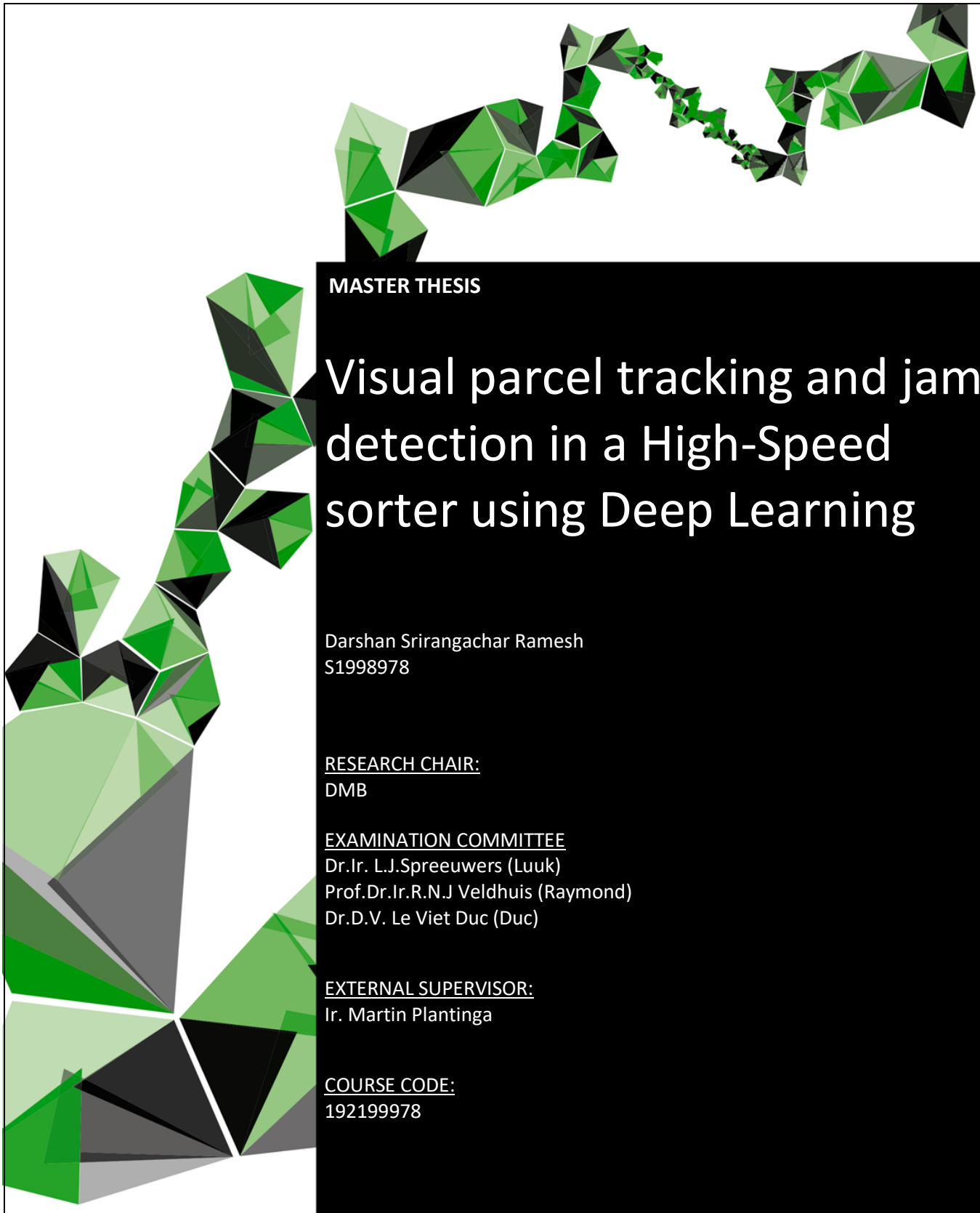


MASTER THESIS

Visual parcel tracking and jam detection in a High-Speed sorter using Deep Learning

Darshan Srirangachar Ramesh
S1998978

RESEARCH CHAIR:
DMB

EXAMINATION COMMITTEE
Dr.Ir. L.J.Spreeuwers (Luuk)
Prof.Dr.Ir.R.N.J Veldhuis (Raymond)
Dr.D.V. Le Viet Duc (Duc)

EXTERNAL SUPERVISOR:
Ir. Martin Plantinga

COURSE CODE:
192199978

UNIVERSITY OF TWENTE.

Acknowledgment

First of all, I would like to thank my daily supervisor Dr.Luuk Spreeuwers for his immense support and guidance throughout the project. I cherish the discussions we have had which always directed me to take the right steps in this project. It was a grateful experience to have worked under him. I am very much humbled by his simple nature, thirst for knowledge and most importantly the interaction style. I really hope he gets the best teacher award every year. I would also like to thank Dr. Duc Le for his detailed review of this research and also for the general discussions on my report, which would be very useful for my future. It was a wonderful experience and given another opportunity, I would also like to work under him. Finally, I would like to thank Dr. Raymond for his patience and encouragement by being a part of the graduation committee.

From the company, I would like to thank my Manager, Erik Van Dartel for providing an opportunity for pursuing this project and also for his immense support throughout. I would like to thank my mentor Martin Plantinga for providing me constant feedback in the informative meetings. Working under him has equipped me with the right skills in AI technology and its application. I am very much indebted to him for the amount of time he has invested in me, which has made me grow as a good researcher and as a young professional. I also extend my thanks to the New Technology team for hearing my presentations and offering insights during the course of this thesis.

I must express my gratitude to my brother Karthick, for his constant support and motivation throughout the thesis period and I am eternally grateful to him for providing a roof In Eindhoven. Without him, It would have been a tough and very expensive stay in Eindhoven.

I would like to thank Asif, Navin for their support and friendship. I am extremely happy to have found you people as my friends at the university and I always cherish the little moments that we have had during these 2 years. My sincere gratitude goes to a very close friend Meghashree, for being there during my frustrations and pushing me to achieve more.

Finally, thanks to my Father and Mother for believing In me and extending the support In every possible manner while studying in the Netherlands.

Contents

1. INTRODUCTION:

Vanderlande's conveyor belt-based sorting systems for the warehousing solutions are high-speed sorters, typically running at an adjustable speed range of 2-3m/s with a sorting capacity of 15,000-24,000 parcels per hour. These systems are complex and rely on a number of sensors to detect, track and direct the parcels to its desired destination. These warehouses are usually highly heterogeneous in nature: different conveyor belt types, colors, varying backgrounds, and varying illumination. The parcels can also be very diverse and can occur in all materials, colors, and dimensions [1]. It is possible that such parcels can be located very close to each other(0-50mm) to attain high sorting capacity. However, there can also be a large number of parcels that are of the same kind.

In Vanderlande's sorting systems, parcels are sorted, by placing them inside a squared shaped material called as Transport and Storage Unit (TSU). All TSU's share the same characteristics as the color and dimension as seen from Figure 2b. During sortation, it was observed that these TSUs get stuck in the system leading to blockages or jams in the system. Unable to detect such an anomaly leads to undesired effects like flying TSUs as seen from Figure 2a. As a result, the sorting capacity decreases due to often system stoppage or halts.

Solutions like using dedicated traditional photoelectric cells (PEC) and proximity sensors to detect and count the number of parcels in the area where jams are more observed are usually employed. Drawbacks such as missing the TSU counts and communication overhead in such complex settings exist. Also, they are costly if they are used just to detect jams in the system during high-speed sorting. Moreover, another important aspect to consider is the positions where these sensors would be placed in the search area. Arriving at an optimal location requires thorough testing leading to an increase in the test time. Also, as the number of search areas increases, so does the need for dedicated PECs or proximity sensors.

The solution, we propose is the reusability of the IP surveillance cameras in the warehouses which are generally RGB with at least 15FPS. With such RGB cameras, traditional computer vision techniques or deep learning techniques could be utilized to detect and track the TSUs in the scene. Additionally, since we reuse the existing cameras, the only additional cost would be on the required computing power. Also, testing time would be comparatively reduced because of the wide potential jam areas a camera could cover.

1.1. Overview of the sorter:

The entire sorting system is made up of a number of components as from Figure 1 below. The component level explanation is out of the scope and hence only the required details are described in Table 1

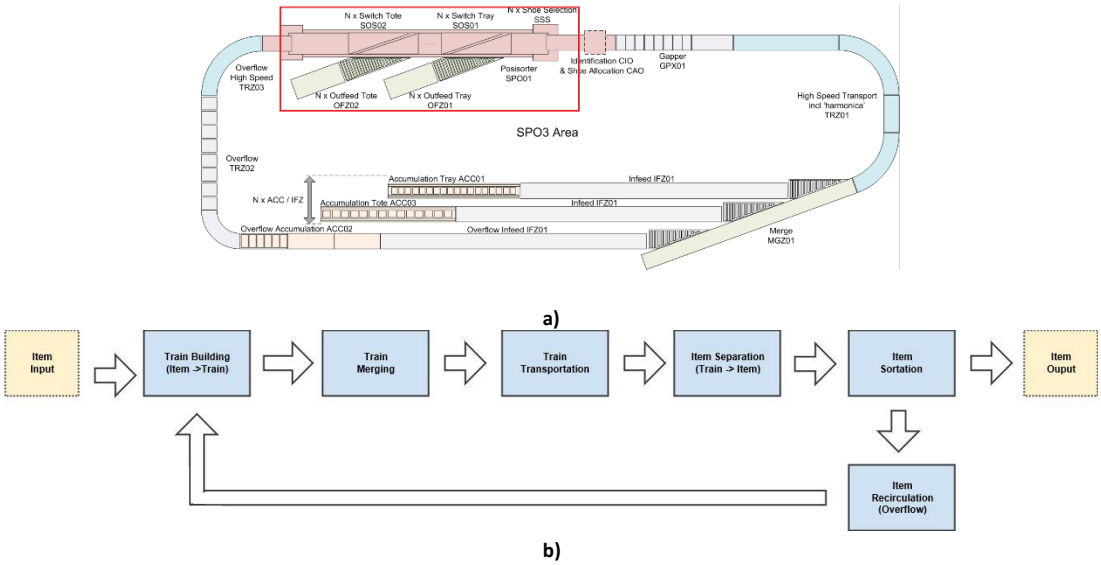


Figure 1- a) System Overview b) Process flow of the system.

Component	Abbreviation	Responsibility
Accumulation Tray	ACC	The area where the parcels are fed into the system
Infeed Zone	IFZ	Parcels are collected from the accumulation tray and is transported to the next section.
Merge zone	MGZ	All parcels from the IFZ are merged into a single train of parcels.
Grapper	GPX	Consists of a number of belts that are of the same length used to create the required gap between the parcels.
Identification & Shoe Allocation unit	-	A barcode scanner identifies the parcels and allocates the shoe accordingly.

Posisorter	SPO01	An area where the incoming parcels are sorter using shoe attached to the belts.
Shoe	-	These are installed on the running sorter and is responsible for pushing the parcels into the Outfeed Zones (OFZ)
Outfeed Zones	OFZ	Consists of high-speed rollers and belts which collects the parcels from the sorter
Live Rollers	LR	Rollers rotating at a certain speed in the OFZ.
Overflow Zone	-	Unsorted parcels are sent to this area which again is fed back via IFZ.

Table 1- Component level explanation

Process flow:

Trains (collection of TSUs) are built in the accumulation zone (ACC) and transported to the infeed zone (IFZ). The gap between TSUs is made as small as possible on the IFZ. Trains are then released onto the collector belt in the merge zone (MGZ). The gap between trains is minimized to reach capacity while preventing trains to crash into each other. Physically alignment of items is needed in the transport zone (TRZ) to correct the orientation of trays after the merge. The gap control zone (GPX) is responsible for creating defined gaps between trays. Required gaps are determined by the quality of the downstream sorting process. Trays are then separated and delivered to the Posisorter zone (SPO01). Before entering the SPO01, on the charged belt the trays are identified (barcode, dimensions, exceptions) and synced with the shoes on the sorter. The speed of the charge belt is as same as the sorter. Now, the TSU destination is looked up and shoes are allocated for the sortation action. The allocated shoes are switched to move it to the outfeed (OFZ). When the sortation is not possible trays will go the overflow and fed again into the system via most upstream ACC/IFZ. This process flow can be seen in Figure 1(b).

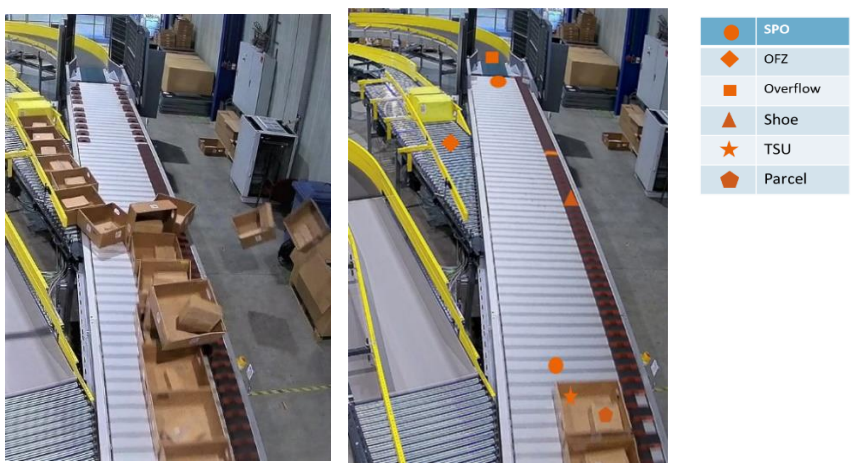
As mentioned before, the whole system operates on variable speeds. As the incoming parcels increases in the ACC, the SPO01 can automatically increase its speed to match the required sorting capacity resulting in the decrease of distance between parcels. Here, SPO3 area consists of the SPO01 and OFZ, which is our area of interest, where the blockages or jams are prominent. In the entire paper, jams and blockages are used interchangeably. Also, to avoid any confusion SPO01 is always referred to as sorter.

Problem:

It is observed that during a)during the handover of TSUs from the sorter to OFZ b)during the alignment on the live rollers in the OFZ., the parcels get stuck (no motion of parcels relative to the moving belts/rollers) in the system creating a blockage due to a few known reasons. If there is no human influence on the system in the creation of such blockages, the reasons can be broadly classified into two categories, though not limited to, is shown in Table 2. Failing to identify such an anomaly, could result in a catastrophic phenomenon shown in Figure 2a. Figure 2b gives information about the components in the system.

Failure categories	Reasons
Mechanical	Wear and tear, belt slippages, sensor failures.
Software	Bugs, network issues.

Table 2- Failure categories and reasons



a) **b)**

Figure 2- a) Jam in the system created manually b) Individual component in the region of interest

1.2. Current Solution:

To mitigate this problem, several experiments were carried out within Vanderlande to reuse some of the existing PECs like the Chute Full PEC and the Divert Verify PEC seen from Figure 3(b). Chute Full PEC is used to check if the OFZ has the maximum number of TSUs and no sortation is carried out until the OFZ is empty. Divert verify PEC is used to check if the TSUs are sorted correctly. If Chute Full PEC is triggered no sortation is carried out and the divert verify PEC must ideally see the unsorted TSUs on the sorter moving to the overflow zone.

Using a fixed travel time for the TSUs between the sorter and chute full PEC as a base to detect jams failed due to the constraints related to extra communication between the controllers just for jam detection. Also, it was observed that the fixed travel time varied as the speed of the live rollers in the OFZ varied and also due to the varied timings between load and no-load TSUs. Furthermore, using the divert verify PEC alone provided unreliable results.

Hence dedicated Photo Electric Cells (PEC) were placed in the OFZ, strategically as shown in Figure 3 (marked in two red circles). These positions were arrived by experimenting by placing the PECs at different positions with a varying speed of the sorter and the rollers in the OFZ. Yet, the experiments were inconclusive if the PECs could count the correct number of passing TSUs in the OFZs.

PECs detect and counts the number of parcels using an infrared transmitter and a photoelectric receiver. Due to very less distance or gap (order of few mm) between the parcels, it was observed that the PECs failed to count the parcels when they were close to each other during high-speed sortation. The effect of such false values leads to delay in the detection of blockages in the system which brings down the sorting capacity with frequent blockages. Also, as the number of OFZ's increases so does the number of PECs. A risk vs cost analysis was already analyzed by the company between the standard PECs vs proximity sensors (RTQ-P4221) and the former was chosen due to the latter being expensive per OFZ.

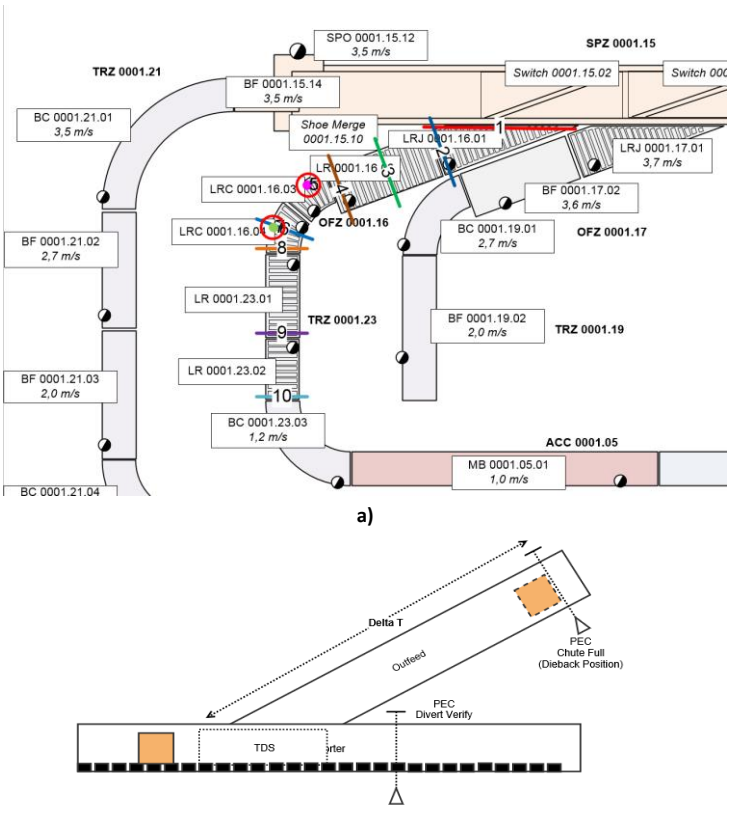


Figure 3- a) PEC placement b)Other PECs on the Sorter(Divert Verify PEC) and in OFZ (PEC EOS Dieback)

1.3. Proposed solution:

The proposed solution is to reuse an inexpensive RGB camera that would be available in the warehouses, placed in such a way that it's viewing angle covers the SPO3 area as seen in Figure 2. Using one of the cameras, we explore classical computer vision and the recent deep learning techniques to detect and track the parcels. With tracking, count of TSUs entering and exiting the OFZ would be known and also could detect jams based on the location of the parcels. During a jam, a parcel would have almost the same location in certain consecutive frames,

which directly depends on the quality of detection. With such techniques, behavior of the TSUs during jams could be analyzed, such as, obtaining the angle of rotation of the TSUs (with or without load) when being handed over to the OFZ by the shoes, checking for the alignment of TSUs on the sorter with respect to some reference point in the image plane.

With this, we believe that having the cameras in the sorting facility we could replace the PECs solely dedicated to detecting jams in the OFZs. One such camera with a good viewing angle could cover at least 2-3 OFZs reducing the need for PEC pair per OFZ. This also would bring down the testing time dedicated to finding a *sweet spot* for the PECs in the OFZ to be able to detect all the TSUs.

Upcoming sections explore the classical vision techniques and the discussion leans towards deep learning-based techniques while mentioning the advantages and disadvantages of techniques experimented. Section 2 has information on classical vision object detection and tracking techniques. While Section 3 gives an overview of the recent State of the Art (SOTA) deep learning object detection techniques which are broadly classified as single-stage and two-stage detectors. Section 4 deals with related work and section 5 provide information about the camera placements in the facility. The discussion about the experiments performed and the corresponding observations made both on detection and tracking are mentioned in section 6.

2. COMPUTER VISION TECHNIQUES

2.1.1. Object Tracking

The aim of an object tracker is to generate the trajectory of an object or objects over time by locating its position in every frame of the video. It is a trajectory estimation problem. This has been an important computer vision problem due to its application in both academia and industry with commercial potential. Though tracking is a well-studied problem, it remains a challenge in many aspects [5]. Over the years, research in the field of trackers applied to specific classes like pedestrians or faces has made great progress. But, trackers for generic objects continues to be a difficult area of research since these objects may change their appearance over a period. This is due to changes in the light, noise in the image, low visibility, changing motion patterns, occlusions are few challenges.

2.1.2. Classification of trackers

Tracking can be broadly classified as Single Object Tracking (SOT) and Multiple Object Tracking(MOT). SOT is used to track to single objects In the scene though it contains multiple objects and does not account for tracking new objects entering the scene. Meanwhile, MOT is largely partitioned to locating multiple objects, maintaining their identity, and yielding their individual trajectories given an input video. Both MOT and SOT focuses on designing sophisticated appearances models and motion models to deal with factors such as scale changes, out-of-plane rotations, and illumination variations. Meanwhile, MOT has to deal with two extra tasks: determining the number of objects in the scene over time and maintaining their identity[4]. As mentioned earlier, since SOT tracks a single object, modeling the tracker to handle occlusions is simpler than that is for MOT due to additional challenges. They are frequent occlusions, similar appearance, initialization and termination of tracks and interactions among multiple objects [4]. Also, SOT can be employed to track multiple objects by instantiating multi-single object trackers. By doing so, different trackers may lock onto the same object which has a similar appearance.

MOT trackers solve such problems by modeling the interaction between objects by maintaining a joint likelihood or posterior for all the objects. Such a joint space is usually computation heavy and hence the MOT problem is transformed into a one-to-one mapping constraint [6]. Now, the objective of MOT trackers is to find an optimal sequential state of all the objects, which can be modeled by performing maximal a posterior estimation (MAP) from the conditional distribution of the sequential states given all the observation [4]. Generally, finding optimal sequential states or estimating MAP falls under two categories. One is by designing a probabilistic model in a two-step iterative process, predicting, and updating. Latter is designed by building a Dynamic model and the former by building an observation model. Second is by designing a deterministic optimization model which is to directly maximize the likelihood function over a set of available observations or minimize an energy function.

2.1.3. Problems in object detection and tracking

While designing a robust tracker, either SOT and MOT, common challenges to be solved are as follows

- Object modeling: One of the major issues in finding a suitable visual description that makes the object distinguishable from other objects and the background.
- Changes in appearances and shape: This is majorly dependent on the camera viewing angle. Also, objects tend to change their shapes during tracking, specifically in pedestrian tracking, where humans are considered as deformable objects. There is also the perspective effect, wherein objects closer to the camera appear to be big and appear to be small as they move farther away from the camera.
- Illumination changes: This is one of the challenging issues in object tracking in vision wherein an object can have a different appearance in different artificial lighting conditions. Lighting conditions also change depending on the time of the day, weather condition and seasons making it more challenging in designing a powerful tracker.
- Shadows and reflections: This is a classical vision problem where the shadows and reflections would be classified as an object if these are not considered while modeling the background and foreground features. Reflections can cause problems when objects are moving on smooth surfaces.
- Occlusion: Again, as mentioned earlier Section 1, occlusion is strongly dependent on the viewing angle. It can be categorized as self-occlusion and inter-object occlusion. Self-occlusion occurs when one part of the object is occluded by another part, while inter-object occlusion occurs when two objects being tracked are occluded by each other. The trackers must be capable enough to handle such occlusions by maintaining the individuality of objects under occlusion.

