Investigation of Multi-Object Tracking Techniques for the Robotic Application

By

Giro Ele

Supervisor: Prof. Goldie Nejat

April 16, 2022

B.A.Sc. Thesis

Division of Engineering Science
UNIVERSITY OF TORONTO

Abstract

The lack of robot applicable MOT models with SOTA performance was to be resolved by this thesis work. The real-time ability was improved by developing an inference speed test and creating a faster and smaller model. Knowledge distillation was applied to maintain tracking performance, which would be lowed by the reductions required to reach real-time. The result was a reduced TransTrack model that operated at 18Hz and achieved 34.5% MOTA performance. The model trained normally only had 33.1%. This achievement shows there is benefit in applying knowledge distillation to MOT training. However, the chosen approach produced insufficient performance. The proposed model was only achieved a far lower MOTA tracking performance than the original model. Its performance was also worse than non-SOTA real-time MOT solutions. Therefore, the proposed model is not robot applicable.

Acknowledgements

I would first like to thank Professor Goldie Nejat for the opportunity to work on this project and contribute to the Autonomous Systems and Biomechatronics Laboratory.

I am grateful to the Ph.D. Students, Angus Fung and Aaron Tan, who were a great help throughout the research project. Giving me useful insight, feedback, and direction.

Finally, I would like to thank my parents for cultivating me into who I am. Instilling into me the courage so I even when I don't know right from wrong, I can always fulfill my duty to the bitter end.

Table of Contents

List of Abbreviations, Figures, and Tables

List of Abbreviations

FPS	Frames per Second
MOT	Multi-Object Tracking
SOTA	State of the Art

List of Figures

List of Tables

1 Introduction

In this thesis, I attempt to address the lack of robot applicable models that have SOTA MOT performance. Most MOT research ignores creating systems that are overall ready for robot use. Accurate tracking is an important feature in robotics. In high-speed autonomous driving, it allows for trajectory predictions to prevent collisions. In this thesis, I attempt to address the robot application of SOTA MOT models. The focus on reaching the real-time requirement.

Robotic problems, such as navigation and motion planning, require an awareness of the agent's surrounding objects and an ability to predict their movement [1][2]. For that reason, the Multi-Object Tracking (MOT) task is vital [3]. This task aims to produce estimates of the spatio-temporal trajectories of multiple objects. The result is all objects within an input (eg. video or depth point clouds) being uniquely identified and tracked throughout time. The use of the state-of-the-art transformer architecture [4][5] has produced good performance on the MOT task [6;7-12]. The reason for this is that dominant MOT methods keep object detection as a fixed, independent process (tracking-by-detection). This leads to poor tracking performance, like tracking failures when objects are occluded, deformed, or change appearance [1][13]. Transformers leverage the sequential temporal structure of the tracking problem, where each visual feature and embedding developed are relevant in future processing [6]. Transformer-based MOT solutions have shown the ability to perform well in these failure cases. However, the models of this new paradigm do not ensure their accessibility to robotic applications. Additionally, the MOT problem is considered complex and not completely solved. State-of-the-art solutions are frequently progressing the area of study but often lack a connection to the real-world applications. For example, many state-of-the-art solutions focus solely on video input for 2D MOT. However, robotic systems are often multi-modal (combine multiple sensors) to have robust vision [1]. Also, these systems' ability to operate in real-time, a term meaning at an acceptable speed, are often not evaluated, or considered. The focus often laying on their proposed system's tracking performance. In many cases this increases the computational load and reduces inference speed.

This research will develop a novel real-time MOT model to address the lacking evaluation of the innovative transformer-based systems' capabilities with regards to the robotic scenario. TransTrack, a state-of-the-art deep learning MOT model, will be leveraged as the foundation [4]. It has successfully applied the transformer to the tracking problem but has not explored producing an effective real-time solution. The initial sub-objective to achieve our goal was to reproduce the results of the initial model. Next, changes were made to improve TransTrack's speed. This will come at the cost of tracking performance. Knowledge distillation for MOT is proposed to improve this reduced model's performance through leveraging the original fully trained model. Finally, the proposed model's performance will be evaluated.

A python implementation of TransTrack was used. An inference speed test was developed and used in producing our proposed real-time model. The training and evaluation of our models will be over the benchmarking dataset MOT17 [14]. It includes videos with annotations for tracking. Our model's tracking performance was determined by using the multiple object tracking accuracy (MOTA) [15].

The contribution of this research will include the following: (1) we will purpose a novel online MOT model, (2) evaluate knowledge distillation for MOT training, and (3) evaluate the performance of the MOT transformer architecture in the robotic scenario (real-time). Beyond the proposed model, the procedure outlined will be transferable to future SOTA models. This is a step in developing robotic perception. A field that becomes more important as we move towards a future where robotics systems operate in diverse real environments and have significant responsibilities.

2 Literature Review

2.1 MOT

The multiple object tracking (MOT) or multi-object tracking task is the process of taking video input to identify all objects and how their positions change with time [16]. Each video frame will have all objects identified with a bounding box, in the image domain. These detections are related across frames using IDs that uniquely identify the object's trajectory. The difference with the single object tracking (SOT) task is that a single object is tracked given a known appearance.

The evaluation of MOT algorithms does not have a single clear metric. Instead, a set of metrics are used that were employed by many benchmarking tools, like the MOT17 dataset [14]. Multiple Object Tracking Accuracy (MOTA) [17] is the most accepted metric that gives an indication of overall performance of the tracker. It incorporates 3 sources of error. The number of false positives, false negatives, and identity switches. The latter is when a prediction switches to tracking a different ground-truth. Multiple Object Tracking Precision (MOTP) is a metric that gives a measure of localization precision. It is how well any true positive result overlaps with the assigned ground truth. MOTP is more related to the accuracy of individual detections and does not say a lot about tracking performance. IDF1 is a measure that determines the ability to preserve the track identity over the entire sequence.

2.2 MOT Connection to Robotic Applications

MOT is essential to autonomous driving.[18] Localization is often not enough when in high-speed situations. Instead, understand and predicting the trajectories of objects is necessary to maintain safe operation.

Multi-modal is necessary because of the need to gain information about objects in the vehicle frame. This is often competed using multiple cameras, lidar, or other sensor inputs.

Traditional object tracking used data association after objects were detected. Many employing appearance models to leverage the help maintain tracks of the right object. Recent models use

deep learning approaches that become better at avoiding issues such as identity switches and lacking tracking performance. These are more computationally expensive and require datasets for their training.

The limitation of current robotic applicable technique is that MOT performance is not solved yet. Using previous outdated models will produce inferior results, that could jeopardize the safety in robotic operations. However, there is not consistent work in adapting newly proposed techniques for robotic applications.

