents inference speed as its main benefit. Our model reached the highest FPS on both datasets, with 35.70 frames per second. It outperforms CenterTrack (FPS = 17.50 in MOT17), which is a baseline regarding the use of centers and displacements to perform MOT, LMOT (FPS = 28.50 in MOT17), which stays in second place, and MOTR by far (FPS = 7.50 in MOT17), which is another important work in literature as it introduced the application of transformers to MOT.

It is important to note that our model maintained constant FPS between MOT17 and MOT20 datasets, since it does not depend on the number of objects per frame. Unlike work in the literature, which tracks every object, we applied a channel-wise encoding for each frame, which proved to be an effective solution to integrating temporal information into transformer-based models. Thus, OneTrack-M FPS remained the same even in the crowded scenario of MOT20.

When comparing OneTrack-M tracking metrics with other works, we see it presented a competitive performance, but with some room for improvements. For HOTA, our model achieved 65.10% for MOT17 and 78.30% for MOT20, which is around 0.50 pp higher than the second-highest performing model, BoTSORT. A high HOTA suggests that both detection and association processes are performing well. For IDF1, our model configuration achieved 79.61% in MOT17 and 79.60% in MOT20, higher than the second highest performing models, tied at 79.50% for both StrongSORT and BoTSORT in MOT17, and respectively 77.00% and 77.50% in MOT20x. A high IDF1 indicates a strong performance in matching detected objects to their true identities. Regarding the number of ID switches (IDS), a lower metric implies that the tracking algorithm is good at maintaining the identity of tracked ob-

Table 2. Comparison of Results between established models in the literature on the MOT17 and MOT20 datasets.

	MOT17					MOT20				
Model	HOTA	MOTA	IDF1	IDS	FPS	HOTA	MOTA	IDF1	IDS	FPS
OneTrack-M	**65.10**	67.95	**79.61**	**1090**	**35.70**	**78.30**	52.40	**79.60**	**688**	**35.70**
Bytetrack [Zhang *et al*., 2022]	63.10	80.30	77.30	2196	11.80	77.80	61.30	75.20	1223	17.50
CenterTrack [Zhou *et al*., 2020]	52.20	67.80	64.70	3039	17.50	40.90	-	45.10	1568	-
MOTR [Zeng *et al*., 2022]	-	65.10	66.40	2049	07.50	-	-	-	-	-
MOTRV2 [Zhang *et al*., 2023]	62.00	78.60	75.00	-	06.90	76.20	60.30	72.20	-	-
TrackFormer [Meinhardt *et al*., 2022]	-	74.10	68.00	2829	07.40	68.60	54.70	65.70	1532	-
TransTrack [Sun *et al*., 2021]	54.10	75.20	63.50	3603	10.00	-	-	-	-	-
BoTSORT [Aharon *et al*., 2022]	64.60	**80.60**	79.50	1257	06.60	77.80	**63.30**	77.50	1313	-
StrongSORT [Du *et al*., 2023]	64.40	79.60	79.50	1194	07.10	73.80	62.60	77.00	770	-
FairMOT [Zhang *et al*., 2021]	-	73.70	72.30	3303	25.90	61.80	54.60	67.30	-	-
LMOT [Mostafa *et al*., 2022]	-	72.00	70.30	3071	28.50	76.30	-	76.80	-	5.35
DiffusionTrack [Luo *et al*., 2024]	60.80	77.90	73.80	3819	21.05	72.80	55.30	66.30	4117	-

jects, meaning the tracker is reliable in associating the correct IDs across frames. For this test, our model presented around 100 fewer occurrences of ID switches in comparison to the second highest, StrongSORT (1090 vs 1194 in MOT17 and 688 vs 770 in MOT20).

In terms of MOTA, our model shows a rather big discrepancy from most of the other works, with up to 12.65 percentual points difference below the best model, BoTSORT, in both datasets. This is compensated with the other metrics, but it is still interesting to understand why this single metric is lacking. Particularly, MOTA is a metric that counts false positives and false negatives in the detection step, while the others do not care so much for false tracking. Competitive metrics except by MOTA indicates that OneTrack-M can effectively track objects it finds, but it is seeing some either wrong or redundant detections, affecting the MOTA metric. While our model still presents a MOTA performance near important works in literature, like CenterTrack and MOTR, it is behind best models, where BotSORT stands out. However, to achieve a robust detection in all tracking metrics, BotSORT applies several compensation techniques for tracking like advanced Kalman filters for individual trajectories, object re-identification, and camera movement compensation. All these techniques impose a high computational cost to this model, which results in the worse FPS as seen in MOT17 dataset (FPS = 6.60).