- Information loss: Caused by projecting the 3D world on a 2D image plane.
- Drifting problem: Degradation of the appearance model (features of the object to be tracked, color, points, etc) caused by the inaccuracy in the estimation of foreground and background leads to the poor partitioning of foreground and background. This is mostly seen during object -to-scene occlusion and the trackers tend to lock onto the background. This is termed as a Drifting problem[7].
- Low resolution: As the area of a ground truth bounding box reduces, detection becomes hard.

2.1.4. **Criteria**

Though there are no universal criteria to differentiate between the methods employed by researchers to create a robust MOT tacker, we can still group them under three criteria which are as follows [4]

1. Initialization methods
2. Processing mode
3. Learning or training mode
4. Type of output

2.1.4.1. **Initialization method**

The task of detecting the objects and establishing the correspondence between the object instances across frames are either performed separately or jointly [8].

Detection-Based Tracking (DBT)

Figure 4a represents the DBT, wherein the objects are detected prior to the linking of trajectories in consecutive frames. This is commonly termed as "tracking-by-detection". Most of the MOT trackers are based on this framework and is being used very often as a commercial solution. In this strategy, object detection is first applied, which is object-specific like pedestrians, vehicles, etc in each frame to obtain the object hypothesis. This is usually generated using appearance modeling or motion modeling which will be discussed later. Post the detection, the trajectories are generated using the MOT trackers, which links the detection hypothesis to trajectories[4]. It must be noted that the performance of such trackers highly depends on the quality of the detectors employed. Such object detector models are commonly pre-trained on huge datasets.

Detection-Free Tracking (DFT)

Figure 4b represents the DFT, where the objects to be tracked are initialized manually in the first frame. The object region and correspondence are jointly estimated by iteratively updating object location and region information obtained from the previous frame. Localization is done in the subsequent frames [4]. One of the advantages is that the strategy is free of pre-trained models. Table 3 briefs about the difference between these strategies.

Item	DBT	DFT
Initialization	Automatic	manual
No. of objects	Varying	fixed
Applications	A specific type of objects	Any type of objects
Advantages	Handles newly added and disappearing objects	Free of object detectors
Drawbacks	Performance depends on object detectors	Manual initialization

Table 3 – Difference between DBT & DFT

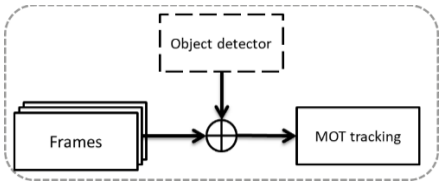


Figure 4a- DBT

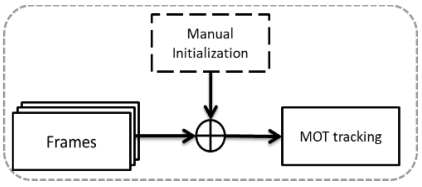


Figure 4b-DFT

2.1.4.2. **Processing mode:**

Based on the way the observations from the frames (past or future) are processed to track the object in the current frame, two strategies are employed.

Online

Tracking of objects is done by processing the observation information obtained from the previous frame to handle the tracking in the current frame. The frames or sequences are handled in a step-wise manner thus online tracking is also named as sequential tracking. Based on the up-to-time observations, trajectories are produced on the fly [4]. They will not use future frames to improve the results. These are referred to as recursive methods.

Offline

While online tracking trackers use the previous frame's observation information to update the current frame's observation information, offline tracking-based trackers make use of past and also the future observation to predict the trajectory which improvises the tracking prediction. These are used in tracking objects in a recorded stream. Recorded videos of a game where the analysis of opponent team players is necessary for the team's strategy, is one such example. Here the not only frames from the past can be used but also from the future can be used to make more accurate tracking predictions. Since they require observation from the future frames, memory and computations are high for such trackers. This can be solved by splitting the frames into smaller chunks and inferring the results from each chunk. These are sometimes referred to as non-recursive methods. The results of such a tracking system are the object's location and their IDs.

2.1.4.3. **Learning or training mode:**

Online learning trackers

Online learning trackers are usually manually initialized in the required frames. These are general trackers since they learn about the object to be tracked online or during the run time just by utilizing a bounding box around the object. These trackers consider the object as a positive example that is inside the bounding box and rest as a negative example.

Offline learning trackers

The trackers need the training to learn about the objects In an offline manner and do not learn anything during run time. These trackers are more object-specific like trackers for pedestrians or cars etc.

2.1.4.4. **Type of Output/ Inference**

Based on the output of a tracker, the MOT trackers can be classified into deterministic models and probabilistic models

Deterministic

These are the trackers in which no randomness is involved in the development of the future states of the system. Thus, it produces the same output from a given starting condition or initial state. The object movements are assumed to follow some trajectory prototypes [9]. Such prototypes could be learned either in an online or offline fashion. But such an approach is more suitable for the task of offline tracking [4] because it requires at least a time window of observations for the tracker to associate observation belonging to an identical object into a trajectory. Such approaches define the correspondence cost as a combinatorial optimization problem where one solution is to use greedy search methods to obtain one-to-one correspondence [5]. The idea is to find the object's associations in different frames optimally. Bipartite graph matching, dynamic programming, etc. are few frameworks to mention.

Probabilistic

Trackers employing this approach rely on the probability of object movements. The object to be tracked is represented as either one or many points [9]. The general idea is that such trackers calculate or predict future states based on the variety of probabilistic reasoning based on the existing or current observation. Kalman filter and its variants (extended Kalman filter) are one such example used in modeling linear and non-linear motion of objects, respectively.

2.1.5. **Object tracking taxonomy**

Two major issues to consider while developing a tracker are

1. Effectively measuring the similarity between the objects in frames &
2. Based on the above measurement, recovering the identity information.

The first one is addressed in modeling the object's appearance, motion, interaction, exclusion, and occlusion. The second one is considered as an inference problem as discussed in section 2.1.4 (4). Mathematically the similarity between two observation *i* and *j* can be written as

$$S_{ij} = G\ (o_i,\ oj),$$

where o_i and o_j are the visual representation of different observation and function G (.,.) measures the similarity between the two of them.

The object tracking framework falls under two categories: deterministic methods and stochastic methods. Deterministic models try to reduce an optimization problem by solving the similarity cost function in an iterative way. Functions like Sum of Squared Differences (SSD) and kernel-based cost functions.

- SSD – Based cost function is defined as the summation of squared differences between the current image patch and the template
- A kernel-based cost function is defined as the distance between two kernel densities.

Deterministic methods tend to be more computationally efficient by has a drawback of solutions getting trapped in local minima. While stochastic methods view the object tracking problem as Bayesian inference problem and model the underlying dynamics of tracking as state spaces. Such Bayesian inference models need to generate multiple hypotheses to estimate and propagate the posterior distribution of the state. But such methods, due to the generation of multiple hypotheses, requires more computation and is subjected to the curse of dimensionality in a high-dimensional state space [22].

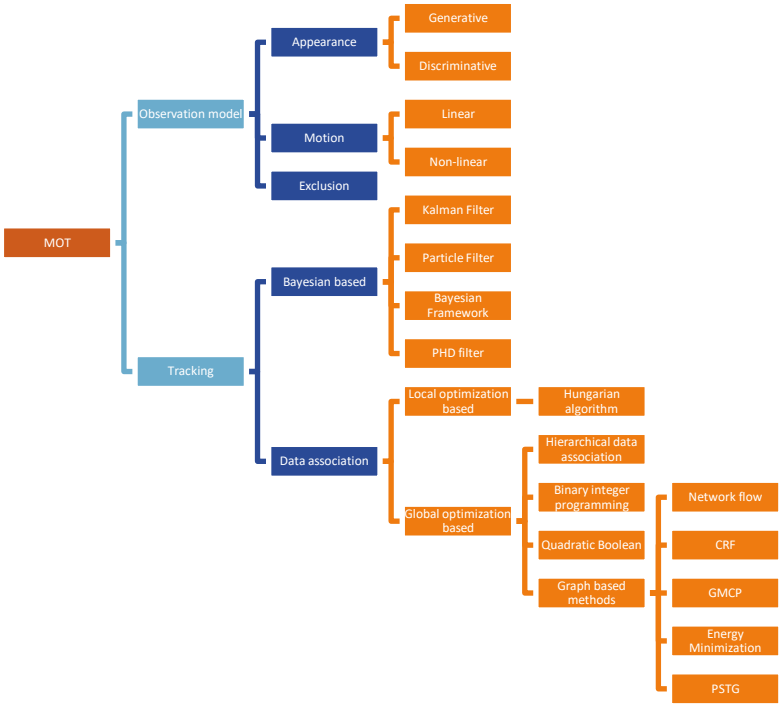


Figure 5- Object tracking taxonomy [10]

In general, MOT consists of an observation model and a tracking process. Observation models try to describe the unique features or descriptors (like HOG, SIFT, SURF) of objects and the tracking process tries to link these observations to the object's trajectory also known as tracklets. Such tracklets can be considered as a recursive problem where their current state is estimated only using information from the previous states and as a non-recursive problem where the current state is estimated by considering information from both previous frames and future frames.

Based on such a tracking process, the MOT problem can be easily classified as Bayesian-based tracking methods and data association-based tracking methods. Upcoming sections briefly will introduce the concepts and related work briefly.

2.1.5.1. **Observation models**

Observation models represent the distinct feature or characteristics of the objects. The characteristics may be, the object's appearance, velocity, location, etc. Features such as color, shape, gradient, motion, texture are conventional features used to describe the objects. Based on the tracking needs, either single appearance models are employed, or multiple cues are fused together to make a robust tracking system. Using such different

features and representations different observation models can be created. Such features distinguish the objects from the background. They are classified as appearance models, motion models, exclusion model and integrated models as seen from Figure 5.

Appearance model

These models are built either discriminatively or generatively. The generative models represent moving objects such that the tracking problem is restricted to searching an optimal state that results in an object appearance which is most like the appearance model. While determinative models consider the tracking problem as a binary classification problem based on an optimal decision boundary that distinguishes the object from the background [10].

Generative models

These models are created using features like color histogram, Histogram of Gradients (HOG), Gaussian Mixture Models (GMM), Hidden Markov random filed models (HMM). In general, the most desirable property of a visual feature is its uniqueness so that the objects can be easily distinguished in the feature space [7]. These models are used to learn the appearance of objects. Such models are updated online to adapt to its appearance changes. Features could be grouped as either local or region based [4]. [7] classifies features as gradient-based, texture-based, color-based, spatio-temporal based and mixed multiple fusion-based features. We shall focus more on [4]'s classification principles since it gives an easy to understand guide of core concepts involved in tracking.
Local features such as Kanade-Lucas-Tomashi (KLT) is considered as one such good local feature to search and track objects. Motion clustering & estimation of camera motion can be employed to generate short trajectories. Solutions from [11], [12] utilize optical flow model to link detection into short tracklets prior to the data association [4]. Solutions in [13], [14] encodes the motion information for building similarity scores for data association of the same objects having the temporal information. These features are efficient but are sensitive to occlusions and out-of-plane rotation.

Region-based features are wider in range compared to the local features. Information from the bounding box, segmentation information, etc. are examples. Color histogram, raw pixel template, HOG, region covariance matrix, Probabilistic Occupancy Map (POM), gait features fall under this category. Gradient-based features like HOG can describe the shape of an object which is robust to illumination changes but sensitive to occlusion and deformation. Also, the region-based features are more robust to certain transformations as they carry more information but at a cost of more memory and computation. Choosing features to track is the utmost importance since the performance of the trackers also depends on it.

Discriminative models

Based on the above models these models are built using classifiers in the feature space to distinguish between the object's feature and background features are known as discriminative models. Such models can be either fed with a single cue or multiple cues. Using a single cue such as in [15] based on color histogram, Bhattacharya distance is employed to develop a similarity score or fit a Gaussian distribution between two color histograms Also, fusing multiple cues is a non-trivial task and is active research field. By using multiple cues, one can develop a robust appearance model. Boosting, concatenation, summation, cascading, etc. are few methods being used to fuse information from various features [4]. General methods are using Support Vector Machines (SVM), random forest, boosting, etc.

Motion models

Also known as dynamic models, used to describe the movement of objects. This information is of importance since it can be used to predict the potential position of objects in the future frames by reducing the search space. Such motion modeling is built on an assumption, that is, the objects move smoothly in the real world and hence it is the same in the image space. Abrupt motions are not considered while modeling the motion. Based on this, the motion models are divided into linear and non-linear motion models.

Linear motion models

This motion model is based on the constant velocity assumption as in [16]. The velocity of the tracklets is estimated using first-order linear regression assuming that the motion is linear over a short time. Based on assumptions like velocity smoothness, position smoothness, and acceleration smoothness, these models are built. The velocity smoothness model is constructed by constraining the velocity of objects in successive frames to change smoothly. [17] implements this as a cost term which is the summation over N frames consisting of M objects. Positional smoothness is modeled by knowing the discrepancy between the observed position and the estimated position. Here, the smoothness is modeled by fitting the Gaussian distribution of the estimated position

with the observed position. Acceleration smoothness is modeled using both velocity and positional smoothness models by also considering the acceleration of objects.

Non-linear models

Real-world objects move around mostly in a non-linear fashion and hence most trackers are modeled using non-linear models. Though the linear models establish the dynamics of the object it cannot deal in some cases. A more accurate motion affinity between tracklets could be produced using these models. According to [4],[18] given two tracklets (t_1 and t_2) of the same object linear motion models produces low probability to link them while non-linear models can introduce another tracklet (t_0) which can explain the gap between t_1 and t_2 when the object moves in a non-linear fashion. According to [10], these models can describe in an online approach to learn non-linear motion patterns and create robust appearance models for multi-target tracking in a tracklet association framework. Also, a Relative Motion Network is constructed by considering the motion context (relative movements) between multiple objects.

Interaction models

These models are designed to capture the influence of an object on the other object. One such scenario can be observed in a crowd of people walking. A person's velocity, destination, and trajectory may change based on the other person. Such information is crucial in fields like crowd motion pattern analysis which could improve the performance of a tracker. These models are further divided as Individual force and group force which is modeled as social force models. These individual forces have their own attributes and assumptions which are considered while modeling complex trajectories.

Exclusion models

These models are designed based on the fact that no two distinct objects occupy the same physical space in the real world. This is classified further based on the generated hypothesis either by the detection or multiple trajectory generation as detection level exclusion and trajectory level exclusion, respectively. Former one hypothesizes that two different detection responses in the same frame cannot be assigned to the same target while the latter hypothesizes that two trajectories cannot be infinitely close to each other. Such models penalize the cost term in case of a violation of the hypothesis. It's graph-based modeling where exclusion models are constructed as nodes representing the detection responses and each node is connected only to other detection nodes that exist at the same time as the node itself. Post the nodes construction, the label assignment task is maximized with respect to the exclusion, such that the connected nodes have different labels. According to [10], multiple graphs are constructed to model the spatial-temporal, relationship, appearance, exclusion and propagate labels in the graphs.

Such models belong to detection level exclusion models. Meanwhile, in trajectory level exclusion models, two close detection hypotheses having different trajectory labels are penalized in a way to suppress one trajectory label. This penalty term varies as per the requirement. In [20] it is inversely proportional to the distance between two detection responses with different trajectory labels while in [19] it is proportional to the spatio-temporal overlap between two trajectories.

Apart from modeling the strategic mentioned above, different techniques could also be employed to model the tracking problem. But the above-mentioned strategies give an overview of methods being used or explored by the community.

2.1.5.2. ***Tracking methods***

DBT is the most commonly used paradigm by the community for tracking objects. This paradigm has two common steps: detection, wherein a pre-trained object detector gets the information about the object's locations and observation models are built upon it. Secondly, tracking, where these different observation models are linked with different targets. Such linking (observation models to targets) process can be categorized as Bayesian theory-based and data association based.

Such Bayesian theory-based trackers can be related to dynamic modeling and observation modeling. The dynamic models correspond to the tracking strategy while the observation model provides the observation measurements related to the object's states. Such a framework works on the predict-update mechanism. The predict step estimates the posterior probability distribution of the current state by integrating into the space of the last object state via the dynamic model. In simple words, it estimates the current state of an object based on all the previous observations given to it. The update step then updates the posterior probability distribution of states based on the obtained measurements under such an observation model. But in practice, computing the state distribution is not straight forward and hence some assumptions have to be made. By making such an

assumption we can arrive at an approximate solution and not an exact analytical solution. Approximate solutions would be a fit in cases where there are multiple objects to be tracked and integrating the sets of states or state distributions of every object in a higher dimension is large and makes it difficult to arrive at an exact solution.

Bayesian theory-based

1. **Kalman filter:** In a linear system where the object states and noise are represented in terms of Gaussian distribution, Kalman filter is proved to an optimal estimator. It is a recursive predictive filter based on the state space techniques and recursive algorithm. The *predict* state of the filter is used to extrapolate the position of objects in a new frame based on the constant velocity constraint. Such predictions are associated with new measurements or are used to trigger detectors. A correction step uses such detection as measurements and *updates* the filter state.

   This is an efficient way to address MOT problems when the objects to be tracked are smaller in number. As the number of objects increases, it is more prone to problems like identity switches which are hard to correct due to the recursive nature of this method. Since it's an unimodal Gaussian assumption these filters cannot represent multiple hypotheses simultaneously. Such problems are tackled by Particle filters, which is based on Monte Carlo integration methods. Kalman filter fails to operate in a system where the object states are non-linear, Extended Kalman filter is used to solve such problem. In [21], Kalman filter is used to estimate the background image by modeling the color of each pixel is done by one filter but fails to handle the illumination changes since illumination tends to be distributed in a non-gaussian way, which is against the basic assumption of such filter.
2. **Extended Kalman filter:** As mentioned earlier, when the objects in an environment are moving in a non-linear fashion, such filters could be one of the possible solutions. This filter approximates the non-linear states of the objects using the Taylor series expansion.
3. **Particle filter:** These filters have an advantage over the other two filters mentioned above in terms of finding an approximate solution in a non-linear and multi-model distribution system. These are based on the Monte Carlo sampling technique where the underlying principle is representing the required posterior density function by a set of random particles or samples with associated weights and propagating multiple hypotheses between frames. It is a recursive Monte Carlo statistical computing method, which is used to solve a Bayesian estimation problem assuming that the measurement models are corrupted by noise which may be non-gaussian and multimodal. Though these filters have achieved considerable success in the tracking literature they still face problems due to the suboptimal mechanism while sampling the states which leads to the sample degeneracy & impoverishment problem [7]. Sample impoverishment refers to a state where the particles with high weights are selected many times statistically. This is observed in the case of small process noises and resampling is used to reduce the sample degeneracy. It is also observed that these filters do not perform well in the dynamic system with very small noise or if the observation noise has a very small variance. Regularized Particle Filter (RPF) and Kernel Particle Filter (KPF) are proposed with some success. Although the filters can handle multi-model states, the computational complexity increases exponentially as the number of state parameters increases in high dimensional state space.
4. **Bayesian Framework:** This framework works by estimating the unknown state, S_t at a time t from sequential observations $O_{1:t}$ perturbed by noises. The key idea is to approximate the posterior probability distribution (PDF) by a weighted sample set where each sample consists of element $S^{(n)}$ which represents the hypothetical state of an object and a corresponding discrete sampling probability $\prod^{(n)}$ such that Its summation is one.
5. **PHD filter:** The Probability Hypothesis Density (PHD) filter is a multi-target filter for recursive estimating the number and state of a set of targets given a set of observations. This is different from the traditional multi-target tracking which considers target states and their observation as Random Finite Set (RFS). Multiple Hypothesis Tracking (MHT), Joint Probabilistic Data Association Filter (JPDAF) or Probabilistic MHT(PMHT) are based on RFS where the individual targets are represented as a set-valued state and collection of observation as a set-valued observation. While PHD filters work by propagating in time the intensity of targets RFS instead of the full multi-target posterior density. According to [23], the Sequential Monte Carlo implementation of (SMC-PHD) filter suffers from high computation cost as it requires a large number of particles and relies on the clustering mechanism to provide state estimates. But the main drawback is its introduction of inaccuracies in the estimates by the clustering step and complex computation. A closed-form solution to such a problem has been alleviated using Gaussian Mixture PHD (GM-PHD) wherein the clustering step is completely replaced by Kalman filter equations but is restricted to linear-Gaussian targets. Extended Kalman filter (EK-PHD) is an extension for using estimating the target states in a non-linear dynamic system.

Data association based

Under this framework, tracking is seen as an optimization problem aiming to find the maximum a posteriori (MAP) solution to MOT. Such a framework is well suitable for offline tracking where the observations from past and future are required. Given such observations, this type of method tries to associate observations belonging to an identical object into a trajectory globally using cost functions. This method formulates the tracking problem as selecting and clustering detected objects over time.

1. **Local optimization-based data association:** Methods which considers only a few frames for solving association problem (a couple of past frames) falls under this category. They are bipartite graphs and their variants. Hungarian algorithm is one such example that computes the affinity between tracklets and detection observations. In [24], Zhong et al consider the target tracking as a bipartite graph matching problem, the nodes of the graph correspond to the targets in two neighboring frames and the edges correspond to the degree of the similarity measure between the targets in different frames. According to [10], an affinity measurement based on the position, size and color are measured and an optimized Hungarian algorithm is employed to associate object hypothesis and detection responses in the neighboring frames.
2. **Global optimization-based data association:** These methods consider a batch of frames for optimization, unlike local optimization which considers only a few frames. Dynamic and linear programming are applied to solve the optimization problem. Methods to solve data association globally are based on a graph-like structure which includes methods like hierarchical data association, quadratic Boolean problem-based and binary integer programming-based method. These methods include many more approaches under them, for which we direct the readers to [10], [4], [21].

Most of the core concepts being utilized by the community have been discussed so far. As mentioned earlier DBT is the most commonly used tracking paradigm, hence it is necessary to discuss object representation and object detection methods on a high level by not getting in-depth about it. By doing so it enables the readers to understand better about the core concepts and process of designing an object tracking system.

2.2. Object representation

In order to track an object, its visual representation plays a vital role. Such representations motivate the choice of object detection and tracking algorithms. Object representation is seen as a combination of shape and appearance representation. Though it could be used individually or in combination, based on the external factors like application domain, purpose and end goal determine how objects should be represented. Nevertheless, having information about different ways of representation is valuable information for developing a good tracking solution. Hence, a brief overview or explanation about the shape representation and appearance representation is discussed in the following section.

2.2.1. Shape representation

Based on the application domain and the purpose of object detection and tracking, the representation of the objects is chosen. Such representations carry their own advantages and disadvantages based on the application it is being used for. Usually, ways to represent objects are as points, geometric shapes, silhouettes, contour, articulated shape models and skeletal models. A short explanation of each representation is discussed. Figure 6 represents the various shape representations.

Points

The object of interest in a scene can be either represented as a single point or a collection of points as seen from Figure a &b. Representing objects as a cluster of points would cause problems while tracking multiple similar objects in the scene. Association of such points in every frame would be a difficult task leading to missed detections.

Geometric shapes

The most common way to represent an object (rigid or non-rigid) and widely used in the community is either as a rectangle or an ellipse. This can be seen in Figure c &d. According to [5], using such representation might cause tracking problems since a part of the background is also included inside these rectangles or ellipse.

Contour and Silhouette

Representation of objects with their boundaries separated from the background (background subtraction) results in either contours or silhouettes seen in Figure h & i. This is a flexible representation of any complex or non-rigid objects.

Articulated shapes

Such representation is employed when different body parts are held together by joints (Figure e). Such representations are helpful in scenarios where health monitoring is involved. For example, using such representation, the health of cows, pigs, etc. can be monitored based on the behavior of their individual body parts.

Skeletal shapes

Using the silhouette of objects and applying the medial axis transform, it is possible to extract an object skeleton (Figure f). Such a representation is being used to estimate human pose for activity recognition. This is considered an important problem in understanding human activity in a video and most of the deep learning-based techniques employ such representation.

