DEPARTMENT OF INFORMATICS

RHEINLAND-PFÄLZISCHE TECHNISCHE UNIVERSITÄT
KAISERSLAUTERN-LANDAU

Master's Thesis in Informatics: Artificial Intelligence

Efficient Object Tracking For Autonomous Driving

Dheeraj Varma Chittari

DEPARTMENT OF INFORMATICS

RHEINLAND-PFÄLZISCHE TECHNISCHE UNIVERSITÄT
KAISERSLAUTERN-LANDAU

Master's Thesis in Informatics: Artificial Intelligence

Efficient Object Tracking For Autonomous Driving

Author:	Dheeraj Varma Chittari
Supervisor:	Prof.Dr.H.C. Andreas Dengel
Advisor:	Abdul Hannan Khan, M.Sc.
Submission Date:	15.11.2023

I confirm that this master's thesis in informatics: artificial intelligence is my own work and I have documented all sources and material used.

Kaiserslautern, 15.11.2023 Dheeraj Varma Chittari

Acknowledgments

I would like to thank my supervisor Prof.Dr.H.C.Andreas Dengel for giving me this opportunity to do a master's thesis under the Smart Data and Knowledge Services department at the prestigious institute Deutsches Forschungszentrum für Künstliche Intelligenz(DFKI), Kaiserslautern.

I express my sincere gratitude to Mr.Abdul Hannan Khan, M.Sc., for his guidance and mentorship throughout the thesis. The discussions and inputs have helped the thesis to finish on time.

I am unconditionally indebted to my mother Mrs.Neeraja Kumari, my father Mr.Sesha Varma, my sister Mrs.Likitha Varma and my grandmother Mrs.Sujana Prabha. I am deeply thankful to Mrs.Hiranmai for all the helpful discussions and support during the entire master's course.

Abstract

In computer vision, object detection and tracking refers to identifying and monitoring objects present in the scene, thus allowing systems to analyze their surroundings through visual inputs. Real-world applications, such as autonomous cars at Tesla and surveillance systems at Honeywell, leverage these techniques to drive their products. Current research trends involve the development of attention-based Vision Transformers to enhance accuracy and reduce tracker latency for real-time processing.

This master's thesis explores the best practices and ongoing trends in the deep learning based object detection and tracking domain. To enhance the object tracking efficiency, we proposed several research assumptions and conducted experiments on the popular CenterTrack framework to justify these assumptions. As the underlying feature extractor plays a primary role in the object tracking framework, we experimented with deeper and denser parallel connection networks, specifically with the ResNet and HRNet backbones for tracking. Recently, the object trackers employing the Vision Transformer architectures are outperforming the convolution block-based architectures. Therefore, we integrated the RepMixer blocks from the FastViT framework into CenterTrack's head network to capture the long-distance dependencies of objects efficiently. Further, changes were made to the tracker module to harness unused tracks and unmatched detections for enhanced object tracking. Finally, we implemented a novel motion in depth estimation network in the CenterTrack tracker, which gives a precise estimate of the collision time for autonomous vehicles. Overall, the research assumptions and the corresponding experiments presented in this thesis reveal new potential directions for researchers to enhance the efficiency of deep learning-based object trackers.

Contents

1 Introduction

1.1 Overview

Object detection and tracking is a technique of uniquely identifying and tracking the key objects in images or video frames. It enables machines and computers to process the objects present in the scene and analyze their movement. This high-level understanding of real-world scenes is necessary for creating intelligent systems that can interact with the environment by taking visual inputs.

In the 1990s and early 2000s, object detection and tracking relied on hand-crafted key features and mathematical modelling. In the 2010s, deep learning models came into prominence and had a significant impact since then. Object detection models have advanced from traditional methods to convolutional-based and Transformer-based models. Meanwhile, object-tracking models have advanced from motion estimation and optical flow models to deep learning-based associations. The current trends in deep learning-based object detection and tracking are:

- Reducing the latency of the object detection and tracking frameworks to enable real-time processing has gained popularity over the last few years. This enables applications like autonomous systems to perform fast split-second decisions.
- Since the inception of attention mechanism [1], models based on Vision Transformer (ViTs) [2] and Hybrid architectures [3] have outperformed the convolutional-based models [4].
- In the last few years, the domain has moved towards multi-object tracking from a single-object, which involves tracking several objects simultaneously with the use of deep association methods.
- Semi-supervised and unsupervised learning methods for object detection and tracking are gaining popularity to reduce the dependency on labelled data.

Object detection and tracking techniques play a crucial role in multiple real-world applications such as autonomous vehicles, surveillance, augmented reality, and robotics. Companies such as Tesla and Waymo rely on these algorithms to enable self-driving cars to perceive their surroundings and plan navigation accordingly as in figure 1.1. Companies like Honeywell and Axis Communications use these technologies to identify any suspicious activity. Sporting clubs use these techniques to analyze individual player's movements in the field.

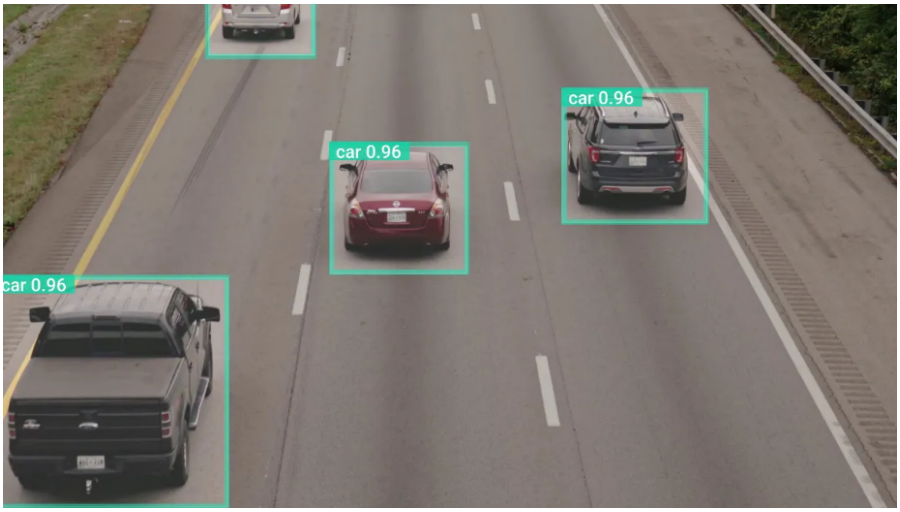

Figure 1.1: Identifying the surrounding vehicles for proper motion planning in autonomous driving [5].

1.2 Motivation

Real-time processing is becoming significantly crucial for object detection and tracking systems due to the demand for faster responses. The low latency of the object trackers is crucial for systems like autonomous driving systems and robotics that need to make quick decisions and respond immediately based on visual data. For example, a self-driving car coming across a child on the road has to immediately come to a halt.

Most of the real-time trackers employ deep learning-based architecture. By relying on neural networks, the trackers achieve strong cues for differentiating between objects. In general, deep learning based object trackers consist of a feature extractor network (or an object detector) and a tracking module. The object detector network produces object detections present in the images. The tracker network estimates object locations and uses these detections to form associations across the frames.

The underlying architecture is critical for balancing the trade-off between tracking accuracy and latency. Deeper networks can generate features that can be robust for tracking in difficult conditions like occlusion or low light. But they require a high computational cost. On the other hand, shallower networks result in a much faster feature extraction but may lack object tracking accuracy. So careful consideration of the architectural design allows for optimizing the trade-off between tracking speed and latency.

As the title of the thesis suggests, *efficient object tracking*, we focus on the strategies that can optimise the tracking latency while maintaining the accuracy. As there is no defined procedure to achieve this trade-off, we formulated several hypotheses and accordingly performed the experiments such as analysing the impact of feature extractors, analysing the impact of the hybrid models and analysing the impact of associating low score detections, trying to find out the best possible technique to answer our research problem of finding the trade-off between

tracking speed and latency.

While numerous deep learning-based multi-object tracking algorithms exist [6, 7, 8], CenterTrack [9] is a prominent object-tracking algorithm for autonomous driving scenarios and human pose estimation tasks. Though centertrack has given remarkable results, there is a continuous pursuit for improving its tracking efficiency. There are several reasons for choosing CenterTrack [9] framework for our experiments. It has generated state-of-the-art performance in various object-tracking benchmarks [10], achieving the best results in tracking accuracy and latency. It adopts a unique approach for tracking objects by estimating the center points of objects analogous to the optical flow technique. This particular approach offers a different perspective than the traditional bounding box association-based trackers. Exploring the center point estimation technique can offer the edge in accuracy, handling of occlusions and precise object localization. It is developed to work in real-time situations, thus making it suitable for low-latency applications such as autonomous systems and robotics. It is an open-source GitHub project [9], providing open access to the source code implementation and allowing users to customize the algorithm according to their needs. The open source facilitates the integration of several components, such as modifying and extending the backbone, neck, head and tracker networks, to further enhance the tracking accuracy and improve the latency. By using the CenterTrack [9] as a base framework, the thesis aims to leverage its advantages and strengths to develop a highly accurate tracking solution that meets real-time requirements.

1.3 Thesis Structure

The thesis begins with the abstract that describes the high-level synopsis of the thesis. The chapter 1 introduction, describes the overview of the deep learning-based object detector and tracker followed by the motivation to perform the research and experiments. The chapter 2 background, contains the history of the computer vision domain and traditional and deep learning-based computer vision techniques. Then various types of deep-learning object detectors are explained in detailed. The chapter 3 related work, presents motion and appearance modelling in object tracking. Followed by listing all the object tracking paradigms and finally describing the CenterTrack [9] architecture. The chapters 4, 5, 6 and 7 contain the information related to several experiments conducted during the thesis. Each chapter begins with a short motivation for performing the experiment, then describes the procedure followed by the experiment setup and results. Finally, the chapter 8 states the conclusion of the thesis.

2 Background

Computer Vision (CV) is a branch of artificial intelligence and machine learning that deals with enabling machines to analyze and interpret the visual world through images and videos. The main goal of computer vision tasks is to understand the scene and context through visual inputs and take necessary actions similar to how a normal human behaves [11]. Some of the real-world applications of computer vision are:

- Autonomous Driving: Scene understanding as presented in figure 2.1, is a key factor in self-driving vehicles that can be achieved through computer vision algorithms [7, 8, 12] making transportation more convenient and efficient.
- Surveillance: Computer vision algorithms play a significant role in security and surveillance analysis. It can detect unusual behaviours in the scene, identify objects precisely and track their movement [4, 14].
- Augmented Reality: Modern gaming devices and gadgets are equipped with augmented reality and virtual reality technologies as in figure 2.2, to enhance the gaming experience for the users [15].
- Healthcare: Computer vision algorithms can be used to analyse Magnetic Resonance Imaging (MRI) and X-ray scans and can detect fractures and tumours through medical image analysis [17].
- Manufacturing: Computer vision enables robots and other autonomous bots to help organise the logistics, navigate the equipment and identify the defects, thus automating the monotonous work and increasing the efficiency [18].

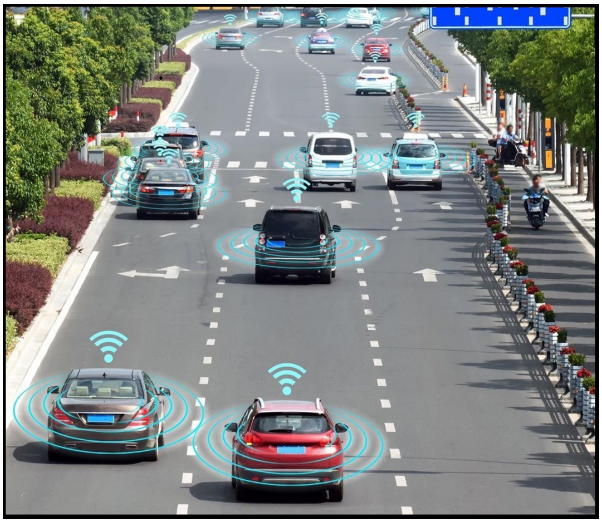

Figure 2.1: Self-driving vehicles motion perception about its surroundings [13].

Figure 2.2: Augmented Reality in gaming [16].

2.1 History of Computer Vision

Computer vision history can be traced back to the 1960s, with researchers started developing techniques for image analysis. Here are some of the important contributions towards computer vision:

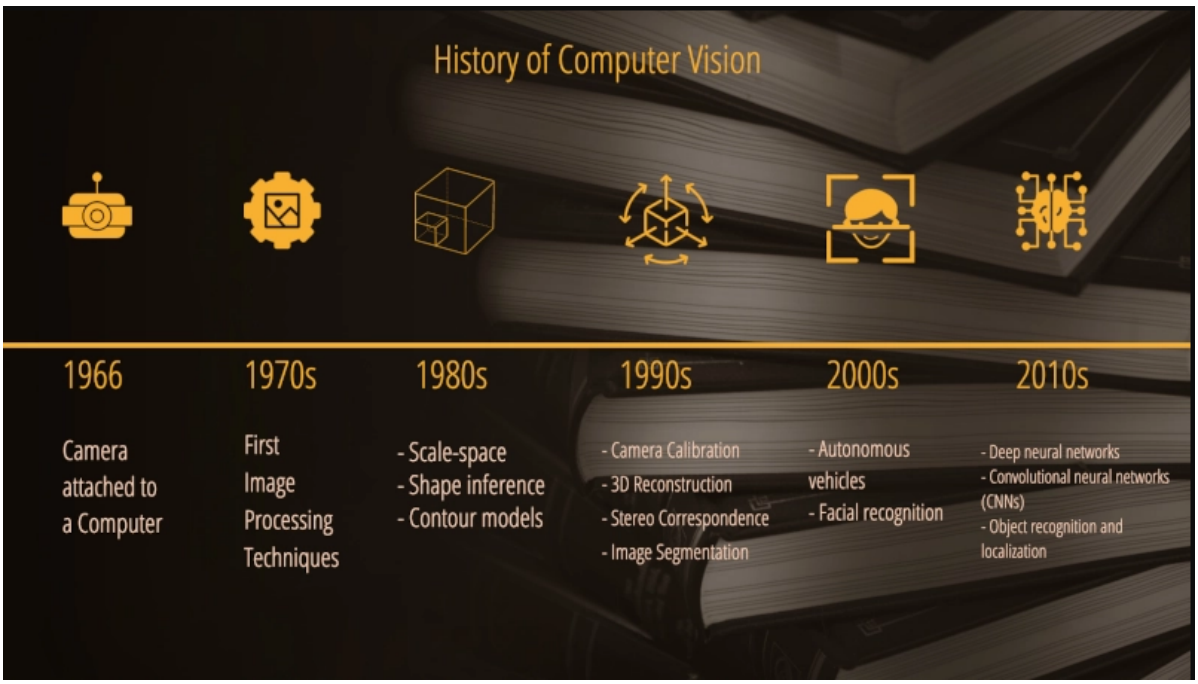

Figure 2.3: Timeline of major contributions in the fields of computer vision [19].

- 1960s-1970s: In this period, scientists were involved in image processing by computers aimed to detect lines and simple shapes [20].
- 1980s-1990s: Research began in the direction of object recognition and motion analysis through key features present in images [21].
- 1990s - 2000s: In this period, camera calibration, 2-dimensional and 3-dimensional image processing and scene reconstruction techniques were developed [22].
- 2000s – Present: There has been significant progress in computer vision in this period, mainly due to the large data availability [23, 24], huge memory and computational capabilities of the computers. Deep learning and convolutional neural networks [25] revolutionized the field of computer vision.

2.2 Traditional Computer Vision Techniques

Traditional computer vision techniques are a set of algorithms based on mathematical modelling and principles to extract necessary information from images and videos. These

techniques do not use deep learning-based neural network architectures [25, 26] to process the data. Some of the traditional computer vision techniques utilise the Gaussian and the Sobel filters [27] for edge detection, and Histogram of Oriented Gradients (HOG) [28] methods for feature extraction as in figure 2.4, template matching for object recognition tasks, thresholding techniques based on the intensity of its pixels for image segmentation tasks and Optical flow and Lucas-Kanade [29] methods for object tracking.