2.3 MOT Approaches and Example Solutions

MOT systems have two popular approaches. They are **tracking by detection** and **joint detection and tracking**. Tracking by detection is the dominate approach applied by state-of-the-art MOT algorithms. It involves first using an object detector on individual frames. This results in the localization of all the frames' objects, often as bounding boxes. The second step is to then associate these detected objects between frames using some method. A simple example is using the Intersection over Unions (IoU), which determines the overlap between areas. An advantage of this approach is the ability to separately improve the detection and association segments [19]. For example, easily applying a new state-of-the-art object detector into the design of a MOT system. However, the tracking ability is impacted because of the information lost when only associating detected objects. For example, visual features of each frame. The association step can be completed considering object motion. SORT [20] used a Kalman filter [21] to create predictions of a detected object's location in a future frame from previous frames. It then applied the Hungarian algorithm to associate predictions with detections of a current frame's objects. DeepSORT [22] uses a convolutional neural network (CNN) to find a different association metric that incorporates motion and appearance similarity.

Joint detection and tracking is another popular approach that differs from the previous approach by completing detection and tracking simultaneously, in one-shot. The main benefit being that information is available between frames. These are often created by building off an object detector. An example of this is FairMOT [23] which used CenterNet as a backbone and

built a branch that learned to use appearance (re-identification) embeddings for data association.

These joint methods usually only operate on the current and previous frame. This would affect their ability to track objects that are occluded for longer than the number of frames considered at each interval. In this event, the track is deleted. To account for possible reappearances of that object, many trackers employ track rebirth. This is a method to store previously eliminated tracks for a given amount of time. They are then reopened when that object reappears.

An online MOT model takes in input sequentially and update tracks using current and past information. This is different from offline MOT models, that have access to all data and usually produce a batch tracking output. Robotic applications where MOT is necessary are in the online domain. A “real time” MOT model is one that can inference at a high FPS (>= 20HZ).

For training and benchmarking the two real-world datasets MOT16 or MOT17 are often used [24]. They include several video scenes where pedestrians are tracked and annotated with bounding boxes surrounding them.

2.4 Transformer Architecture

The transformer [4] architecture can capture long-range dependencies and sequential information. This was originally designed for the NLP problem but became popular in computer vision. Recently, MOT systems were based around transformers to better exploit the spatial-temporal relations between sequential frames. This includes information about the object’s appearance previously extracted being useful in later tracking inference. Transformer-based MOT systems include TransTrack [25], Trackformer [6], PatchTrack [26].

Transformers use attention functions. They take a set of key-value pairs and a query as input. The output is determined by a weighted sum of the given values. The weight is computed from a compatibility function of the query to the value’s corresponding key. The specific function used in transformer architecture is the scaled dot-product attention. The dot products of the query and keys are computed then divided by square root of the keys’ dimension. This is

because of a hypothesis that large key dimension will result in large dot products. Finally, a softmax function is applied given the final weights.

Self-attention and cross-attention blocks are used in the encoder and decoder of the transformer architecture. The self-attention blocks all have values, keys, and queries come from the same place. Cross-attention does not have this constraint. Also note in figure 1 that attention functions grow in complexity with the sequence length.

A standard transformer can be seen in figure 1. It shows a single layer of encoder-decoder structure. The left and right grey boxes respectively. Each encoder incorporates two sub-layers, a self-attention mechanism followed by a fully connected feed-forward network. The decoders have 3 sub-layers, a self-attention block, cross-attention block, then a fully connected feed-forward networks. Positional encodings are embeddings inputted at the bottom of stacks of encoders and decoders to give position information in the sequence of the tokens.

Multi-head attention refers to using the linearly projecting the single attention function N times, then concatenating and projecting into the result. This allows for different representations subspaces at different positions to be jointly attended to.

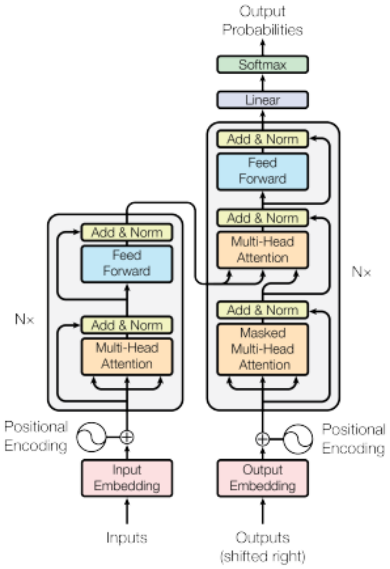

Table 1: Maximum path lengths, per-layer complexity and minimum number of sequential operations for different layer types. n is the sequence length, d is the representation dimension, k is the kernel size of convolutions and r the size of the neighborhood in restricted self-attention.

Layer Type	Complexity per Layer	Sequential Operations	Maximum Path Length
Self-Attention	$O(n^2 \cdot d)$	$O(1)$	$O(1)$
Recurrent	$O(n \cdot d^2)$	$O(n)$	$O(n)$
Convolutional	$O(k \cdot n \cdot d^2)$	$O(1)$	$O(log_k(n))$
Self-Attention (restricted)	$O(r \cdot n \cdot d)$	$O(1)$	$O(n/r)$

Figure 1: Transformer Architecture (left), Attention Function Complexity Comparison (right) [4]

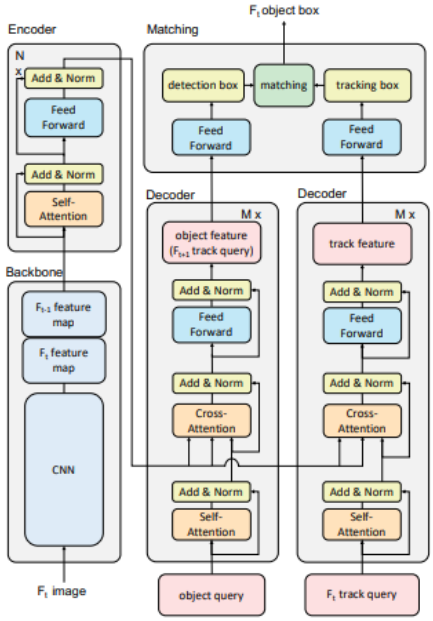

Figure 3: **The architecture details of TransTrack**. First, the current frame image is input to CNN backbone to extract feature map. Then, both the current frame feature map and the previous one are fed into encoder to generate composite feature. Next, learned object query is decoded into detection boxes and object feature of the previous frame is decoded into tracking boxes. Finally, IoU matching is employed to associate detection boxes to tracking boxes.