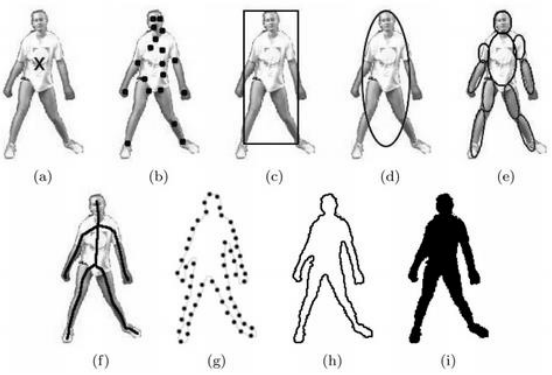


Figure 6- Various shape representation [5]

2.2.2. **Appearance representation**

Appearance representation is very similar to the shape representation but encodes more information about an object's appearance like histograms, Gaussian distribution, etc. Few ways to represent objects are using probability densities, templates, active appearance models and multi-view appearance models [22].

***Probability densities*:** This represents the probability of some random variables falling in a particular range of values. Histograms are examples of this. Such density functions can either be parametric like Gaussian distribution or non-parametric like histograms. Such calculations could be made by using above shape models like contours for example, which gives the outline or boundary of objects. From the shallow region of such contours, information about the pixels could be obtained. A color histogram is unaffected by pose change or motion and used as a reliable metric for matching after occlusion. But they do not contain any position information. The color correlogram is one variant of the color histogram where geometric information and color information are encoded according to predefined geometric configuration.

Templates: Geometric shapes and silhouettes are the best examples for such representation, whose shapes are used to form templates. Such templates encode both spatial and appearance information. They aim to represent objects with a set of pre-defined models. But using template matching in tracking framework has proven to be difficult in cases where the appearance of objects changes drastically either due to viewpoint variation of illumination. Hence such representations are more suitable, in tracking framework, where the objects pose does not vary.

Active appearance models: These models are formed by representing objects as a set of landmarks either on the boundary or inside the object region such as gradients, color, texture, etc. To arrive at such models, usually a training phase in required to learn the shape and associated appearance from the given set of examples. These models contain statistical models of shape ad grey-level appearance of the object of interest. They can embed both shape and texture into their formulation and thus allowing tracking simultaneously the outline of the object and its appearance. [25] mentions that Stegmann, using such models was able to successfully track objects by applying a deterministic approach and search algorithms to each frame.

Multi-View appearance models: Models encodes different view of objects by generating a subspace from given views using Principal Component Analysis and Independent Component Analysis (ICA) [5]

2.3. Object Features

At the low level, an object can be represented simply by the intensity value of its pixels. At the middle level, it can be represented by some features like color, texture, etc. At the highest level, it can be represented by a global feature vector which can be boosted from many features. In general, a tracking framework exploits a global feature vector to measure the similarity between target and candidate objects [25]. Since these features represent objects, they must be invariant to certain transforms like illumination, degradation, etc. Features such as edges, texture, color, motion features like optical flow, gradient features like Histogram of Gradients (HOG), Local Binary Pattern (LBP), are some of the commonly used features to represent the object of interest. The choice of object representation strongly influences feature selection.

Color models are widely being used in the literature since it strongly descriptive nature describes the object of interest. RGB (Red-Green-Blue) color spaces are used in general, but this model is not perceptually uniform. Hence most of the time HSV (Hue-Saturation-Value) color space is used, which is uniform. While texture models also play an important role, which is less sensitive to illumination changes. It measures the intensity variations of the surface of objects while representing regular patterns in an image. These are classified as structural and statistical. Morphological operations, adjacency graphs belong to structural classification while 1-D grey-level histograms, co-occurrence metrics, grey-level differences, and multi-resolution filtering methods [25].

Motion models such as optical flow measure the displacement of pixels in a region between consecutive frames using displacements vectors under the brightness constancy assumption. Lucas-Kanade were the people behind the success of optical flow calculation which computes it robustly over multiple scales using a pyramid scheme.

2.4. Object detection

Object detection is the first step in the object tracking process, which is used to localize and locate the object of interest in the frame or scene. Such detections can either be done in every frame or can be done when the object first appears in the scene. Detections can be achieved either by using hand-crafted features like HOG, Scale Invariant Feature Transforms (SIFT) features, etc. and then using a classifier like SVM to classify the objects based on the application and domain. These were the traditional vision approaches to solving object detection problems. HOG+SVM, Haar cascades were vision and machine learning-based object detection frameworks that were very popular until the rise of Deep Learning. Deep learning (supervised learning) is a part of machine learning which utilizes convolutional neural networks and has solved or trying to solve many vision-related problems easily with superior performance compared to vision-based algorithms. But such models are computation and memory hungry. But since the evolution of hardware, we have a lot of memory and computation at our disposal to work on such models and deploy as either web services or as a mobile phone application.

The following sections cover a few most popular methods used in Computer Vision.

2.4.1. Point Detectors

In this method, interest points in an image are detected. Such interest points must be stable under changes in the scene illumination and camera viewpoint.

Corners of objects are viewed as interest points. The intersection of two edges is a corner in which the direction of two edges changes. Corners are important features because these are the regions in which there are variations in the large intensity of the gradient in all possible dimensions and direction. It also includes interesting points such as ending line, maxima on a curve, minimum or maximum local intensity points. Vision-based detectors used majorly such detectors to find such interesting points that could help in locating objects.

One such method is the Harris detector, which is a 3-step process to find the corners in an image. The algorithm determines a window that contains a possible corner by calculating the intensity variation in both X and Y direction and is represented as matrices. With each such windows found, a score R is computed. Finally, depending on such values, the windows are classified as windows that contain an edge, flat or corner based on a threshold. A large value indicates a corner and negative values indicate an edge. Non-Maximum Suppression (NMS) is used to select the best or optimal corners. A better version is the Shi-Tomasi Corner detector which is built on the Harris corner detector were there a slight change in the criteria for determining a corner. Table 4 explains the fundamental difference in selecting the value of R. These are rotation invariant but not scale-invariant. A corner in the small image within a small window can be seen as a flat edge/line when it is zoomed in the same window. Hence advanced feature detectors like SIFT, SURF (Speeded-Up Robust Features), FAST(Features from Accelerated Segment Test), BRIEF(Binary Robust, Independent Elementary Features), ORB (Oriented FAST and BRIEF) are being used. Among these, SIFT and SURF are patented algorithms that can be used only for experimental and academic purposes.

Harris	Shi-Tomasi
$R = \det(M) - k(\text{trace}(M))^2$ M is intensity over small window $\det(M) = \lambda_1 \lambda_2$ $\text{trace}(M) = \lambda_1 + \lambda_2$ where λ_1 & λ_2 are eigen values of M	$R = \min(\lambda_1 \lambda_2)$

Table 4- Harris corner detector vs Shi-Tomasi corner detector criteria

2.4.2. Background Subtraction

The key idea of this method is to build a robust background model representing the background of a scene for object detection by separating the foreground and background. Such a background model serves as a referenced and must be updated continuously and must not contain any moving objects in the scene initially. Then each frame is compared to this model and any significant change in an image region w.r.t the background model indicates moving object. These are very simple and computation friendly but are very sensitive to the changes in the environment. Methods like Frame Differencing Region-based or spatial information, Hidden Markov models (HMM) and Eigenspace decomposition are few methods uses background models to detect objects [26].
There are recursive and non-recursive techniques.

Recursive techniques base the background model on every frame by recursively updating the background model. Thus, the result may get influenced by the input frames processed in the distant past. Kalman filter, Mixture of Gaussian (MoG) are a few examples. This requires less memory but are prone to error due to constant model updating.

While non-recursive techniques store a buffer with the last few frames and the model is based on only those saved frames in the buffer. They use the sliding window approach for background estimation. The relevance of the background model is updated faster and needs more memory as the buffer gets larger [27]. Frame differencing, mean and median filters are few examples.

2.4.3. Segmentation:

The goal of segmentation methods is to partition the image into perceptually similar regions. Such a process helps to locate objects and their boundaries. Segmentation algorithms try to solve two problems, the criteria for a good partition and the method for achieving efficient partitioning. Mean shift clustering, graph cuts, active contours are few popular algorithms. Graph cut based algorithms consider the segmentation problem as an optimization problem where every pixel of the images is represented as a node. The vertices between nodes and sources are set to a weight-related to the data where sources represent the labels for each pixel (foreground or background). The graph cut method then completely separates the source and sink nodes and leaves the nodes connected to either source or sink to indicate that the pixel corresponds to the respective label. Hence this is considered an optimization problem in polynomial time. Applications like analysis of moving vehicles are done using such a method.

Active contours, on the other hand, tries to find an object outline in an image. This is also popularly known as snakes. A snake is pulled towards image features like lines and edges by the guidance of external constraints combined with the influences by image forces. This method especially finds the boundaries or contours with the help of user interaction since the knowledge about the required contour shape is needed beforehand [5].

2.4.4. Supervised learning:

Initially, all object detectors relied on mechanisms to align the 2D/3D model of the object on the image using handcrafted features such as edges, key points, or templates. But with the machine learning algorithms can process such rich hand-crafted visual features using classifiers or regressors such as random forest, SVM, decision tree, etc. By training a classifier to learn different object views and appearances using supervised methods, object detection can be achieved. Such handcrafted features were diverse as Haar Wavelets [Viola et al], edgelets [Wu and Nevatia, 2005], shapelets [Sabzmeydani and Mori, 2007], histograms of oriented gradient [Dalal and Triggs, 2005], bags-of-visual-words [Lampert et al., 2008], integral histograms [Porikli, 2005], color histograms [Walk et al., 2010], covariance descriptors [Tuzel et al., 2008], linear binary patterns by Wang et al. [2009], or their combinations [Enzweiler and Gavrila, 2011] [28]. Deformable Part Model (DPM) and its variants were the most popular detectors before the rise of Neural nets-based object detectors, which are inspired by the human visual cortex. Here instead of hand-engineered features, the neural nets learned itself the filters and used no pre-

processing, making them independent from prior knowledge and human effort. The authors in [28] observe that, in less than five years, the Deep Convolution Neural Networks, a class of deep feedforward artificial neural networks has almost wiped out the very rich literature on visual descriptors.

2.4.5. Optical Flow:

A technique to find moving objects in video frames. Optical flow is the most widespread depiction of motion. Horn and Schunk computed displacement vectors using brightness constraint which assumes brightness constancy of corresponding pixels in consecutive frames. This method can get information about object movement and thus distinguishes the moving objects from the background. But such detection has prone to errors like sensitive to noise and requires more computational power.

2.4.6. Temporal differencing:

It is a method employed when the camera is in motion such as in cars. Detection of objects is done by taking the differences of consecutive frames pixel by pixel. In cases where a camera is constantly moving, modeling both motion of car and objects at the same time is challenging, since they such motions are correlated. [29] proposes to model the motion of the camera initially and then apply the background subtraction method. This method does not work well when an object moves slowly since no major change can be detected between frames.

2.5. Object tracking:

Tracking has been classified into various categories. In [30] it is divided into 5 classes namely feature-based, segmentation based, estimation based,appearance-based and learning based, which are self-explanatory and are closely related to the discussions made in the above sections.

But the following classification is widely being used in the community and is referred to in [5],[9],[31] & [32]. Most of the state of the art trackers can fall under these categories. The classification is based on the "tracked targets" which can be represented as points of interest or as silhouettes.

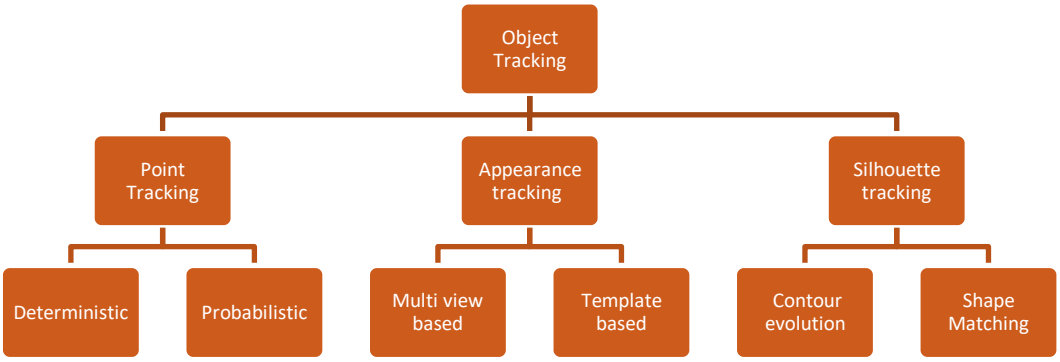


Figure 7 – Object tracking classification

2.5.1. Point tracking:

This applies to methods where the detected objects are represented as feature points and tracking of such points is based on the previous object state which could include their position and motion. Such a tracker tries to associate point in one frame to the next frame based on the observation of object state in the previous frame. This is considered as a point correspondence problem. The point tracking approach is further divided into Deterministic and statistical approaches. The main difference between this two-division is the way the approaches solve the problem of minimizing the correspondence cost. Correspondence cost is defined as the process of matching a point from the *t-1* frame to the *t* frame. And according to [5], the correspondence/association problem is hard to solve in complex situations like occlusions, misdetections and hard to handles cases like entry and exit of objects. General rules followed to minimize the correspondence are

- Assuming the object's location does not change drastically in consecutive frames.
- There is always an upper bound constraint on the velocity of the object.
- Assuming that the object's velocity does not change drastically in consecutive frames.
- The velocity of objects in a small neighborhood is constrained to be similar.
- Objects are rigid.

2.5.1.1. Deterministic approaches:

As discussed in section 2.1.4 (4), these approaches consider a system in which no randomness is involved in the development of future states. This produces the same output from a given initial condition or state. This can be applied to the object tracking by associating points across frames. Association is based on the previous object states. Tracking is enabled under the assumption that object movements follow some trajectory prototypes which could be learned offline or online. There are several methods to prototype such trajectories. One such method used in tracking people is creating the prototype offline by using the ground truth data. Certain rules or assumptions are made for prototyping such trajectories as mentioned above. The main disadvantage lies in such assumptions. This method could be problematic if the used rules are incorrect for the object's behavior in the scene.

In [5], in the case of people tracking, which is done in an offline manner, they define an energy function to compute the correctness of people trajectories. They define four rules for prototyping the trajectories based on the above general rules and define energy for each assumption. The complete energy is then the weighted combination of all such energies. This energy predicts the pedestrian locations in the next frame in such a way that the pedestrian moves to the locations that minimize this energy. The training phase is involved to learn the weights that make the predicted pedestrian tracks match corresponding tracks in ground truth data. A loss function is involved to compare the quality of prediction with the ground truth. Since they follow certain rules to track the pedestrians, they may be incorrect in complex movements like during occlusion and obstacles. Also, the calculated velocity is only correct if the pedestrian is always detected. The pedestrian tracking showcased in [5] were the experiments done offline using only simple sequences.

In [33], the authors implement a tracking algorithm based on a HOG descriptor. FAST is used to detect the points of interest. Each point is associated with a HOG descriptor. A similarity matrix is constructed using the HOG points located in the consecutive frames to determine a couple of matched points. These points are then tracked, to determine the object movement. Occlusion Is handled by comparing the object's speed and displacement distance of point trajectories of occluded objects with those objects in the previous frame to split the bounding box of occluded objects. But HOG descriptor's reliability decreases significantly if the contrast between the considered object and its background is low [9]

2.5.1.2. Probabilistic Approaches:

This approach is majorly used in the community to track objects based on the probability of object movements using states. Kalman filter is one of the most popular methods. This has been discussed in Section 2.1.5 under the Bayesian framework.

In the paper by K. Robert [34], detects vehicle headlights during night time and track the pair using Kalman filter. Detection is done by generating hypotheses of headlights than using traditional template matching or using morphological operations. This is done because the authors mention that such operations are not practical for traffic scenes and camera exposures. Discussion about blob area changes due to badly oriented headlights is discussed. During tracking, they assume that the route taken by the vehicles is linear and when the vehicles deviate from the path, they re-initialize the filter. In [35] authors present an algorithm to track mobile objects in different scene conditions like cluttered environment, occlusions, etc. The tracker is based on estimation, multi-feature similarity measures and trajectory filtering. A rich feature set like distance, area, shape ratio, a color histogram is defined for each object to be tracked for matching in consecutive frames. The best matching object and its state are estimated using a Kalman filter. These two pieces of information are fused to update the position and size of the tracked object. Yet, this is difficult to achieve good performance since the object trajectories are usually fragmented due to occlusions and misdetections. Hence, they propose a trajectory filtering which aims at removing the noisy trajectories and fusing fragmented trajectories belonging to the same object.
In cases of non-linear motion, extended Kalman filter or particle filter could be used as discussed in section 2.1.5.

*2.5.2. **Appearance/Kernel tracking:***

Kernel tracking or appearance tracking approach is based on computing the motion of objects represented by a primitive object region. Usually, motion models are used to tracked objects, but it also depends on the purpose of tracking. In case of analyzing only the object behavior motion information is enough for tracking but in the case where identification of the object is needed, information about the object's appearance is also important. This approach is divided into two groups: multi-view-based appearance models and template-based models.

2.5.2.1. *Template-based:*

It is a popular choice in the appearance model and has been used for a long time due to its simplicity. They are employed to track both single and multiple objects.
Features like color or image intensity can be used to form a template. They contain both spatial and appearance information. It works on the principle of matching the reference template with the given image. They look for a specific pattern in an image where the templates are generated from the previous frame. This is an easy solution for single object tracking since it is not necessary to consider the interaction between object and background and with other objects. But this is a brute force method that could be time-consuming for complex templates.

In multiple object tracking, since the interaction between object and background needs to be considered, this method is not very efficient. Hence in [36], authors consider an image as a set of layers, where the number of objects determines the number of layers in the image including an additional background layer. In each layer, is contains models such as layer appearance and a motion model corresponding to the object being represented. The background layer is used to compensate for any background motion so that the motion of an object can be calculated from the compensated image. Occlusions are handled this way explicitly.

Another solution is to use a Bayesian decision theory to track movements and detect occlusion [37]. A similarity score is calculated using the color intensity and color histograms as a feature representation for each detected pair. If this matching score is higher than a certain threshold, it is considered as a matching pair in two frames and templates are updated and tracked. If the score is below the threshold, it is further investigated for occlusion. In such cases, objects are split into different subparts and the similarity score is calculated for these parts. If one object part scores high enough while other scores low an occlusion is detected and the pair is still considered to be a match, although the template is not updated. But this fails to work under full occlusion since the object would be fully occluded. Also, this method is not reliable in cases of poor illumination, weak contrast. Also, this method in [37] is tested on video sequences that are not complex to prove its effectiveness.

In [38] authors make use of Haar and LBP combined with online boosting to detect and track humans. The image is divided into cells and the Haar-like features are applied in each cell to detect people. Each detected person is divided into 2*3 blocks and each block is further divided into 9 sub-regions. For each region, LBP is calculated and these features are used to track people. Both Haar and LBP are combined with online boosting. Such a mechanism of applying both Haar and LBP is considered a weak classification. Such weak classifiers cluster samples by assuming a gaussian distribution of considered features. Such an online boosting scheme can help systems to adapt to problems that could occur during an online process like occlusion, lighting changes, etc. But in [38] authors fail to clearly state the criteria to train such classifiers.

In some cases, the image region to be tracked is represented by histograms. Trackers such as the Mean shift uses such histograms to find the area in a video frame which is locally most like a previously initialized model. The gradient ascent method is used to shift the tracker to a new location that maximizes a similarity score between the model and the given image region. Such histograms are represented in the form of probability density functions. The target object is regularized by spatial masking with an asymmetric kernel.

2.5.2.2. *Multi-view approach:*

These are the approaches applied to track objects in a multi-video camera system. Since the template is used to represent the object to be tracked, such representation is usually based on the latest observation. Such representation models consider the object from one view only. This can cause a problem with more complex objects that may appear different from different views. Since we are dealing with a single camera view, we shall skip the discussion of such approaches. But we encourage the readers to [39]-[41].

2.5.3. ***Silhouette tracking:***

Sometimes representing objects with simple geometric shapes can be inadequate for complex shaped objects like hand or shoulders. Similarly, representing humans as skeleton models, cylindrical models may be a bad representation. But with silhouette-based methods, it is possible to represent objects accurately. The objects can be represented in the form of a color histogram or object contour. They are formed or created using the object models from previous frames during tracking. Hence the problem in tracking such contours becomes a matching problem. These are also known as region tracking. This approach is further divided into shape matching and contour evolution. While shape matching methods search for the object silhouette in the current frame, contour tracking evolves from an initial contour to its new position in the current frame by using either state-space models or direct minimization of some energy functions [9].

2.5.3.1. Shape matching:

In [42], the authors propose a novel method to detect objects using shape similarity between two objects. The moving object shape is described by a Gaussian distribution of RGB color of moving pixels and edge points. They define a reference smallest circle for a given moving blob. The circle is uniformly sampled into a set of control points. For each control point, again a set of concentric circles of various radii are used to define the bins of appearance models. Inside such bins, a gaussian color model is computed for modeling the color properties of the overlapping pixels between a circle and a detected blob. For a given control point, one-dimensional distribution is calculated. The appearance model of the blobs is defined by the normalized combination of distributions obtained from control points. The authors further sample the reference circles with 8 control points. They define a formula to describe the 2D shape description by collecting and normalizing corresponding edge points for each bin. This model is rotation and translation invariant. But the approach is analyzed where there are only two people in the scene.

2.5.3.2. Contour tracking:

These are also called as boundary tracking wherein a new contour in the current frame is evolved from the previous frame contour as an initial contour. This method is also robust to illumination changes since it used edge-based features for tracking. This is faster than shape matching since this method requires less information i.e., the area of boundaries is less than including the whole object region [5].
The authors in [42] proposed graph-cut based active contours (GCBAC) for object contour tracking. It is a two-step process wherein the first step, candidate contours are produced by taking the difference between the current and previous frame and this candidate contour is taken as initialization in the second step by applying GCBAC to the current frame directly. A predefined threshold is set and is compared with the amount of difference taken in the first step. And if the difference is less than this predefined threshold, authors consider this nonmoving object. Information gathered by using frame differencing removes the background pixels from object contour but cannot produce good contours when the objects are not moving too quickly between the frames. Hence, this approach cannot model occlusions very well.

In [43] Knag, Cohen & Medioni propose contour tracking by combining the Kalman filter and tracking algorithm based on an extended greedy snake technique. Contours of objects are represented by a set of control points called Snaxels. Object's centroid is calculated using the control points. Now, a contour is represented by its centroid and the control points represented as vectors relative to the centroid coordinate. Kalman filter is then used to estimate the new centroid position in the next frame. New control points are now calculated based on the new centroid coordinate. A greedy snake technique is applied to reconstruct the contour of moving objects. They define two energies regarding the Snaxels, internal and external. The internal energy determines the shape of the contour and external energy prevents contour from shape changes and holds the shape of it intact such that it represents target boundaries. Calculated control points get updated with the neighbor point having minimum energy. As mentioned, they employ a Kalman filter to update the centroid position. Combining this with energy values will result in tracking objects with high speed and large displacement.

2.6. Shadow removal techniques:

Shadows cause major problems while using foreground segmentation methods. Since shadows are cast from objects, they are hard to eliminate. This is a major problem in vision algorithms like object recognition, object segmentation, scene understanding, object tracking, etc. Employing shadow removal techniques improves the performance of foreground segmentation and the accuracy of object detection is improved. Sometimes, they are even considered objects, due the similar properties between the object and its shadow. This is because shadows tend to move along the object being tracked, in similar patterns and directions and thus get detected as a part of the object. The shadow areas appear as surface features and corrupt the original object area, resulting in misclassification of the object of interest and bias in the estimation of object parameters [25]. When a bounding box is used to represent the object inside it, due to shadows, a large portion inside such a box may contain shadows and the representation of objects becomes incorrect and as a result, processing of such misclassified information may lead to poor or incorrect results in the tracking or detection pipelines.