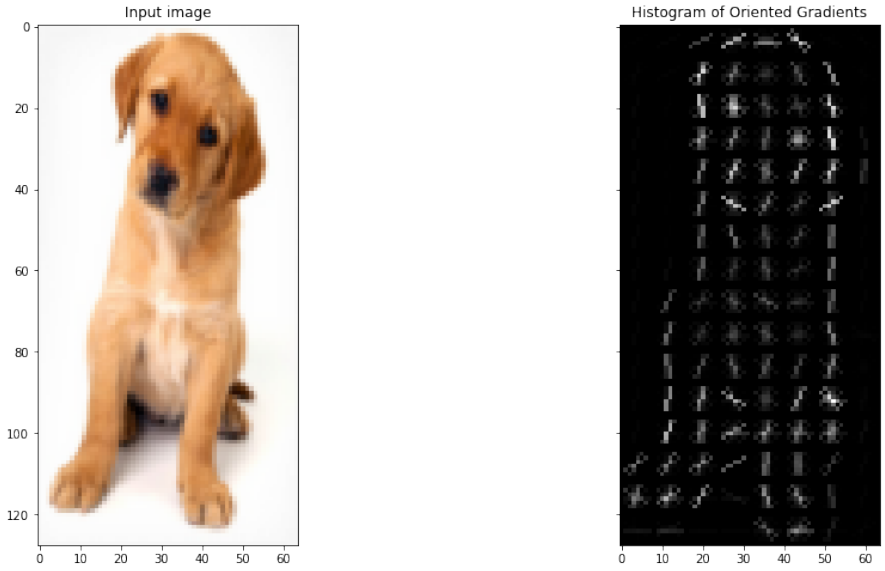

Figure 2.4: Extracting features of an object through a Histogram of Oriented Gradients [28] technique [30].

These traditional techniques are computationally efficient and require minimal data as compared to deep learning methods, making them ideal for low-resource scenarios. These techniques are interpretable as they are based on mathematical formulation [27, 31]. However, traditional techniques require manual feature engineering by an expert and usually struggle to learn the complex pattern in the data thus leading to an inferior performance. Deep Learning-based computer vision techniques can overcome these downsides and produce superior results.

2.3 Role of Deep Learning in Computer Vision

In recent times, the advancements in deep learning have revolutionized the way computer vision tasks are perceived. Especially, Convolutional Neural Networks (CNNs) [25] are the backbone for groundbreaking achievements in the field of image processing and computer vision. Convolutional Neural Networks [25] have the ability to extract and learn the core features from the image data. They are capable of modelling end-to-end learning by mapping the inputs to the corresponding outputs. They can learn very complex interpretations and

patterns in the data that can be used to solve real-world challenging tasks such as object detection and object tracking.

Object detection is the task of classifying and identifying the bounding boxes (location) of the objects present in an image. Deep Learning based object detectors [4, 32, 33] can identify several objects of varying shapes and sizes within the same image and can accurately localize objects (bounding box coordinates or centre points), thus obtaining information about the object's class, size, position, and orientation as in figure 2.5. Deep learning-based object detectors such as Faster R-CNN [4], YOLO (You Only Look Once) [33], and SSD (Single Shot MultiBox Detector) [32] can analyse the visual streams in real-time making them employ for applications like self-driving vehicles and surveillance systems.

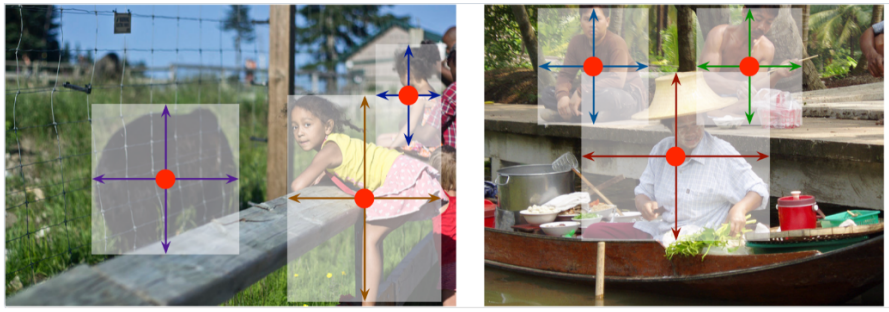

Figure 2.5: Identifying Object's size, position and orientation using CenterNet [34].

Object tracking is calculating the position and velocity of objects over time in continuous image sequences or videos. Deep Learning-based object trackers [6, 7, 8] can handle challenging situations, such as intensity changes, lighting variations and occlusions, more efficiently than traditional methods. They can distinguish various object classes and thus can perform multi-object tracking, allowing multiple objects to be tracked simultaneously. They can be used in real-time, making them suitable for applications like activity recognition, robotic movements and traffic surveillance as in figure 2.6.

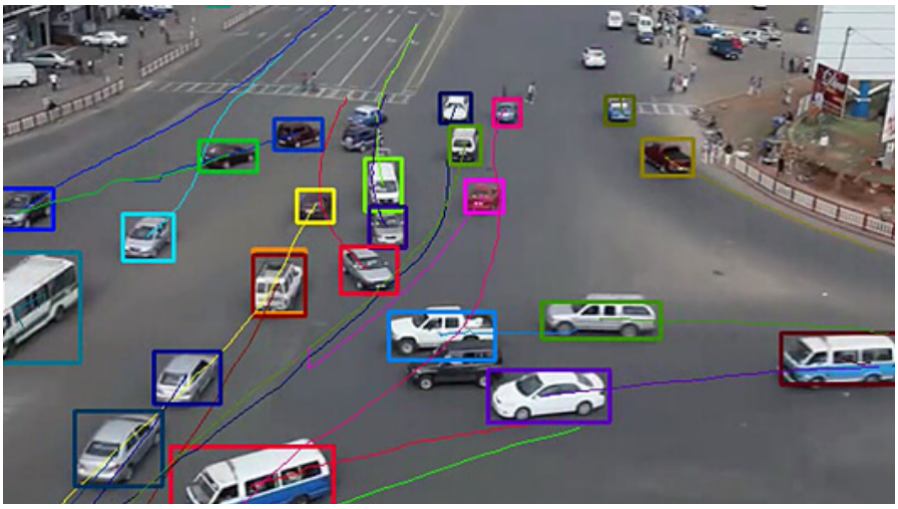

Figure 2.6: Traffic analysis using deep learning [35].

2.4 Deep Learning-based Object Detectors

Prominent frameworks in the history of object detection and their proposed timeline are described in the figure 2.7.

With the emergence of the deep learning paradigm, object detection algorithms relying on deep learning techniques have surpassed the traditional ones in their performance. The subsequent sections review the deep learning-based object detection methodologies and their working principles in detail.

2.4.1 Two-Stage Object Detectors

In two-stage object detection algorithms, as the name suggests object detection happens in two stages. First, the potential candidates are identified as object proposals and in the later stage these proposals are refined for classification and localization.

Region-based Convolutional Neural Network(R-CNN)

In 2014, Girshick et al proposed a Region-based Convolutional Neural Network (R-CNN) [37] architecture for object detection. Initially, the input image is passed to the selective search algorithm [38] to generate object proposals. The proposals are then fed into the pre-trained Convolutional Neural Networks (CNNs) [25] to get the object embeddings. These representations are passed to the Support Vector Machine (SVM) to get the class probabilities. A regression model is trained separately to improve the localization accuracy.

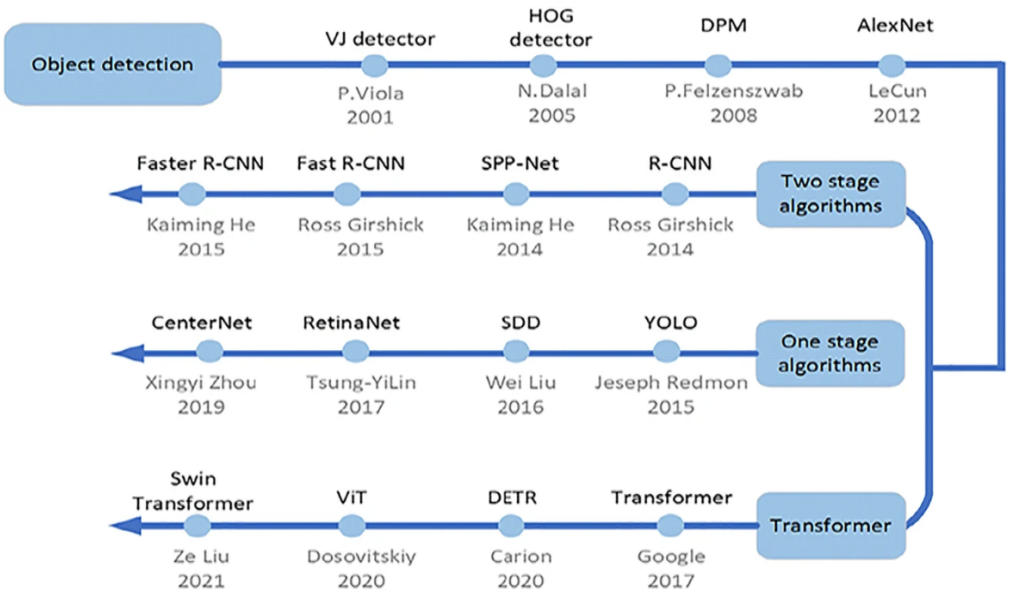

Figure 2.7: Timeline of object detection algorithms based on their working principle[36].

Fast R-CNN

In 2015, Girshick et al proposed another framework that is Fast R-CNN [39] that integrates the Region-based Convolutional Neural Network (R-CNN) [37] object detection modules into a single pipeline. The Region of Interest (RoI) pooling layer is introduced to obtain the same dimension object proposals. These proposals are passed to the pre-trained Convolutional Neural Networks (CNNs). Softmax activation and bounding box regressor are added to the last layer of the architecture for classification and object localization respectively as shown in figure 2.8. However, the model is not end-to-end trainable because of the selective search algorithm [38] used for object proposals.

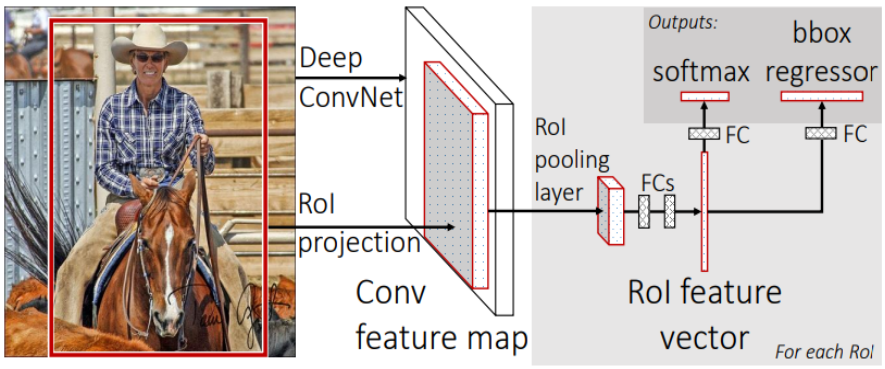

Figure 2.8: Architecture of Fast R-CNN object detector[39].

Faster R-CNN

Ren et al, introduced a Faster R-CNN [4] object detector to overcome the downsides of Fast R-CNN [39]. Instead of a traditional selective search algorithm [38], a deep learning-based Region Proposal Network (RPN) is introduced for object proposals. Faster R-CNN [4] is end-to-end trainable. The workflow of Faster R-CNN is as follows:

1. First, the input image is passed to the pre-trained Convolutional Neural Networks [25] to extract the feature maps.
2. The feature maps are passed to the Region Proposal Network (RPN) as in figure 2.9. RPN uses a sliding window approach to extract object proposals using anchor boxes at three different scales and aspect ratios [4].
3. The Non-Maximum Suppression (NMS) technique is used to handle similar overlapping proposals. NMS rejects all the proposed boxes with a confidence score of less than a certain threshold. In the next stage, the proposal boxes that have a high Intersection of Union(IOU) score greater than 0.5 with the highest confidence score proposal boxes for that region are ignored [4].
4. Then Region of Interest (RoI) pooling is applied to the object proposals to obtain the exact size outputs.
5. The outputs are fed into several fully connected layers followed by the classification and regression head.

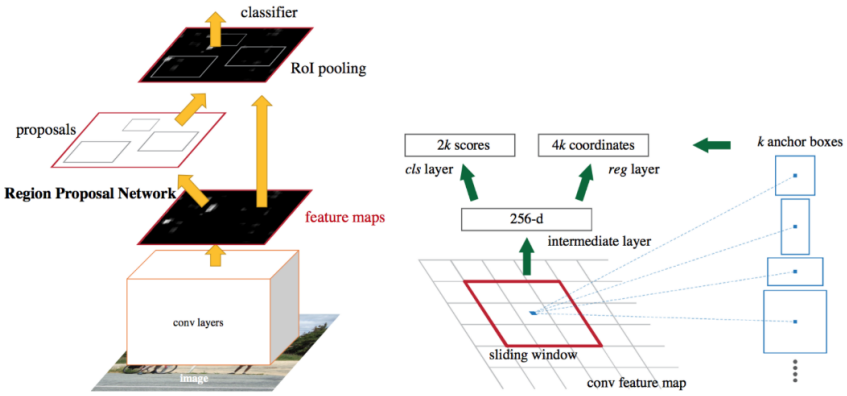

Figure 2.9: Faster R-CNN High Level Architecture [4].

Mask R-CNN

In 2017, He et al introduced Mask R-CNN [40] which extends the Faster R-CNN [4] to perform image segmentation tasks. An additional mask head is added to the Faster R-CNN

[4] network to capture the instance and semantic segmentation. To overcome the quantization errors due to Region of Interest (RoI) pooling, they proposed to use Region of Interest (RoI) alignment. RoI alignment is based on the bilinear interpolation technique for mapping the feature back to image coordinates [40].

2.4.2 Single-Stage Object Detectors

Unlike two-stage object detectors [4, 39], object proposals and refinement happen at a time in the single-stage object detectors [32, 33].

You Only Look Once (YOLO)

The You Only Look Once (YOLO) algorithm[33] is a very fast real-time detector. YOLO [33] doesn't have a region proposal step and it infers from the limited set of grid samples. Thus, it has a very efficient run time compared to two-stage object detectors. The algorithm works as follows:

1. The image is initially divided into several grids of cells as in figure 2.10. Each cell serves as a candidate for identifying the objects.
2. If there is a likelihood of an object being detected in the cell then the objectness score is calculated.
3. It then predicts the class and bounding box coordinates for the object present in the cell.
4. After finding the class, location, and objectness score, it performs a post-processing Non-Maximum Suppression (NMS) step to eliminate the overlapping detections and only one box with the highest confidence score localizes the object.

Single Shot Multibox Detector(SSD)

The Single Shot Multibox Detector (SSD) [32] is one of the early deep learning-based object detection algorithms to use the concept of feature pyramids (considering objects at various scales) for the accurate detection of different sizes of objects. SSD [32] employs VGG16 network [41] for feature extraction and adds several convolution blocks [25] on top of VGG16 network [41] of varying sizes. Each layer can be considered as a varying-scale pyramidal structure as shown in figure 2.11.

Single Shot Multibox Detector uses anchor boxes for every feature hierarchical layer to obtain the potential objects [32]. These objects are then refined using classification and regression heads. Non-maximum suppression is applied to eliminate the redundant detections.

RetinaNet Detector

The RetinaNet [42] is a single-stage dense object detection algorithm. The two novel building blocks used in this algorithm are focal loss and the Feature Pyramid Network (FPN) [43].