Figure 2: TransTrack Architecture [25]

2.5 Transformer-based MOT Solutions

Often transformed-based MOTs use a CNN to obtain a feature map of frame images. These feature maps are then used as the input for the transformer encoder-decoder structure. An example is TransTrack's architecture, shown in figure 2, where the previous and current frame's feature maps are inputted. They use two decoders. One decoder takes a learned object query input and decodes detection boxes for the current frame. This object query is a set of trained parameters. This branch was developed following DETR [27], a transformer-based object detector. The object query looks for objects of interest in the key, the feature map inputted in cross-attention. This decoder's object feature map is the high dimensional output of decoder before refining into bounding boxes. It stores the location and appearance of objects in the frame. The previous inference step's object feature map is used as a "tracking" query for the second decoder. It is used to decode tracking bounding boxes, an approximation of where previous seen objects should be on the current frame. Association between the detection and

tracking bounding boxes using Intersection over Union matching and the Hungarian algorithm [28] to produce the current step's final output. This explains how TransTrack can complete the MOT task.

TransTrack's training loss is the weighted sum of 3 terms. The first is a focal loss comparing the predicted classifications and category labels ground truth. The next 2 are a L1 loss and generalized IOU loss comparing the normalized center coordinates, height, and width of bounding boxes from the model and ground truth.

The TransTrack model did not investigate its real time capability, which is an essential demand of practical MOT applications. Transformers have already been known for their complexity and large resource demand. Additionally, the MOT case results in larger sequence length in the attention blocks, which is quadratic to the complexity per layer. On top of that, TransTrack reached relatively higher results by applying dual decoders branches.

Trackformer [6] is another transformer-based MOT that built off deformable DETR. Two decoder branches were not used. Instead, the track queries, found like in TransTrack, are merged with a static object query. This combined query is then used in a single decoder branch to produce MOT results. While computationally less expensive, this system produces worse results than TransTrack (MOTA 65.0 vs 75.2).

A variety of other transformer-based MOT methods have been created [29], with complying developments. However, they are unable to produce results that rival the peak MOT methods. At best being comparable. Additionally, these methods do not consider their practical application with an analysis of its real time ability.

These systems have proven to have satisfying tracking accuracy. However, none of these studies have investigated the real time ability of the transformer-based tracker. In practical application of these MOT systems, the inference speed is an essential demand.

2.6 ByteTrack

ByteTrack [30] is a tracking by detection system that uses a novel association method, BYTE. It associates every detection box. First high confidence detections boxes are mapped to tracks

using motion similarity (eg Kalman Filter). Then the remaining unmatched tracks are matched to low confidence detections boxes. ByteTrack was created by applying this Byte to a state of the art object detector, YOLOX [31]. It is a recent MOT method and has produced state of the art results according to key metrics, as well as maintain a high FPS inference speed. This means that it is practical for the real-world application.

2.7 3D MOT and multi-modal

MOT benefit from camera-lidar input. Camera input allows for 2D information with high resolution. However, lidar (point cloud) input gives 3D information at a high accuracy. Sensor fusion is the process that enables these different inputs to be used in the same model. This approach is very common in the 3D MOT specialty [32,33,34]. 3D MOT involves the ability to return the trajectories of objects in the 3D world frame. This usually takes the form as a bounding box (3D). What was previously in this document is also referred to as 2D MOT. While some techniques are similar between 2D and 3D tasks, there are several differences between the tasks. For example, the generally accepted benchmarks differ, being Waymo Open Dataset [35], KITTI [36] or nuScenes [37].

For 3D MOT, many solutions follow the protocol described in [3][38] that defines metrics for 3D MOT like average MOTA (AMOTA). A difference with 3D MOT is that tracking occurs with 3D bounding box, instead 2D MOT's 2D bounding boxes.

2.8 Knowledge Distilling and Addressing Computational Complexity

Knowledge distilling [39] is the method to better train a smaller model, based on the learning of a larger one. The idea is that using the "soft targets" generated from a larger model, will allow the smaller model to train better that if ground truths were used. The procedure [39] introduces for distillation is to determine a subset of the full training set, called the transfer set. For each case, compute the soft target distribution using the large model with a high temperature in its softmax output layer. This results in a softer probability distribution over classes. Continue training using a high temperature, then once completed swap back to a temperature of 1.

A modification offered to increase the distilled model's ability to compute correct labels is as follows. Two different objective functions are created, and the distilled model is trained on its weighted average. The first takes the cross entropy of the soft targets, while the second is the cross entropy with the correct labels. It is recommended to have the former use a high temperature, while the latter has a temperature of 1.

This however is ultimately a strategy that allows for more efficient training, without clear indications on producing an effective smaller model.

In the NLP field, distilling a transformer and reproducing the same results was possible with a 1/40 sized distilled model [40]. They followed work in [39] and improved upon the CodeSearchNet challenge's leader at the time [41].

For the single object tracking task, recently [42] introduce "exemplar transformers". Overall applying transformers to maintains relatively state of the art results, while also having high inference speed, 47 FPS. In comparison, it determined that the other state of the art methods all produced less than 10 FPS and are not real-time. Also, it produced this high FPS inference on CPU, while most inference speed calculations use relatively good single GPUs. This was possible because of the exemplar attention's changes to the normal scaled dot-product attention. It resulted in a $O(Ek^2ND^2)$ computational complexity from the original $O(N^2D)$. Here N is the associated with the spatial size (HW) which is far larger than the D term, feature dimension. Therefore, reducing the complexity to be quadratic in D is the reason for the lower computational load. However, as mentioned this was not designed for the MOT task.

Additionally, changing the number of encoder-decoder layers in the transformer architecture will reduce tracking performance and the computational load [25]. 6 layers are originally used, so experimenting with fewer layers might allow these systems to reach real time inference quality. Also, reducing the inputted image dimensions inputted into the system will reduce the computational complexity, while again trading off for performance.

3 Methods

3.1 Choosing SOTA Transformer-based MOT model

A state-of-the-art transformer-based MOT model was selected as the basis. These are presented with papers. Most supply an implementation of their design. The search can begin in a variety of places that host these papers. An example of these are paperswithcode.com and arxiv.org [3,43]. This search is to be completed using keywords relevant to the task such as: MOT, 3D MOT, real-time, online.

A good indicator for the impact of these papers, is the number of times they are cited. This is because the current SOTA models often used in subsequent papers for their benchmarking studies. Following this trail will lead to understanding the current SOTA. This is limited to models created within the last year.

Factors that affected the model choice were tracking performance, size, and amount of training required. Additionally, models that better met the robotic scenario are desired. This would eliminate additional work to develop a fully robot applicable solution.