Deterministic models are employed to solve such issues and it has been proven that these models outperform statistical-based models in a noisy environment. A category of statistical models used to solve this issue is statistical nonparametric (SNP), which considers the color consistency of the human eye to detect shadows. This is discussed in [21] and is applied to various traffic systems. Shadow removal technique using GMM is discussed in [44] where authors instead of using color consistency employ the stability of states in the GMM to

determine shadows. A GMM is used for background modeling and follows a multistage approach wherein at each stage the pixels are filtered out which cannot be shadow pixels. The stages are initial showdown pixel reduction where the pixels having attenuated intensity than the background are considered as shadow candidates. Blue ratio test considers the fact that the blue sky is responsible for outdoor shadows and there is a high ratio of blue in the picture. Albedo ratio segmentation which is the ratio of the difference between two neighboring pixels and ratios of differences between foreground and background pixels. Ambient reflection correction, the difference between foreground and background pixels are calculated. Body-color segmentation, The true color of an object is calculated using a dichromatic reflection model. Verification, a technique to match the various surfaces with their expected body colors and classifies regions lying in shadows. Also, instead of using color consistency, the stability of states in the GMM to determine shadows can be used. The assumption is that shadow states are less than background states but are more stable than the foreground states. Hence, in [45] authors have used several foregrounds and shadow states instead of two group states. Then converged states are copied into a Gaussian mixture shadow model to prevent them from being overridden by foreground states [25]. In [46], Cucchiara et all presents a method exploiting HSV color space to extract moving objects, ghosts, and shadows. The statistical assumptions about the shadow region are that brightness and saturation are reduced while hue properties remain the same. The ghost objects are separated using optical flow as they do not have any motion field. In [47], the authors used texture analysis for shadow detection with the assumption that textural properties remain the same in shadow-regions. Gabor function does the texture analysis. Such methods can be employed only in the indoor environment since this method is good only to identify weak shadows. This method is also computationally expensive.

2.7. Occlusion handling techniques

MOT trackers must be able to associate uniquely between frames while tracking. Establishing such association or correspondence is hard for the complex environment where there would be frequent obstacles and occlusions. It is even more difficult if the objects in the scene are similar in appearance. The trackers need to effectively discriminate such similar-looking objects. There are numbers of literature that classify occlusions in a different way, but every occlusion can be grouped under self-occlusion, inter-object occlusion, object to background occlusion which is self-explanatory.

Kalman filter is used as a solution to track multiple objects and the tracking process becomes more difficult as the number of objects increases since mistakes in track estimation become more frequent in such scenarios. Hence particle filter is used to solve the problem. Lao and Zheng [48] propose a solution to the tracking problem where the objects show the same movement pattern and are very similar in appearance. They propose an algorithm that explores the correlations among the targets in a statistical online fashion and embeds both the correlation and most recent observation into sampling to improve the searching efficiency [25]. Also, another strategy employed is built on the assumption that a part of the object is still visible when an occlusion happens which holds in most of the cases. Hence, most of the approaches rely on this assumption and utilize the visible part to infer the state of the whole object.

The object being tracked is usually divided into grids uniformly such that it divides into several parts and affinity is computed based on the individual parts. The affinity of occluded parts should be less, and the trackers would be informed about these affinities and adopt only unconcluded parts for estimation. Also, based on feature point clustering is applied to detect the occlusion. This assumes that feature points with similar motion should belong to the same object. This works when at least certain parts of the object are still visible [47].

Strategies like “buffer and recover” are also proposed in the literature wherein the observations are buffered when occlusion occurs, and states are remembered before occlusion. Such object states are recovered based on the buffered observations and the stored states before occlusion. This strategy is used in [48], where the authors keep the trajectory alive for 15 frames when occlusion happens and extrapolates the position to grow the dormant trajectory through the occlusion. In case when the object reappears, the track is triggered again, and its identity is maintained.

There is also a “hypothesize and test” strategy where initially an occlusion proposal is hypothesized and testing such a proposal is done according to observation models. One such proposal can be seen from [49] where authors generate occlusion hypotheses based on occludable pairs of observations, which are close and with a similar scale. In [38], the hypothesized observations along with the original ones are given as input to the cost-flow framework and MAP is conducted to obtain the optimal solution [4].

In [50], Ess et al proposed an approach for autonomous navigation and path planning targeting multi-person detection and targeting. Occlusion maps are initially generated that contains a region of occluded by both pedestrians and static objects. Extended Kalman Filter (EKF) is used to reidentify the object once it becomes visible again when occlusion ends.

2.8. Tracking pipeline

The tracking pipeline can be grouped into either a top-down approach or a bottom-up approach. The choice of either of the approaches is application and domain-specific. Figure 8 shows elements involved in developing a tracking pipeline using a top-down and bottom-up approach. Explanation of individual elements is beyond the scope this paper and readers are encouraged to refer to [21] & [25]. Hence following section, briefly touches upon the core concepts involved in these two architectures that are or were used by the research community.

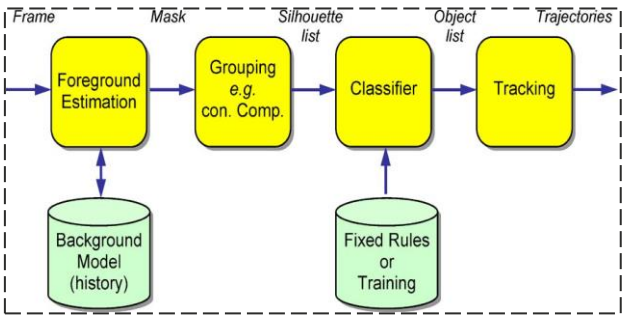


Figure 8 (a)- Top-down approach [21]

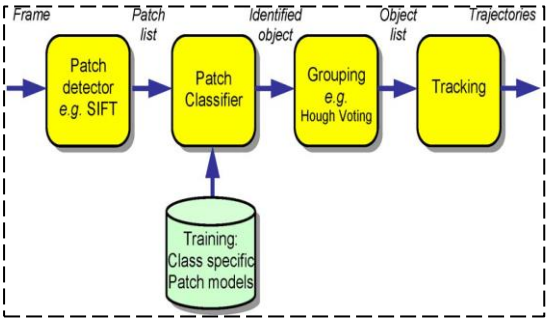


Figure 8(b) – Bottom-up approach [21]

2.8.1. Top-down approach:

In this approach, a statistical model estimates foreground pixel and is grouped together which are then classified/. Classification then uses prior information about the object classes to assign a class label. Since the pixels are grouped into objects early during the processing, it is referred to as a "top-down" approach. This uses methods like foreground subtraction techniques such as background subtraction, frame differencing, Kalman filter, GMM or MoG, and graph cuts to group the background and foreground pixels forming contours of objects to be tracked. This is followed by clustering and classification of objects, where machine learning techniques are used to generate discriminative classifiers from training data and assign class labels to unseen data. Classification of similar-looking objects is usually done using distance measures of features by calculating Manhattan distance, Euclidean distance, Mahalanobis distance, Bhattacharya distance, etc. [51,52]. Since these feature vectors usually have a higher dimension, not all dimensions are necessarily independent. Hence dimensionality reduction is applied to reduce the data to a significant dimension and helps in speeding up classification processing. Distance measure and dimensionality reduction are used on original training data while clustering would provide more meaning to such data by grouping data points together. Clustering techniques groups the training samples into a specified number of groups based on the distance between features [21]. Classifiers map the unknown features to a class, or no class based on the features the classifiers are trained on. Nearest neighbor, SVM, Bayesian or probabilistic frameworks are few examples for classifiers.

2.8.2. Bottom-up approach:

This approach involves the detection of parts objects as its first step followed by classification before they are grouped into objects. The combination of these objects into valid objects and trajectories is the final step of this approach. This is the approach followed in the object recognition task. Features are extracted using interest points like Harris corners, SIFT, SURF, HOG, gradient location, and orientation histogram (GLOTH) which form good descriptors. This is followed by boosting methods to improve the performance of simple classifiers such as AdaBoost classifiers which are weak classifiers. During training weights for weak classifiers are learned and during every round of training changes the weights of training images to forces the classifier to be trained on more difficult examples. Such weighted weak classifiers result in a strong classifier. This algorithm is robust to overfitting. Methods to model the objects are also used such as explicit shape models like k-fans, implicit-shape model (ISM) and alphabets. Such techniques directly model the spatial relationship between parts of objects detected.

2.9. Remarks:

Initial experiments using the top-down approach such as background subtraction, optical flow, and contour tracking of TSU proved to be computationally efficient but with more false positives. These false positives were due to poor background modeling (static background subtraction) and severe occlusions in the scene. Also, threshold values used for background subtraction varies as the illumination changes and cannot be well generalized. The experiments to detect objects were done using feature extractors like SIFT & SURF but proved to

be not efficient since it was difficult to reason about the certain descriptors it produced as interest points. These features don't work when there are changes in the lighting conditions and blur in the video.

The fundamental problem with the traditional approach is that it is necessary to choose or hand-pick the features that are important in each given image. And as the number of classes increases, feature extraction and classification become difficult. Also, it is hard to reason about the features which describe the different classes of objects the best. Such decisions are usually made empirically, which are long and cumbersome process whose performance would be still sub-optimal. Also, existing hand-engineered features are very designed specific, like for pedestrian detection and face detections. Such features and methods were not targeted for generic object detection. Since the rise of deep learning, the performance of such hand-crafted features is saturated and now research on vision-based object detection has reached a plateau post-2010 (Figure 9b). According to [52], research progress in vision-based object detection was slow during 2010-2012 but smaller performance gains were obtained by building ensemble systems and minor variants of methods that had some success earlier. This was due to techniques like sliding window strategy used to generate candidate bounding boxes that were redundant, inefficient, and inaccurate and were not able to bridge the semantic gap by combining manually engineered low-level descriptors and discriminatively-trained shallow models [56]. A later point in time, R. Girshick and team proposed the concept of a Region-based proposal technique based on the features extracted from raw data, automatically, using convolution neural networks (CNN), which marked the research progress of CNN based object detectors.

Deep learning introduced the concept of end-to-end learning where the machines are shown the good number of examples that are annotated with the classes of objects present in each image (Supervised learning problem). Since then the research towards CNN based object detection has evolved at an unprecedented speed. Figure 9a gives an idea about the performance of deep learning versus the traditional methods. Also, key factors for such acceleration in research are

- Availability of large dataset – Large Scale annotated training data such as ImageNet.
- Parallel computing power – Graphic Processing Unit (GPU) clusters, more recently Tensor Processing Unit (TPU) by Google which is a math library designed to solely work on Google's TensorFlow deep learning framework.
- Advances in the efficient neural net architecture designs such as VGG, GoogLeNet, ResNets, and ResNeXt which has shown significant classification and detection performance in the increasing order, respectively.
- Introduction of normalization techniques like Batch Normalization, Layer Normalization, Instance Normalization and recently Group Normalization by Facebook has proved to increase the efficiency of the deep neural nets while keeping the training time low.
- The introduction of dropouts and data augmentation techniques have relieved overfitting problems.

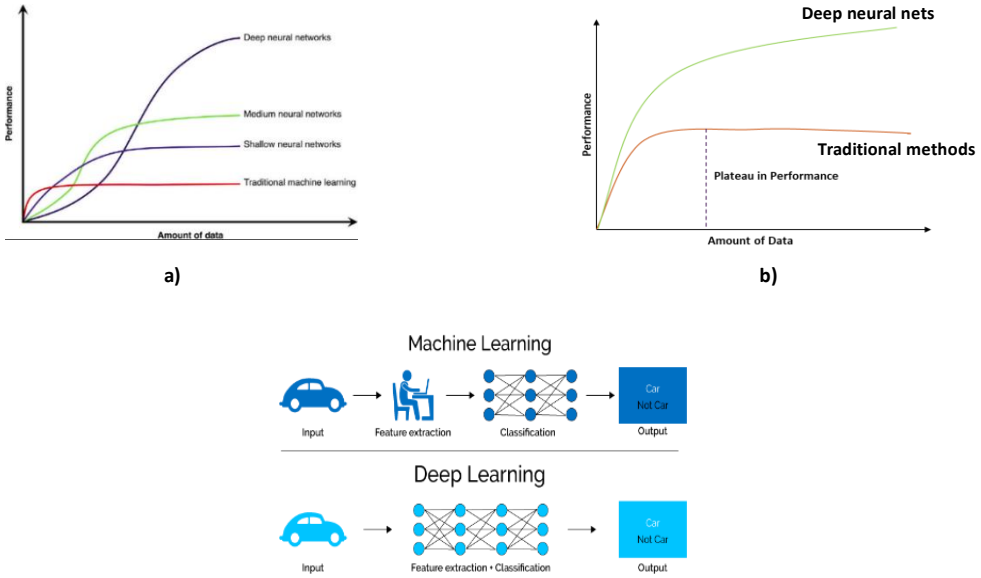


Figure 9 a) & b)- Amount of data vs performance plot for deep learning and traditional methods, c)Difference between machine learning and deep learning approach [59]

3. DEEP LEARNING TECHNIQUES:

3.1. Short history

The term “Deep learning” has gone through various incarnations since its inception in the 1950s which was proposed by McColloch and Pitts, as a binary classifier where the weights of the neural networks were updated manually. Following that was a Perceptron algorithm from Rosenblatt in 1950 that the neural nets learned to update the weights automatically but couldn’t solve the XOR problem as they were linear classifiers. Post this event, the research stagnated until the work on the backpropagation algorithm. However other reasons were like limited computing power to train even a few layers of the perceptron, lack of large training datasets, overfitting of training and insignificant performance compared to other machine learning tools. But now, a vast amount of computing power is at our disposal such that deep layers such as Resnet 101 or Inception networks can be trained on ImageNet within an hour.

Deep learning only took off since 1998 when Yann LeCun et al introduced a paper: “Efficient BackProp” which is considered as a stepping stone in deep learning, which induced the functionality of “learning” in a computational model. But as per [54], 2012 was the actual year of rebirth of the convolutional neural network when A. Krizhevsky et al [55] introduced AlexNet, a Deep convolutional Neural Network (DCNN) which achieved record-breaking image classification accuracy in the Large-Scale Visual Recognition Challenge (ILSRVC). Since then tremendous growth has been observed in research towards object recognition, classification, instance level, and semantic level segmentation. This is because a DCNN learns the high-level features in a robust way of an image automatically and thus allowing computational models to learn the complex, subtle and abstract representation where the hand-crafted features failed to do so. Figure 10(a) shows the amount of research being made on object detection since 2012 and these are deep learning-based generic object detection techniques presented. Also, authors in [54] divide the object detection progress into two historical periods: traditional object detection period (before 2014) and deep learning-based object detection period (after 2014). Milestones in generic object detection algorithms from hand-crafted feature representation until the recent deep learning generic object detection techniques can be seen from Figure 11. Though deep learning techniques perform very well, they have their own limitations or disadvantages which are mentioned in the next section. Since the analysis of its limitations in greater detail is out of the scope of this discussion, they are best taken with a grain of salt.

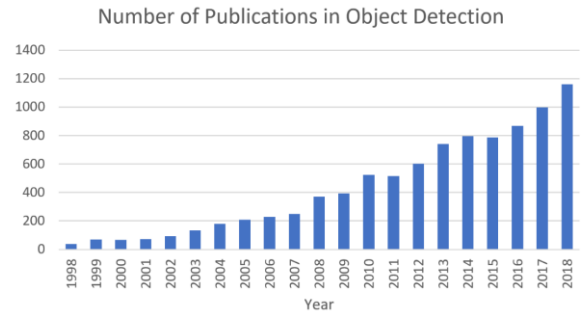


Figure 10 (a)- Object detection papers published from 1998-2018 [54]

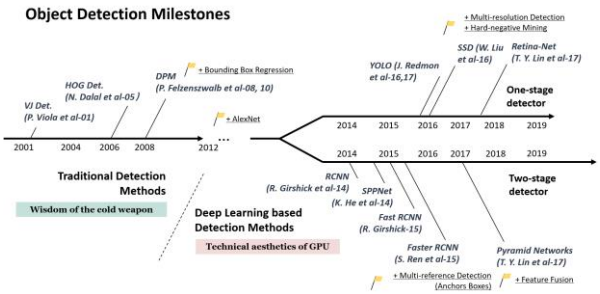


Figure 10(b) – Road map of object detection techniques [54]

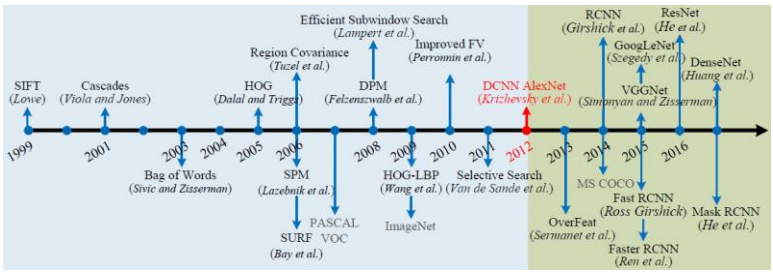


Figure 11(a)- Milestones in object detection and recognition [54]

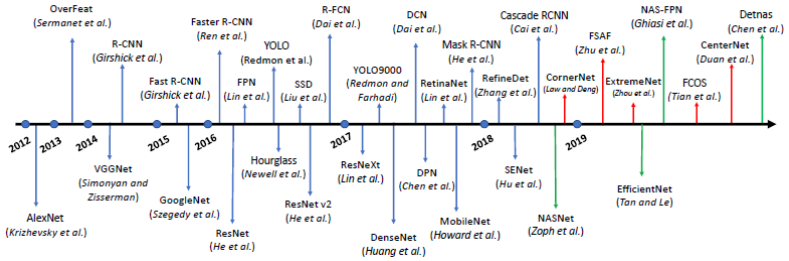


Figure 11(b) – Milestones in deep learning-based object detection [143]

3.2. Advantages of deep learning over traditional methods:

- Hidden information in the input data can be disentangled via multi-level nonlinear mapping.
- Learning hierarchical feature representation by a hierarchical multi-stage structure automatically.
- Increased performance and expressive capabilities.
- Some computer vision problems can be recast as higher-dimensional data transform problems and can be solved from a different point of view.
- CNN architectures can be used to jointly optimize several related tasks (regression and classification in multi-task learning)
- Superior performance in image classification, object recognition, segmentation tasks & SLAM.
- Super flexibility since DCNN models can be re-trained on custom datasets for the given use case where traditional vision techniques tend to be more domain-specific.
- Training time can be reduced by using transfer learning techniques wherein a model already pre-trained on the huge dataset (ImageNet) can be used to train & fine-tune on custom dataset instead of training from scratch which is time-consuming.
- Data augmentation techniques (e.g., horizontal & vertical flipping, random cropping, etc) can be employed to increase the amount of dataset.
- DCNN models are the translation, scale-invariant but to some extent rotational invariant.

3.3. Disadvantages of deep learning techniques

- In general, the amount of labeled dataset required is high.
- Higher the training dataset, higher memory, and processing power required.
- Transfer learning can lead to catastrophic forgetting. This is observed when a network is trained on a primary task and then trained on a similar secondary task tends to learn the secondary task quickly but at the expense of forgetting the first [60]
- The dependency of CNNs on initial parameter tuning to avoid local minima.

3.4. Definition & current trend:

The goal of the generic object detection is to not only determine the presence of many predefined object categories and its instances in a given image but also to locate those objects and extent of each instance. Since there are various rigid and non-rigid bodies in images, the researchers are highly interested in localization of structured (e.g., cars) and articulated bodies (e.g., humans, cows) than the unstructured objects (cloud, river, and grass). Such an object's location and extent of instances are coarsely defined using a rectangular bounding box, pixel-wise segmentation mask or a closed boundary [56]. Most of the literature contains object detection algorithms that use bounding boxes to evaluate such algorithms. Such understanding is only restricted to image-level and there is a need to understand images on the pixel level. As a result, current research on understanding images on a pixel level is gaining more traction and hence future challenges will resolve problems related to the pixel level. Such an understanding is termed as instance segmentation and semantic segmentation. Instance segmentation aims at differentiating different instances of the same object class while semantic segmentation does not differentiate the instances. Figure 12 shows the difference and it is clear that generic object detection is like semantic segmentation. Research is also moving towards detecting the key points on the object of interest to understand the pose of interesting objects in the scene. Such developed algorithms are mainly targeted to analyze the pose of humans but could be extended to any articulated objects such as animals to monitor their behavior and health (e.g., pigs).

Our main focus is on object detection and not on segmentation. Hence, the explanation of segmentation algorithms is beyond the scope of this discussion and for the sake of completeness Table 5 summaries major research papers in both the mentioned fields.

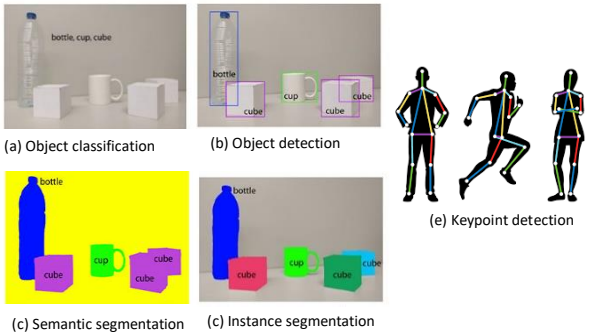


Figure 12 – a) Image classification, b) object detection, c) semantic segmentation, d) instance segmentation, e) Keypoint detection [57,58]

Class	Papers
Object detection	**Single stage detectors**- SSD [61], YOLO [62], RetinaNet [63]
	Two stage Detectors- SPPNet [64], Fast R-CNN [65], Faster R-CNN [66], Feature Pyramid Network (FPN) [67], Region- based Fully Convolutional Networks (R-FCN) [68], DetNet [69]
Instance/Semantic Segmentation	Mask RCNN [70], U-Net [71], SegNet [72], DeepLab [73], PANet [74], FPN [67], Fully Convolutional Network (FCN) [75], ParseNet [76], Convolutional & Deconvolutional Networks [77]

Table 5- Various SOTA papers on generic object detection and segmentation

3.5. Detection framework:

In this section, we will review a few object detectors referred to in Figure 11b. All the recent object detectors are based on one of such baseline detectors, attempting to improve on one or more aspects. The object detectors are classified as single-stage and two-stage detectors based on the network architecture and its way of working which is summarised in Figure 13. Two-stage detectors include a pre-processing step for region proposal and hence the name two-stage. This is termed as a "coarse-to-fine" process[54]. Single state or regional proposal free detectors considers object detection problem as regression and classification problem and does not separate detection proposal making the pipeline single stage. This is termed as a "complete in one step" process. These two approaches are correlated and are bridged by the introduction of a concept called "anchors", by the Faster RCNN paper[66].