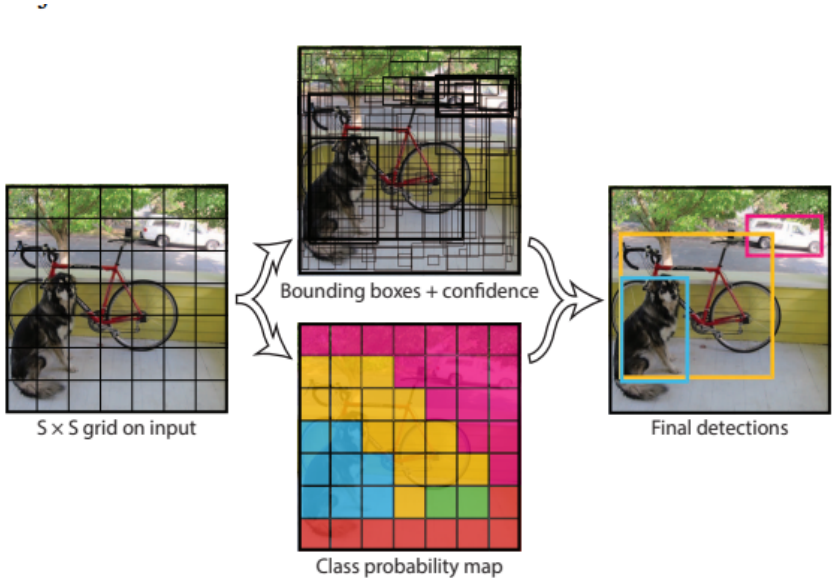

Figure 2.10: High level overview of You Only Look Once(YOLO) detector[33].

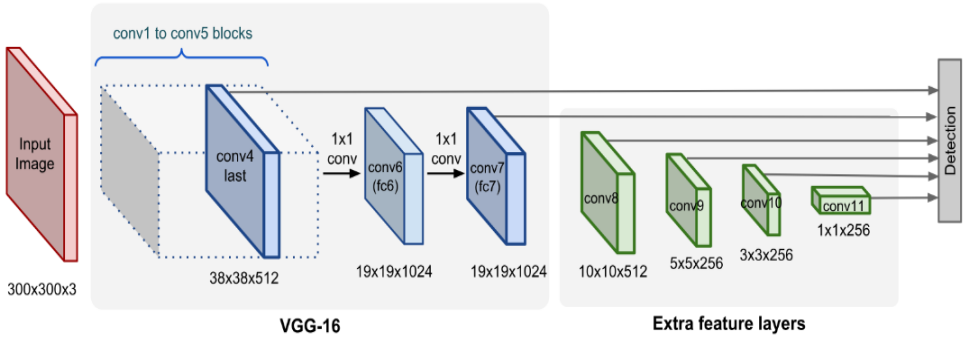

Figure 2.11: Architecture of Single Shot Multibox Detector [32].

Feature Pyramid Network [43] is a multiscale feature representation strategy. FPN [43] integrates feature maps from different layers of a network to create a pyramid of feature maps with different spatial resolutions. The lower levels contain finer-grained details useful to detect smaller objects whereas the top level contains the most semantically rich features. This pyramid structure allows the model to infer objects of multiple scales and various sizes.

Class imbalance is a major problem in image classification and object detection. To address this problem to a certain extent, focal loss is introduced in the RetinaNet detector [42]. Focal Loss down weights the easy-to-classify examples thus giving significance to hard-to-classify examples during training. It helps the model learn from difficult cases and improve accuracy.

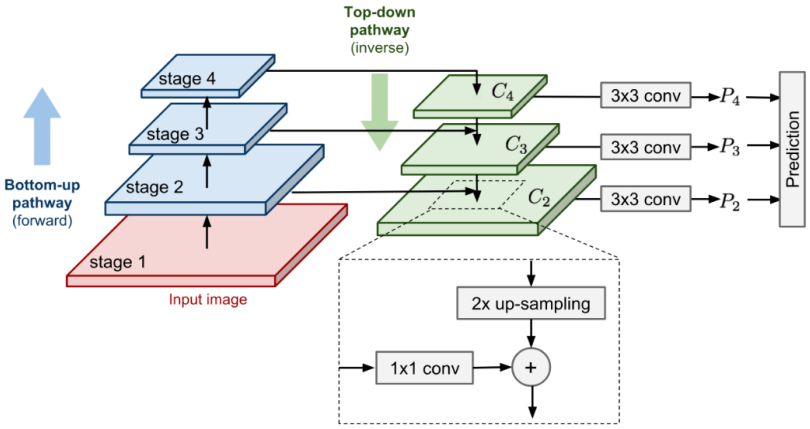

Figure 2.12: Architecture of RetinaNet Detector [44].

RetinaNet [42] generates anchor boxes on every level of the Feature Pyramid Network [43] as shown in figure 2.12, and predicts the class confidence score and bounding box coordinates for each anchor box. It uses non-maximum suppression to eliminate the overlapping predicted boxes.

CenterNet Detector

CenterNet [34] object detectors treat objects as points and detect object centers. CenterNet [34] is a very efficient object detection algorithm that does not employ object proposal networks [4, 40] or anchor box [4, 32, 42] techniques. It predicts a heatmap for the object's center and then regresses the bounding box coordinates. Thus, it is suitable for real-time applications.

CenterNet [34] takes the input image and extracts the feature maps. It employs Law and Deng [45] algorithm for keypoint predictions. The predicted keypoints are in the form of a heatmap. Every pixel in the heatmap gives the likelihood of the center point location of the objects belonging to that class. Each class has a heatmap. The bounding box head predicts the coordinates for the objects. It also employs the offset head responsible for mapping the predicted coordinates in feature maps back to the image space.

2.4.3 Transformer-based Object Detectors

Transformers [1] are the deep learning architecture utilizing the attention mechanism [1] in their encoder-decoder networks. This architecture has been widely used in the Natural Language Processing (NLP) domain. The similar architecture is adapted to the computer vision tasks. The main idea behind using transformers in computer vision is to use their ability to capture long-range information present among the features in an image. This adoption of the attention mechanism [1] has been quite successful in many image and video downstream tasks such as image classification [46, 2], object detection [47] and image segmentation [2].

Vision Transformer

The Vision Transformer (ViT) [46] is one of the early models to adapt the transformer architecture for computer vision tasks. It is a general architecture modelled for image classification tasks and also can be extended to other vision tasks. The working principle of the vision transformer is as follows:

1. Image Partition into Patches: The image is divided into smaller fixed-size patches. The patches are flattened and linearly embedded into a lower-dimension vector representation as shown in figure 2.13.

2. Positional Embeddings: Positional embeddings preserve the positional information of the patches. The positional embeddings are added to the above patch embeddings as in figure 2.13.

3. Transformer Encoder Block: The core part of the vision transformer is the encoder block. It consists of multiple transformer layers with each layer consisting of self-attention mechanisms [1], feed-forward networks, layer normalization and residual connections.

4. Classification Head: Transformer encoder output is passed to a classification head containing several fully connected Multi-Layer Perceptron (MLP) layers. The classification head classifies the input image using the softmax layer at the end.

Detection Transformer(DETR)

DETR (Detection Transformer) [47] is a transformer-based object detector. It is the first end-to-end trainable object detector to use transformer architecture [1]. DETR [47] removes the dependencies to have anchor-based techniques [4, 32, 42] or region proposal networks [4, 40] in object detection algorithms. It has opened up a new direction for the object detection paradigm.

The Detection Transformer is composed of a feature extractor, several encoder-decoder blocks, object queries and uses bipartite matching loss [47] as in figure 2.14. The image is processed by a feature extractor to obtain features. These features are then passed to the encoder blocks to learn the relationship among the features. The decoder takes these

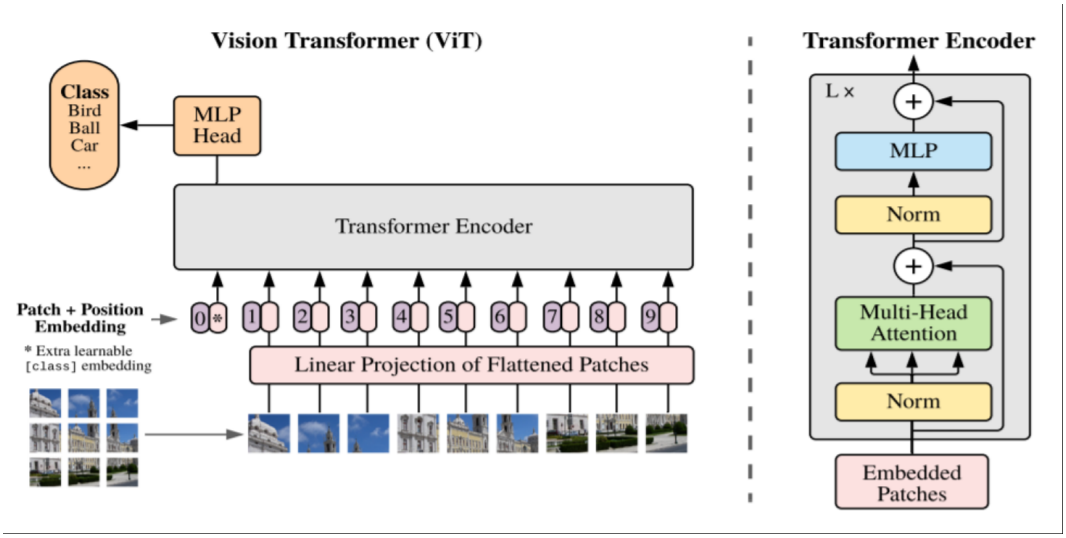

Figure 2.13: Architecture of Vision Transformer[46].

learnt representations and uses trainable object queries to attend to specific regions of the image usually that contains an object. The decoder then predicts bounding boxes and class probabilities for each object query. DETR [47] uses a bipartite matching strategy to assign model predictions to the ground-truth objects and refines the bounding box regressor and classification score.

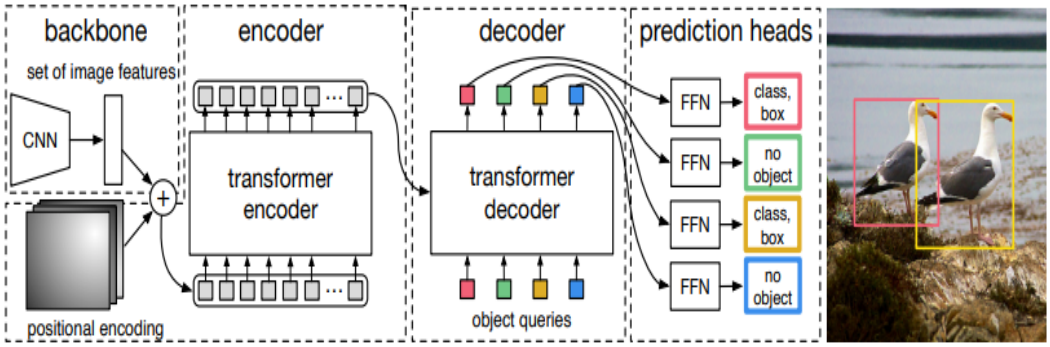

Figure 2.14: Architecture of Detection Transformer[47].

Hierarchical Vision Transformer using Shifted Windows

Hierarchical Vision Transformer using Shifted Windows [2] is a vision transformer-based object classifier that can be used for several vision downstream tasks. The key components introduced by the Swin Transformer [2] are the shifted window-based self-attention and hierarchical transformer architecture.

Shifted window-based self-attention is introduced to reduce the high computational cost required to calculate the self-attention [1] among the image patches. The shifted window-based self-attention computes the attention scores [1] within the local regions of image patches rather than all pairs of image patches. Hierarchical transformer architecture is analogous to the image pyramid networks [43] to process the image at multiple scales. It allows for the effective flow of information across different scales and captures global dependencies present in the image.

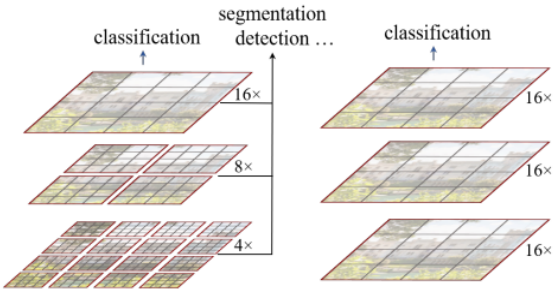

Figure 2.15: Comparision of Swin Transformer's[2] image division technique to standard vision transformer[46] image patch division.

In Swin Transformer [2], the input is partitioned into patches, and each patch is treated as an individual token. These patches are then flattened and fed into the model. It hierarchically processes the patches at different scales as in figure 2.15. Swin transformer uses shifted window-based self-attention to reduce the time complexity of attention mechanisms [1]. The patches pass through several transformer blocks, where each block contains shifted window-based self-attention and Multi-Layer Perceptron layers to capture the local and global contextual information as in figure 2.16. The output of the transformer blocks is passed through a classifier to predict the class probabilities for the input image.

2.4.4 Multi-Layer Perceptron-based Object Detectors

The widely used paradigms for object detection in deep learning are based on Convolutional Neural Networks (CNNs) [4, 32, 33] and Vision Transformers (ViT's) [47]. These frameworks have highly complicated architectures with a lot of intricate details and very sensitive hyper-parameters. It is not an easy task to understand and make any further modifications to these algorithms. Multi-Layer Perceptron-based object detectors [3] try to address this problem by reducing the complexity of the algorithms by employing only Multi-Layer Perceptron (MLPs) blocks and proving to a large extent they are as competitive as the other paradigms in this object detection domain.

Multi-Layer Perceptron-Mixer

Multi-Layer Perceptron Mixer (MLP-Mixer) [3] is one of the early architectures to use only Multi-Layer Perceptron blocks for image classification tasks.

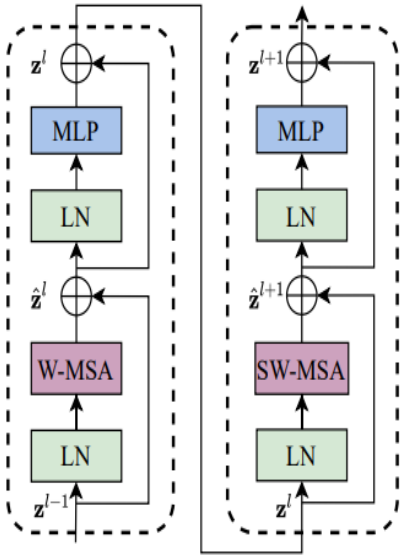

Figure 2.16: Swin Transformer Blocks with regular and the shifted window configurations [2].

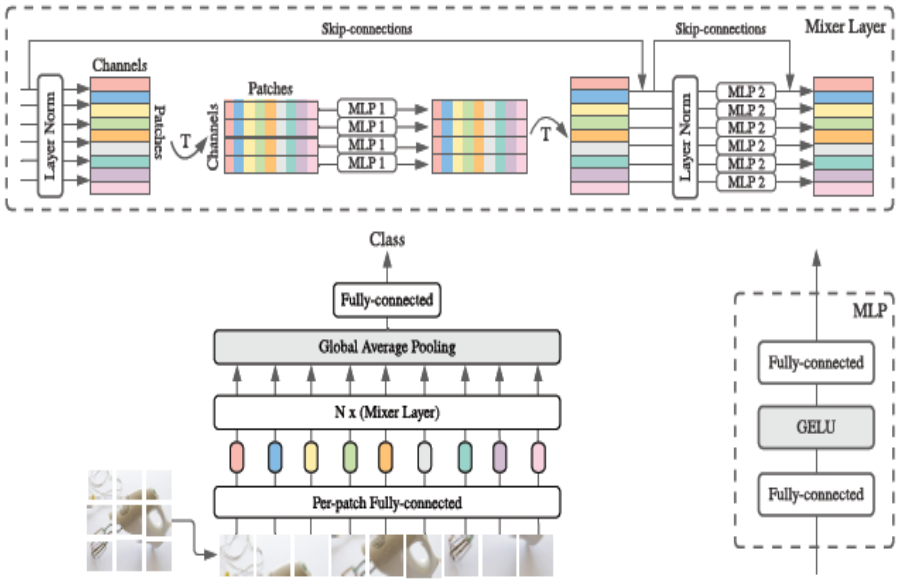

Figure 2.17: MLP-Mixer Architecture [3].

In the MLP-Mixers [3] the input image is divided into a set of patches or tokens. Each token is passed to a fully connected layer to extract token representations. Then, these token embeddings are passed to mixer layers. The mixer layer is the core component of the MLP-Mixer algorithm [3]. Token and channel mixers are the two different kinds of mixer layers that learn the global and local contextual information respectively. Each mixer contains fully connected layers and activation for non-linearity. The MLP-Mixer layers also employ skip connections and layer normalization for gradient flow and stable training [3]. After a series of token mixing and channel mixing layers, the obtained embeddings are fed into fully connected layers followed by the softmax classifier for image classification. The architecture of the MLP-Mixer [3] is in figure 2.17.