The chosen model's training was reproduced to ensure the implementation is accurate. This was not done in its entirety because of limited compute.

3.2 Inference Speed (FPS) Evaluation Procedure

The FPS evaluation script can take in any video and completes the MOT task. It computes the elapsed time of each inference. After 50 frames, an average inference speed (FPS) is calculated.

At each iteration, outside of the timed segments, the output annotations are projected onto each image and saved. They show the predicted bounding boxes and the track identifications for the unique object followed. These are also color coded. Finally, these images are used to generate a video.

When using video input, the FPS evaluation script simulates live operation. This means that frames are missed during the time the model takes to produce an iteration's output. This simulates an application of MOT in the robotic scenario. These outputted annotated videos are

used as a qualitative check. The performance during tracking failure cases like occlusion can be observed. The video input selected was from outside of the dataset the models were trained on.

It was also helpful to see the elapsed time of specific sections of the model. This was implemented by printing out elapsed times, and tabulating from the log manually. This method was used because of the difficulty in passing information between the dozen programs necessary in the model's operation.

Inference speed refers to the time from image taken to annotations produced. The pre-processing of each image frame, before it is inserted into the model, is treated as part of the model's inference speed. Additionally, the output section excludes producing each iteration's annotated image, since the bounding boxes themselves are sufficient for completing the MOT task. The annotated image would serve little purpose in a robotic pipeline.

3.3 Reduced the Original (Large Model) to Approach a Real-Time Solution (Small Model)

I experimented with various configurations to have the chosen SOTA model become real-time. This relied on the previous FPS evaluation script.

Reducing the model to reach real-time is beneficial over finding a real-time MOT solution, because it could ideally maintain the SOTA's improved tracking performance. Real-time MOT solutions are usually behind in terms of tracking performance and research attention.

This procedure does not have specific guidelines. All that is desired is that real-time (>25Hz) is achieved. A strategy used was to not train each experimental model. This allowed for more rapid experimentation to see how model changes affect inference speed. The number of parameters for each model should be recorded. This can be informative and is expected to be correlated to the overall speed of the model.

Also, understanding the selected model's architecture and how it is implemented is crucial. An IDE that incorporates breakpoints is helpful. This will give you the ability to step through the

implementation of the model. A better understanding of the model's implementation can be gained, especially if it is separate throughout a dozen programs in various subfolders.

One modified model should be selected to go forward with. This will be called the "small model", the reduced model developed from a chosen SOTA "large model".

3.4 Implement Knowledge Distillation and Train Model with Both Approaches

Now I developed a knowledge distillation technique for MOT to improve the small model's training.

The implementation I propose is altering the MOT17's ground truth dataset. A dataset is created by making the fully trained large model generate annotations from the training and validation samples. This includes predictions for both the bounding boxes and class confidence. The bounding boxes would now become sub pixel, and the class confidence is the large's models confidence instead of a one hot encoding of the detected class. Note that the only class is pedestrians in MOT17.

This technique leverages the success of KD on object classification, with its similar methods. Compared to normal training, they suggested transferring more in-depth features then just the hard targets. Instead of just learning to reproduce a task, it learns off the "soft targets" as well. This was represented as an additional loss term that compared the probability distribution each model found over all possible classes. The success of KD in this object classification implementation motivates my approach.

A dataset is to be built instead of producing the "soft targets" at each iteration. This will avoid additional time in training the models. The evaluation of the KD trained model, will be on the ground truth dataset.

The small model is trained with and without the proposed KD implementation. Once fully trained, they will be evaluated on the MOT17 ground truth validation set. Their MOTA scores will allow for the comparison of their tracking performance. As well as the FPS evaluation script being used to produce qualitative comparisons from their produced demonstrations.

Additionally, the large model's results, after similar training, will be taken or produced. This will allow for seeing how well the KD method did at recuperating the original model's performance. The results will also be compared to non-SOTA real-time models. Another qualitative comparison can be completed.

3.5 Unexplored Multi-modal Ability

Within the scope of this thesis was also developing the proposed model to be multi-modal from the original SOTA 2D MOT solution. This work was not done. In this section, I discuss the procedures outlined for that task.

This proposed model would take both video and point cloud input, which is normal for multi-modal MOT models [1]. We will then design and train a multi-modal model that is an improvement on TransTrack. Finally, the developed model's performance would have been compared to existing 3D MOT models, like EagerMOT and JRMOT [1][27]. These were selected because of their success with the multi-modal MOT task.

Multi-modal fusion techniques would need to be investigated in the designing step. The KITTI dataset would be used for training and evaluating the final model [36]. It includes camera and lidar recordings with annotations for object detection and tracking. Instead of MOTA, average MOTA (AMOTA) [4][38] is the metric commonly used for 3D MOT.

4 Results and Discussion

4.1 FPS Evaluation

This table shows the results of experiments to reduce the TransTrack model. Empty entries refer to the default settings being applied. Each experiment had the changed model's inference speed evaluated in terms of FPS. The number of trainable parameters was also found. The table is setup to show the impact of each individual model component change and is not an exhaustive log of all experiments.

	Experiment # (Blank refers to default settings)								
Components	1 (Large Model)	2	3	4	5	6	7	8	9 (Small Model)
largest dim of inputted image	800	800	800	800	800	800	800	800	800
# of encoding layers		1	3						1
# of decoding layers		1	3						1
OD Backbone				resnet18					resnet18
# of feature levels					1				1
Embedding size (hidden dim)						64			64
Dim of feedforward layers							512		512
# of query slots								250	250
# of params (million)	47.2	32.5	38.3	30.6	40.8	28.3	42.5	47.1	11.5
FPS Evaluation	7.91	10.33	8.36	7.49	9.23	7.75	7.27	8.01	18.62

Table 1: Reduction Experiments with FPS Evaluation

The largest dimension of the inputted image is the parameter that downsizes frames. I choose 800 pixels because memory limitations made operating on full sized images impossible (1920 x1080). This is using the 8GB GPU chosen as the robotic equivalent compute. For this thesis, I will treat 800 pixels as the default largest dimension.

The experiments that changed the number of encoding and decoding layers produced large improvement to inference speed. On average the other sections resulted in a 0.9 Hz improvement. The minimal layers experiment produced a 3.28 Hz change, which displays its significant impact.

Selecting a different object detection backbone resulted in a small improvement in FPS. This is expected from resnet18 having a faster inference speed than its alternative configuration resnet50. The number of feature levels refers to the number of intermediate layers taken from the backbones forward pass, as input into the transformer. Decreasing this initial input causes a decrease of many subsequent calculations. This matches the found result, where this component contributes to a large 2Hz increase. The embedding size, dimension of feedforward layers, and number of query slots have been shown to produce relatively minor improvements in inference speed. The number of parameters for the model appears to not directly correlate to lower inference speed. This can be seen by comparing experiments 3 and 6.