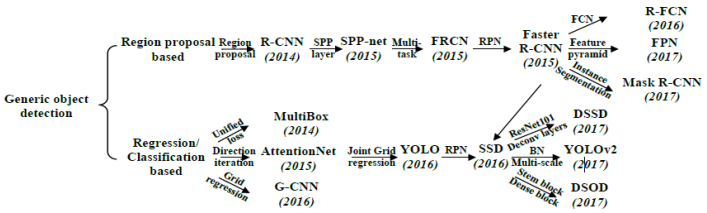


Figure 13 [78]- Object detection framework

3.5.1. Two-stage detectors

These detectors have a two-step process: proposing regions & classification+bounding box regression. The purpose of the region proposal is to present the classifier with class-agnostic rectangular bounding boxes to locate the ground-truth instances. The assigning of a class to each of the proposed regions is done by the classifier while the regressor fine-tunes the coordinates of the proposed boxes. This is based on the human's attention mechanism which firstly scans the whole image coarsely and then focuses on the region of interest. Such region proposal approach can be seen in Region-based Convolutional Neural Networks(R-CNN), Fast R-CNN, Faster R-CNN, R-FCN and Mask R-CNN[70]

3.5.1.1. Region Proposal

Several approaches were designed to propose regions such as measuring objectness, Constrained Parametric Min-Cuts (CPMC), Selective Search, Prime Object Proposal with Randomized Prim, EdgeBoxes and many more. They were also evaluated their effect on the detector's performance. It was observed that out of all these approaches, the best speed and recall were given by Selective Search and EdgeBox. Such approaches had fundamental problems with the speed compared to newer detectors being proposed at that time (Fast RCNN). Hence it was observed was in the design of the region proposal part in the detection pipeline. Post this, some work on deep learning-based approaches such as [Erhan et al., 2014, Szegedy et al, 2014] were designed to propose regions but they were not end-to-end trainable for detection. Faster RCNN introduced by Ren et al., 2015 showed that using the same backbone architecture as used in Fast RCNN for classification could be used to generate proposals as well and they termed it as Region Proposal Network (RPN). The following sections will discuss more on the evolution of Faster RCNN with its pros and cons.

3.5.1.2. Evolution of Faster RCNN

- **Region-based Convolutional Neural Networks(R-CNN)**

This was proposed by Girshick et al., 2014 with the main idea of using selective search as its first step to identify a manageable number of bounding box object region candidates (RoI, region or interest). Based on these ROIs, features are extracted using CNNs from each region independently for classification which can be seen in Figure 14.

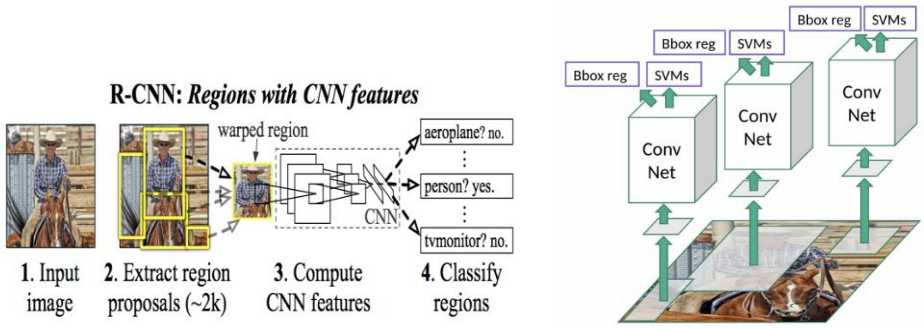


Figure 14 [80] – R-CNN architecture

Region Proposal: R-CNN adopts a selective search to generate about 2000 region proposals for each image which are category-independent and may contain targets objects with different possible sizes. The selective search methods rely on bottom-up grouping and saliency cues to propose accurate bounding boxes of various sizes quickly reducing the search space in object detection.

Feature extraction: In this stage, each region proposal is warped and cropped to have a fixed resolution for the CNN feature extractor which is of 4096-dimensional feature vector as its final representation. At this point, we can employ either transfer learning technique (discussed later) or training from scratch (not recommended sometimes) to extract features. Using transfer learning, the pre-trained CNN models can be used as a starting point for training, which accelerates the training process by learning quickly compared to training CNN from scratch. These pre-trained models are usually trained on large datasets like ILSVRC (ImageNet) and can be fine-tuned for specific domains.

CNN model in R-CNN is fine-tuned on warped proposal regions for K+1 classes: K refers to the number of classes with an additional background class. During this stage, CNN learns the hierarchical structure, semantic and robust feature representation for each proposal of the image.

Classification and localization: One forward pass through the CNN generates a feature vector for every image region and is consumed by a binary SVM trained for each class independently. These linear SVMs are again pre-trained for a specific category for multiple classes. Difference region proposals are then scored on a set of positive regions and negative regions (background). Such scored regions are then adjusted with bounding box regression and filtered by a greedy Non-Maximum Suppression (NMS) to output the final bounding boxes for preserved object locations. Positive samples are proposed regions with Intersection over Union (IoU) overlap with a pre-determined threshold (>=0.3 as per [80]) and negative samples are irrelevant. In order to reduced the localization error, the regressor is trained to correct the predicted detection window on bounding box correction offset using CNN features.

Bottleneck:

Training an R-CNN model is expensive in space and time since extracted features from 2k regions per image has to be stored. The time taken to process even a small dataset with deep networks like VGG16 is very long. Also, obtained region proposals from the selective search have high recalls, the proposals are still redundant and time-consuming (approximately 2 seconds to extract 2k proposals for an image) leading to slow detection speed (14s per image with GPU). Also, since the selective search is a fixed algorithm, it does not learn anything and this may lead to the generation of bad candidate region proposals. The whole pipeline involved three models separately without shared computation: CNN for image classification and feature extraction; SVM for identifying target objects and the regressor for tightening region bounding boxes. Training such models are time-consuming and require them to train them separately. R-CNN has a Fully Connect layer (FC), where CNN requires a **fixed-size input image** (244×224 image for AlexNet)leading to the re-computation of the whole CNN for each evaluated region, taking a considerable amount of time in the testing period. Later in the same year, SPPNet was proposed to overcome these problems.

- **Spatial Pyramid Pooling (SPP)-Net:**

He et al. proposed a novel approach called SPPNet. This architecture inspired by the Spatial Pyramid Matching (SPM) approach which takes several finer to coarser scales to partition the image into a number of divisions. Quantized local features are aggregated and represented as mid-level features.

Conventionally, at the transition of the conv layer and FC layer, there is one single pooling layer or no pooling layer. Authors of SPP-net built multiple pooling layers with different scale as shown in Figure 15a. Convolution layers accept input image with variable sizes but the hard requirement of fixed-size images in CNN is only due to the Fully Connected (FC) layer. Since FC layers need a fixed-sized input image, R-CNN has to warp and crop each region proposal into the same size. If any objects exist partly in the cropped region, unwanted geometric distortion may be produced due to warping and results in reducing the classification accuracy and is more prone to error when the scales of objects changes. SPP-net solves this issue by resuing feature maps of the 5th convolutional layer (conv5) to project region proposals of arbitrary sizes to fixed-length feature vectors. The layer post the conv layer is referred to as the SPP layer. This layer enables CNN to generate a fixed-length feature representation irrespective of the input image/region of interest without rescaling it as in R-CNN. Thus, when used for object detection, the feature maps are computed only once from the entire image and then fixed-length arbitrary regions are generated for training the detectors avoiding repeated computing features. SPPnet processes the image at conv layers for only one time while the R-CNN process it at conv layers for 2000 times because of 2000 region proposals. It is 20 times faster than it's predecessor, R-CNN, without the loss in detection accuracy. So, if the number of feature maps in conv5 is 256, then by taking a 3-level pyramid, the final feature vector for each region proposal post the SPP layer has a dimension of 5376 [256× (4^2+2^2+1^2)]. Due to this architecture, SPPNet produces a correct estimation of different region proposals in their corresponding scales with significant improvement in the detection by sharing computation cost before the SPP layer among different proposals[78].

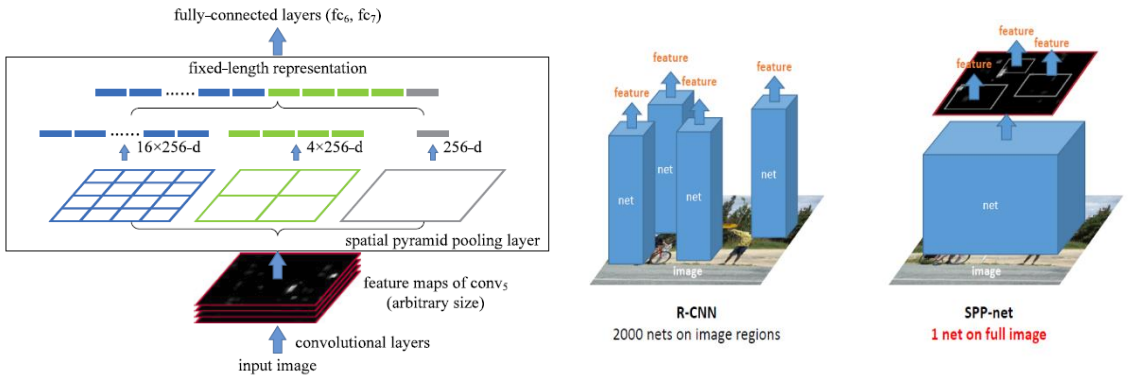


Figure 15 a)– SPPNet with Pyramid {4×4,2×2,1×1} **b) Difference between R-CNN & SPPNet [64,82]**

Bottleneck:

Though the SPPNet improves the detection speed, it has drawbacks like the training is still multi-stage and fine-tuning is done only on FC layers while it ignores all previous layers. Also, end-to-end training is still not possible due to the presence of SVM and regressor after the FC layer. The conv layers before the SPP layer cannot be updated using the fine-tuning algorithm introduced in [64]. In the succeeding year, Fast R-CNN was introduced and solved this problem. Since the feature vectors are stored in hard drives, it occupied a large amount of memory to train the regressor. An R-CNN with VGG16 as its backbone has a test time of 47s per image using a GPU which is slow.

- **Fast R-CNN**

Girshick [65] proposed Fast R-CNN which solved the problems associated with SPPNet and R-CNN resulting in state of the art detection speed and quality in 2015. This architecture unified three independent models into one jointly trained framework thus increasing the shared computation and training. This enabled researchers to train this network en-to-end and obtain higher performance results than that of SPPNet and R-CNN. It achieved a detection speed of 200x faster than R-CNN[54]. Since the conv layers before the SPP layer cannot be updated by the fine-tuning algorithms, the accuracy drop was obvious. By the introduction of a multi-task loss on classification and bounding box regression, Fast R-CNN was able to increase the detection accuracy and now it was trainable end-to-end.

The network architecture is shown in Figure 16. It is similar to the SPPNet, the whole image is fed to the conv layers to out feature maps. Now, instead of the SPP layer, the RoI pooling layer was introduced between the last conv layer and the first FC layer to produce a fixed-length feature vector for each region proposal. This can be considered as a special case of the SPP layer with only one pyramid level. Each feature vector is further fed into FC layers and then branches into two sibling output layers, the softmax classifier, and bounding box regressor. The softmax classifier produces softmax probabilities for all K+1 categories and regressor encodes refined bounding box coordinates. It was found that Softmax performs better than SVMs. This network is sometimes referred to as the detection network in the literature.

Parameters of these layers are optimized via a multi-task loss except for the generation of region proposals (still selective search) in an end-to-end fashion.

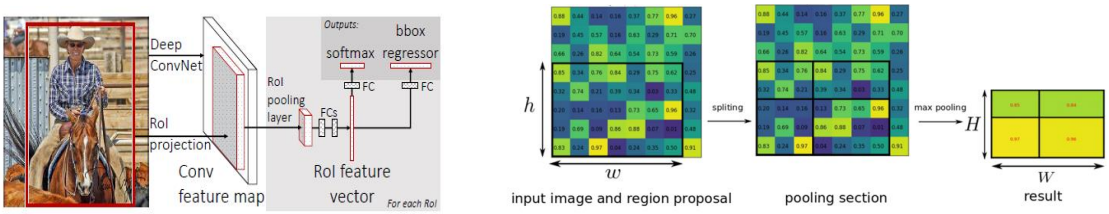


Figure 16 **a) Fast R-CNN** **b) RoI Pooling layer [65,82]**

RoI pooling:

This is a layer used for object detection where for each proposed RoI from the conv layer, it takes a section of the input feature map that corresponds to it and scales it to some pre-defined size (e.g., 2×2, 7×7). The scaling is done in 3 steps. It first divides the region proposal into a fixed dimension based on the output size mentioned above. Then it finds the largest value in each section. In the last step, the copying of these max values to the output buffer. As a result, we can get a list of corresponding feature maps with fixed size from a list of rectangles of different sizes. It is important to note that the dimension of the RoI pooling output does not depend on either the size of the input feature map nor on the size of region proposals. But, it is solely determined by the number of sections the region proposal has to be divided into. It is to be noted that at this point, the feature size reduces and increases the depth of such features. An advantage is a gain of processing speed. This is due to the fact that there will be multiple object proposals on the frame and one can reuse the same feature map for all of those proposals. Thus, the whole system can be trained in an end-to-end manner. This also introduces some error due to quantization of the region being divided unequally which was solved by introducing RoI Align in Mask R-CNN[70] which is built on top of Faster R-CNN for instance segmentation.

Example: From Figure 16b, a feature map size is 8×8 with output size after RoI pooling is 2×2 with RoI or proposal of 5×7. Now the feature map will be divided according to the equation h/H × w/W, where (w,h) [7×5 in our case] is the feature map size and (H, W) [2×2 in our case] is the output of RoI pooling size. The area for each pooling area becomes 2×3 or 3×3 after rounding. Max pooling is done within each section. It has to note that the RoI's size doesn't have to be perfectly divisible by the number of pooling section (2×2 in our case).

Multi-task loss:

This loss helps the network to train classification and bounding box regression jointly. It is defined in equation 1.

$$\mathrm{L}(p,u,t^u,v) = \mathrm{L}_{\mathrm{cls}}(p,u) + \lambda[u \geq 1]\mathrm{L}_{\mathrm{loc}}(t^u,v) \qquad (1)$$

Where $\mathrm{L}_{\mathrm{cls}}(p,u)$ = - log p_u calculates the log loss for ground truth class u and p_u are derived from the discrete probability distribution p= (p_0,....., p_c) over K+1 outputs from the last FC layer. $\mathrm{L}_{\mathrm{loc}}(t^u,v)$ is defined over predicted offsets $t^u = (t_x^u, t_y^u, t_w^u, t_h^u)$ and ground truth bounding box regression targets indicated as $v = (v_x, v_y, v_w, v_h,)$, where (x,y,w,h) denotes the two coordinates of the box center, width and height respectively. $[u \geq 1]$ is known as the Iverson bracket indicator function employed to omit all background ROIs. A smooth L1 loss is employed to provide more robustness against the outliers and to eliminate the exploding gradients. It helps to fit the bounding box regressors as

$$L_{loc}(t^u, v) = \sum_{i \epsilon x,y,w,h} \text{smooth}_{L1}(t_i^u - v_i) \quad (2)$$

Where

$$\text{smooth}_{L1}(x) = \begin{cases} 0.5x^2 & if|x| < 1 \\ |x| - 0.5 & otherwise \end{cases} \quad (3)$$

In spite of such improvements, the region proposals are done using Selective Search which is external to Fast R-CNN and also the FC layers which is a test time bottleneck. Authors have used truncated Single Value Decomposition (SVD) to reduce the number of FC connections, in turn, reducing the test time with negligible drop inaccuracy. As mentioned earlier, Fast R-CNN is more accurate and faster than R-CNN and SPPNet. VGG 16 based Fast R-CNN can be trained 9× times faster than R-CNN which is 213× faster at test time. Compared to SPPNet, this network can be trained 3× faster and is 10× faster [65]. The dataset was trained on Nvidia K40 GPUs. Figure 17 compares the training and testing time of all the three architectures discussed. This Figure is collective information taken from [80], [64] & [65] which gives an idea of how fast the Fast R-CNN is compared to its predecessors.

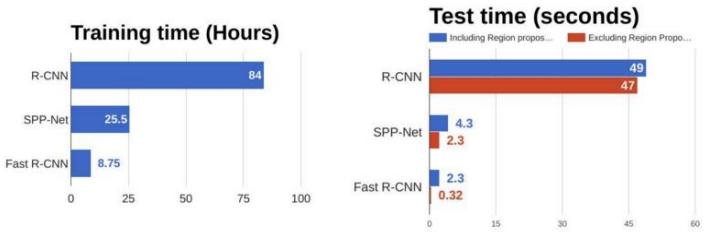


Figure 17 – Training & testing time comparison of R-CNN, SPPNet, and Fast R-CNN [83]

Bottleneck:

The test time bottleneck is solely because of the region proposals which can be inferred from the above Figure, where excluding the region proposals reduced the test time to 0.32 seconds per image to 2.3 seconds affecting Its performance. The region proposal approach and the detection network are decoupled which hurts the performance if there are false negatives. So, the obvious question was at that time, if the region proposals can be achieved using convolutional networks which could be trained.

- **Faster R-CNN**

Despite the attempt to improve the accuracy, training and testing time, Fast R-CNN had a testing time bottleneck and this was mainly due to the Selective Search used to generate region proposals. This is also a performance bottleneck since this cannot be trained. Ren et al.,[66] solved this problem by introducing Region Proposal Network (RPN), an efficient fully convolutional approach that learns the "objectness" of all instances and accumulates the proposals to be used by the detector part [28]. In simple terms, this is very similar to Fast R-CNN, an image is fed to the convolutional layers to produce the feature maps. Now, instead of using selective search, RPN is used to get the proposals. The authors introduced the concept of "anchors" (discussed later), which are nothing but boxes with various scale and aspect ratio which are translation invariant. These proposals are then reshaped using a detector network which includes the RoI pooling layer, softmax classifier, and bounding box regressor. This RPN shares the convolutional layers with the detection network (Fast R-CNN) boosting the test time and accuracy to the new SOTA performing network. The marginal cost for computing proposals is small (e.g, 10ms per image).

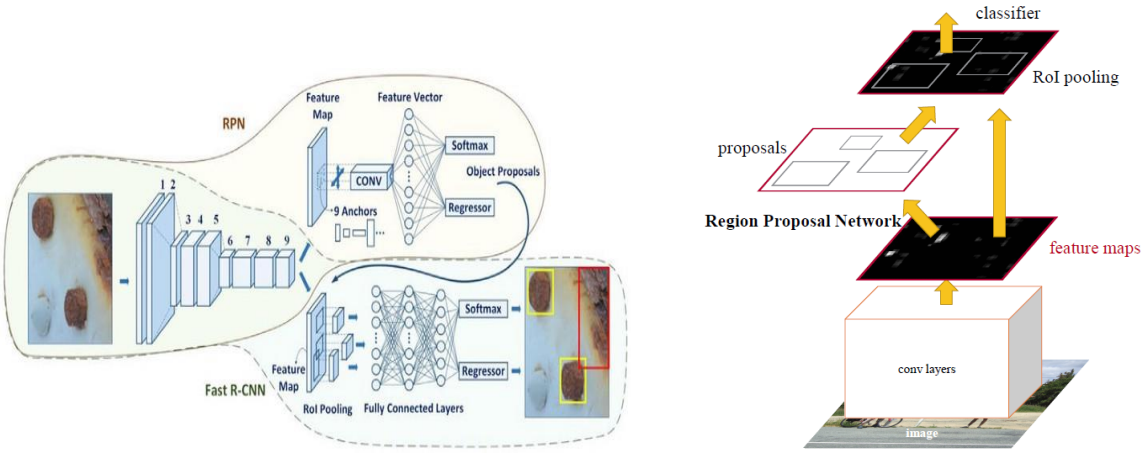


Figure 18- Faster R-CNN architecture [84,66]

Region Proposal Network (RPN):

The architecture is shown in Figure 19. RPN is a fully convolutional network that employes sliding window for each location over the feature map. At each location "k=9" anchor boxes are used having 3 scales (128×128, 256×256, 512×512) and 3 aspect ratios (1:1, 1:2, 2:1) for generating proposals. The specific conv layer (last layer usually) is used by the RPN and the preceding layers are shared with the detection network (Fast R-CNN) as shown in Figure 18. Low dimensional vector with specific dimension (varies as per the backbone architecture, 512-d for VGG16) is obtained in each sliding window and is further sent to sibling FC layers consisting of classifier and regressor which are 1×1 conv layers. The sliding window is of the format n×n and n=3 is chosen by the authors due to the receptive fields of the backbone networks used during that time (ZF =171, VGG=228 pixels [66]). The class agnostic classifier outputs 2k scores, if there is an object or not for k boxes and regressor outputs 4k coordinates (box center and w,h) of k boxes. Hence, the output of the RPN will have WHk anchors in total for a given W×H feature map. Rectified Linear Unit (ReLU) is used as an activation function here to increase the non-linearity.
With these WHk region proposals are fed to the Fast R-CNN network or detection network for further processing where RoI pooling and class-specific classification and regression takes place. Since the proposed regions will be highly overlapping, NMS is used to reduce this number (6000 to 300 top proposals). The region proposals also contain cross-boundary regions which are ignored during training and do not contribute to the loss. If failed to ignore such region proposals or anchors (outliers) introduces large, difficult to correct errors and training does not converge. But during testing, such proposals are clipped to the image boundaries [66]. Also, since there is feature sharing between RPN and detector networks training them independently will modify their convolutional layers in different ways. Hence the authors propose a 4 step alternating training scheme that alternates between fine-tuning for the region proposal task and later fine-tuning for object detection by freezing the proposals. By doing so the network converges quickly and creates a unified network with convolutional features that are shared between both RPN and detection networks.

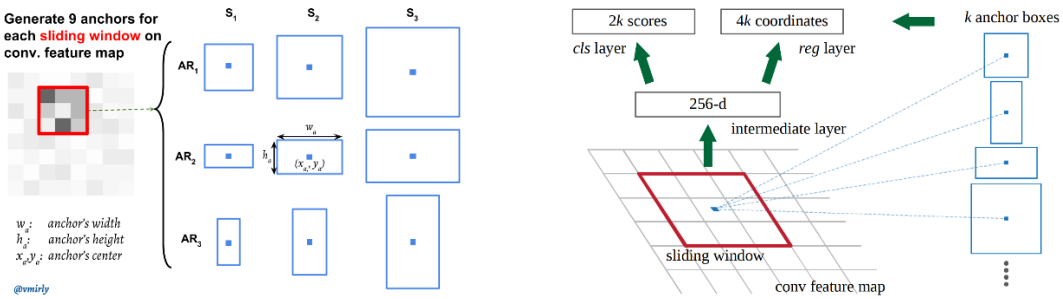


Figure 19 – RPN in Faster R-CNN [66]

Regression of true bounding boxes is achieved by comparing proposals relative to the anchors. Hence the loss function is defined as

$$L(p_i, t_i) = \frac{1}{N_{cls}} \sum_i L_{cls}(p_i, p_i^*) + \lambda \frac{1}{N_{reg}} \sum_i p_i^* L_{reg}(t_i, t_i^*) \quad (4)$$

where p_i indicates the predicted probability of the i^{th} anchor being an object.The ground truth label p_i^* is 1 if the anchor is positive, 0 otherwise. T_I has 4 parameterized coordinates of the predicted bounding box while t_i^* is related to the ground truth box overlapping with a positive anchor. L_{cls} is a binary class log loss and L_{reg} is a smooth L_1 loss. These are normalized by the N_{cls} (mini-batch size = 256) and N_{reg} (number of anchor location 2400) The term λ is the balancing parameter to make L_{cls} and L_{reg} roughly equally weighted. They also show that the accuracy remains constant for the values above 10 and a slight decrease in values (negligible) for less than 10. Feature sharing is also important since it
Now, the whole Faster R-CNN network can be trained end-to-end by using backpropagation and Stochastic Gradient Descent(SGD) algorithms in an alternating training manner. State-Of-The-Art results were obtained improving accuracy, training and testing time with near real-time object detection up to 5 fps using a GPU which is very much faster than R-CNN, SPPNet, and Fast R-CNN. The speed is also dependent on the classification network or called a backbone network used in Faster R-CNN. Paper has used ZF and VGG 16 to compare the accuracy results on various datasets like COCO, VOC PASCAL open-sourced datasets.