MetaFormer Is Actually What You Need for Vision

The authors proposed a novel concept called MetaFormer [48] which is a general architecture for deep learning-based computer vision tasks. It provides an abstract idea from transformers [46] without specifying the token mixer. Over the last few years, the transformers-based architecture models [2, 47] have outperformed the deep convolution-based computer vision frameworks. So far, the efficacy has been attributed to the powerful attention-based [1] token mixer module. Recent research suggests that Multi-Layer Perceptrons (MLPs) [3] can be used as a replacement for this module without compromising on performance.

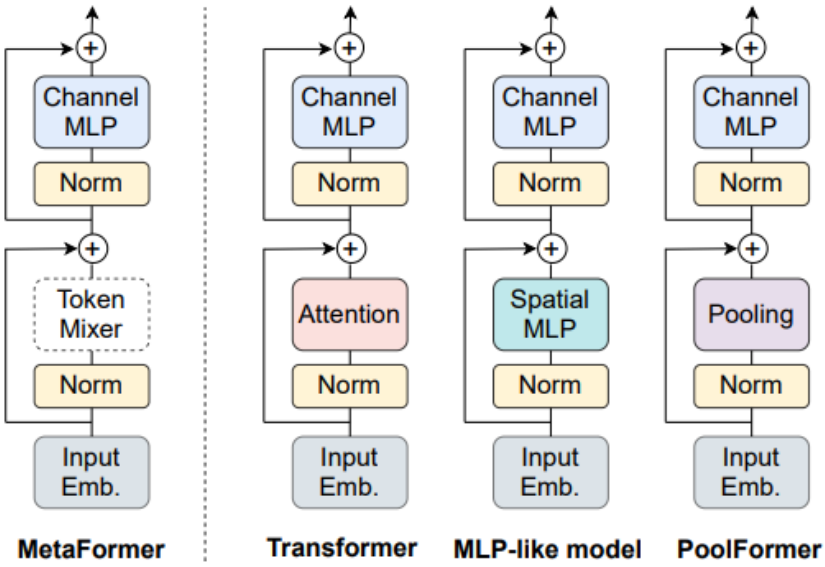

Figure 2.18: General architecture of the MetaFormer along with the attention [1], MLP and pooling based token mixers [48].

The success of transformer-based based architectures is possibly attributed to their general architecture instead of the specific token mixer module. The authors of MetaFormer [48] propose a new model called PoolFormer, as presented in figure 2.18. It uses pooling layers in place of attention blocks and achieves significant results on the ImageNet-1K [23] classification

benchmark. Thus, the general architecture of transformers plays a key role than the specific token mixer network. The results suggest that emphasis should be on the overall architecture, The results suggest that emphasis should be placed on the overall architecture, instead of concentrating on a single component like the token mixers.

3 Related Work

Computer vision tasks such as autonomous driving, traffic analysis, and pedestrian surveillance need to understand the scene and context from a video. In order to analyze a video sequence, we first identify key objects and track their movements over time to estimate their trajectories. This technique is known as Object Tracking. Traditionally, object tracking has been following the key points through space and time.

There is single and multi-object tracking. In single-object tracking, the focus is on analysing a single item in the scene such as analysing a player's movement in a football game. The multi-object tracking The process of multi-object tracking is analyzing a group of objects that fall under the same category. For instance, analysing the vehicle traffic in a particular area. Object trackers can be categorised as online and offline trackers. The online trackers are real-time trackers and have no access to future frames, whereas the offline trackers have access to all the frames in a batch. They are accurate and used for offline video analysis.

The three kinds of object-tracking paradigms are:

1. Traditional tracking: Define and estimate the key points from frame to frame by applying traditional machine learning algorithms.

2. Tracking by Detection: Obtain object detection from a standard detector and then perform the data associations through motion and appearance modelling.

3. Simultaneous Detection and Tracking: The main objective is to model the object detector as the tracker.

Object tracking is one of the difficult tasks in computer vision mainly because of the challenging real-world scenarios such as tracking in low-light conditions, dealing with heavy occlusions or similar appearances, and handling motion blur due to fast object motion or low camera frame rate. Currently, state-of-the-art object trackers focus on enhancing object tracking accuracy by employing robust object detectors and modelling the motion and appearance of the objects accurately.

3.1 Motion Modelling

Motion modelling deals with the dynamic behaviour of an object to predict the position of an object in future frames. The main aim is to reduce the search space of the object thus reducing the time complexity of tracking. Most object trackers possess some kind of motion modelling. Various techniques and filters to achieve motion modelling are:

3.1.1 Kalman filter

Given a series of observations, the Kalman filter [31] estimates the object's state over time. The state of an object is typically represented by the object's position and velocity. This filter models linear projections for a single system. Some of the object trackers employing the Kalman filter are SORT [6] and DeepSORT [7].

3.1.2 Particle filter

The state of an object is constituted as a set of particles. Each particle represents a possible state. The particle filter uses Bayesian statistical inference to update the particle distribution in each frame based on new observations. This method models non-linear projections for a single instance and is more robust than the Kalman filter [31]. Condensation algorithm (Conditional Density Propagation) [49], particle swarm optimization trackers [50] employ particle filter.

3.1.3 Optical Flow

Optical Flow is a computer vision technique to determine the motion of objects by following the displacement of pixels between consecutive image frames. 3.1. This assumes the intensity of an object point remains constant over time and uses this information to estimate the velocity and direction of motion. This method estimates all key points at a time. There are two types of Optical Flow based on a number of selected key points. Sparse Optical Flow selects a few key points and tracks their movement. Examples are the Kanade-Lucas-Tomasi(KLT) tracker [29] and the Mean-shift trackers [51]. On the other hand, the dense optical flow tracks the motion of all points.

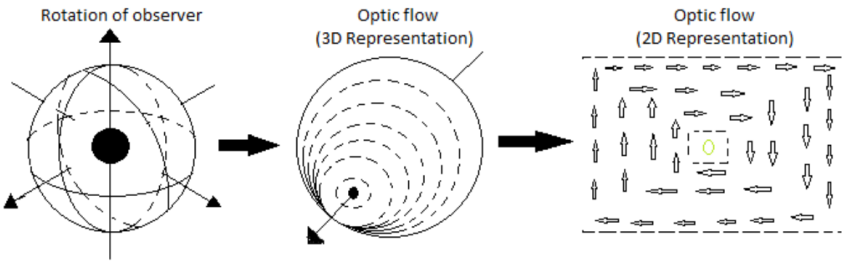

Figure 3.1: Optical flow illustration experienced by a rotating observer. The length and direction of each arrow give the direction and magnitude of the optical flow in each location [52].

3.1.4 Deep Learning based Recurrent models

Recently, recurrent models like Long Short Term Memory(LSTM) [53], and Gated Recurrent Unit(GRU) [54] as shown in figure Figure 3.2, are used to model the motion of objects. Neural

Particle Filter[55] algorithm employs this technique.

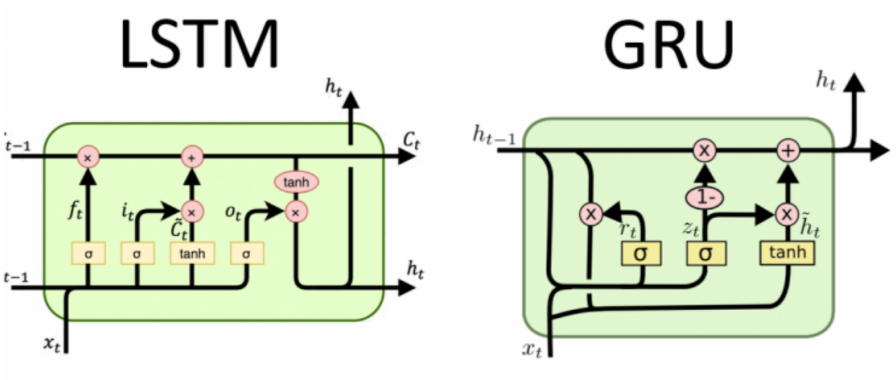

Figure 3.2: LSTM and GRU architecture overview [56].

3.2 Appearance Modelling

Generally, the appearance is modelled as a search retrieval problem. For a given object, we want to find the top matching similar objects in the current frame. We use a similarity learning technique in this scenario. In similarity learning, we minimize the distance for those that are similar objects(true pair) and maximise the distance for dissimilar objects(negative pair).

$$Loss = (Y)(-\log(Y_{pred})) + (1 - Y)(-\log(1 - Y_{pred}) \tag{3.1}$$

The different types of losses employed to model similarity learning are contrastive loss, triplet loss, and group loss. The above equation represents contrastive loss formulation with standard notations.

3.2.1 Correlation filter

Another approach is to use a filter to calculate to correlation score of the object. Using the same filter, identify the object in the next frame that gives a high correlation score. In multi-object tracking, each object requires one separate filter. This method has high time complexity. Some of the tracking algorithms employing correlation filter techniques are High-Speed Tracking with Kernelized Correlation Filters [57].

3.3 Single Object Tracking

Single-object tracking aims to trace the motion of one specific target object as it moves across the frames. Trackers like GOTURN [58] and Learning Correspondence from the Cycle-Consistency of Time [59] perceive tracking as a correspondence-matching task. To reduce high time complexity, they make certain assumptions about the object and it lowers

the search space. Most recent trackers, such as ROLO - Recurrent YOLO [60], models both the motion and the appearance of the object. Recurrent models, including Recurrent Neural Networks(RNNs) [26] and Long Short Term Memory(LSTMs) [53] focus on the motion of an object, whereas Convolution Neural Networks(CNNs) [25] give the appearance information.

3.4 Multi Object Tracking

The multi-object tracking focuses on multiple objects that usually belong to a single category like pedestrians or vehicles and tracks their movement across the frames. Traditional approaches such as High-Speed Tracking with Kernelized Correlation Filters, the detection and tracking of point features employ an algorithm like SIFT [61] to initialize keypoints. The points are tracked across the frames using motion models. They are relatively fast but not very accurate. Following the surge of deep learning algorithms, a new tracking paradigm known as Tracking by Detection has emerged. The trackers in this category employ auto object detectors to receive the localized positions of the objects of interest. As part of data association, then match the obtained detection with the predictions drawn from motion or appearance modelling. Usually, Intersection of Union(IOU) similarity measure values are passed to the Hungarian algorithm [62] to perform the Bipartite matching between the detected and predicted object bounding boxes. Algorithms such as SORT [6], DeepSORT [7], and ByteTrack [12] work on this principle. They are complex and slow but fairly accurate than the trackers using traditional approaches. Nowadays, another paradigm, Simultaneous Detection and Tracking is drawing a lot of attention because of its nature to model the detector as a tracker that enhances the speed of object tracking. Tracktor [8] and CenterTrack [9] work on this principle. They take the previous frame object detections as object proposals or heatmaps as inputs and then regress or estimate the new position of objects under the assumption that the objects have not moved very far or not occluded. These are relatively fast and accurate. However, one needs to account for the assumptions, additionally employing the re-appearance model in case of heavy occlusions.

3.5 CenterTrack Architecture

CenterTrack [9] is a state-of-the-art multi-object tracker that considers the objects as points instead of bounding boxes. It presents a novel approach by simultaneously estimating object centers and regressing their size for bounding boxes. This technique simplifies the object tracking process by having an additional task of estimating the optical flow of the object. All of these are achieved in a single network and do not require a separate tracker network. CenterTrack [9] achieves competitive results in various tracking benchmarks [10] as it processes a rich set of object features obtained from the previous and the current frames as shown in the figure 3.3.

The overall architecture of the CenterTrack framework is shown in figure 3.4. The various components present in the CenterTrack [9] framework are:

Figure 3.3: CenterTrack framework with inputs and outputs [9].

- Input: The input to the CenterTrack [9] model can either be a series of individual images or a video. These images are the inputs that the model will learn and generalize to perform object tracking.
- Backbone Network: The backbone network is a crucial component responsible for feature extraction. The Deep Layer Aggregation (DLA34) [63] is the default backbone for this network.
- Neck Network: The neck network comes after the backbone network and is used to refine the features generated by the backbone. Here, the Deep Layer Aggregation with upsampling (DLAUp) is used as the neck network. It involves upsampling and downsampling layers to further improve the feature representations.
- Head Network: The head network is responsible for predicting the final results, such as object heatmaps, tracking offsets and regressing width and height information from the object centre. Here, a convolutional block [25] is used as the head network.
- Track Association: It uses a greedy approach to associate the tracks with predicted detection boxes at every frame by using the learned tracking offsets from the object's center point.
- Feature Augmentation: In CenterTrack, the current frame features are augmented with the feature from the previous image as well as the previous frame object's heatmap for rich feature information. This is to provide the model with rich information to handle cases where objects may be occluded temporarily.

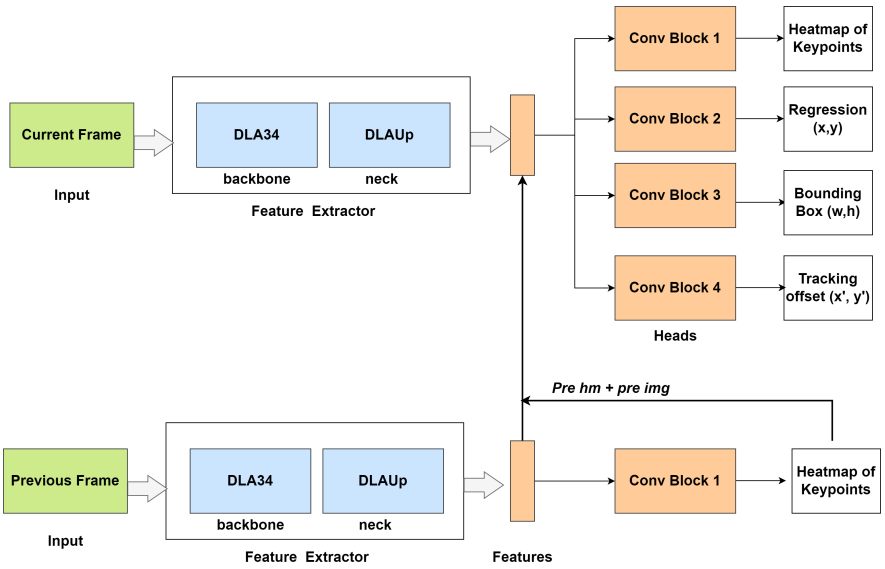

Figure 3.4: CenterTrack framework high-level architecture.

4 Feature Extractors for Tracking

This section starts with a motivation behind researching with comprehensive feature extractors in the CenterTrack [9] tracker. It describes the procedure to replace the default feature extractor of the CenterTrack [9] tracker with state-of-the-art alternatives. Next, a detailed experimental setup is presented, including the key findings and observations.

4.1 Motivation

The performance of a deep learning based object tracking framework depends on the underlying feature extractor network, and refining the feature extraction capabilities could enhance the overall tracking accuracy. We experimented with multiple feature extraction backbones, such as Deep Residual Network (ResNet101) [64] and High-Resolution Network (HRNet-W32) [65], to investigate if deeper and high-resolution backbone could improve the overall performance of an object tracker.

4.2 Procedure

To analyse the impact of the image feature extractor on the CenterTrack [9] object tracking performance, we replaced the default Deep Layer Aggregation (DLA) [63] backbone network with a Deep Residual Network (ResNet101) [64] and Deep High-Resolution Network (HRNet) [65] backbone. The feature extractors are interchangeable modules in CenterTrack's [9] architecture.

ResNet [64] is a convolutional neural network [25] architecture that has skip connections and residual blocks as in figure 4.1 in order to allow for the training of deeper networks. The skip connections maintain the information flow in very deep networks. The ImageNet [23] pre-trained ResNet101 [64] backbone is downloaded from the official PyTorch's [66] implementation. The ResNet101 [64] backbone network is then integrated into the CenterTrack's [9] architecture by adjusting the channel dimensions of the backbone outputs to match the neck inputs. The figure 4.2 shows the ResNet101 [64] backbone in the feature extractor block, while the rest of the components of the CenterTrack framework [9] are kept constant.