MOT20 results showed to be consistent in comparison with MOT17 for all models. OneTrack-M was able to outperform other models on the metrics of HOTA, IDF1, IDS and FPS in both scenarios. The challenging crowded scenario of MOT20 proportionally reduced MOTA for all models from MOT17 results, as crowded scenarios are more prone to errors like false negatives due to occlusions, but HOTA increased for all models due to a higher number of tracked objects with lower IDS.

Finally, the results highlight two primary advantages of the proposed model. First, its notable inference speed (35.70 FPS) enables real-time operation for practical scenarios (e.g., 30 FPS footage) and represents the best processing time among the compared models. This performance was achieved despite utilizing a transformer, owing to the simplification of the process into a single pass and the efficiency of other stages. Second, the model demonstrated a robust capability for trajectory maintenance, validating the relevance of the mechanisms developed during its conception for correlating objects over time. Although our model did not presented the best MOTA performance, an important metric to capture detailed error occurrences in detection, it is well-suited for environments constrained by computational capacity or demanding minimal inference time.

4.5 Model Characteristics Test Results

These tests aim to validate some of the decisions made for the final version of the model. As MOT17 is the main benchmark dataset of this work, larger and widely used to compare related work, we have chosen it to set our model. In this part of the work, we evaluate different window sizes, training with and without TMP, different configurations for the feature extractor, and the application of a usual positional embedding (like the one used in ViT [Dosovitskiy *et al*., 2021]) or the proposed Channel Wise Encodings to provide context for each crop in the sequence of the transformer network.

4.5.1 Window Size Test

In Table 3, it can be seen that processing time is not affected by the adopted window size. This makes sense and it is in line with how the model processes its inputs, stacking all window images along the RGB channel dimension, effectively using more memory to process more images per window. However, it is not advisable to increase this parameter uncontrollably, as more frames make it more difficult for the model to predict displacement, mainly because this displacement should be from the previous frame's position. An increase in ID values is noticeable when receiving context from time instances far in the past. Thus, a value that keeps this error rate acceptable was 5.

4.5.2 TMP Effectiveness Test

To evaluate the capability of this training method, two versions of the model were trained, one training all loss functions at once, as normally done, for 200 epochs. The other version was trained following the training routine proposed by TMP, training each head or task of the model individually, freezing the weights of the other heads, one at a time, followed by a final training with all heads at once. Table 4 shows the results

Table 3. Table of results on window size tested on MOT17.

Window Size	HOTA	MOTA	IDF1	IDS	FPS
1	45.29	45.82	56.38	5145	35.70
2	56.16	58.94	74.39	1449	35.70
3	56.32	61.81	76.97	1365	35.70
4	62.05	64.30	78.41	1253	35.70
5	65.10	67.95	79.61	1090	35.70
10	52.52	56.82	64.06	3095	35.70
20	52.41	56.63	63.92	3193	35.70

of each test. We can see that the model greatly benefits from an isolated training routine for each task. For this test, all other model hyperparameters were kept constant, i.e., window size fixed at 5 images at a time, and the encoding step by ViT uses the B16 model, i.e., the base size of the network, with images cropped to 16x16 pixels.

Table 4. Comparison of results between a model trained with and without the TMP technique tested on MOT17.

With TMP?	HOTA	MOTA	IDF1	IDS	FPS
No	49.91	55.40	72.37	2312	35.70
Yes	65.10	67.95	79.61	1090	35.70

4.5.3 Test of Different Transformer Feature Extractors

One of the essential parts for the good performance of this model is the choice of the architecture responsible for feature extraction. This is because architecture requires this step to understand concepts in both the spatial and temporal axes. Therefore, this part must be robust enough to handle these aspects simultaneously.