Bottleneck:

Faster R-CNN breaks the accuracy and speed bottleneck of the Fast R-CNN but yet there is computation redundancy at the detection stage. The FC layers after the RoI pooling layer do not share among ROIs but instead applied per RoI which makes it time-consuming and the approach slow also increases the parameters. In order to avoid such a costly RoI-wise subnetwork hundreds of times (once per proposal), Dai et al [68] proposed a novel approach where the FC layer after RoI pooling is removed completely and almost all computation is shared over the entire image which will be discussed in next section.
Figure 20 summarises the different R-CNN architectures.

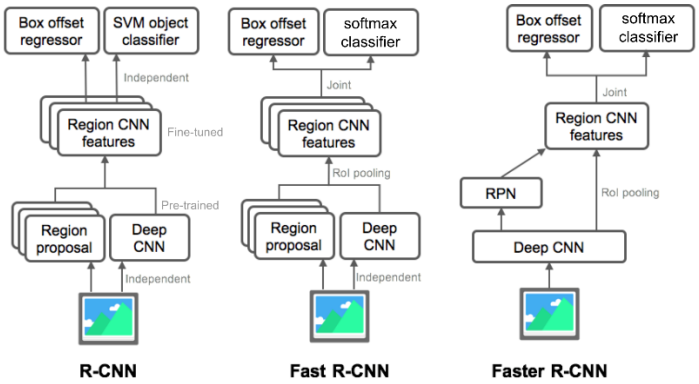


Figure 20 - R-CNN, Fast R-CNN, and Faster R-CNN architectures [79]

Further improvements like R-FCN and light head RCNN are proposed which are based on Faster R-CNN but will be discussed very briefly since we will be employing Faster R-CNN with FPN.

- **Region-based Fully Convolutional Network**

RFCN differs from Faster R-CNN only in the RoI subnetwork since the FC layer after the RoI pooling layer cannot be shared in Faster R-CNN which can be seen in Figure 21. The novel approach was the introduction of position-sensitive score maps as shown in Figure 21b where the network is completely convolutional and no FC layers are involved.

Initial idea was to have to use conv layers to construct a shared RoI subnetwork and to use RoI crops from the last layer of conv features before prediction. Later it was found that this design was a flaw and rendered poor detection accuracy because deeper conv layers are more sensitive to category semantic and less sensitive to translation[56]. But for object detection, localization representations need to respect the translation variance. In other words, an object inside an image should be indiscriminative In image classification but a translation of an object inside a bounding box must be meaningful to object detection. Hence, the idea of a manual embedding RoI pooling layer into a conv layer was discarded due to the possibility of breaking down the translation invariance property at an expense of additional unshared region-wise layers [78].

Based on this experimentation or observations, the authors constructed position-sensitive score maps (Figure 21(b)). Before obtaining position-sensitive score maps, convolution is performed according to k^2 (C+1)-d convolution, where for each class there are k^2 feature maps.
These maps are constructed by using a bank of specialized conv layers which produces a total of k^2 (top-left, top-center,…,bottom-right)position sensitive score maps of objects. These score maps are fixed grid of k×k after which position-sensitive RoI pooling layer is appended to accumulate the responses from score maps as seen from Figure 22. These specialized maps are strongly activated at a specific relative position of an object. So, if a proposed bounding box precisely overlaps with a true object most of the k^2 bins in the RoI are strongly activated and the voting leads to high scores. Also, if the proposed boxes do not overlap precisely with the true object some of the k^2 bins in the RoI are not activated and it scores less in the voting.
In the sense, each feature map is responsible for the output score for a specific part in the k×k grid. When RoI pooling the (C+1) feature maps with k^2 it outputs k^2 (C+1). In each RoI, k^2 position-sensitive scores are averaged to output C+1 d vector and softmax responses are calculated. $4k^2$ d conv layer is attached to obtain class-agnostic bounding boxes.

The pooling is done based on the same area and color criteria. Final scores are obtained by average voting each part of the RoI from the respective filter. It can be also seen from Figure 22 how the positive sensitive maps look, RoI pooling, and average voting works when the proposals overlap and do not overlap with objects. By doing this, this approach introduces some more translation variance to structures that were translation-invariant by construction [28]. This helps the object detection for learning localization representations.
This can also be trained in an end-to-end fashion similar to Faster R-CNN. The loss during training is the same as Fast R-CNN discussed above. Additionally, Online Hard Example Mining (OHEM) is used during training which

selects only top RoIs having the highest loss during backpropagation. 4-step alternative training is used to train RPN and R-FCN. NMS is used with 0.3 as the IoU threshold for post-processing the proposals (300 top proposals). The authors have used RestNet 101 as the backbone, pre-trained on ImageNet (transfer learning).

In spite of this improvement (removal of FC layers), the pipeline performs slightly worse than the Faster R-CNN but is much faster during inference speed. Also, position-sensitive RoI pooling prevents the loss of information at the RoI pooling stage in Faster R-CNN.

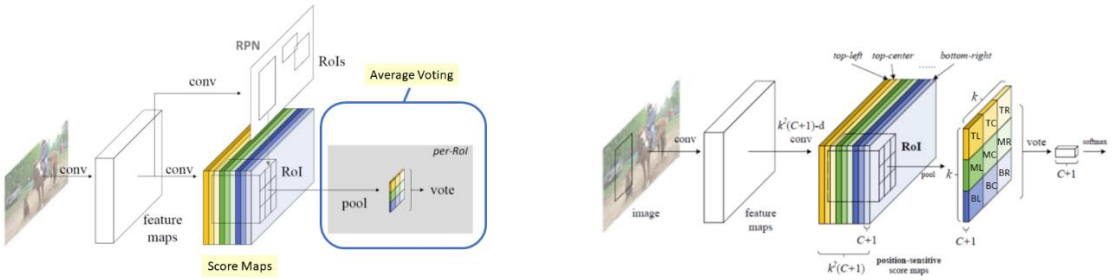


Figure 21 - a) R-FCN architecture **b) Position sensitive Score maps [68]**

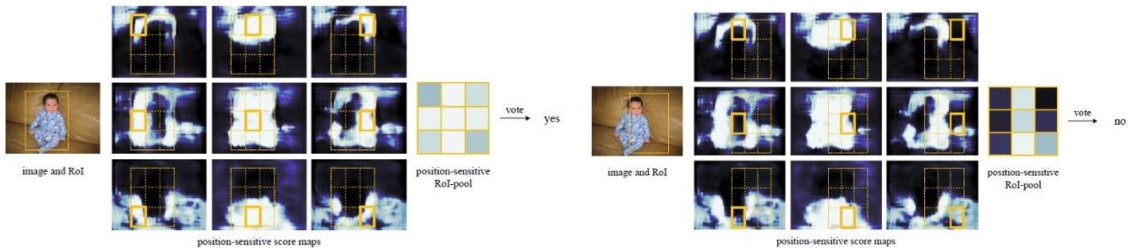


Figure 22- a) When RoI overlap with object correctly(k=3) **b) When RoI do not overlap with object correctly [68]**

- **Feature Pyramid Network (FPN):**

In 2017, Lin et al.. proposed Feature Pyramid Network[67] on top of Faster R-CNN. Prior to this, feature pyramids built on top of image pyramids (featurized image pyramids) were widely used In object detection systems which improved scale invariance [64] seen in Figure 23(a). But by doing so, it requires more training time and memory. But most of the techniques considered only a single input scale to represent high-level semantics which increased robustness to scale changes, and the image pyramids were built during test time. By doing so, it introduced inconsistency between train/test-time inference. Further, conv layers were used to build the feature pyramid using feature maps of different spatial resolutions (Figure 23(c)) which introduce large semantic gaps due to different depths. Because of this, it couldn't reuse the higher resolution maps of feature hierarchy resulting in missed detection of small objects in the scene. These are usually in single-stage detectors like SSD.

FPN takes a different approach as seen from Figure 23(d) which has a bottom-up top-down architecture. The bottom-up pathway is usually a feed-forward backbone network like ResNets and the top-down pathway is again conv layers. Such pathways combine low resolution and semantically strong features with high resolution using several lateral connections. This feature pyramid holds rich semantics at all levels and can be built quickly from a single scale input image without much-sacrificing speed and memory.

The bottom-up pathway produces feature maps and downsamples at each layer with a stride of 2. The output of the last layer of each stage will be used as a reference set of feature maps. In the top-down pathway, the higher resolution features upsampled spatially coarser, but semantically stronger feature maps from higher pyramid levels. The spatial resolution is upsampled by a factor of 2 using the nearest neighbor. Each lateral connection merges the feature maps of the same spatial size from the bottom-up pathway and the top-down pathway. In order to reduce the channel dimension 1×1 convolution layer is appended to the upsampled maps after each layer in the bottom-up pathway. The feature maps from both top-down and bottom-up pathways are merged by element-wise addition. This process introduces some aliasing in the feature maps and to avoid that a 3×3 conv layer is appended to each merged map. This is repeated until the finest resolution is generated. This depends on the layers of the backbone network. In ResNet has {C2,C3,C4,C5} layers and the corresponding feature maps extracted from each top-down pathway is named as {P2,P3,P4,P5} respectively as shown in Figure 24. C1 is called the stem of the network and is not considered because the spatial dimension of it is too large and it slows down the process too much. Outputs of all the feature maps (P2 to P5) will have a dimension of 256-d since they share the same classifier and regressor.

Faster R-CNN [66] is a combination of an RPN network and a Fast R-CNN network where the features share the same conv nets. We already know that in traditional Faster R-CNN, the RPN is a small 3x3 conv layer that produces different anchor scales with different aspect ratios from the given feature map, scaled to the original image size with two 1x1 conv layer for class agnostic classification and bounding box regression. This is called the head of the network. Hence, in FPN, the authors have adapted the RPN by replacing the single-scale feature map with their FPN. They have attached the same design to all the levels of feature maps but with a single scale assigned to each feature pyramid level with different aspect ratios. The authors have observed a similar performance with sharing and not sharing of the parameters of the heads across all feature pyramid levels.

Further, since Fast R-CNN is a region-based object detector that uses RoI pooling to extract the features in single-scale feature maps, authors have assigned ROIs of different scales to the different pyramid levels to adapt Fast R-CNN network to the FPN. The formula to assign a pyramid level P_k based on the width and height of the RoI (w.r.t input image to the network) is discussed below. The authors have also introduced two fully connected (fc) Multi-Layer-Perceptron predictor heads of 1,024-d each before the final class-specific classification and bounding box regression layers. The authors have proved empirically that this architectural change brings more accuracy improvements with little or no computational overheads than the traditional predictor heads in the Fast R-CNN which can be seen from Figure 26. Traditionally, in the Faster R-CNN with ResNet (single scale feature extractor) based models like in [107], the predictor heads in the Fast R-CNN network is usually the last convolutional layers, like conv5 layers which are shared between the RPN and Fast R-CNN network. RoI pooling is performed before conv5_1, say C4 (refer appendix Figure 8 where 'x' indicates the number of layers) and on this RoI pooled feature, all layers above conv5_x, say from C5 onwards, are adopted for each region which plays the role of *fc* layers as in VGG-16. Finally, the final classification layer is replaced by the two sibling layers.

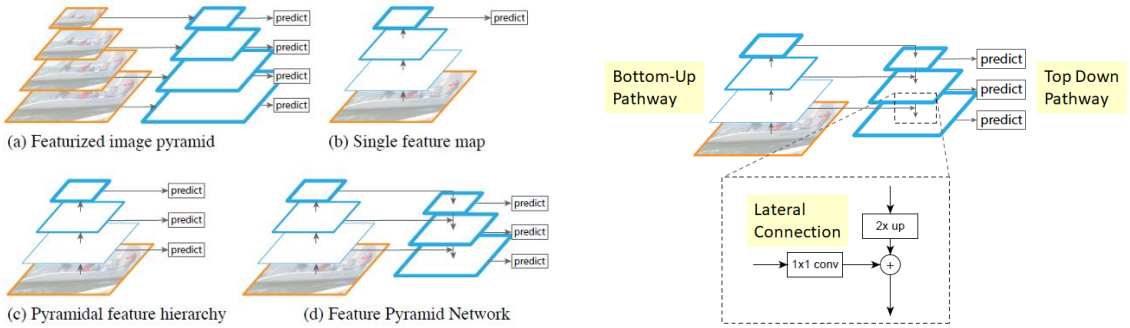
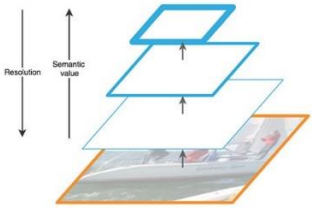


Figure 23- a) Different architectures used in detection **b) FPN architecture [67]**



c) Resolution vs semantic values [85]

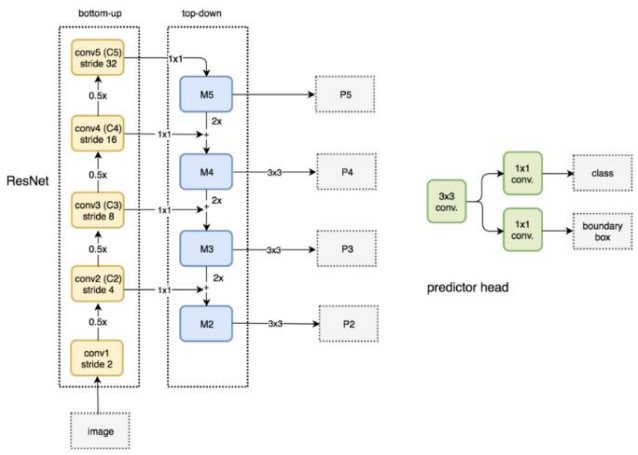


Figure 24 – ResNet 50 FPN [85]

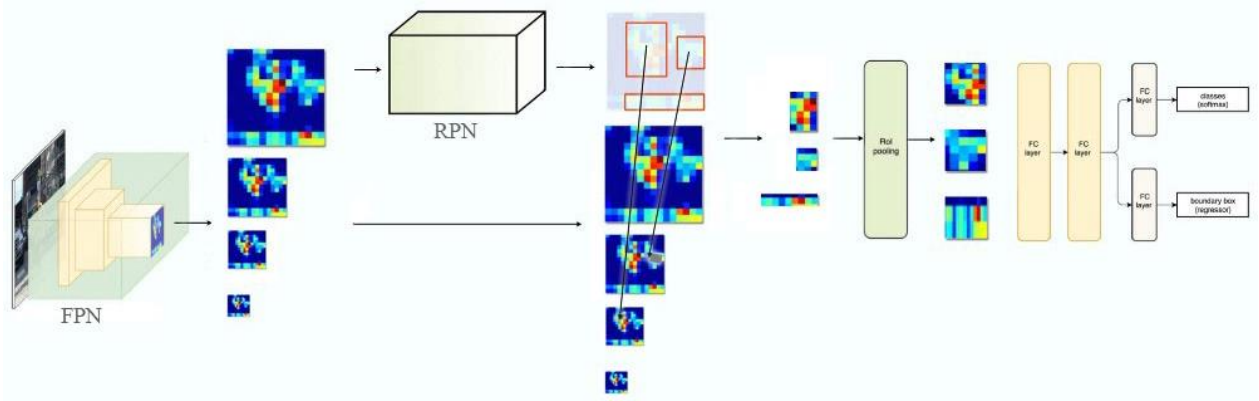


Figure 25 – Faster R-CNN pipeline with FPN [85]

Faster R-CNN	proposals	feature	head	lateral?	top-down?	AP@0.5	AP	AP_s	AP_m	AP_l
(*) baseline from He *et al.* [16]†	RPN, C_4	C_4	conv5			47.3	26.3	-	-	-
(a) baseline on conv4	RPN, C_4	C_4	conv5			53.1	31.6	13.2	35.6	**47.1**
(b) baseline on conv5	RPN, C_5	C_5	2*fc*			51.7	28.0	9.6	31.9	43.1
(c) **FPN**	RPN, $\{P_k\}$	$\{P_k\}$	2*fc*	✓	✓	**56.9**	**33.9**	**17.8**	**37.7**	45.8

Figure 26 – Object detection results using ResNet-50 FPN (c) in Faster R-CNN evaluated on the COCO minival set with traditional ResNet-50 as backbone (*, a) [67]

Figure 25 shows the FPN pipeline employed in Faster R-CNN, where now instead of generating a single feature map, the pyramid of feature maps is generated and fed to the RPN. Since the RoI or proposals can be of different size in an image, based on the size of the RoI feature map layer in the most proper scale is used to extract the feature patch. This makes sense since we have different scales of features, there is no need to apply different scales and aspect ratio of anchors in every point of a feature map in every pyramid. Single anchor scale size will be used in each pyramid but with a different aspect ratio (eg. P5 can use only anchor size of 64×64 with three aspect ratio as in Faster R-CNN). The authors provide the following equation to choose a pyramid or feature map based on the width and height of RoI.

$$k = \lfloor k_0 + log_2\left((\sqrt{w/h})/224\right) \rfloor$$

where k_0 = 4 and k is the P_k layer in FPN used to generate feature patch.
Authors have performed various ablation studies on the importance of lateral connections, pyramid representation, comparisons with single scale baseline, fixed RPN proposals for ResNet based backbone architecture and have shown empirically about their importance. More information is available in [67]
This has shown the SOTA results In the detection accuracy since R-FCN, especially for small objects in the scene, while maintaining almost the same speed as Faster R-CNN with VGG16. The work is also extended for segmentation and is out of the scope for the discussion.

Further improvements were brought to these pipelines like Li et al., [86] introduced Light Head RCNN which increases the speed of detection in RFCN by making smaller detection head and thin feature maps. Dai et al., 2016 proposed introduced RoI warping based on bilinear interpolation. He et al. introduced RoI Align which addresses the problem in RoI pooling's misalignment. Deformable R-FCN by Dai et al., 2017 improvised the detection accuracy by introducing flexibility to the Position sensitive RoI-Pooling. Cornernets by Law et al [87], where object detection is purely based on the two corners (top-left and bottom-right) they have eliminated the requirement of anchor generation which is common to both Single and Two-stage detectors and have achieved new accuracy level for one stage detectors.

3.5.2. **Single-stage detectors:**

The region proposal-based object detectors are currently leading in object detection benchmarks since Faster R-CNN and accuracy improvements are made over time which is based on such region proposals. Despite such progress, these approaches are still computationally expensive for mobile devices with limited computation capabilities and limited storage. Hence the research is being made to develop a unified strategy that considers the object detection problem as a regression problem. These detectors predict class probabilities and bounding box offsets from full images with a single feed-forward CNN network without region proposal generation or post classification [56].

The two most popular approaches are You Only Look Once (YOLO) [62] by Redmon et al., Single Shot Multibook Detector (SSD) [61] by Liu et al. Upcoming sections will discuss these detectors briefly.

- **Before SSD & YOLO.**

Szegedy et al.,[88] were the very first ones to explore object detection using CNN. They formulated it as a DNN based regression [78] named as DetectorNet. They generated a binary mask for the images and extracted detections with bounding boxes. The famous at the time AlexNet was used with the softmax classifier layer replaced by a regressor. Using this, prediction of background pixels over a coarse grid was done by a network while objects bounding box prediction was done by four additional networks. The grouping process was used to convert the detected masks into bounding boxes. But it had difficulties in handling occlusion and did not scale up to multiple classes. Also, these networks required training per object type and mask type.

Sermanet et al.,[89] proposed modern one-stage detectors which are fully convolutional deep networks called OverFeat, which used multi-scale sliding window fashion for detecting objects in a single forward pass fashion such that computation was shared by the overlapping regions. OverFeat outputs a grid of feature vectors each of which represents a slightly different context view location within an image and therefore predicting the presence of an object which is then used to produce bounding boxes. Since they use multiple scales of an image, multiple features are extracted and significant improvement in the detection accuracy is observed by providing up to six enlarged scales of original image through the network and aggregating them one by one. This resulted in an increase in the number of evaluated context views. This network was faster than R-CNN but was less accurate due to the difficulty in training a fully conv net during its time. Many more detectors were also proposed like AttentionNet [90], Grid-based CNN (G-CNN) [92], MultiBox [91]. But they all had problems dealing with either difficulty in training or were not able to scale well with the multiple classes in an image. Due to their way of functioning, the pipeline is also termed as Unified Detectors. More information on popular single-stage detectors like YOLO, SSD, RetinaNet & CornerNet will be discussed in the upcoming sections.

- **You Only Look Once (YOLO)**

It is a single-stage or unified detector which considers the object detection as a regression problem from raw image pixels to spatially separated bounding boxes with associated class probabilities. It is different from the two-stage detectors where the region proposal layer or network is completely removed, and prediction is made using a small set of candidate regions. They do so in a global context, unlike two-stage detectors where predictions are based on features from the local region. Hence, YOLO no longer requires the per-region based classification network which makes this faster than region-proposal based methods.

The basic idea of YOLO is to divide a given image into an S×S (S=7) grid such if the center of an object falls into a grid cell, that grid cell is responsible for detecting that class of object. Each grid cell predicts B bounding boxes (B=2) and associated confidence score for those boxes where each bounding box consists of 5 predictions (x, y, w, h, and confidence score). (x, y) is the center of the grid cell relative to the whole grid, (w, h) are the predicted width and height values relative to the whole image dimension. The confidence score is the Intersection Over Union (IOU) between ground truth box and predicted box. Also, each grid cell predicts the conditional class probabilities. It is important to note that the probability of an object being in a grid cell solely depends on if the centroid of the object lies in the grid cell, which also eliminates counting objects multiple times in different grids. But by doing this initial version of YOLO failed to detect objects that were very close to each other.

The model was designed using 24 convolutional layers followed by 2 FC layers. Alternating 1×1 was utilized to reduce feature space. This was a custom GoogLeNet. Fast YOLO had fewer convolutional layers, 9 instead of 24 and fewer filters in those layers.

Figure 26 shows the idea behind YOLO. For simplicity S=4 in Figure a (left one) with predicting three-class (car, light, and pedestrian).

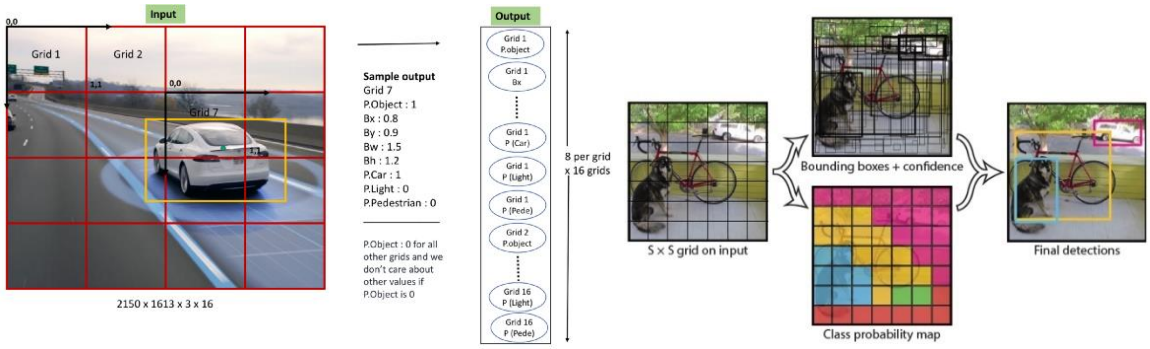


Figure 26 a) & b)YOLO grid representation [93, 62]

Unlike region proposal loss, this introduces new loss functions to be optimized which is Sum of Squared Error (SSE) calculation for predicted (x,y,w,h and confidence score) with an additional parameter λ which is a fixed value which keeps the model stable by not pushing the gradients towards zero from the grid cells whose confidence score tends towards zero. Also, YOLO struggles to localize objects correctly and produces more localization error compared to the region proposal based detectors. This was obvious from the grid cell division which was a coarse division of bounding box scale, aspect ratio, and location. It struggles to generalize to objects in new aspect ratio configurations and due to multiple downsampling of features produces coarse features. But, in spite of problems, this network was faster than two-stage detectors, processing at 45FPS and a fast version and Fast YOLO processing at 155FPS on VOC datasets.