Deep High-Resolution Network (HRNet) [65] is a convolutional neural network [25] used specifically to maintain high-resolution representations throughout the network. This is achieved through maintaining various parallel connections at multiple resolutions. In recent times, the HRNet [65] has been used extensively [68, 69] as a feature extractor for tasks like object detection, segmentation and human pose estimation tasks. We replaced the CenterTrack's [9] backbone with an HRNet [65] feature extractor model pre-trained on the

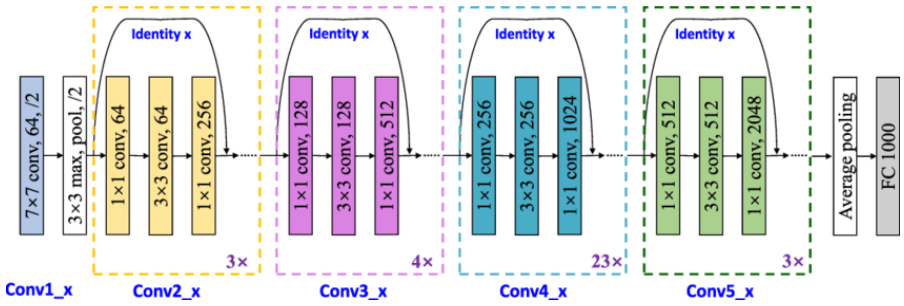

Figure 4.1: Block diagram of ResNet101 feature extractor[67].

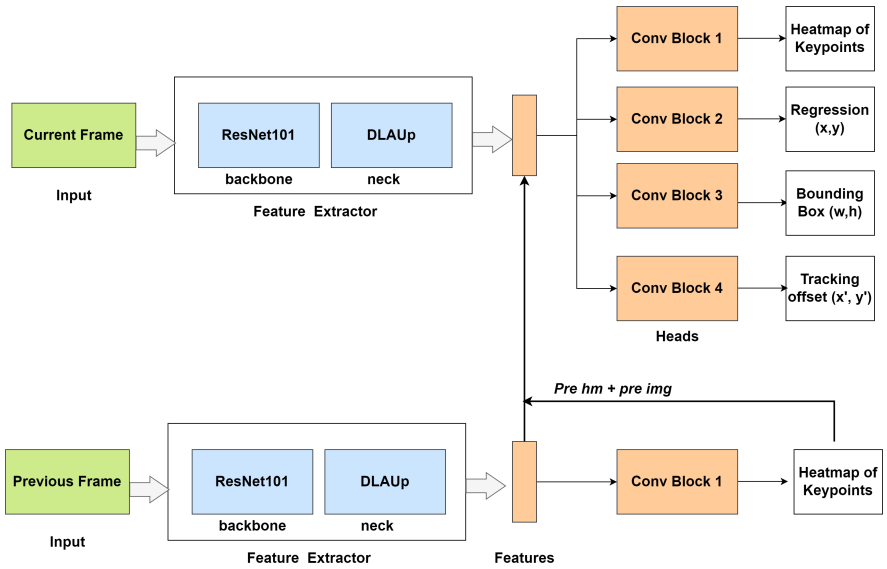

Figure 4.2: CenterTrack framework with ResNet101 [64] as a backbone network.

ImageNet [23] dataset. HRNet-W32 [65] has high and low-resolution convolutions as in the figure 4.3 aligned in parallel in four different stages. The figure 4.4 shows the HRNet-W32 [65] backbone in the feature extractor block, while the rest of the components of the CenterTrack framework [9] are kept constant.

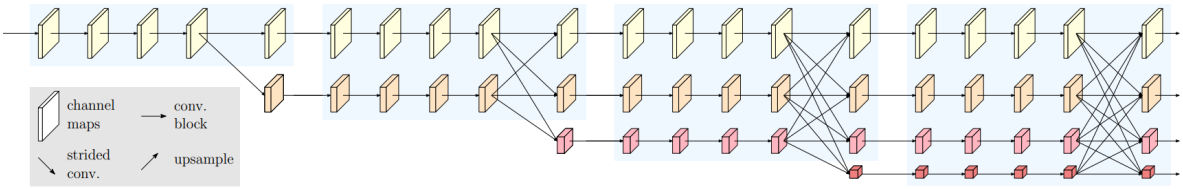

Figure 4.3: Four stages of HRNet feature extractor with different resolutions in each layer[65].

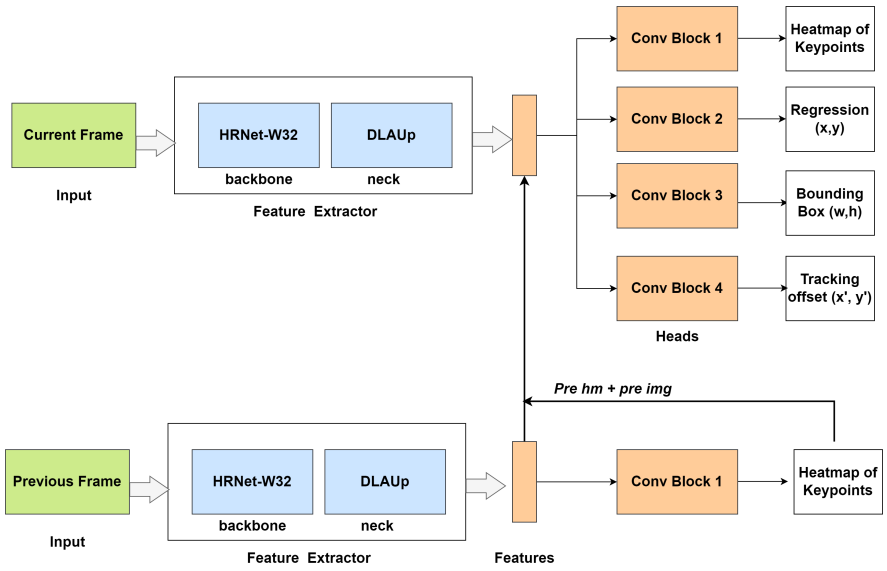

Figure 4.4: CenterTrack framework with HRNet-W32 [65] as a backbone network.

4.3 Experiment Setup

The following section lists the datasets required to perform this experiment and the metrics used to evaluate the results. The complete details about the configurations needed to execute the experiment are presented in the below 4.3.3 subsection.

4.3.1 Datasets

Deep learning based experiments in Autonomous Driving extensively leverage two comprehensive datasets: KITTI [70] and NuScenes [24]. The KITTI [70] dataset has an extensive

Split	Scenes	Frames
Train	21	8,254
Test	29	7,481

Table 4.1: Details of the KITTI Tracking dataset [70] train and test splits.

collection of real-world scenarios captured by sensors mounted on a moving vehicle. Whereas, the NuScenes [24] dataset augments the experimental capabilities by offering a rich set of diverse and complex urban driving scenarios. These datasets contain annotated data for tasks like object detection, tracking, and segmentation. This gives a solid foundation for evaluating the performance of autonomous driving algorithms.

KITTI Tracking Dataset

The KITTI Tracking Dataset [70] is a popular benchmark dataset among object tracking and motion perception scenarios in autonomous driving [8, 9]. KITTI stands for *Karlsruhe Institute of Technology and Toyota Technological Institute* [70]. The KITTI tracking dataset [70] contains a large collection of real-world traffic scenarios captured from a moving vehicle equipped with cameras, Light Detection and Ranging (LiDAR) and Radio Detection And Ranging (RADAR) sensors [71] to calibrate the surrounding object's detection and tracking information. It covers a broad range of traffic scenarios and objects such as cars, pedestrians, and cyclists.

The KITTI Tracking dataset [70] includes data from multiple sensors:

- Velodyne Light Detection and Ranging (LiDAR) Sensor: 3D point clouds obtained from a Velodyne HDL-64E LiDAR [72].
- RGB Camera: Images captured from a stereo colour camera.
- Precise Location Calibration Sensors: Global Positioning System (GPS) and Inertial Measurement Unit (IMU) data.

The train and test split of the KITTI Tracking dataset [70] are in table 4.1. There is no predesignated validation set. There are no publicly available ground truth annotations for the test data. The annotation labels are presented in the table 4.2.

The KITTI Tracking dataset [70] is used for 2D and 3D object detection, tracking, depth estimation and object pose estimation.

NuScenes Dataset

NuScenes dataset [24] is a relatively new dataset introduced in 2019 specialized for computer vision tasks such as autonomous driving. The nuScenes dataset [24] is a large-scale dataset that provides sensor data for perception and planning tasks.

The details regarding the train, validation, and test instances are presented in the table 4.3. These scenes are recorded from the different cities in the USA and Singapore [24]. These

Label	Description
frame	Represents the frame number within a video sequence
track id	It is a unique tracking id given to an object
type	Describes the class of the object
truncated	It is a float representing the object leaving the frame
occluded	Whether the object is fully visible or not
alpha	It is an observation axis of an object
bbox	Bounding Box for an object
dimensions	Object dimensions in x,y and z dimentions
location	Object location in the camera coordinates
rotation	Rotation around the Y axis in the camera coordinate system
score	Object detection score (only in inference)

Table 4.2: KITTI Tracking dataset annotations format [70].

Figure 4.5: Ground Truth train samples of KITTI Tracking dataset [70]. The primary focus of this dataset is the vehicles.

Figure 4.6: Ground Truth test samples of KITTI Tracking dataset [70]. The main focus of this dataset is the vehicles.

Split	Scenes	Frames
Train	700	1.4 million
Validation	150	15,000
Test	150	15,000

Table 4.3: Details of the NuScenes dataset [24] train, validation and test splits.

keyframes are densely annotated with object bounding boxes, tracking information and instance segmentation masks. The test set is used only for evaluation purposes and does not provide ground truth annotations. In addition to the sensor data and annotations, it also provides a range of other information, such as calibrated sensor data, ego vehicle poses, segmentation maps and scene metadata [24].

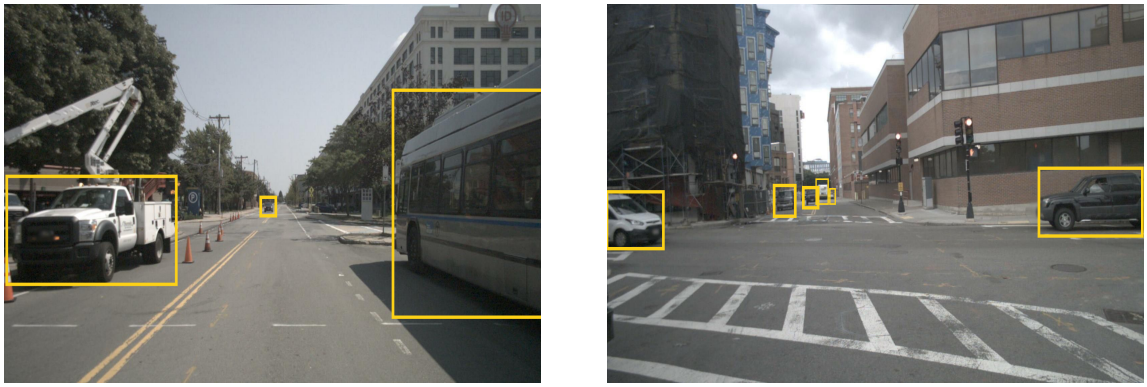

Figure 4.7: Front camera ground truth train samples of NuScenes dataset [24]. NuScenes captures various types of traffic vehicles in different lighting conditions.

In CenterTrack[9] tracker, the NuScenes data [24] is used for pretraining the model for different downstream tasks such as 2D bounding box estimation, 3D depth estimation and object tracking.

4.3.2 Evaluation Metrics

We used Multi-Object Tracking Accuracy (MOTA) [73] to evaluate our experiments, as it is a commonly used metric for multi object tracking. The MOTA is calculated as in the equation 4.1,

$$MOTA = 1 - \frac{(FN + FP + IDSw)}{GT} \tag{4.1}$$

The terms present in the above formula 4.1 are:

- False Positives (FP) - Tracking an item not belonging to the target class.
- False Negatives (FN) - Failing to track an item that belongs to the target class.

Figure 4.8: Back camera ground truth train samples of NuScenes dataset [24]. NuScenes captures various types of traffic vehicles in different lighting conditions.

- Identity Switches (IDSw) - Tracking an object with a wrong identity that is the identity belongs to a different object.
- Ground Truth (GT): It represents the total number of Ground Truth objects.

Other widely used object tracking efficiency metrics are:

- Floating Point Operations (FLOPS): It represents the total floating point computational complexity of a deep learning model. The lower the FLOPS the better the model efficiency on the underlying hardware system.
- Multiply-Accumulate Operations (MACS): It quantifies the total number of multiplication operations the model requires. Most of the deep learning models employ matrix multiplication operations in their computations. Therefore, the number of MACS is directly proportional to the model efficiency.
- Inference Time: In order to employ the tracking model in real-time sensors, the inference time must be as low as possible.

4.3.3 Experiment Details

We employed the default CenterTrack [9] framework implementation for this experiment. As described in the above procedure only the backbone network of the CenterTrack [9] tracker is replaced with ResNet101 [64] and HRNet [65]. The rest of the architecture is unchanged with Deep Layer Aggression [63] upsampling layers acting as neck and convolution block [25] as the head network.

Adam [74] is used as an optimizer with a learning rate of 1.25×10^{-4}. We experiment on the 2D KITTI [70] multi-object tracking dataset with the default input resolution of 1280 $\times$ 384 pixels (width and height) in training and inference. As the KITTI [70] dataset does not provide explicit validation data, the training data is split into almost equal parts for training and validation. The framework's default data augmentation such as random colour

jittering, horizontal flipping and cropping are employed. The remaining hyperparameters are unchanged. The experiment is trained on a single Nvidia [75] A100 GPU with 4 CPUs for 70 epochs with a batch size of 32 which took about 5 hours to train. Nvidia [75] V100 single GPU with 2 CPUs is used to infer the model.

4.4 Results

The CenterTrack [9] model with ResNet101 [64] backbone experiment on the KITTI [70] dataset has resulted in a sub-optimal performance when compared to the CenterTrack's [9] default Deep Layer Aggression (DLA34) [63] backbone, resulting in a decrease of 19.2 % on the car class (or 16.4 points reduction in the MOTA accuracy). Whereas the Deep High-Resolution Network (HRNet-W32) [65] has given a better accuracy over 12.3 % (or over 8.2 MOTA points) increase with respect to the ResNet101 backbone [64]. However, the HRNet [65] experiment also could not match the accuracy of the CenterTrack [9] baseline model. The comprehensive tracking results in comparison to different backbones are presented in the table 4.4

Feature Extractors	Class	MOTA (↑)	FP (↓)	FN (↓)	IDSW (↓)
DLA-34 [63]	Car	83.02	933	906	23
	Pedestrian	61.49	397	1317	13
ResNet101 [64]	Car	66.38	686	2805	197
	Pedestrian	40.93	248	2354	47
HRNet-W32 [65]	Car	74.8	931	1660	166
	Pedestrian	53.4	187	1845	58

Table 4.4: Clear MOT 4.3.2 evaluation for the CenterTrack [9] with different backbone networks on the KITTI Tracking dataset [70].

The CenterTrack [9] model's efficiency, as measured by Floating Point Operations (FLOPS), Multiply-Accumulate Operations (MACS) and inference time, with respect to the ResNet101 [64] backbone is below par compared to the centertrack's baseline model having DAL34 [63] as its backbone, with an increase in FLOPS by 114%, increase in MACS by about 117% and 85% increase in inference time as well. A similar trend is observed with the HRNet-W32 [65] backbone as well. A detailed comparison of the CenterTrack model [9] efficiency with various backbone networks is presented in the table 4.5. The plots 4.9 and 4.10 depict the Multi-Object Tracking Accuracy [73] of the CenterTrack [9] framework when using different backbone networks. The plots present the tracking performance for the pedestrian and car classes over 70 training epochs. The evaluation is performed on the KITTI dataset [70].