Overall, the “small model”, shown as experiment 9, produces an inference speed of 18.62 FPS. This is an 11 FPS improvement from the large model. The model has not met the required real-time capability, but I went forward using it as the proposed model. It will be used in the knowledge distillation experiments. Its severe reductions were expected to impact its training capacity, resulting in worse tracking performance. Therefore, the model would benefit from an effective method of training that leverages the original fully trained TransTrack (experiment 1).

The below graph depicts the large and small model. Their forward pass is separated into 4 categories to illustrate where the inference speed improvements lay. The largest section of the large model is the “main section” which refers to all processes separate to the backbone, and backbone feature preprocessing steps. This separation helps illustrate that backbone selection alone, with more SOTA object detectors, is not sufficient to make a MOT model real-time. The large model shown here, would only reach 10 FPS inference speed under an ideal scenario where the backbone took 0 time. This is not promising regarding the fact that most SOTA object detectors operate with a similar inference speed to the resnet18, used in the small model.

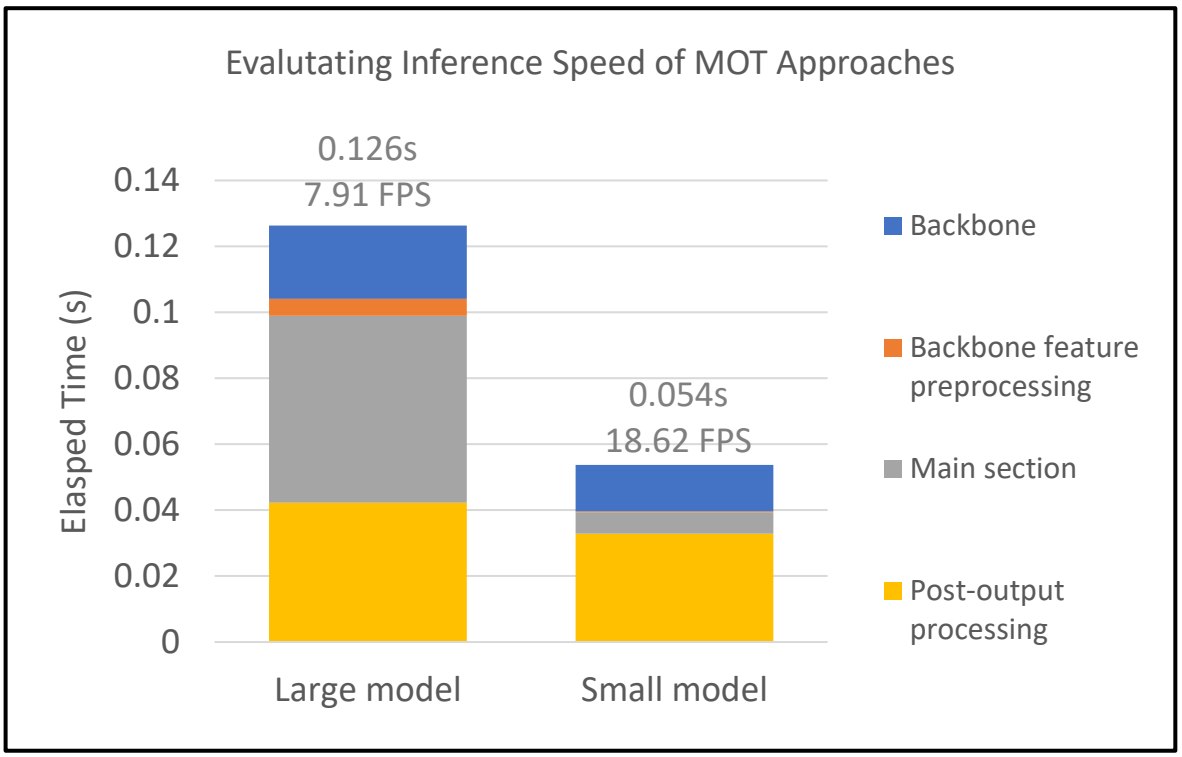

Figure 3: Segmented FPS Evaluation of MOT Approaches

Another result of this FPS evaluation was that it was observed that if the model “goes cold”, inference speed decreases. This refers to the time spent outside the models forward pass will decrease the forward passes speed when eventually started. This was seen with the FPS evaluation program, where annotating images were done after each iteration. This was excluded from the inference speed timing, but still impacted the speed of the model.

4.2 Reduced Model Architecture

The result of reducing TransTrack to reach real-time is the “small model”. It was produced by experimenting with changes to the components of TransTrack. A comparison to the original TransTrack model, now referred to as “large model” is shown in the below table.

Components	Large Model	Small Model
Object Detection Backbone	resnet50	resnet18
Number of feature levels (intermediate layers of backbone)	4	1
# of encoding layers in the transformer	6	1
# of decoding layers in the transformer	6	1
Number of query slots	500	250
Size of the embeddings (hidden dimension of the transformer)	256	64
Intermediate size of the feedforward layers in the transformer blocks	1024	512

Table 2: Proposed Reduced TransTrack Architecture

This below table compares the number of trainable parameters each model incorporates. It displays the 76% reduction between large and small models. This will affect the small model’s ability to learn to solve the MOT task as effective as the large model.

Model Name	Number of Parameters
Large model	47203410
Small model	11545594

Table 3: Proposed Reduced TransTrack's Trainable Parameters Comparison

4.3 Training Improvement from Knowledge Distillation

Loss curves were produced from the small model, with and without KD, from their 150 epochs of training. This training took 6 hours per 100 epochs using a single 8GB RTX 2080 GPU.