To solve these issues further improvements were made YOLO9000 or YOLOv2 [94] where the network from YOLO was replaced by a simper network called DarkNet19. Also, newer concepts like batch normalization, removal of a fully connected network, used the concept of anchor boxes (borrowed idea from region-based detectors), dimension clustering, multi-scale training, direct location prediction, the light-weighted base model and using fine-grained features. Due to the availability of newer datasets like COCO and ILSVCR, YOLO9000 was trained to predict 9000 classes, hence the name. Further, more design tricks were added on top of YOLOv2 inspired by the recent research in object detection known as YOLOv3 [95]. It uses logistic regression instead of SSE for confidence score calculation, removal of softmax layer for class prediction, ResNet based DarkNet model, multi-scale prediction and skip-layer concatenation in the network design.

These subsequent versions [94,95] have paid more attention to decreasing the localization error for small objects by making further improvements over their custom network, DarkNet. [95] is the latest version, YOLOv3 with DarkNet-53 as its backbone has achieved new performance levels in both detection and accuracy on a relatively newer COCO dataset than VOC dataset are is faster than the SSD variants.

- **Single Shot MultiBox Detectors (SSD)**

SSD improvises on YOLO in several aspects such as a) use of default anchor boxes with varying aspect ratios for adjusting varying object shapes b) use of anchor offsets for bounding box locations c) small convolution filters for predicting categories d) usage of pyramid features to predict at different scales. Observing the intrinsic problems of YOLO [62], Liu et al [61] proposed SSD, which is faster than YOLO with competitive accuracy with its region-based detector counterparts. This was one of the first attempts at using feature pyramid hierarchy of convolution nets for efficient object detection of various sizes of objects. Fast and high-quality detection is obtained by combining the ideas from RPN, YOLO, multi-reference, and multi-resolution techniques. The main difference between any other detectors and SSD is that former detectors run detections only on their top layers while the latter detects objects at different scales on different layers of the network [54]. SSD architecture is shown in Figure 27. It is seen as a pyramid representation of images at different scales. Hence, large feature maps at the early layer can capture information about the small objects and small coarse-grained feature maps can detect large objects as well. This enables the SSD to detect objects at every pyramidal layer targeting objects of different sizes in an image.

Unlike YOLO, SSDs do not divide the image into grid cells but instead predict the offset of predefined anchor boxes (default boxes) with different scales and aspect ratios and their confidence score for every location of the feature map as seen from Figure 28. Each box has a fixed size and positive relative to its associated cell. Since feature maps at different levels have different receptive field sizes, the corresponding anchor boxes at each level are rescaled so only one feature map is responsible for detecting objects in one scale, which can also be seen in Figure 28. The cat is associated with GT is with a feature map of 8×8 (lower level) while the dog is detected in 4×4(higher level) feature map size. The network tries to optimize the weighted sum of localization loss such as Smooth L1 and confidence loss such as softmax and the detection results are usually obtained by post-processing step, NMS on multi-scale refined bounding box [78]. It uses hard negative mining to construct the negative set of examples or anchors by selecting the easily misclassified negative examples. The model picks the top candidates for such that the ratio of negative to positive examples is at most 3:1.

SSD has advantages in both accuracy and detection speed on VOC and COCO and is slightly better than the traditional YOLO [62] processing at 59FPS for an input image size of 300×300 (SSD300). This can be only be achieved by regressive trial and error method by fine-tuning vast hyperparameters like carefully choosing the default anchors, huge dataset, and more data augmentation. The model starts to converge only with a larger dataset and carefully chosen data augmentation techniques while the region-based detectors converge quickly for the small dataset and by using pre-trained models for training on the custom datasets. Despite this, SSD cannot deal with small objects in the image (objects with lesser than 32×32-pixel area). Also, since SSDs must sample from a dense set of boxes, their performance is lower in COCO compared to the region-based detectors. This is because latter methods perform predictions from a sparse set of proposals.

Various ablation studies on image resolution, data augmentation techniques are made by the authors and have compared the results with Faster R-CNN [66]. Authors conclude that data augmentation is indeed necessary to boost the performance of the network and without this, the accuracy drops significantly by 8-9 points (VOC

2007 evaluation mAP). More default box shapes lead to slightly better accuracy improvements (2-3 points, VOC 2007).
Also, they conclude that Faster R-CNN is more competitive on smaller objects than SSD due to its RPN.

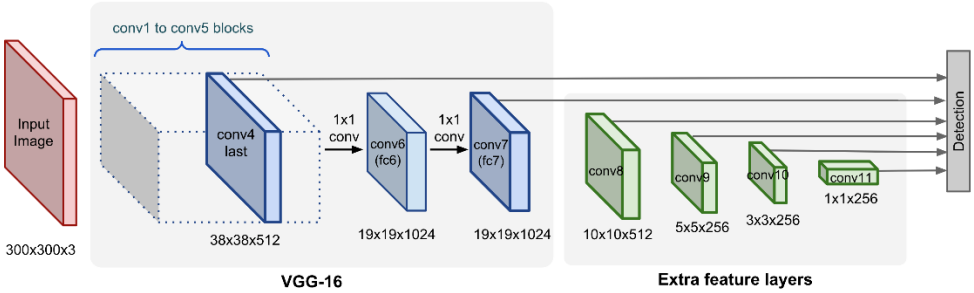


Figure 27 – SSD architecture [96]

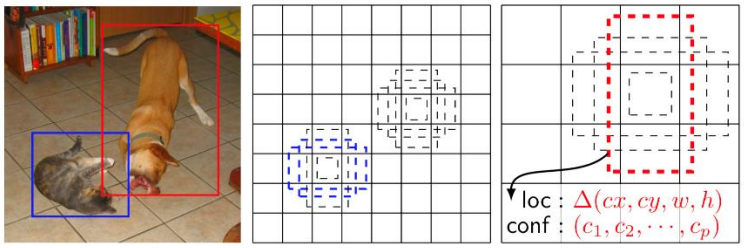


Figure 28 a) Image with GT boxes **b) 8×8 feature map** **c)4×4 feature [62]**

- **RetinaNet:**

Though SSD's being high speed and simple, they have trailed the accuracy of two-stage detectors for years [54]. Lin et al., were the very first to discover that the reason behind such a hinderance is the foreground-background class imbalance encountered during training. Observing this, they proposed a novel loss function calculation which is reshaping of cross-entropy loss function called "focal loss". This loss assigns more weights on hard, misclassified examples and down-weights easy examples during training. This enables the network to train on a sparse set of hard examples and prevents the network to be overwhelmed due to the vast number of easy negative examples.
Binary cross-entropy loss is given as for binary classification is given as

$$CE(p,y) = -y \log p (1-y) \log(1-p) \qquad (5)$$

Where $y \in \{0,1\}$ indicating the ground truth binary label and $p \in [0,1]$ indicates the confidence score.
(5) can be modified for convenience as

$$\text{Let } p_t = \begin{cases} p & if\ y = 1 \\ 1-p & otherwise \end{cases} \quad \text{then } CE(p,y) = CE(p_t) = -\log p_t$$

Now the Focal loss is defined as

$$FL(p_t) = -(1-p_t)^{\gamma} * \alpha_t * \log(p_t)$$

Where γ is called as focusing parameter and α is called balancing parameters.

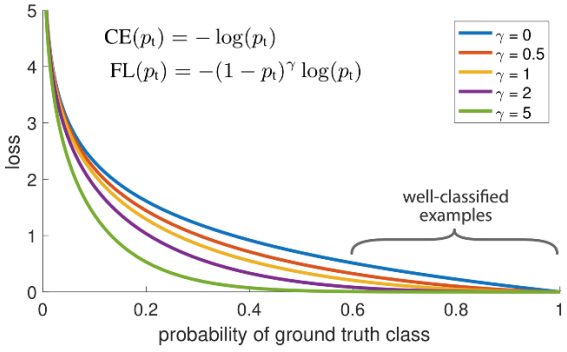


Figure 29 – Focal Loss [63]

The term $(1-p_t)^{\gamma}$, with $\gamma \geq 0$ explicitly adds at this weighting factor to each term in CE loss such that the weight is small when p_t is large thereby down weighting easy examples.

The authors have used FPN based architecture for one stage detection with Focal loss in its classification subnet and smooth L1 loss to its box regression subnet as shown in Figure 30. Modest changes were made to the FPN such as pyramids from P3 to P7. Instead of downsampling, strided convolution was used to compute P6 and P7 was additionally included to increase the detection of large objects. Also, anchors of sizes $[2^0, 2^{1/3}, 2^{2/3}]$ were added at each pyramid level with three aspect ratios (like in Faster R-CNN FPN) totaling to 9 anchors per level. Anchors were assigned to ground-truth boxes using the IoU threshold of 0.5 and to the background in [0, 0.4). It is important to note that each anchor has at most one object box and its corresponding class entry is set to 1 based on the IoU. Anchors that do not obey the IoU threshold [0,0.4) are not used during training.
Classification subnetwork was constructed using 4 Fully Convolutional Network of 3×3 each with C filters followed by ReLu activations. These are further followed by another set of 3×3 layers with KA filters where K is the object class and A (9) refers to anchors. A class-agnostic bounding box regressor, similar to the classifier but with a 4A output layer. During training, the total focal loss is computed as the sum of loss overall anchors which are normalized by the number of anchors assigned to ground-truth boxes. At most 1k top-scoring anchors per FPN level are decoded post thresholding. Post-processing like NMS was used to obtain the final detections.
Ablation study on varies values of α and γ were performed to study the effects of focal loss on AP. It was observed that as γ increased, more weights were concentrated on the hard-negative examples and the majority of the loss was dominated from a small fraction of samples. Hence the proposed focal loss discounted the effect of easy negatives and focused more on hard negative examples. Studies on OHEM vs FL were also done and authors have recorded a 3.2AP gap with OHEM and FL proving the efficiency of FL.
Speed vs accuracy study was performed by authors using ResNet-50 and ResNet-101 layers with various single-stage and two-stage networks trained on COCO. They observed a significant improvement in the AP while maintaining speed. Especially, the accuracy of RetinaNet-101 with 600-pixel input matched Faster-RCNN[67] which ran at 122ms per image compared to 172ms (measured on Nvidia M40GPU). Effects of varying anchor scales and their aspect ratios were studied and concluded that even though anchors with single scale and singe aspect ratio performed well, using 3 scales and 3 aspect ratios had an accuracy gain of 4 points. And, no further gains were observed in the number of anchors was beyond 6-9. Also, for faster inference, they noted that there was only one operating point at which RetinaNet with ResNet-50-FPN improved over ResNet-101-FPN at a resolution of 500-pixel input.

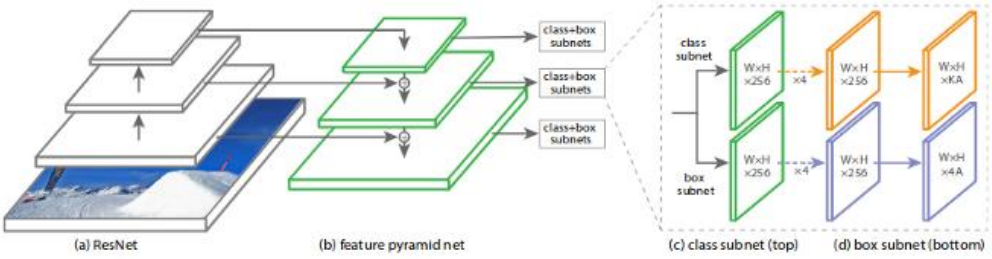


Figure 30 – RetinaNet [63]

- **CornerNet**

Hei and Jia introduced CornerNet [97] very recently and was presented in the prestigious CVPR 2019 conference. They developed this pipeline without any anchor boxes which still makes them single-stage detectors. The detector predicts the two corner points of the bounding box by localizing those two pairs. A convolution network is used to predict two sets of heatmaps to represent the corner location (top-left and bottom-right) of objects of different categories and to predict embedding vector per each detected corner. Hence, embeddings of two predicted corners from the same object are small. Using regression tighter bounding boxes are produced. Final bounding boxes are obtained by using all the embeddings, heat maps and predicted offsets during regression as post-processing. Since the network predicts two sets of heatmaps, each set of heatmap has C channels (categories) and of size H×W without background channel. And each channel is a binary mask that indicates the locations of the corners for a class. Each corner is associated with one ground-truth value and it is the positive location and all other locations are considered negative. The penalty associated with the negative locations is reduced within the radius of the positive location during training. This is done because these pairs of false corner detections, which are close to their respective ground truth, could still produce bounding boxes that sufficiently overlap with the ground-truth.

This radius is determined by the size of objects by ensuring that a pair of points within a certain radius would generate a bounding box with at least *t* IoU with the ground truth. The authors have set this value to be 0.3 in their entire experiment. Having the radius information, 2D unnormalized Gaussian is used to calculate the amount of penalty reduction to be associated with the predicted corners during training, whose center is at a

positive location with the *sigma* being 1/3rd of the radius. Authors have defined a modified version of focal loss (discussed above) with certain other hyperparameters that control the contribution of each point. Since predicted heatmaps are often downsampled compared to the ground truth heatmaps they affect the predicted boxes by introducing slight offsets. To compensate for that, authors have let the network learn them per spatial location. During training, smooth L1 Loss is used at ground truth corner locations.
Since multiple objects are present images, multiple corners may be detected. The problem is to correctly identify a pair of corner points belonging to each other. They have solved this issue by predicting embedding vectors for each top-left and bottom-right corner such that the distance between their embeddings should be small. Hence grouping is done based on the distances between embeddings of top-left and bottom-right corners. Since only the distance between embeddings is considered, the actual values of embeddings are unimportant. They use *pull* and *push* loss from [98] to train the network to group corners and these are applied only at the ground truth corner location.

Authors notice that there was often no visual evidence for the presence of corners. To determine if a pixel is a top-left corner, they max-pool along the corner's row horizontally and max-pool along column vertically and finally sum these values. So, they propose a novel corner pooling approach to better localize the corners. These corners are localized explicitly by encoding prior knowledge.

Inspired from [99], they use the hourglass network and add their modifications to it. Hourglass models/ networks first down samples input features by series of conv layers followed by max-pooling. Then upsampling of features back to their original resolution is done by series of another set of conv layers. Skip layers are added to preserve the lost information during pooling to the upsampled feature. This network captures both local and global features in a unified single network. The higher level of feature information can be obtained by stacking such hourglass modules. Authors here used two hourglass structures with their own modification as shown in Figure 31. During training data augmentations and PCA are used with network input resolution as 511×511. Ablation studies are conducted showing the importance of corner pooling, location penalty via Gaussians, branch importance in the network. With each new addition just mentioned above increased the accuracy but also revealed that the prediction of corner heatmap as a bottleneck. They have compared the results with the most modern detectors like RetinaNet and newer versions of two-stage detectors and found that this approach beats every other detector and its competitor, RetinaNet by 1.4 points but they have not discussed the processing speed of the network.

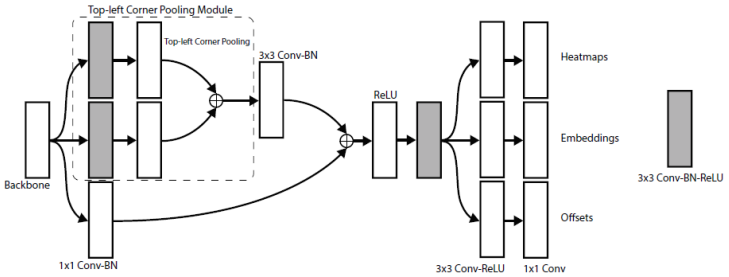


Figure 31 a)

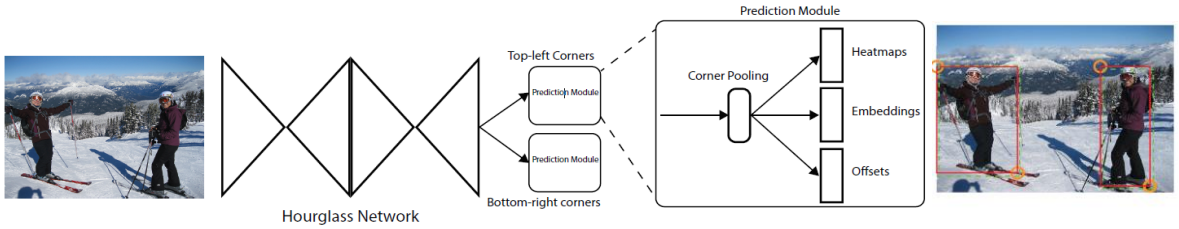


Figure 31– b) CornerNet architecture [99]

Up until now, we have discussed some of the SOTA object detectors, but the datasets and metrics used to evaluate such models were just named along in the explanations (PASCAL VOC, COCO dataset). The next section goes through the terminologies and metrics used to evaluate such models briefly.

3.6. Datasets and evaluation metrics:

3.6.1. ***Datasets***

Datasets play a key role in the progress of image analysis such as object recognition, detection, and segmentation in the field of vision and deep learning. They have been considered as one of the most important factors for the recent considerable progress in the field of deep learning. This has aided the researchers to build complex models that capture the richness and diversity of objects in images that are found in the real world. Recently, significant breakthroughs in the field of object recognition have been observed and have shown unprecedented performance. This is due to the rise of large-scale datasets with millions of images. Though datasets were available to the community since last 10 years, there has been significant growth in the numbers very recently due to three main challenges for object recognition task, PASCAL VOC (Visual Object Class), COCO (Common Objects in Context) and ImageNet Large Scale Visual Recognition Challenge (ILSVRC). Over the years there has been a significant addition of images since the beginning of such challenges. Table 6 gives some information on the number of images each challenge has released year wise with the object classes contained in those images.

- **PASCAL VOC**

Two versions are used for the evaluation of modern object detectors, VOC07 and VOC12. The former has 5k training images + 12k annotated objects and the latter has 11k training with 27k annotations with 20 different object classes. Due to the emergence of larger datasets like ILSVRC and Microsoft's COCO, this dataset has become a test-bed for new detectors [54].

- **ILSVRC**

Commonly referred to as ImageNet, is the largest open-source dataset available today which is used by most of the generic object detectors to train on and release them as pre-trained models. It has 200 classes of objects which contain 517k images with 534k annotated objects [54]. This dataset has been criticized because the objects in the images are large and well centered which makes the dataset atypical of real-world scenarios [56]. Researchers introduced COCO dataset which contains segmentation annotations, to solve ILSVRC's problem.

- **MS COCO**

Most widely used the dataset for generic object detection which is complex with new challenges such as smaller objects, various cluttered environments, heavy occlusions and has 80 classes. This has its own evaluation metric which most of the object detector papers discuss the performance on. This has become the de facto standard for performing training, validating, and testing. Also, very recently in 2017, the Test-Dev split is the default test data and most of the recent papers on object detection report their model's performance on this dataset for a fair comparison.

Challenge	Object class	Number of Images			Total number of annotated objects	
		Train	Val	Test	Train	Val
PASCAL VOC						
VOC07	20	2,501	2,510	4,952	6,301	6,307
VOC08	20	2,111	2,221	4,133	5,082	5,281
VOC09	20	3,473	3,581	6,650	8,505	8,713
VOC10	20	4,998	5,105	9,637	11,577	11,797
VOC11	20	5,717	5,823	10,994	13,609	13,841
VOC12	20	5,717	5,823	10,991	13,609	13,841
ILSVRC						
ILSVRC13	200	395,909	20,121	40,512	345,854	55,502
ILSVRC14	200	456,567	20,121	40,512	478,807	55,502
ILSVRC15	200	456,567	20,121	51,294	478,807	55,502
ILSVRC16	200	456,567	20,121	60,000	478,807	55,502
ILSVRC17	200	456,567	20,121	65,500	478,807	55,502
COCO						
COCO15	80	82,783	40,504	81,434	604,907	291,875
COCO16	80	82,783	40,504	81,434	604,907	291,875
COCO17	80	118,287	5,000	40,670	860,001	36,781
COCO18	80	118,287	5,000	40,670	860,001	36,781

Table 6 – Statistics of used datasets for object detection [56]

*3.6.2. **Metrics or evaluation criteria***

Detection speed, precision, and recall are the three fundamental metrics models are evaluated for performance. The most commonly used metric is Average Precision (AP) introduced in VOC 2007 which is derived based on precision and recall. Precision is the ratio of true object detections to the total number of objects the classifier predicts. The recall is the ratio of true object detections to the total number of objects in the dataset.
Precision-Recall or simply from the PR curve one can observe the performance of models and derive AP. For calculating Precision and Recall four values are calculated, True Positives (TP), False Positives (FP), True Negatives(TN) and False Negatives (FN).

True Positives and False Positives are calculated using the IoU thresholds. For simplicity, if the IoU is set to 0.5, then any object detection having the IoU greater than or equal to this value is considered positive and everything else is false positive. For calculating Recall, negatives are counted by measuring only False Negatives output of models i.e., the objects that the model missed to detect. Since models predict the bounding boxes with a certain threshold for every detection, by varying these confidence thresholds, one can change whether a predicted box is Positive or Negative. Based on these TP and FP the precision and recall are computed as a function of confidence threshold. Different pairs of (P, R) are obtained by varying the confidence scores making the precision to be considered as a function of recall from which AP is calculated [56]. Also, in the context of COCO, there is no distinction between AP and mAP (applies the same to AR). Hence, in the literature, one can
Since the number of objects in a given image is known from the ground truth, we can calculate TP and FP which is the calculation of Precision. IoU with the ground truth for every Positive detection of the model is calculated and is compared with the predefined set threshold (say, 0.5) and calculate the number of correct detections for each class. Precision is calculated as

$$\text{Precision} = \frac{TP}{TP + FP}$$

The recall is derived from the above, since we already calculated the correct predictions and missed detections, and is calculated as

$$\text{Recall} = \frac{TP}{TP + FN}$$

So, if the model has a precision close to 1, then whatever the model has predicted is indeed a correct prediction. And, if the recall is close to 1, then the model has positively detected all the classes of objects in the given dataset.

AP Is computed for each object category in a category-specific manner. In order to compare the performance of models overall object categories, the mean average precision is used (mAP) which is nothing but the averaged value overall object categories. This is usually used as the final metric for the evaluation of the performance. Localization accuracy is measured by checking the IoU between the predicted boxes with the ground truth with certain predefined threshold, say, 0.5. If the overlap is greater than equal to 0.5, the object was detected successfully otherwise it is marked as missed. Most of the papers still evaluate performance using this criterion. But post the availability of the COCO dataset, it has it's own metrics which helps to evaluate the models in a more precise manner. In VOC evaluation, the threshold was set to 0.5 and evaluations were made based on only this one value over 20 categories. But with COCO 2017 metrics pushed this further to evaluate over 10 such threshold values on 80 different categories (AP 0.5:0.05:0.95), which is a primary challenge metric. Hence in most of the literature, accuracy evaluations would be based on either standalone COCO metrics or combination of VOC and COCO. VOC evaluation is declining but is still used as a test-bed to evaluate the models during its initial developments.
MS COCO AP is obtained by averaging APs over multiple IoU thresholds between 0.5 and 0.95 which indicates the coarse-to-prefect localization. This depicts the accurate object localization and tends to reward models that perform better at precise localization. Alongside the varying threshold, the metric also considers the area of objects in the image for better understanding about models' performance on varying sizes of objects which are given as

- **AP** : mAP averaged over 10 $IoU_{0.5:\,0.05:\,0.95}$
- **$AP^{IoU=0.5}$** : AP at IoU=0.5
- **$AP^{IoU=0.75}$** : AP at IoU=0.75 (strict metric)
- **AP^{small}** : AP for small objects with area<32^2
- **AP^{medium}** : AP for small objects with 32^2 < area < 96^2
- **AP^{large}** : AP for small objects with area > 96^2
- **$AR^{max=1}$** : AR given 1 detection per image
- **$AR^{max=10}$** : AR given 10 detection per image
- **$AR^{max=100}$** : AR given 100 detection per image

$AR^{small,\ medium,\ large}$ is same as $AP^{small,\ medium,\ large}$

Figure 32 shows the improvements in the AP from generic object detectors.