From this experiment, we conclude that though the ResNet101 [64] network has more layers, larger model size and thus more parameters as presented in the table 4.5 compared to Deep Layer Aggression network [63], the object tracking results from the table 4.4 are sub-optimal to the CenterTrack's [9] default architecture. Whereas the High-Resolution Network [65] network has parallel connections at multiple resolutions and resulted in a better performance

Feature Extractors	FLOPS (↓)	MACS (↓)	Model Size (↓)	Inference Time (↓)
DLA34 [63]	105.56 G	52.65 G	19.81 M	28.4 ms
ResNet101 [64]	224.63 G	111.98 G	98.22 M	52.7 ms
HRNet-W32 [65]	164.15 G	81.77 G	30.53 M	44.6 ms

Table 4.5: Efficiency evaluation of CenterTrack model [9] with various backbone networks.

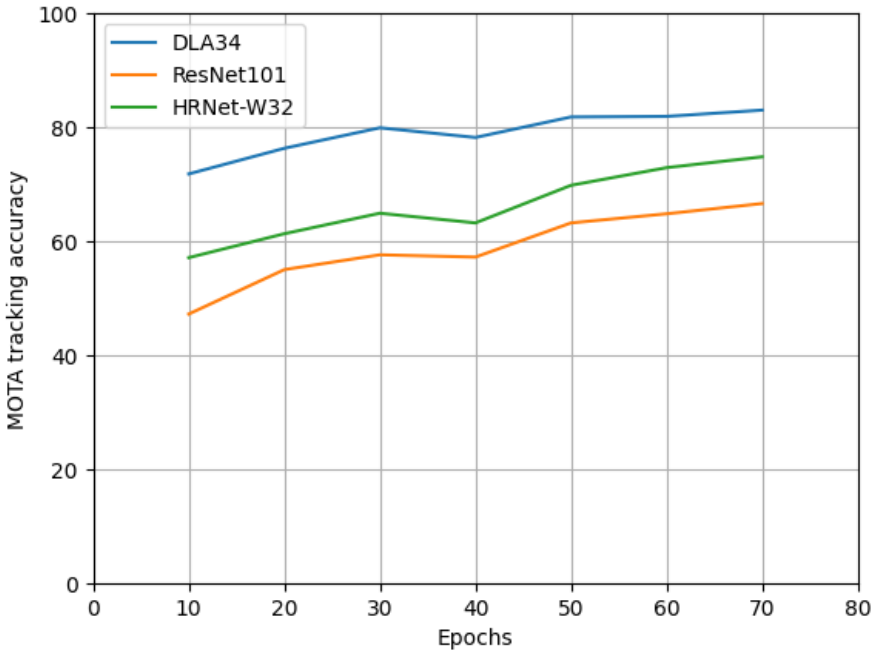

Figure 4.9: Car MOTA with DLA34 [63], ResNet101 [64] and HRNet-W32 [65] backbones on the KITTI dataset [70].

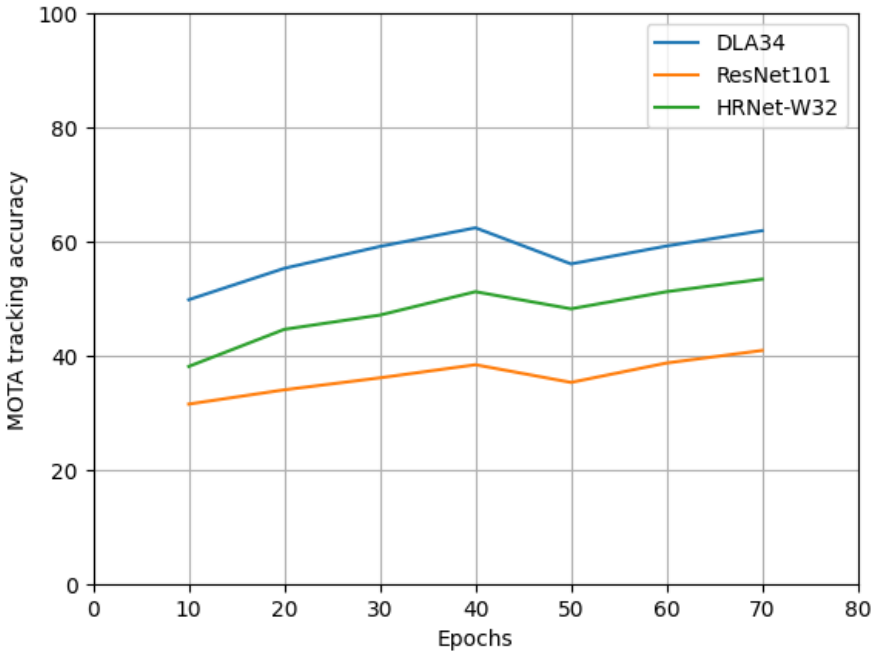

Figure 4.10: Pedestrian MOTA with DLA34 [63], ResNet101 [64] and HRNet-W32 [65] backbones on the KITTI dataset [70].

compared to ResNet101 [64] but below par with respect to the DLA34 [63] backbone. This shows that the presence of parallel connections plays a key role in feature extraction and can be attributed to using appropriate feature pyramid networks like Deep Layer Aggression with upsampling layers [63] suitable to the backbone is a key factor of the object tracking framework.

5 MetaFormers and Object Tracking

This section contains the motivation behind exploring the hybrid vision transformer [46] architectures for CenterTrack's [9] head network. The subsequent section outlines the procedure required to substitute the default convolution [25] head blocks with FastViT's [76] RepMixer Blocks. Following this, the experimental setup and a concise conclusion summarizing the key insights derived from this experiment.

5.1 Motivation

This particular experiment is based on the trade-off between the accuracy and the efficiency of object trackers. In view of the real-time autonomous systems with resource-constrained environments, where efficiency is the key factor, our motive is to modify the CenterTrack [9] head network by replacing convolutional blocks [25] with the hybrid vision transformers blocks present in the FastViT [76] network.

FastViT [76] has the RepMixer blocks (reparameterized blocks) that are more memory efficient than standard convolutional blocks [25]. The RepMixer blocks [76] employ a structural parameterization technique involving depthwise separable convolutions [77] and a feed-forward network that makes it highly memory efficient. The main goal of the experiment is to demonstrate that RepMixer blocks [76] can be used as an efficient replacement for convolutional blocks [25] in the context of computer vision tasks.

5.2 Procedure

Hybrid Vision Transformers models [76, 78] employ both the convolutional [25] layers and attention [1] layers in computer vision downstream tasks such as image classification, object detection and depth estimation because the attention layers can capture the long-range dependencies among the features and the convolutional layers are known for extracting local features from images. This type of combination allows hybrid vision models to learn complex and long-range dependent representations of objects that can lead to improved object detection performance which in turn gives higher object tracking accuracy.

In this experiment, we replaced some of the convolutional layers [25] present in the head network of CenterTrack [9] with FastViT's [76] RepMixer blocks. RepMixer blocks [76] use structural reparameterization to decrease the memory access time and have been more efficient than convolutional [25] blocks. As shown in figure 5.1, we modified the CenterTrack's [9] head network with RepMixer blocks and the rest of the architecture is kept unchanged 5.2.

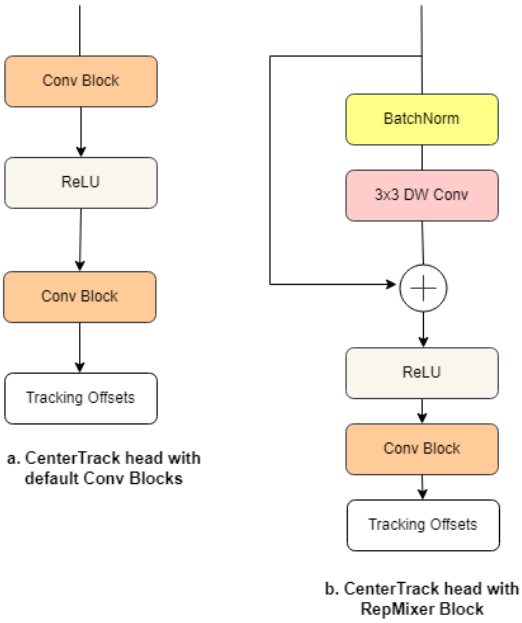

Figure 5.1: The CenterTrack's default head network vs modified head with RepMixer Block[76].

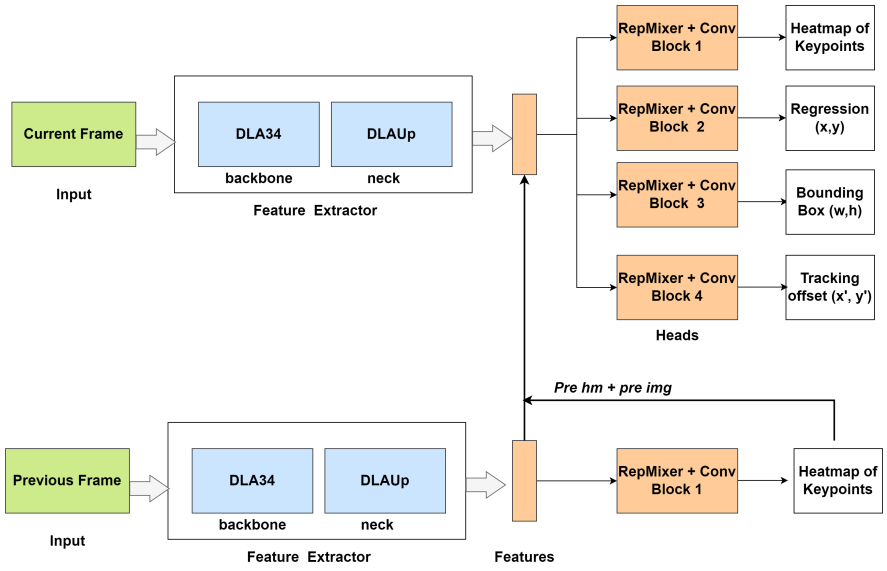

Figure 5.2: CenterTrack framework [9] with Reparameterize Mixer Block [76] and conv layers in the head network.

5.3 Experiment Setup

We used the default CenterTrack [9] tracker configuration except as described in the above procedure, only the head network of the CenterTrack [9] tracker is replaced with the RepMixer block of FastViT network [76]. The rest of the architecture is unchanged with the Deep Layer Aggression, DLA34, [63] as the backbone, upsampling layers with the DLA34 acting as the neck network.

KITTI [70] tracking dataset with an image resolution of 1280 × 384 pixels is used. Adam [74] optimizer with a learning rate of 1.25×10^{-4} is employed during the train. Horizontal flipping, colour changing and random cropping are used as data augmentation techniques. The experiment is trained with a batch size of 32 on a two Nvidia [75] A100 GPU with 4 CPUs for 70 epochs, that ran about 2.5 hours. Nvidia [75] V100 single GPU with 2 CPUs is used to infer the model. The model is inferred on the Clear Multi-Object Tracking (Clear MOT) [73] metrics and computational efficiency metrics as defined in the section 4.3.2.

5.4 Results

The CenterTrack [9] model with RepMixer head [76] experiment on the KITTI [70] tracking dataset has resulted in comparable tracking accuracy outcome with a significant improvement in the tracker efficiency as measured by the Floating Point Operations (FLOPS), Multiply-Accumulate Operations (MACS) and inference time compared to the default convolution head. There is a 33.33% reduction in the FLOPS, a 32.7% reduction in MACS and a 3.5% decrease in the inference time with respect to the CenterTrack [9] baseline model. The detailed Clear MOT [73] tracking results in comparison to RepMixer [76] head network are presented in the table 5.1. A comparison of the CenterTrack model [9] efficiency with convolution block head and RepMixer [76] head network is presented in the table 5.2. The plot 5.3 depicts the tracking accuracy, as measured by MOTA metric [73], on CenterTrack [9] frameworks involving RepMixer [76] head, towards the car and pedestrian classes on KITTI [70] dataset at different epochs.

Head Networks	Class	MOTA (↑)	FP (↓)	FN (↓)	IDSW (↓)
Conv Block [25]	Car	83.02	933	906	23
	Pedestrian	61.49	397	1317	13
RepMixer Block [76]	Car	80.48	1165	959	17
	Pedestrian	54.82	661	1350	15

Table 5.1: Clear MOT 4.3.2 evaluation for the CenterTrack [9] with RepMixer head [76] on the KITTI Tracking dataset [70].

From this experiment, we conclude that employing RepMixer [76] head in place of the convolutional blocks [25] results in a significant improvement in the model efficiency, 33.33% reduction in the FLOPS and a 3.5% decrease in the inference time 5.2, as they are designed to achieve an attention-like mechanism for computer vision tasks but with lower computation

Head Networks	FLOPS (↓)	MACS (↓)	Model Size	Inference Time
Conv Block [25]	105.56 G	52.65 G	19.81 M	28.4 ms
RepMixer Block [76]	70.33 G	35.05 G	19.23 M	27.5 ms

Table 5.2: Efficiency evaluation of CenterTrack model [9] with convolution and RepMixer [76] head.

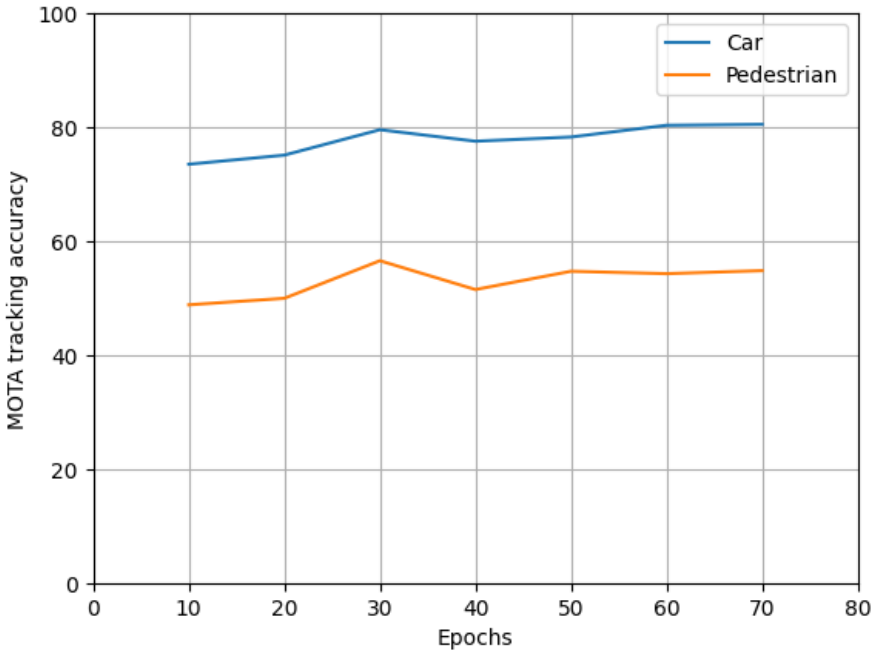

Figure 5.3: Car and pedestrian tracking accuracy of the CenterTrack [9] with RepMixer head [76] over 70 epochs on the KITTI Tracking dataset [70].

costs through structural reparameterization of attention blocks [76]. Even though there is a slight decrease in the Multi-Object Tracking Accuracy (MOTA) by around 3%, the trade-off offered by the significant improvement in memory and computation cost can outweigh the small decrease in the tracking accuracy when it comes to employing them in the fast real-world autonomous systems.

6 Harnessing Unused Tracklets and Unmatched Detections for Enhanced Object Tracking

In this section, we outline the underlying motivation for our research experiment that emphasis on harnessing the potential of unused tracklets and unmatched detections to augment the overall object-tracking accuracy of the CenterTrack [9] framework. The changes required to the tracker module are presented, outlining a detailed procedure for implementing this idea. The section further contains the experimental configuration, followed by a comprehensive summary of key results and findings obtained from the experiment.

6.1 Motivation

This experiment is based on the research question of analysing the impact of unmatched detections present in the tracker module of the CenterTrack [9] framework. The tracking module plays a vital role in object-tracking scenarios. Like other object trackers [6, 7, 8], the CenterTrack [9] utilizes the high-confidence detection boxes and discards the rest of the low-confidence objects. Instead of ignoring the low score detections, ByteTrack [12] initially matches the high score confidence detections to the tracks based on motion modelling. Later, it compares the remaining low confidence detections to unmatched tracks in order to recover false negatives as in figure 6.1, thus improving the overall object tracking accuracy. In this experiment, we perform the concept of matching low-score detections with unmatched tracks for the CenterTrack's [9] tracking module to recover objects that have low detection scores and augment the tracking accuracy.