This also brings a relevant question to the overall inference time of the system. This model part is responsible for almost all of the pipeline's time consumption. Thus, applying a network with many parameters greatly impacts the model's processing time per second. This is well reflected in the obtained results, shown in Table 5.

For more details on the configuration of each ViT version, they should be referenced from the original work in Dosovitskiy *et al*. [2021], but briefly, the main difference for the Large models is a larger number of parameters and layers, while the 16 and 32 number in models refer to the size at which the images are cropped. This implies that the 32 models have less detail representation capability as they need to summarize more information in each input token to the encoder sequence. Conversely, this larger size per crop means fewer elements per image, resulting in a shorter inference time.

Table 5. Comparison results of the model varying the version of the Vision Transformer being used tested on MOT17.

ViT Version	HOTA	MOTA	IDF1	IDS	FPS
ViT-Base-16	65.10	67.95	79.61	1090	37.70
ViT-Base-32	60.31	65.47	71.23	1322	60.18
ViT-Large-16	68.02	75.02	80.05	1108	08.11
ViT-Large-32	65.02	71.02	81.17	1243	21.88

4.5.4 Contextual Embeddings Test

This test aims to validate the method presented in this work regarding the embedding that incorporates spatial and temporal context. During preprocessing, the image sequence is stacked in the channel dimension, and applying these representations to each channel promotes spatial-temporal perception by the model. A common practice for this part of transformer models includes using spatial embeddings, either through fixed sinusoidal functions that set values for each position in the original image or through learnable representations with weights adjusted during training. The results of this test are presented in Table 6.

Table 6. Comparison results of the model varying the embedding method used. Tested on MOT17.

Embedding Method	HOTA	MOTA	IDF1	IDS	FPS
Positional	44.68	45.21	58.22	4866	39.50
Channel Wise	65.10	67.95	79.61	1090	35.70

The results indicate that the model does not identify temporal features when handling input data, and only positional embedding is used. Even with an inference window of 5 images, there are difficulties in properly connecting the movement of objects in the scene, resulting in tracking failures, as shown by the high incidence of identification switches.

4.6 Qualitative Analysis Results

Figure 9 shows a sequence of images with their respective inferences. It demonstrates a problem called ID theft. Due to the method used to associate IDs over time, this phenomenon becomes present in the inferences, where a sufficiently large error in the displacement value for two very close objects can cause one of them to receive the ID of the other, while it becomes necessary to create a new ID for the one that lost its identifier.

Figure 9. Example of an ID theft case. Note that between the second and fourth image, the object with id=2 loses it and receives the ID of the object next to it as a new value.

Next, Figure 10 demonstrates a model limitation for objects at medium to long distances from the camera. Here we can see a person on a bicycle that is not detected over several frames while other elements appear to be positioned close to the person in a two-dimensional notion, however, due to the perspective generated by the image, the model does not detect them.

Another interesting case, demonstrating the model's ability to maintain the trajectory despite occlusions, is in Figure 11. In this image, there is a person completely behind the person in front of the image. In the second frame, the model even loses the detection of this person, but in the last one, the ID is recovered, as the same one that had been lost, despite the case showing severe occlusion.

Figure 10. Example of an object too far for detection and tracking.

Figure 11. Demonstration of model robustness in a scenario of object occlusion.

In summary, while the model can show some weak points, which can and should be considered for future works, it is compensated in higher metrics (besides MOTA) and especially in high FPS, which can allow for better practical usage of this model, which was not available with some of the better performing models, scoring less than 10 FPS, meaning in a 30 FPS video, it would be missing a third of frames for processing.

5 Conclusion and Future Works

This work brings relevant contributions to the field of object tracking. These innovations focused on improving two main aspects: inference time and training stability. Regarding inference time, the use of transformers allowed the abstraction of the temporal context in such a way that the network is responsible for relating the parts of the network, without requiring an internal step of temporal contextualization. That is, instead of modeling the problem as each element of the transformer's input sequence being an image from the frame window, they all become a single image with multiple channels. From the implemented encoding, which provides these channels with temporal information, the model can consider everything as a single image. In addition, there is no need for a decoder to infer objects and their trajectories, requiring only a few more layers at the output to process the generated attention maps.

In terms of training stability, the TMP method proved to be an option with the potential to improve the training of MOT models in general. Making the process more stable and generally improving the results obtained.