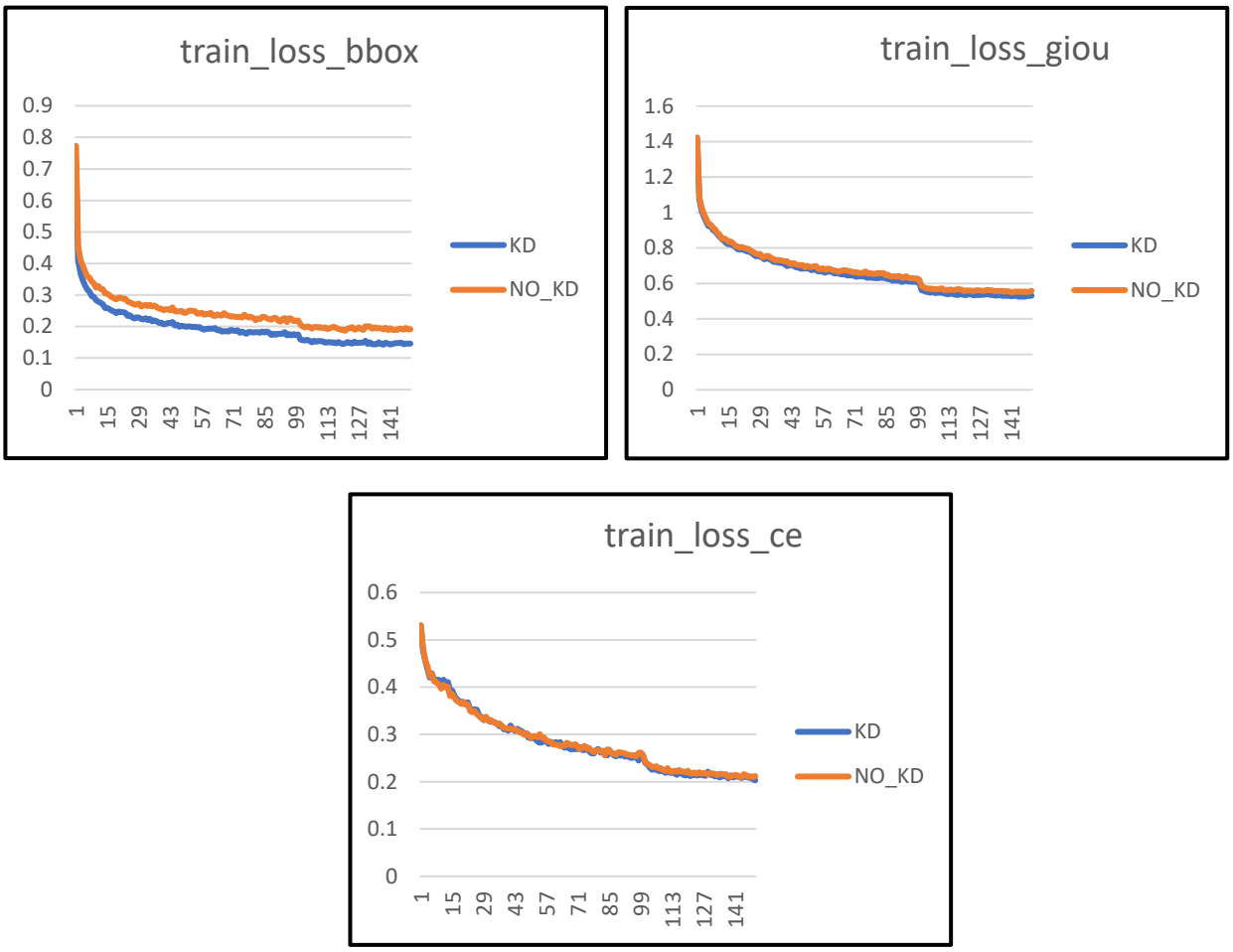

Figure 4: Training Loss Curves Comparing KD and Normal Training Approaches

These curves show that the small model can learn better from the proposed KD method than using the original method (relative to each of their datasets). However, the effect of using knowledge distillation is relatively small.

The bounding box (bbox) loss produces the greatest improvement with KD. The generalized IOU (giou) and class error (ce) loss in comparison have a very slight improvement. This can be understood because the overall loss gives a greater weight to the bbox loss, relative to the others (5, 2, and 2).

A checkpoint from the model could not be used as a starting point for training. This is because significant changes were made to the model. Our training was completed, with no prior information passed into the weights, except backbone pre-training from the torch implementation.

4.4 Quantitative Analysis of Tracking Performance

The below table compares the fully trained small models and large model (supplied checkpoint). The fully trained original TransTrack model chosen for this comparison was one that was not pre-trained and completed 500 epochs of training on MOT17. This allowed for a more direct comparison, because I could not pre-train the small models from limitations in compute.

Models	MOTA	MOTP	FPS
Large (Supplied; 500 epochs)	61.90%	80.00%	7.9
Small + KD (150 Epochs)	34.50%	24.00%	18.6
Small + No KD (150 Epochs)	33.10%	24.40%	18.6

Table 4: Tracking Performance of Fully Trained Large, Small (KD + No KD) Models

The 150th epoch was chosen as the fully trained point for the small models, because of lacking compute and little improvement observed. A minor increase in tracking performance was observed between this point and the 250th epoch (34.5% to 35.7%) of the KD trained small model.

Comparing the MOTA scores of these models show that the reducing the original TransTrack model had a significant effect on its MOT performance. Without knowledge distillation it roughly halved in performance.

The success or failure of these small models can be determined from comparing to other real-time MOT solutions evaluated on MOT17. These are mostly not SOTA and do not have peak

tracking performance. Some examples are FairMOT and ByteTrack, which both are real-time. Their MOTA scores are sufficiently high >75%[30,23]. This could partially be attributed to large amounts of pre-training, but as shown in TransTrack, no pretraining only resulting in 15% decrease in MOTA scores [25]. When compared to a 60% MOTA score, the small models are confirmed to be significantly lacking. In addition, the non-SOTA real-time models achieve ~30 FPS, reaching the real-time requirement the proposed small model did not. It should be noted that their inference speed testing is not clearly explained so there is uncertainty.

The methods to adapt a SOTA MOT solution presented in this thesis is insufficient, with these better performing real-time MOT solutions as a standard. However, the proposed idea is still promising. These real-time approaches are separate from the SOTA that focuses on tracking performance. An adaption procedure is more pro-active than waiting for real-time or robot applicable approaches to be developed, that might or might not be able to match up to the SOTA MOT models' tracking performance.

The proposed implementation of KD was unable to significantly improve the training of the small model. There is only a relatively small 1.4% MOTA increase, while their MOTP scores are similar.

This can be understood by the relatively shallow proposed implementation of knowledge distillation. In the object classification example offered by the authors of KD, they suggested transferring features from the large model to the small [39]. This was an additional loss term that compared the probability distribution each model found over all possible classes. The layer before the softmax that produces a one hot encoding. Intuitively, this method allows the smaller model to gain more rich information about the task.

The softmax output does not have the same benefit in this implementation because there is only one class, pedestrian. This can explain the limited improvement. The performance in multi-class MOT could show better results, but pedestrian only datasets are the standard for the MOT task. Additionally, the proposed MOT KD method only other method to supply better information is from the subpixel bounding box annotations. However, this does not convey significantly more information than the hand-made ground truth annotations.