6.2 Procedure

In the CenterTrack [9] framework, the object detection module keeps track of the list of objects identified in a particular frame. The tracker module has information regarding the tracks that are coming from the previous frame and also is responsible for correctly identifying the current frame's object detections with the previous frame tracks and making an association (that is to maintain constant track-id for the same object). It uses a greedy matching technique to find the correct association pair. If there are unmatched detections with a confidence score greater than a certain threshold then new tracks are spawned whereas the unmatched tracks are kept for a certain number of frames before discarding to potentially recover any of the occluded objects.

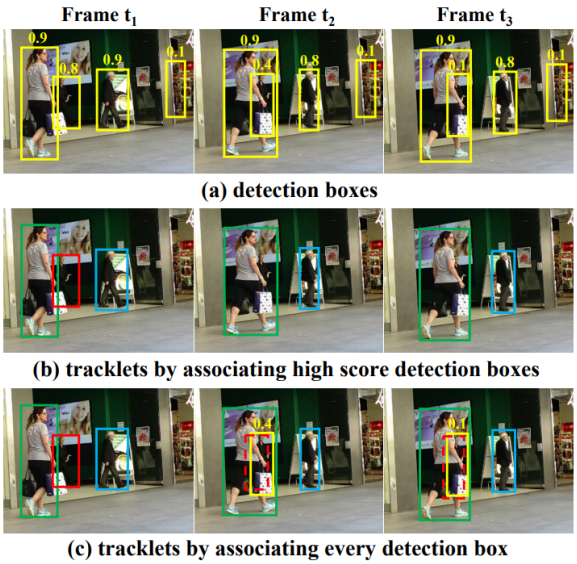

Figure 6.1: Inference of ByteTrack[12] model by associating both the high score and low score detections.

We made a modification to the tracker module of the CenterTrack [9] framework in order to match the low-score unmatched detection in the current frame with their possible corresponding tracks. Track association of low-score object detections is implemented through the following steps:

- Unmatched Tracklets Collection: All the unmatched tracks obtained from the previous frame are stored.
- Unmatched Detections Collection: All the unmatched detections in the current image are collected.
- Association: Associate low score (less than a certain threshold) unmatched detections to the leftover tracks in the current frame based on the closest distance matching, thus regaining the objects with a very low detection score to their corresponding tracks.
- New tracks initialization: Finally, create new tracks for those objects that did not match any tracks and have a confidence score greater than a certain threshold.

6.3 Experiment Setup

In this experiment, most of the components are kept constant as they are in the CenterTrack [9] tracker, except that the object tracking module is modified to handle unmatched detections with the unmatched tracks coming from the previous frame. KITTI [70] tracking dataset as presented in the section 4.3.1 is employed with a default image resolution is used. DLA34 [63]

is used as the backbone with Deep Layer Aggression [63] upsampling layers acting as neck and convolution block [25] as the head network. The experiment is trained on two Nvidia [75] A100 GPU with 4 CPUs for 70 epochs which ran about 3 hours. Nvidia [75] V100 single GPU with 2 CPUs is used to infer the model. The model is inferred on the Clear Multi-Object Tracking (Clear MOT) [73] metrics as defined in the section 4.3.2. Since there is no change in the training network, the model efficiency metrics, like Floating Point Operations (FLOPS), Multiply-Accumulate Operations (MACS) and the total model parameters remain the same.

6.4 Results

The CenterTrack [9] model with the modified tracker module, to include the unmatched detections, on the KITTI [70] object tracking dataset has slightly performed better with an improvement of 1.2% of multi-object tracking accuracy of the car class present in the KITTI [70] dataset, over the CenterTrack [9] baseline model measured by the Multi-Object Tracking Accuracy (MOTA) metric. The detailed Clear MOT [74] tracking results in comparison to the baseline model are presented in the table 6.1. The graph 6.2 represents the tracking accuracy, as measured by MOTA metric 4.3.2, on CenterTrack [9] frameworks with the updates tracker module, towards the car and pedestrian classes on KITTI [70] dataset at different epochs.

CenterTrack Framework	Class	MOTA (↑)	FP (↓)	FN (↓)	IDSW (↓)
Baseline Model	Car	83.02	933	906	23
	Pedestrian	61.49	397	1317	13
Improved Tracker Module	Car	83.99	856	882	18
	Pedestrian	61.93	418	1272	15

Table 6.1: Clear MOT 4.3.2 evaluation for the CenterTrack [9] model with low score detection associations on the KITTI Tracking dataset [70].

From this experiment, we can observe a notable decrease in the false negatives tracking values, with a 2.6% drop in the car class and a 3.4% drop in the pedestrian class of the KITTI [70]tracking dataset. Despite the lack of significant improvement in multi-object tracking accuracy as presented in the table 6.1, the technique to include low score detections can be applied to various deep learning-based trackers that adapt the tracking by detection approach [6, 7, 8]. Instead of completely ignoring these low-scoring detection objects, the approach presented in this section associates them with unmatched tracks in the current frame. This could potentially improve the overall object tracking [73] of the object trackers present in various domains such as autonomous driving, player movement analysis and robotics.

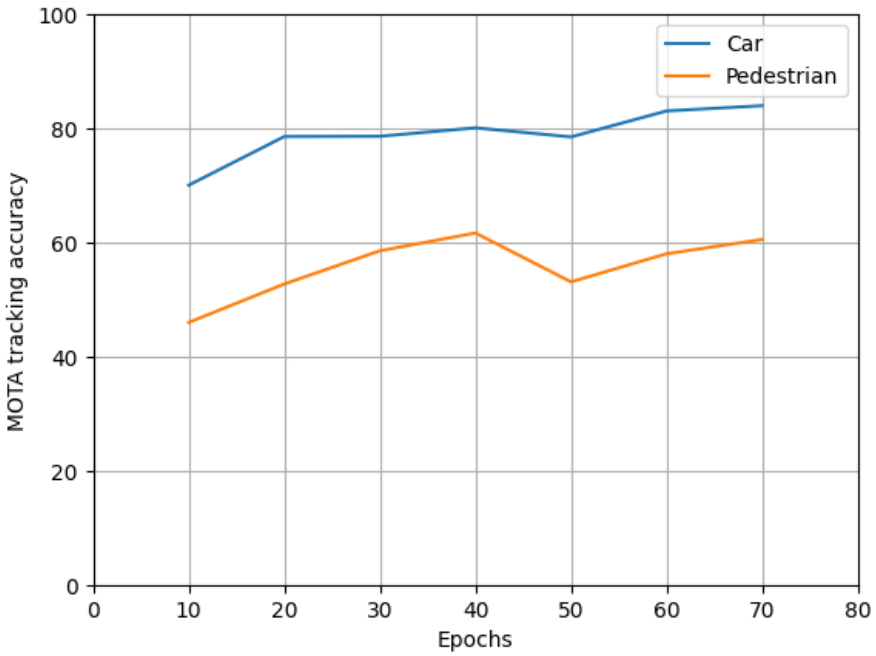

Figure 6.2: Car and pedestrian MOTA of the CenterTrack [9] with low score detections over 70 epochs on the KITTI Tracking dataset [70].

7 Time To Collision Estimation With Object Tracking

This section begins with a motivation behind our research question, examining whether the CenterTrack [9] tracker, equipped to process comprehensive data from the previous and current frames at a time in a video sequence, can enhance the performance of Time-To-Collision (TTC) estimation for autonomous systems. The section presents the idea and subsequently outlines the procedural steps for its implementation, followed by a detailed experimental setup and presentation of results obtained from the experiments.

7.1 Motivation

In Autonomous Driving scenarios, the Motion-In-Depth (MiD) refers to the movement of an object in between the camera frames. It reveals the relative change in depth over the consecutive frames for an object in the scene and is analogous to the optical flow estimation for a key point in the image. This information is crucial for estimating Time-To-Collision (TTC) of autonomous systems. We formulated a novel research insight to adapt the CenterTrack [9] tracker to estimate the motion-in-depth information for a given object. The extensive information that this framework processes at each time frame will lead us to very accurate results in estimating the final time to collision with the surrounding objects in the context of a self-driving vehicle.

The existing frameworks, Stereo Piecewise Rigid Scene Model [79] and Object Scene Flow for Autonomous Vehicles (OSF) [80] frameworks consider stereo images and classifies each pixel to estimate the TTC, whereas the CenterTrack [9] framework accounts for the previous frame's information in addition to the current frame's features in a given video sequence and estimates TTC per each object. Thus, it is much faster as it requires only an additional head network to estimate the MID values.

7.2 Procedure

To estimate the Time-To-Collision (TTC) information, a novel *depth_ratio* head is developed in the CenterTrack[9] network, as shown in figure 7.1. The new head network is used to estimate the Motion-in-Depth (MiD) information that is proportional to the Time-To-Collision. Motion-in-Depth (MiD) is the change in the 3D depth of the object in between the two consecutive frames. The formula to calculate the MiD is given in the equation 7.1, where n is

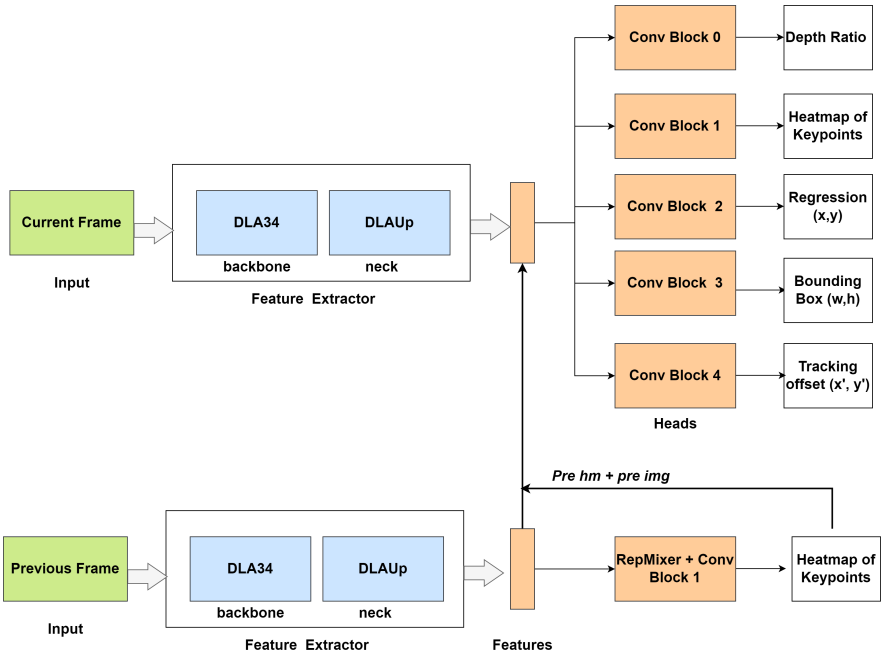

Figure 7.1: CenterTrack framework [9] with an additional motion in depth network.

the estimated depth ratio value and n_{GT} is the ground truth depth ratio value for an object.

$$MiD = |log(n) - log(n_{GT})|_1 \times 10^4 \tag{7.1}$$

The primary hypothesis is that the CenterTrack [9] tracker is designed to process the input from the two consecutive frames (the previous frame and the current one) that can lead to the precise estimation of depth ratio information compared to a single image input network.

Depth-Ratio head is implemented in the CenterTrack [9] framework through the following steps:

- Ground Truth Calculation: The ground truth values for the depth ratio are calculated from the 3D depth information. Most of the autonomous driving dataset [70, 24] capture this information through cameras and sensors [71, 72]. The depth ratio for an object in an image can be determined using the depth information obtained from the two consecutive frames. Several edge cases where the previous frame is missing and the current frame is the first frame in the video sequence are ignored.

- Data Loaders: Used the baseline model's data loader as it is, except that the new field *depth_ratio* is added to the COCO [81] annotations of the corresponding datasets.

- Architecture of the Model: A novel *depth_ratio* head network is added along with existing head networks to predict a continuous motion in depth value. The CenterTrack's [9] backbone and neck feature extractors are kept constant.

- Loss Function and Optimizer: The weighted L1 loss (mean absolute error) is employed to estimate the loss and the default Adam optimiser is used to ensure that the model learns to estimate the Motion-in-Depth information effectively during the training process.

7.3 Experiment Setup

For this experiment, most of the object detector and tracker components remain unchanged, with an additional head network being introduced to capture the $depth_ratio$ information in the CenterTrack [9] framework. We ran the experiments on the KITTI [70] tracking dataset as presented in the section 4.3.1. Deep Layer Aggression (DLA34) [63] is used as the backbone with upsampling layers acting as neck network. Convolution block [25] is employed for the head networks. This setup is trained on two Nvidia [75] A100 GPU with 4 CPUs for 70 epochs which ran about 3.5 hours. Nvidia [75] V100 single GPU with 2 CPUs is used to infer the model.

7.4 Results

Summary of results of Motion-in-Depth (MiD) values using CenterTrack network [9] on KITTI [70] dataset is presented in the table 7.1, along with the other state-of-the-art techniques such as Stereo Piecewise Rigid Scene Model [79] and Object Scene Flow for Autonomous Vehicles (OSF) [80]. As the name suggests, the Stereo Piecewise Rigid Scene Model uses a piecewise scene model to capture and estimate the motion of objects present in a scene, generating a better 3D scene flow estimation [79]. In the OSF [80] framework, the object 3D motion and the shape are calculated simultaneously, thus it presents the autonomous system to account for the surrounding objects accurately. In general, the frameworks that estimate the optical flow precisely for the key points present in an image would yield better results.

Our approach of adding an additional head network to the CenterTrack [9] framework has yielded significantly better results in estimating the depth ratio information on the KITTI [70]

Experiment	MiD (↓)
Stereo PRSM [79]	124.0
OSF [80]	115.0
BiTTC [14]	120.0
BiTTC (*) [14]	124.0
Ours	115.9
Ours (*)	107.5

Table 7.1: Comparision of the MiD values on the KITTI tracking dataset [70]. (*) indicates a special case of considering only those objects that are moving towards the ego vehicle.

dataset compared to the other state-of-the-art models. It has achieved a 13.3% improvement over the Stereo Piecewise Rigid Scene Model [79] framework and around 6.5% better results than the Object Scene Flow for Autonomous Vehicles (OSF) [80] model when considering the special case of objects moving towards the ego vehicle. Practically, the vehicles moving away from the ego vehicle will never collide with it and estimating the time to collision for those vehicles is unnecessary. Thus, passing additional information from the previous frame would generate a better optical flow estimate and can predict the motion in depth information accurately. It is highly crucial in motion perception scenarios such as autonomous vehicles or robotics to estimate the scene by avoiding collisions.

8 Conclusion and Future Work

8.1 Conclusion

Current object-tracking frameworks [8, 12] are yet to fully adapt attention-based transformer architectures for feature extraction and object association. Those incorporating transformer architecture exhibit significant tracking latency, making them unsuitable for real-time deployment [82, 83]. There are several challenges in handling partially occluded objects or motion blur, where the object detection confidence score is significantly low, affecting the overall tracking accuracy. In the autonomous driving and robotics domain, the precise estimation of time to collision with nearby objects is crucial in designing the tracking algorithms. The majority of existing techniques depend on information from a single current frame that cannot accurately estimate the surrounding context and involves per-pixel evaluation that incurs a large inference time to estimate the motion in depth information.

This master's thesis presents several research hypotheses that aim to address these challenges to a certain extent, substantiating them through conducting experiments and discussing their outcomes and results. To increase tracking accuracy and efficiency, a strategic modification to tracker architectures involves the integration of attention blocks to capture long-distance dependencies among objects. One of the ways of addressing issues related to partially occluded objects having low confidence scores requires employing unmatched tracks in the current frame to recover these detections. Moreover, accurate time-to-collision estimation is achieved by employing an object-tracking framework that processes information from previous and current frames, offering more context to estimate the motion in depth.