As future work, we plan to evaluate OneTrack-M in other MOT datasets. Besides, evaluating other extractor models instead of the Vision Transformer could be a promising path, but it requires additional work to change the functioning of the proposed architecture mechanisms in such a way that it maintains its characteristics and allows effective training of the model. Finally, we suggest investigating the feasibility of TMP for other MOT models that have similar processing, that is, perform both MOT stages - detection and tracking - in a unified way. This is also true in other fields where there are models that also perform related tasks, but may end up interfering with the mutual learning of the model.

Declarations

Acknowledgements

We also wish to thank the Computing Institute - IComp/UFAM and the Intelligent Systems Lab - LSI/UEA for providing the necessary resources and facilities to carry out this study.

Funding

No monetary funding was provided for the development of this work.

Authors' Contributions

All authors contributed equally to this work. All authors read and approved the final manuscript.

Competing interests

The authors declare that they have no competing interests.

Availability of data and materials

The datasets generated and/or analyzed during the current study are available on the MOT Challenge website, specifically at https://motchallenge.net/data/MOT17/ and https://motchallenge.net/data/MOT20/.

References

Aharon, N., Orfaig, R., and Bobrovsky, B.-Z. (2022). Botsort: Robust associations multi-pedestrian tracking. DOI: 10.48550/arxiv.2206.14651.

Bashar, M., Islam, S., Hussain, K. K., Hasan, M. B., Rahman, A. B. M. A., and Kabir, M. H. (2022). Multiple object tracking in recent times: A literature review.

Bernardin, K. and Stiefelhagen, R. (2008). Evaluating multiple object tracking performance: The clear mot metrics. *EURASIP Journal on Image and Video Processing*, 2008. DOI: 10.1155/2008/246309.

Bewley, A., Ge, Z., Ott, L., Ramos, F., and Upcroft, B. (2016). Simple online and realtime tracking. In *2016 IEEE International Conference on Image Processing (ICIP)*, pages 3464–3468. DOI: 10.1109/ICIP.2016.7533003.

Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., and Zagoruyko, S. (2020). End-to-end object detection with transformers. DOI: 10.1007/978-3-030-58452-8$_1$3.

Caruana, R. (1997). Multitask learning. *Machine Learning*, 28(1):41–75. DOI: 10.1007/978-1-4615-5529-2$_5$.

Dendorfer, P., Rezatofighi, H., Milan, A., Shi, J., Cremers, D., Reid, I., Roth, S., Schindler, K., and Leal-Taixé, L.

(2020). Mot20: A benchmark for multi object tracking in crowded scenes. DOI: 10.48550/arxiv.2003.09003.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. (2021). An image is worth 16x16 words: Transformers for image recognition at scale. DOI: 10.48550/arxiv.2010.11929.

Du, Y., Zhao, Z., Song, Y., Zhao, Y., Su, F., Gong, T., and Meng, H. (2023). Strongsort: Make deepsort great again. DOI: 10.1109/tmm.2023.3240881.

Jaward, M., Mihaylova, L., Canagarajah, N., and Bull, D. (2006). Multiple object tracking using particle filters. In *2006 IEEE Aerospace Conference*, pages 8 pp.–. DOI: 10.1109/AERO.2006.1655926.

Jonathon Luiten, A. H. (2020). Trackeval. Available at:`https://github.com/JonathonLuiten/TrackEval`.

Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., and Guo, B. (2021a). Swin transformer: Hierarchical vision transformer using shifted windows. DOI: 10.1109/iccv48922.2021.00986.

Liu, Z., Ning, J., Cao, Y., Wei, Y., Zhang, Z., Lin, S., and Hu, H. (2021b). Video swin transformer. DOI: 10.1109/cvpr52688.2022.00320.

Luiten, J., Os□ep, A., Dendorfer, P., Torr, P., Geiger, A., Leal-Taixé, L., and Leibe, B. (2020). Hota: A higher order metric for evaluating multi-object tracking. *International Journal of Computer Vision*, 129(2):548–578. DOI: 10.1007/s11263-020-01375-2.

Luo, R., Song, Z., Ma, L., Wei, J., Yang, W., and Yang, M. (2024). Diffusiontrack: Diffusion model for multi-object tracking. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(5):3991–3999. DOI: 10.1609/aaai.v38i5.28192.