This result does suggest that building larger MOT datasets from the outputs of MOT models is a possibility. The learning from an artificial dataset was like that of the original. This claim is however weak, because of the low training performance of the KD trained model we cannot understand the upper limits of training possible on. There could be a limit to what a model trained on an artificial dataset could learn. A next step in this regard would be retraining the large model or an equivalent performing model, from scratch, on this artificial dataset.

4.5 Qualitative Analysis of Tracking Performance

Below is an example that shows qualitatively the lacking tracking performance of the small model. The pedestrians detected as #11 and #13 in the first image have their tracks swapped after an occlusion in later frames, like the second image. This is a common issue in low tracking performing MOT models.

Another pair of images are the outputs of the same frames but now annotated by the large model. The identity swaps tracking failures, shown in the previous example, are not present.

A comparison between the outputs of KD and non-KD trained small models is not shown. This is because there is little visual difference in their tracking performance. This is too be expected from the minimal difference in MOTA.

Additionally, it is visually seen that the large model missing frames from its slower inference speed did not affect its tracking performance. It was still very able to outperform the real-time model. This is excluding the complications that having a non-real time model would produce.

Figure 5: Annotated Images of Tracking Failure Example for KD trained Small Model

Figure 6: Annotated Images of Successful MOT example for Trained Large Model

4.6 FPS Evaluation Differences Comparing Local Machine and Google Collab.

Environment	FPS
Local Machine	3.96
Google Collab	1.29

Table 5: Local Machine vs Google Collab Effect on a Model's FPS Evaluation

It was found that the environment used can significantly affect the inference speed of the model. The local machine is equipped with an RTX 2070 8 GB GPU. Google collab supplies a NVIDIA Tesla K80 12 GB that is 4 years older and is slower [44,45]. This difference was observed in the inference speed test producing significantly different results for the same model depending on environment.

All FPS evaluations were completed using the local machine. It's newer GPU would better relate to the available compute on board a robotic system currently. This is also because of limitations in my experimental setup, referring to using the computer I had available.

This GPU hardware improvement in only 4 years shows that improvements in GPU hardware could eventually close the gap of even the large model achieving real-time. However, newer SOTA models continue to be more computationally expensive, so there is still a need to rely on knowledge distillation.

4.7 Key Claims

The key claims this thesis can offer is that the proposed implementation of knowledge distillation was unable to significantly improve the training of a small MOT model. However, the progress shown is evidence of the possibility of KD's application in the MOT space. Additionally, the reduced model produced is not sufficient to be used because of its lacking tracking performance. Finally, the SOTA adaption procedure offered was unable to compare to the current real-time solutions.

5 Conclusion

The thesis work attempted to develop a novel model that is SOTA and robotic applicable. This was not successful. The current proposed model's prospects firmly show that it should not be used. The goal of the work was not reached but the research did allow for gains. It was able to show there is some promise in applying knowledge distillation to the MOT task. The purposed KD method, while shown to produce little benefit, has not explored its true limitations. For example, the maximum possible learning on the artificial dataset. Also, other possibilities for KD MOT training exist. Future work could explore using the large model's intermediate features as a loss. This is a practice present under knowledge distillation for object detection. Another avenue is the work outlined but not explored. This being the development of a method to adapt SOTA 2D MOT solutions to be multi-modal and solve the 3D MOT task.

References

[1] A. Kim, A. Osep, and L. Leal-Taix´e, “Eagermot: 3d multi-object tracking via sensor fusion,” CoRR, vol. abs/2104.14682, 2021.

[2] S. Sharma, J. A. Ansari, J. K. Murthy, and K. M. Krishna, “Beyond pixels: Leveraging geometry and shape cues for online multi-object tracking,” CoRR, vol. abs/1802.09298, 2018.

[3] “Multiple object tracking,” Papers With Code. [Online]. Available: https://paperswithcode.com/task/multiple-object-tracking. [Accessed: 19-Oct-2021].

[4] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention is all you need,” CoRR, vol. abs/1706.03762, 2017.

[5] K. Han, Y. Wang, H. Chen, X. Chen, J. Guo, Z. Liu, Y. Tang, A. Xiao, C. Xu, Y. Xu, Z. Yang, Y. Zhang, and D. Tao, “A survey on visual transformer,” CoRR, vol. abs/2012.12556, 2020.

[6] T. Meinhardt, A. Kirillov, L. Leal-Taix´e, and C. Feichtenhofer, “Trackformer: Multi-object tracking with transformers,” CoRR, vol. abs/2101.02702, 2021.

[7] P. Chu, J. Wang, Q. You, H. Ling, and Z. Liu, “Transmot: Spatialtemporal graph transformer for multiple object tracking,” CoRR, vol. abs/2104.00194, 2021.

[8] Y. Xu, Y. Ban, G. Delorme, C. Gan, D. Rus, and X. Alameda-Pineda, “Transcenter: Transformers with dense queries for multiple-object tracking,” CoRR, vol. abs/2103.15145, 2021.

[9] J. Wu, J. Cao, L. Song, Y. Wang, M. Yang, and J. Yuan, “Track to detect and segment: An online multi-object tracker,” CoRR, vol. abs/2103.08808, 2021.

[10] E. Yu, Z. Li, S. Han, and H. Wang, “Relationtrack: Relationaware multiple object tracking with decoupled representation,” CoRR, vol. abs/2105.04322, 2021.

[11] F. Zeng, B. Dong, T. Wang, C. Chen, X. Zhang, and Y. Wei, “MOTR: end-to-end multiple-object tracking with transformer,” CoRR, vol. abs/2105.03247, 2021.

[12] M. Zhao, K. Okada, and M. Inaba, “Trtr: Visual tracking with transformer,” CoRR, vol. abs/2105.03817, 2021.

[13] N. Wang, W. Zhou, J. Wang, and H. Li, “Transformer meets tracker: Exploiting temporal context for robust visual tracking,” CoRR, vol. abs/2103.11681, 2021.

[14] P. Dendorfer, A. Osep, A. Milan, K. Schindler, D. Cremers, I. D. Reid, S. Roth, and L. Leal-Taix´e, “Motchallenge: A benchmark for single-camera multiple target tracking,” CoRR, vol. abs/2010.07548, 2020.

[15] J. Luiten, A. Ošep, P. Dendorfer, P. Torr, A. Geiger, L. Leal-Taixé and B. Leibe, “Hota: A higher order metric for evaluating multi-object tracking,” International Journal of Computer Vision, 08-Oct-2020. [Online]. Available: https://link.springer.com/article/10.1007/s11263-020-01375-2. [Accessed: 19-Oct-2021].

[16] N. Dardagan, A. Brdjanin, D. Dzigal, and A. Akagic, “Multiple object track-ers in opencv: A benchmark,” CoRR, vol. abs/2110.05102, 2021.