This work uses the popular CenterTrack [9] framework to conduct a series of experiments to validate the above hypothesis and assumptions. CenterTrack [9] tracks objects based on their center points and uses a separate head network to estimate the position of the objects in the upcoming frames. In the initial experiment, the default feature extractor backbone of CenterTrack [9] is replaced with deeper and denser parallel connection networks, specifically ResNet [64] and HRNet [65] backbone networks, to examine the impact of a feature extractor on object identification and tracking. Integration of RepMixer blocks from the FastViT [76] framework into CenterTrack's head network introduces an attention [1] mechanism to capture long-distance object relationships, improving tracking efficiency and capabilities. Modification of CenterTrack's [9] tracker module associates low-confidence object detections with unmatched tracks. Finally, a novel motion in depth estimation network is implemented to estimate the collision time of autonomous vehicles for the CenterTrack framework as it takes features from the previous frame.

The experiment 4, replacing CenterTrack's [9] backbone feature extractor with ResNet [64] and HRNet [65] networks has proved highly inefficient, indicating the need for corresponding modifications in the neck network to align with the underlying backbone network. Integrating the RepMixer [76] block in the CenterTrack head network, which contains structural changes involving depthwise convolution and skip connections, results in reduced computational complexity, as measured in Floating Point Operations (FLOPS), Multiply-Accumulate Operations (MACS) and inference time as presented in 5. Handling the unmatched detections yields a slightly improved object tracing accuracy. The results are listed in the chapter 6. However, this technique has the potential to apply to various object trackers. CenterTrack's [9] novel motion in depth estimation network, as presented in the chapter 7, outperforms other frameworks in terms of inference time and accuracy, substantiating the hypothesis that augmenting the input network with additional previous frame embeddings results in more accurate estimations.

In summary, the research assumptions and corresponding experimental validations presented in this thesis reveal promising directions for researchers seeking to enhance the efficiency of deep learning-based object trackers.

8.2 Future Work

To further enhance the tracking accuracy, researchers can incorporate the Long Short-Term Memory (LSTM) networks to account for modelling temporal information to capture object movement from several frames before. The utilization of attention mechanisms, as presented by integrating RepMixer blocks from FastViT [76], offers opportunities for the researchers to explore the fine-tuning of attention-based methods within the different components of tracking framework, aiming for an optimal balance between computational complexity and tracking accuracy. Handling the challenges posed by low-confidence detections might recover the false negatives and improve overall tracking accuracy. This technique was introduced by ByteTrack [12] and has the potential to apply to different object trackers that rely on tracking-by-detection. The combination of these techniques has the potential to create a unified framework, advancing the efficiency of deep learning-based object trackers.

List of Figures

List of Tables

Bibliography

[1] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. "Attention is all you need". In: *Advances in neural information processing systems* 30 (2017).

[2] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo. "Swin transformer: Hierarchical vision transformer using shifted windows". In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2021, pp. 10012–10022.

[3] I. O. Tolstikhin, N. Houlsby, A. Kolesnikov, L. Beyer, X. Zhai, T. Unterthiner, J. Yung, A. Steiner, D. Keysers, J. Uszkoreit, et al. "Mlp-mixer: An all-mlp architecture for vision". In: *Advances in neural information processing systems* 34 (2021), pp. 24261–24272.

[4] S. Ren, K. He, R. Girshick, and J. Sun. "Faster r-cnn: Towards real-time object detection with region proposal networks". In: *Advances in neural information processing systems* 28 (2015).

[5] gaudenz-boesch. Accessed November 2, 2023. 2020. URL: https://viso.ai/applications/computer-vision-applications-in-surveillance-and-security/.

[6] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft. "Simple online and realtime tracking". In: *2016 IEEE international conference on image processing (ICIP)*. IEEE. 2016, pp. 3464–3468.

[7] N. Wojke, A. Bewley, and D. Paulus. "Simple online and realtime tracking with a deep association metric". In: *2017 IEEE international conference on image processing (ICIP)*. IEEE. 2017, pp. 3645–3649.

[8] P. Bergmann, T. Meinhardt, and L. Leal-Taixe. "Tracking without bells and whistles". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 941–951.

[9] X. Zhou, V. Koltun, and P. Krähenbühl. "Tracking objects as points". In: *European conference on computer vision*. Springer. 2020, pp. 474–490.

[10] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun. "The KITTI vision benchmark suite". In: *URL http://www. cvlibs. net/datasets/kitti* 2.5 (2015).

[11] D. Marr. *Vision: A computational investigation into the human representation and processing of visual information*. MIT press, 2010.

[12] Y. Zhang, P. Sun, Y. Jiang, D. Yu, F. Weng, Z. Yuan, P. Luo, W. Liu, and X. Wang. "Bytetrack: Multi-object tracking by associating every detection box". In: *European Conference on Computer Vision*. Springer. 2022, pp. 1–21.

[13] datahacker. Accessed November 2, 2023. 2019. URL: https://datahacker.rs/best-autonomous-driving-cars/.

[14] A. Badki, O. Gallo, J. Kautz, and P. Sen. "Binary ttc: A temporal geofence for autonomous navigation". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 12946–12955.

[15] K. G. Nalbant and Ş. UYANIK. "Computer vision in the metaverse". In: *Journal of Metaverse* 1.1 (2021), pp. 9–12.

[16] S. Reynolds. Accessed November 2, 2023. 2017. URL: https://therealizers.com/pokemon-go-is-changing-the-augmented-reality-game/.

[17] J. Gao, Y. Yang, P. Lin, D. S. Park, et al. *Computer vision in healthcare applications*. 2018.

[18] P. Khandelwal, A. Khandelwal, S. Agarwal, D. Thomas, N. Xavier, and A. Raghuraman. "Using computer vision to enhance safety of workforce in manufacturing in a post covid world". In: *arXiv preprint arXiv:2005.05287* (2020).

[19] clickworker. Accessed November 2, 2023. URL: https://www.clickworker.com/ai-glossary/computer-vision/.

[20] A. Rosenfeld. "Picture processing by computer". In: *ACM Computing Surveys (CSUR)* 1.3 (1969), pp. 147–176.

[21] M. Intelligence. "1986 Index IEEE Transactions on Pattern Analysis and Machine Intelligence Vol. PAMI-8". In: ().

[22] P. F. McLauchlan. "A batch/recursive algorithm for 3D scene reconstruction". In: *Proceedings IEEE Conference on Computer Vision and Pattern Recognition. CVPR 2000 (Cat. No. PR00662)*. Vol. 2. IEEE. 2000, pp. 738–743.

[23] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. "Imagenet: A large-scale hierarchical image database". In: *2009 IEEE conference on computer vision and pattern recognition*. Ieee. 2009, pp. 248–255.

[24] W. K. Fong, R. Mohan, J. V. Hurtado, L. Zhou, H. Caesar, O. Beijbom, and A. Valada. "Panoptic nuScenes: A Large-Scale Benchmark for LiDAR Panoptic Segmentation and Tracking". In: *arXiv preprint arXiv:2109.03805* (2021).

[25] K. O'Shea and R. Nash. "An introduction to convolutional neural networks". In: *arXiv preprint arXiv:1511.08458* (2015).

[26] L. R. Medsker and L. Jain. "Recurrent neural networks". In: *Design and Applications* 5.64-67 (2001), p. 2.

[27] W. Gao, X. Zhang, L. Yang, and H. Liu. "An improved Sobel edge detection". In: *2010 3rd International conference on computer science and information technology*. Vol. 5. IEEE. 2010, pp. 67–71.

[28] N. Dalal and B. Triggs. "Histograms of oriented gradients for human detection". In: *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*. Vol. 1. Ieee. 2005, pp. 886–893.

[29] C. Tomasi and T. Kanade. "Detection and tracking of point". In: *Int J Comput Vis* 9.137-154 (1991), p. 3.

[30] analyticsvidhya. Accessed November 2, 2023. 2023. URL: https://www.analyticsvidhya.com/blog/2019/09/feature-engineering-images-introduction-hog-feature-descriptor/.

[31] R. E. Kalman. "A new approach to linear filtering and prediction problems". In: (1960).

[32] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. "Ssd: Single shot multibox detector". In: *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14*. Springer. 2016, pp. 21–37.

[33] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. "You only look once: Unified, real-time object detection". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 779–788.

[34] X. Zhou, D. Wang, and P. Krähenbühl. "Objects as points". In: *arXiv preprint arXiv:1904.07850* (2019).

[35] mehul. Accessed November 2, 2023. 2020. URL: https://aidetic.in/blog/2020/10/05/object-tracking-in-videos-introduction-and-common-techniques/.

[36] E. Arkin, N. Yadikar, X. Xu, A. Aysa, and K. Ubul. "A survey: Object detection methods from CNN to transformer". In: *Multimedia Tools and Applications* 82.14 (2023), pp. 21353–21383.

[37] R. Girshick, J. Donahue, T. Darrell, and J. Malik. "Rich feature hierarchies for accurate object detection and semantic segmentation". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2014, pp. 580–587.

[38] J. R. Uijlings, K. E. Van De Sande, T. Gevers, and A. W. Smeulders. "Selective search for object recognition". In: *International journal of computer vision* 104 (2013), pp. 154–171.

[39] R. Girshick. "Fast r-cnn". In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 1440–1448.

[40] K. He, G. Gkioxari, P. Dollár, and R. Girshick. "Mask r-cnn". In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 2961–2969.

[41] K. Simonyan and A. Zisserman. "Very deep convolutional networks for large-scale image recognition". In: *arXiv preprint arXiv:1409.1556* (2014).

[42] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. "Focal loss for dense object detection". In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 2980–2988.

[43] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie. "Feature pyramid networks for object detection". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 2117–2125.

[44] L. Weng. "Object Detection Part 4: Fast Detection Models". In: *lilianweng.github.io* (2018). URL: https://lilianweng.github.io/posts/2018-12-27-object-recognition-part-4/.

[45] H. Law and J. Deng. "Cornernet: Detecting objects as paired keypoints". In: *Proceedings of the European conference on computer vision (ECCV)*. 2018, pp. 734–750.

[46] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, and T. Unterthiner. "Transformers for image recognition at scale". In: *arXiv preprint arXiv:2010.11929* (2020).

[47] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko. "End-to-end object detection with transformers". In: *European conference on computer vision*. Springer. 2020, pp. 213–229.

[48] W. Yu, M. Luo, P. Zhou, C. Si, Y. Zhou, X. Wang, J. Feng, and S. Yan. "Metaformer is actually what you need for vision". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2022, pp. 10819–10829.

[49] M. Isard and A. Blake. "CONDENSATION–conditional density propagation for visual tracking". In: *International journal of computer vision* 29.1 (1998), p. 5.

[50] X. Zhang, W. Hu, S. Maybank, X. Li, and M. Zhu. "Sequential particle swarm optimization for visual tracking". In: *2008 IEEE Conference on Computer Vision and Pattern Recognition*. 2008, pp. 1–8. DOI: 10.1109/CVPR.2008.4587512.

[51] I. Leichter. "Mean shift trackers with cross-bin metrics". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 34.4 (2011), pp. 695–706.

[52] wikipedia. Accessed November 2, 2023. 2016. URL: https://en.wikipedia.org/wiki/Optical%20flow.

[53] S. Hochreiter and J. Schmidhuber. "Long short-term memory". In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[54] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio. "Empirical evaluation of gated recurrent neural networks on sequence modeling". In: *arXiv preprint arXiv:1412.3555* (2014).

[55] X. Ma, P. Karkus, D. Hsu, and W. S. Lee. "Particle filter recurrent neural networks". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34. 04. 2020, pp. 5101–5108.

[56] "character-level-deep-language-model-with-gru-lstm-units-using-tensorflow". In: (). URL: https://www.nablasquared.com/character-level-deep-language-model-with-gru-lstm-units-using-tensorflow/.

[57] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista. "High-speed tracking with kernelized correlation filters". In: *IEEE transactions on pattern analysis and machine intelligence* 37.3 (2014), pp. 583–596.

[58] D. Held, S. Thrun, and S. Savarese. "Learning to track at 100 fps with deep regression networks". In: *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14*. Springer. 2016, pp. 749–765.

[59] X. Wang, A. Jabri, and A. A. Efros. "Learning correspondence from the cycle-consistency of time". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 2566–2576.

[60] G. Ning, Z. Zhang, C. Huang, X. Ren, H. Wang, C. Cai, and Z. He. "Spatially supervised recurrent convolutional neural networks for visual object tracking". In: *2017 IEEE international symposium on circuits and systems (ISCAS)*. IEEE. 2017, pp. 1–4.

[61] H. Zhou, Y. Yuan, and C. Shi. "Object tracking using SIFT features and mean shift". In: *Computer vision and image understanding* 113.3 (2009), pp. 345–352.

[62] H. Kuhn. "The Hungarian Method for the assignment problem. Naval Research Logistics Quarterly". In: *http://dx.doi.org/10.1002/nav.3800020109* 2 (1955), pp. 83–97.

[63] F. Yu, D. Wang, E. Shelhamer, and T. Darrell. "Deep layer aggregation". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 2403–2412.

[64] K. He, X. Zhang, S. Ren, and J. Sun. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[65] J. Wang, K. Sun, T. Cheng, B. Jiang, C. Deng, Y. Zhao, D. Liu, Y. Mu, M. Tan, X. Wang, et al. "Deep high-resolution representation learning for visual recognition". In: *IEEE transactions on pattern analysis and machine intelligence* 43.10 (2020), pp. 3349–3364.

[66] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. "Automatic differentiation in PyTorch". In: *NIPS-W*. 2017.

[67] J. Chen, M. Zhou, D. Zhang, H. Huang, and F. Zhang. "Quantification of water inflow in rock tunnel faces via convolutional neural network approach". In: *Automation in Construction* 123 (2021), p. 103526.

[68] J. Huang, Z. Zhu, and G. Huang. "Multi-stage HRNet: Multiple stage high-resolution network for human pose estimation". In: *arXiv preprint arXiv:1910.05901* (2019).

[69] H. Wu, C. Liang, M. Liu, and Z. Wen. "Optimized HRNet for image semantic segmentation". In: *Expert Systems with Applications* 174 (2021), p. 114532.

[70] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun. "Vision meets robotics: The kitti dataset". In: *The International Journal of Robotics Research* 32.11 (2013), pp. 1231–1237.

[71] D. Göhring, M. Wang, M. Schnürmacher, and T. Ganjineh. "Radar/lidar sensor fusion for car-following on highways". In: *The 5th International Conference on Automation, Robotics and Applications*. IEEE. 2011, pp. 407–412.

[72] V. Lidar. "HDL-64E". In: *online],[Retrieved Jul. 15, 2020]. Retrieved from Internet (10pgs)* (2016).

[73] K. Bernardin and R. Stiefelhagen. "Evaluating multiple object tracking performance: the clear mot metrics". In: *EURASIP Journal on Image and Video Processing* 2008 (2008), pp. 1–10.

[74] D. P. Kingma and J. Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[75] J. Choquette, W. Gandhi, O. Giroux, N. Stam, and R. Krashinsky. "NVIDIA A100 tensor core GPU: Performance and innovation". In: *IEEE Micro* 41.2 (2021), pp. 29–35.

[76] P. K. A. Vasu, J. Gabriel, J. Zhu, O. Tuzel, and A. Ranjan. "FastViT: A Fast Hybrid Vision Transformer using Structural Reparameterization". In: *arXiv preprint arXiv:2303.14189* (2023).

[77] F. Chollet. "Xception: Deep learning with depthwise separable convolutions". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 1251–1258.

[78] H. Zhang, W. Hu, and X. Wang. "Fcaformer: Forward Cross Attention in Hybrid Vision Transformer". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2023, pp. 6060–6069.

[79] C. Vogel, K. Schindler, and S. Roth. "3d scene flow estimation with a piecewise rigid scene model". In: *International Journal of Computer Vision* 115 (2015), pp. 1–28.

[80] M. Menze and A. Geiger. "Object scene flow for autonomous vehicles". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 3061–3070.

[81] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. "Microsoft coco: Common objects in context". In: *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*. Springer. 2014, pp. 740–755.

[82] X. Zhou, T. Yin, V. Koltun, and P. Krähenbühl. "Global tracking transformers". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 8771–8780.

[83] T. Meinhardt, A. Kirillov, L. Leal-Taixe, and C. Feichtenhofer. "Trackformer: Multi-object tracking with transformers". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2022, pp. 8844–8854.