Meinhardt, T., Kirillov, A., Leal-Taixe, L., and Feichtenhofer, C. (2022). Trackformer: Multi-object tracking with transformers. DOI: 10.1109/cvpr52688.2022.00864.

Milan, A., Leal-Taixe, L., Reid, I., Roth, S., and Schindler, K. (2016). Mot16: A benchmark for multi-object tracking. DOI: 10.48550/arxiv.1603.00831.

Mostafa, R., Baraka, H., and Bayoumi, A. (2022). Lmot: Efficient light-weight detection and tracking in crowds. *IEEE Access*, 10:83085–83095. DOI: 10.1109/ACCESS.2022.3197157.

Murad, A. and Pyun, J.-Y. (2017). Deep recurrent neural networks for human activity recognition. *Sensors*, 17:2556. DOI: 10.3390/s17112556.

Ristani, E., Solera, F., Zou, R., Cucchiara, R., and Tomasi, C. (2016). Performance measures and a data set for multi-target, multi-camera tracking. In Hua, G. and Jégou, H., editors, *Computer Vision – ECCV 2016 Workshops*, pages 17–35, Cham. Springer International Publishing. DOI: 10.1007/978-3-319-48881-3_2.

Su, H., Chen, Y., Tong, S., and Zhao, D. (2019). Real-time multiple object tracking based on optical flow. In *2019 9th International Conference on Information Science and Technology (ICIST)*, pages 350–356. DOI: 10.1109/ICIST.2019.8836764.

Sun, P., Cao, J., Jiang, Y., Zhang, R., Xie, E., Yuan, Z., Wang, C., and Luo, P. (2021). Transtrack: Multiple object tracking with transformer. DOI: 10.48550/arxiv.2012.15460.

Wang, C.-Y., Bochkovskiy, A., and Liao, H.-Y. M. (2022). Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. DOI: 10.1109/cvpr52729.2023.00721.

Wojke, N., Bewley, A., and Paulus, D. (2017). Simple online and realtime tracking with a deep association metric. DOI: 10.1109/icip.2017.8296962.

Yang, M., Han, G., Yan, B., Zhang, W., Qi, J., Lu, H., and Wang, D. (2024). Hybrid-sort: Weak cues matter for online multi-object tracking. DOI: 10.1609/aaai.v38i7.28471.

Zeng, F., Dong, B., Zhang, Y., Wang, T., Zhang, X., and Wei, Y. (2022). Motr: End-to-end multiple-object tracking with transformer. DOI: 10.1007/978-3-031-19812-0_38.

Zhang, Y., Sun, P., Jiang, Y., Yu, D., Weng, F., Yuan, Z., Luo, P., Liu, W., and Wang, X. (2022). Bytetrack: Multi-object tracking by associating every detection box. DOI: 10.1007/978-3-031-20047-2_1.

Zhang, Y., Wang, C., Wang, X., Zeng, W., and Liu, W. (2021). Fairmot: On the fairness of detection and re-identification in multiple object tracking. *International Journal of Computer Vision*, 129:3069–3087. DOI: 10.1007/s11263-021-01513-4.

Zhang, Y., Wang, T., and Zhang, X. (2023). Motrv2: Bootstrapping end-to-end multi-object tracking by pretrained object detectors. DOI: 10.1109/cvpr52729.2023.02112.

Zhang, Y. and Yang, Q. (2017). A survey on multi-task learning. *arXiv preprint arXiv:1707.08114*. DOI: 10.1109/tkde.2021.3070203.

Zheng, Y. (2024). Cuestrack:multi-object tracking based on weak clues and trajectory prediction. In *2024 5th International Conference on Machine Learning and Computer Application (ICMLCA)*, pages 321–325. DOI: 10.1109/ICMLCA63499.2024.10754464.

Zhou, X., Koltun, V., and Krähenbühl, P. (2020). Tracking objects as points. *ECCV*. DOI: 10.1007/978-3-030-58548-8_28.

Zhou, X., Wang, D., and Krähenbühl, P. (2019). Objects as points. In *arXiv preprint arXiv:1904.07850*. DOI: 10.48550/arXiv.1904.07850.