[17] “Evaluatingmultipleobjecttrackingperformance ... - Springer.” [Online]. Available: https://link.springer.com/content/pdf/10.1155%2F2008%2F246309.pdf. [Accessed: 01-Feb-2022].

[18] Yurtsever, Ekim, et al. "A survey of autonomous driving: Common practices and emerging technologies." IEEE access 8 (2020): 58443-58469.

[19] X. Chen, S. M. Iranmanesh, and K.-C. Lien, “Patchtrack: Multiple object tracking using frame patches,” CoRR, vol. abs/2010.07548, 2022.

[20] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft, “Simple online and realtime tracking,” CoRR, vol. abs/1602.00763, 2016.

[21] “An introduction to the Kalman filter - crans.” [Online]. Available: https://perso.crans.org/club-krobot/doc/kalman.pdf. [Accessed: 01-Feb-2022].

[22] N. Wojke, A. Bewley, and D. Paulus, “Simple online and realtime tracking with a deep association metric,” CoRR, vol. abs/1703.07402, 2017.

[23] Y. Zhang, C. Wang, X. Wang, W. Zeng, and W. Liu, “A simple baseline for multi-object tracking,” CoRR, vol. abs/2004.01888, 2020.

[24] “Mot challenge,” MOT Challenge. [Online]. Available: https://motchallenge.net/. [Accessed: 01-Feb-2022].

[25] P. Sun, Y. Jiang, R. Zhang, E. Xie, J. Cao, X. Hu, T. Kong, Z. Yuan, C. Wang, and P. Luo, “Transtrack: Multiple-object tracking with trans-former,” CoRR, vol. abs/2012.15460, 2020.

[26] Chen, Xiaotong, Seyed Mehdi Iranmanesh, and Kuo-Chin Lien. "PatchTrack: Multiple Object Tracking Using Frame Patches." CoRR, vol. abs/2201.00080, 2022.

[27] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, “End-to-end object detection with transformers,” CoRR, vol. abs/2005.12872, 2020.

[28] “THE HUNGARIAN METHOD FOR THE ASSIGNMENT PROBLEM.” [Online]. Available: https://web.eecs.umich.edu/~pettie/matching/Kuhn-hungarian-assignment.pdf. [Accessed: 01-Feb-2022].

[29] T. Zhu, M. Hiller, M. Ehsanpour, R. Ma, T. Drummond, and H. Rezatofighi, “Looking beyond two frames: End-to-end multi- object tracking using spatial and temporal transformers,” CoRR, vol. abs/2103.14829, 2021.

[30] Y. Zhang, P. Sun, Y. Jiang, D. Yu, Z. Yuan, P. Luo, W. Liu, and X. Wang, “Bytetrack: Multi-object tracking by associating every detection box,” CoRR, vol. abs/2110.06864, 2021.

[31] Z. Ge, S. Liu, F. Wang, Z. Li, and J. Sun, “YOLOX: exceeding YOLO series in 2021,” CoRR, vol. abs/2107.08430, 2021.

[32] K. Huang and Q. Hao, “Joint multi-object detection and tracking with camera-lidar fusion for autonomous driving,” CoRR, vol. abs/2108.04602, 2021.

[33] A. Shenoi, M. Patel, J. Gwak, P. Goebel, A. Sadeghian, H. Rezatofighi, R. M. Martin, and S. Savarese, “JRMOT: A real-time 3d multi-object tracker and a new large-scale dataset,” CoRR, vol. abs/2002.08397, 2020.

[34] W. Zhang, H. Zhou, S. Sun, Z. Wang, J. Shi, and C. C. Loy, “Robust multi-modality multi-object tracking,” CoRR, vol. abs/1909.03850, 2019.

[35] “Waymo” [Online]. Available: https://waymo.com/open/#. [Accessed: 01-Feb-2022].

[36] “KITTI” [Online.] Available: http://www.cvlibs.net/datasets/kitti/. [Accessed: 01-Feb-2022].

[37] “nuScenes” [Online.] Available: https://www.nuscenes.org/. [Accessed: 01-Feb-2022].

[38] X. Weng and K. Kitani, “A baseline for 3d multi-object tracking,” CoRR, vol. abs/1907.03961, 2019.

[39] G. Hinton, O. Vinyals, and J. Dean, “Distilling the knowledge in a neural network,” CoRR, vol. abs/1503.02531, 2015.

[40] C. B. Clement, C. Wu, D. Drain, and N. Sundaresan, “Distilling transform-ers for neural cross-domain search,” CoRR, vol. abs/2108.03322, 2021.

[41] C. B. Clement, D. Drain, J. Timcheck, A. Svyatkovskiy, and N. Sundaresan, “Pymt5: multi-mode translation of natural language and python code with transformers,” CoRR, vol. abs/2010.03150, 2020.

[42] P. Blatter, M. Kanakis, M. Danelljan, and L. V. Gool, “Efficient visual tracking with exemplar transformers,” CoRR, vol. abs/2112.09686, 2021.

[43] “Global survey,” arXiv.org e-Print archive. [Online]. Available: https://arxiv.org/. [Accessed: 15-Apr-2022].

[44] “Train your neural network model on Google Colab GPU,” Analytics Vidhya, 28-May-2021. [Online]. Available: https://www.analyticsvidhya.com/blog/2021/05/a-complete-hands-on-guide-to-train-your-neural-network-model-on-google-colab-gpu/. [Accessed: 15-Apr-2022].

[45] “GeForce RTX 2070 vs Tesla K80 - compare now,” SysRqmts.com. [Online]. Available: https://sysrqmts.com/gpus/compare/nvidia-geforce-rtx-2070-vs-nvidia-tesla-k80. [Accessed: 15-Apr-2022].

[46] P. Dendorfer, H. Rezatofighi, A. Milan, J. Shi, D. Cremers, I. D. Reid, S. Roth, K. Schindler, and L. Leal-Taix ́e, “MOT20: A benchmark for multi object tracking in crowded scenes,” CoRR, vol. abs/2003.09003, 2020.

[47] X. Zhou, V. Koltun, and P. Kr ̈ahenb ̈uhl, “Tracking objects as points,” CoRR, vol. abs/2004.01177, 2020.

[48] X. Zhu, W. Su, L. Lu, B. Li, X. Wang, and J. Dai, “Deformable DETR: deformable transformers for end-to-end object detection,” CoRR, vol. abs/2010.04159, 2020.

[49] “CrowdHuman” [Online.] Available: https://www.crowdhuman.org/. [Accessed: 01-Feb-2022].

Appendices