Figure 32 – Object detector's accuracy improvements over time

*3.6.3. **Performance evaluation of detectors:***

Since a large variety of deep learning-based object detectors are being released very frequently, it is indeed hard to compare the accuracy, speed, and memory performance of all detectors on the standard benchmarks like VOC/COCO. This is because these object detectors vary in their functionality, for example- single-stage detectors work completely different than the two-stage detectors. Apart from this difference, there are many choices within the architecture of these networks which affect their performance. Choosing the right detector with optimized hyper-parameters which strikes the balance between accuracy and speed is of utmost importance for the real-life application. Choices that impact the performance of detectors are mentioned in Table 7. Amongst them, training parameters are most important because a wrong choice of values would lead to longer training time with no luck in the model's convergence towards its global minima.
It is unwise to compare every detector side-by-side since every experiment in the published papers is carried out in not very similar settings. But having information about such a qualitative comparison would enable the developers to choose an object detector architecture suitable for the targeted application. There has been very less study in such a direction except for the only study by Huang et al., [100] about the tradeoff between accuracy and speed of three main families of detectors (Faster R-CNN, R-FCN, and SSD). Also, analysis by Canzaiani & Paszke [101] provides deep insights regarding the accuracy, memory footprint, parameters, inference time and power consumption of modern backbone networks.

The authors have integrated publicly available detectors into a common platform (TensorFlow) in a unified manner. And they explore the performance of these detectors by varying parameters mentioned in Table 7. One example is seen in Figure 33, where the authors provide information about the accuracy of detectors with different backbones (discussed in the next section).

Architecture or feature extractor selection	VGG16/19, ResNet-50/101/152, ResNeXt, Inception, MobileNets, FPN, model resolution
Datasets used	VOC, COCO, ImageNet, custom data
Hyper-parameter choices during training	Usage of data augmentation, number of proposals during, IoU threshold selection, learning rates, batch size, learning decay rate, optimizers, loss functions, the requirement of batch normalization, NMS threshold configuration

Table 7- Parameters affecting the model's performance

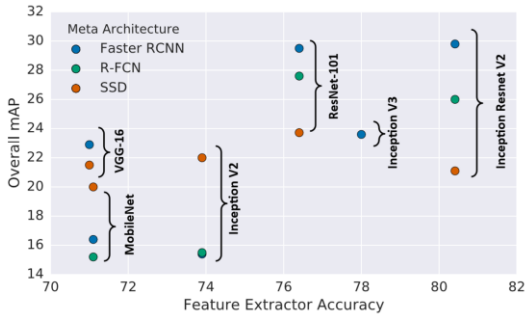


Figure 33– Top-1 accuracy comparison of three object detectors with different backbone trained on COCO. Shown only the results of lower resolution models (300×300) [100]

3.7. CNN architectures for image classification

This section briefs about the different CNN architectures used in the detection frameworks we discussed earlier. Usually, they are termed as engines [54] or backbone network because these are the conv layers that are responsible for the feature extraction which is further fed to the other layers in the pipeline of object recognition, classification or detection. They are LeNet, AlexNet, VGG, GoogLeNet, ResNet, ResNeXt, DenseNet, SENet, DarkNet and many more. The following information gives an overview of these backbones but the goal is not to discuss this to a greater extent.

- **LeNet:** Introduced by the famous, Yann Lecun, LeNet-5 [102] was introduced in 1998 for the handwritten zip code recognition. This was the very first model built using convolutional layers built on 3 main ideas: spacial subsampling, shared weights, and receptive fields. It has 3 conv layers, 2 pooling layers, and 1 FC layer. *Tanh sigmoid* is used as the activation function through the network. It achieved an error rate of less than 1% on the MNIST dataset (handwritten numbers with 32*32 image size)close to the SOTA during the time.

- **AlexNet:** Developed by Alex Krizhevsky et al.,[103] in 2012, similar to LeNet-5 achieved considerable accuracy in the ILSVRC-2012 challenge with an accuracy of 84.7%. It is 8 layer deep has 5 conv layers with variable kernel filter size (11×11, 5×5, 3×3) and 3 FC layers with the Rectified Linear Unit (ReLu) as the activation function. This was the first work to prove that by using ReLU nonlinearity, the training of deep CNNs was much faster than using the activation functions like sigmoid or tanh. The ILSVRC challenge or commonly termed as ImageNet challenge is the de facto benchmark to analyze the performance of image classification and object detection networks. In our context in this discussion, we will be referring to the classification performance of the network. Also, the top-5 error is defined as the percentage of guesses that the classifier did not include the correct class in its top-5 guesses correctly.

  This architecture reduced the top-5 error from 26% to 15.3%. Techniques like data augmentation, the introduction of dropout layers to reduce overfitting, using stochastic gradient descent, momentum and weight decay were used during training.

- **ZFNet**: Later ZFNet[104] was introduced on top of AlxeNet which further reduced the error to 14.8% by tweaking the hyper-parameters of AlexNet with some additional changes in the first layer like using a smaller filter size like a 7×7 instead of 11×11 with decreased stride value. This helped the network retain a lot of original pixel information from the input volume since 11×11 skips a lot of relevant information. This further reduced the number of trainable parameters.

- **VGGNet:** This architecture was proposed to solve the problems of AlexNet, reducing the number of parameters in the conv layer and to decrease the training time further which in turn makes this network deeper than AlexNet. VGGNet[105] has multiple variants, VGG-11, VGG-16, etc. Hence, the overall parameters increased but with the decrease in the conv layer parameters by making multiple 3×3 kernels as building blocks, unlike AlexNet which has variable conv layer kernels like 11×11, 5×5, 3×3. This architecture reduced the number of trainable parameters by approximately 45% leading to faster training time and more robust to over-fitting problems. This architecture introduced the idea of shrinking the spatial dimensions of input down the network and growing the depth by using more number of filters. This has the top-5 ILSVRC-2014 error rate of 7.3%.

- **GoogLeNet:** It is a 22 layer CNN and the winner of ILSVRC-2014 with a top 5 error rate of 6.7% [106]. This showed that by increasing the depth of the network, improves the representational power of it. This has a structure of network inside a network called inception modules, which embeds a 1×1, 3×3, 5×5 convolution

and a max-pooling layer side by side such that their outputs are concatenated at the end by a filter concatenation operation. Any feature map from the previous layer fed to this inception module unit processes it using the above-mentioned filters parallelly. For each cell, these set of filters learns to extract features at different scales from the input. Further improvements were also made to this network and are available as v2 and v3 inception network. Hence the authors showed the idea of CNNs being stacked parallelly from the traditional way of stacking CNN's sequentially. This architecture dramatically reduced the trainable parameters which are 12x lesser than AlexNet.

- **ResNet:** The Deep Residual Networks (ResNet)[107] was proposed by He et al., in 2015 which is substantially deeper than (up to 152 layers) than the previously discussed architectures. This model holds the top position in ILSVRC-2015 with the top-5 error rate of just 3.57% exceeding the human level performance(between 5-10% error rate) In the classification task.

  During the time it was generally accepted that deeper networks were able to learn more complex representations and functions which leads to better performance. It is He et al who observed that adding more layers had a negative effect on the performance of the network since this behavior was not intuitively expected. Prior to this, AlxeNet and GoogLeNet had 19 and 22 layers respectively. However, increasing the depth of the network does not necessarily work by simply stacking layers together due to the vanishing gradient problem, where, as the gradients are back-propagated to the earlier layers, recursive multiplication may cause the gradients to become infinitely small. As a result of this, as a network goes deeper, its performance may be saturated or may start degrading quickly. He et al. argued that deeper models stacked simply with the identity mappings inside a shallow network must not degrade its performance and must perform the same. Simply put, a deeper model must not produce higher training error than its shallower counterparts. They observed that this was a wrong assumption since the existing solvers were unable to produce solutions comparably good or better than the proposed solutions in a feasible time. This alluded to the fact that although with better parameter initialization techniques and batch normalization techniques allows a deep network to converge, they do so with a higher error rate than their shallow counterparts. Hence, stacking more layers would ultimately degrade the model's performance.

  They proposed a solution to this degradation problem by introducing residual blocks of skip connections or identity mappings. This mapping simply takes the output from the previous layers to the layers ahead where the addition of features takes place as seen from Figure 34a. The identity mapping (x) is usually multiplied by a linear projection, 1×1 to expand the channels of it to match the residual. If the dimensions are the same, there is no need for the 1×1 layer. Also, in practice, residual mappings are easy to optimize. The authors[106] here noted performance drop when they trained a network with 1202 layers which showed a performance drop on a CIFAR 10 dataset. Hence, the addition of skip connections in addition to the standard network flow enables the network to simply copy the activations from one ResNet block to another ResNet block or simply from layer to layer preserving information as data goes through the layers. Later, it was empirically found that by adding ReLU and batch normalization before the conv layers inside a residual unit further increases the training speed and offers better performance by allowing gradients to propagate efficiently. This is known as the pre-activation variant residual block shown in Figure 34b (rightmost (e)). With this, the authors in [108] were able to squeeze performance from the deep layers like 1001 layers deep.

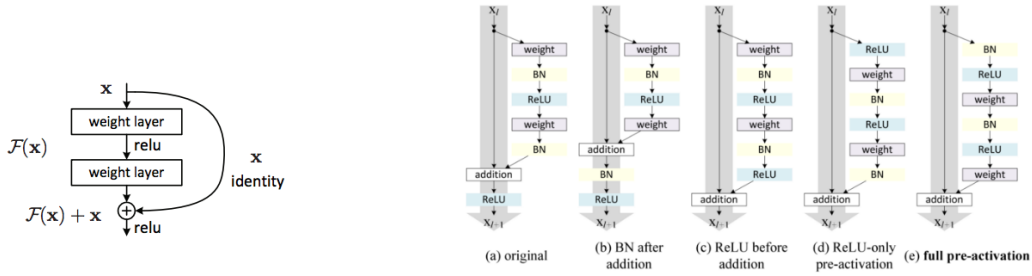


Figure 34 - a) Residual block **b)pre-acivate residual block [107,108]**

This architecture is inspired by the VGGNet as seen in Figure 35. The feature mapping is downsampled periodically by the strided convolution along with the increase in the channel depth. Dotted lines indicate the residual connections where the input is projected via a 1×1 convolution to match the dimension of a new layer or block. Figure 35 b and c, shows the difference between a 34 layer design to a 50 layer design. ResNet-50 is modeled by replacing two-layer of the residual block with a three-layer bottleneck block which again uses 1×1 convolution to reduce and restore the channel depth when calculating 3×3 convolution which further relieves the computational complexity. The authors have demonstrated with experiments that we can now train a 1001 layer deep ResNet which outperforms its shallower counterparts.

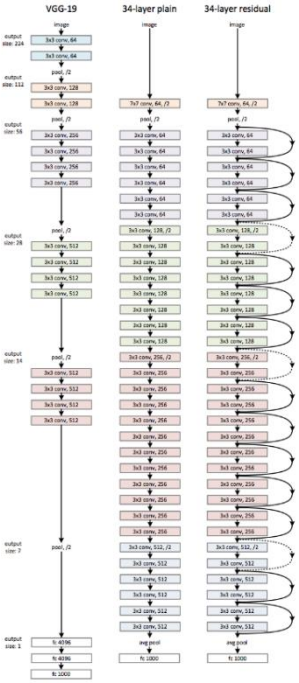
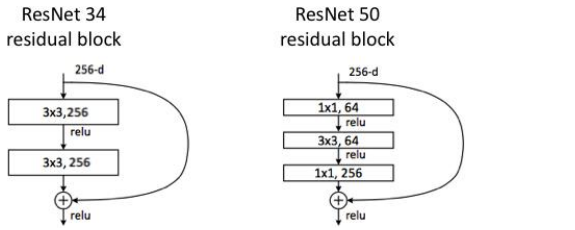
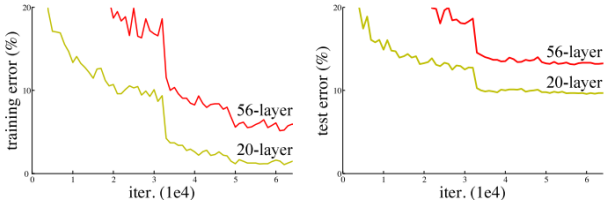


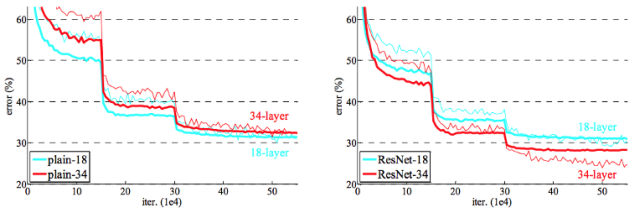
Figure 35– a) VGG, plain and residual connection [107]



b) ResNet-34 architecture [107] c) ResNet-50 architecture [107]



e) training error-plain network (PN) [107] c) testing error-PN [107]



f) training error-residual networks(RN) [107] g) testing error-RN [107]

- **ResNext:** Xie et al [109] proposed this variant which is built on top of ResNet and called ResNeXt. The architecture follows the above mentioned GoogLeNet inception modules where they follow the split-transform-merge paradigm. But the one notable difference is the addition takes place after different paths are merged while inception modules concatenate different outputs depth-wise. Also, the conv blocks share the same topology, unlike inception modules. These differences can be seen in Figure 36.
  In this architecture, the introduced a hyper-parameter called cardinality which is the number of branches or groups. This provides a new way of tuning the model capacity. They have conducted experiments to understand the performance of models with the increase in width, depth, and cardinality and found that by varying the cardinality was more efficient and effective and benefitted model performance. They also suggested that residual connections were helpful in optimizing the network.

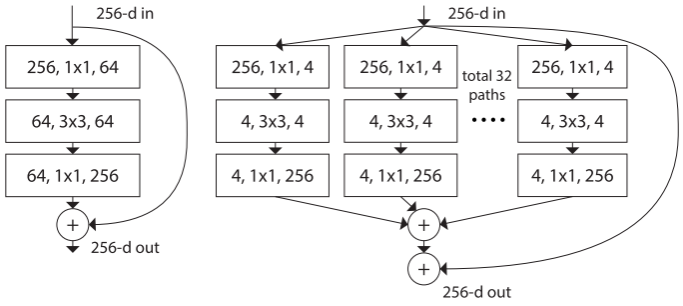


Figure 36 a)ResNet block b)ResNext block [109]

Post these architectures, many more networks are being made public, like NASNet, DenseNet [120], DetNet [69], InceptionResNets [122], Squeeze and Excitation (SE) ResNet [123], etc which have reduced the top-5 error percentage even further. Amongst these Neural Architecture Search Network (NASNet) is interesting research where the CNN architecture was designed by a neural network beating all the previous human-engineered or designed networks. This was done using Google Brain's AutoML, a reinforcement learning approach for the architecture design. Though this requires modest computational capabilities it outperformed other discussed hand-engineered architectures in the ILSVRC challenge. Currently, SE ResNet50 based approaches have achieved a 2.3% top-5 error leading the ILSVRC 2017, which models the interdependencies between channels explicitly such that the network adaptively recalibrates the channel-wise feature responses. Table 8 summarises the architectures discussed.

Network	Feature	#params (approx) (million)	#layers (conv+FC)	Test Error (Top-5)
AlexNet	Deeper network	62	5+2	15.3%
ZFNet	Fixed kernel size	58	5+2	14.8%
VGGNet	Fixed kernel size	138	13+2	6.8%
GoogLeNet	Wider-parallel kernels	6	22	6.7%
ResNet50	Shortcut connections	23	49	3.6%
ResNet101	Shortcut connections	42	100	3.6%
ResNeXT50	ResNet+ cardinality	23	49	3.0%

Table 8[56]- Comparison of various attributes of different architectures (# = number of)

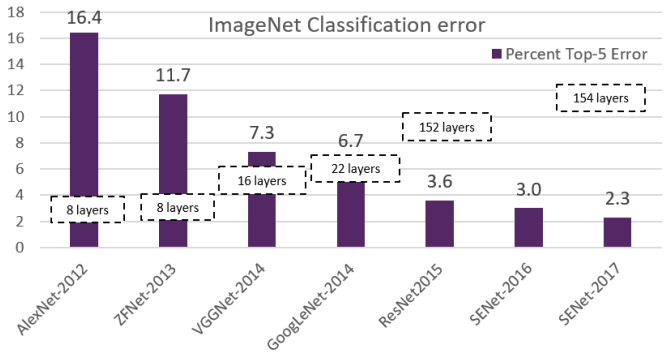


Figure 37- Top-5 percentage error with the number of layers in the network

All the diagrams of the object detectors (single and multi-stage) are provided in the appendix for reference. It also includes the network architecture of feature extractors discussed earlier with the number of conv layers and filters used in the architecture. Readers can refer that to get an overall picture of the detectors and different backbone architectures.

3.8. Platforms/ Frameworks to implement deep learning models

Since these models are made of mainly CNNs they require more hardware computing power, Graphics Processing Unit (GPU) and efficient computing software architectures or packages. The hardware market is mostly dominated by Nvidia with its powerful GPUs with CUDA software libraries. But, there is a number of key players in the computing software packages like Google's Tensorflow, Keras built on top of TensorFlow, Facebook's Pytorch and many more which is seen from Figure 38. The individual framework has its own added advantage and disadvantage. Some frameworks are easy to understand, some have more community support, etc. For example, Pytorch is mainly being used for research purposes since its codebase is pure Python and has more community support while Tensorflow or Keras is used in production which is quite difficult to understand for a fresh researcher in this field.

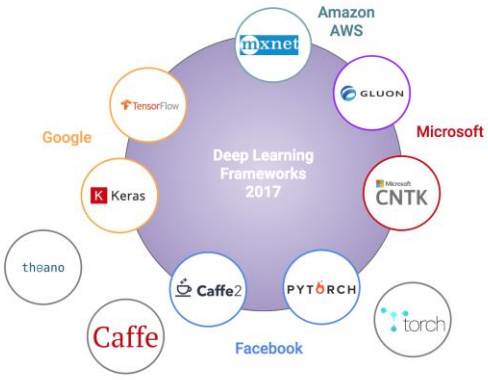


Figure 38 – Open source deep learning framework [128]

3.9. Applications of object detectors

Since these object detectors have significantly improved the performance compared to hand-engineered feature algorithms, either these detectors with or without modification and new CNN based solutions are proposed for applications like pedestrian detection, face detection, text detection, traffic sign-light detection, remote sensing target detection, aerial Imagery, etc. Also, deep learning-based solutions are getting traction in the medical field to detect the early stages of cancer, Alzheimer's disease, Vascular lesions and many more, where CNN based models have surpassed human-level accuracy in detecting such anomalies. Due to its performance, the medical field has become a multi-billion-dollar market for AI technologies.
In this section, the main goal is not to analyze each new domain-specific CNN based solution but rather give an overview of existing solutions to the readers to start with. For more information related to these fields, readers are advised to refer [28], [54], [78] and [113].

Pedestrian Detection: With the success of R-CNN models, traditional methods of detecting the pedestrians such as HOG+SVM, DPM, and Integral Channel Features (ICF) has been replaced with the superior performance of deep learning models. But yet these models had limited success in detecting pedestrians whose areas were less than 32^2 pixels and also due to the low resolution of their features. As a solution, feature fusion, ensemble detection on multiple resolutions are proposed in the literature. Also, to boost the hard-negative detection, integration of boosted decision tree and semantic segmentation are proposed by Zhang et al. and Tian et al. DeepParts proposed by Tian et al make the decision based on an ensemble of extensive part detectors. This has advantages in dealing with partial occlusion, low IoU positive proposals [56]. Based on Faster R-CNN, Liu et al have proposed multi-spectral DNN where complementary information from color and thermal images are combined to detect pedestrians [110].
Specialized datasets are made public to accelerate the research apart from ImageNet, COCO datasets. These include ETH, Daimler DB, TUD-Brussels, Caltech USA, KTTI, GM-ATCI, City Person and EuroCity.

Face Detection: The most famous face detector Viola and Jones trained cascaded classifiers with Haar-like features and AdaBoost which achieved real-time efficiency with good performance but had degraded performance in the real world due to the large variation in the visual representation of human faces. Thus, DPM for face detection was proposed. In order to achieve even a reasonable result, these models required a large number of annotated datasets and high computational expenses. Besides these problems, the features were manually designed and had shallow architecture. Farfade et al [111] have proposed a novel Deep Dense Face Detector (DDFD) which detects faces from multi-view with a wide range of orientations without requiring much pose/landmark annotation. Hung et al have proposed a solution that jointly detects faces and performs landmark localization. This is a unified end-to-end trainable FCN model known as DenseBox [112]. Yang et al [113] proposed ScaleFace, in which the targets are split into much smaller subscale range and different sub-networks are constructed on these sub-scales. These are then combined as one to conduct end-to-end optimization. Specialized datasets available are Face Detection Data Set and Benchmark (FDDB), Annotated Facial Landmarks in the Wild (AFLW), Annotated Face in-the-Wild (AFW), PASCAL Faces, Multi-Attribute Labeled Faces (MALF), Wilder Face, IARPA Janus Benchmark, Un-constrained Face Detection Dataset (UFDD) and Wildest Faces.

In order to detect targets with different orientation (either in text recognition or in aerial imagery), researchers have proposed solutions that are improvements over the existing RoI pooling layer for better rotation invariance. Faster R-CNN and SSD based solutions are used to detect traffic signs or light detection. New techniques such as attention mechanisms and adversarial training are employed to improve detection performance in a complex traffic environment [78]. Logo detection pipelines are constructed using Faster R-CNN. Faster R-CNN is used for commercial purposes in Pinterest which has reported improvements in the user engagement rate in their item-to-item recommendation [65].

4. RELATED WORK

As mentioned in the earlier sections, tracking of parcels in the distribution hubs are usually done using the industry-standard PECs. Its performance depends on the architecture of the sorter, speed of sorting, etc. Solutions towards tracking parcels were developed using vision-based techniques like [114], [115] and [116], with certain limitations and assumptions. Ever since the rise of deep learning, due to its superior performance both in terms of speed and accuracy, research like [117] has proven the success of deep learning-based object detectors in the logistic or parcel distribution hub settings. To the best of our knowledge, this is the only work which is related to our work but with a very different application.

In [114], Karaca and Akınlar employed Lucas-Kanade-Tomasi (LKT) feature tracking algorithm to track the parcels on a conveyor belt to align the parcels in a single line. A parcel's dimensions are computed using 4 stereo cameras which are placed in the entrance of the conveyor belt. A camera is placed above the conveyor belt with a top-down viewing angle. Corner points are further fed to the LKT tracker from the previous frame. Camera calibration is used to project the known 3D points to the 2D image plane. Authors initially used bare LKT trackers to track the corners of the parcels in the scene and concluded that using the corner as a single criterion, tracking would fail in most of the cases and was not sufficient for successful tracking. They proposed a five-step process which involved back projecting of corner points into 2D image plane, feeding the corner points to LKT corner detector to get new coordinates in the current frame, applying feature detection algorithm (Good features to track by Shi-Tomasi) to detect better features based on the eigenvalues, compute new 3D coordinates using refined corner and using edge mapping algorithm to extract edge information from the images. They hypothesize that the center of parcels in x and y directions to be off by certain "mm" which resulted in testing 36 hypotheses for each parcel. They also discuss results obtained by running only the edge detection which still gave some good performance. Figure 39 (a)-(d) depicts their work.

With regard to our work, placing the cameras in a top-down view was not an option but we do believe that with such a top-down viewing angle would require extra undesired infrastructure and also cameras must have been placed in greater height to cover the required area of interest.

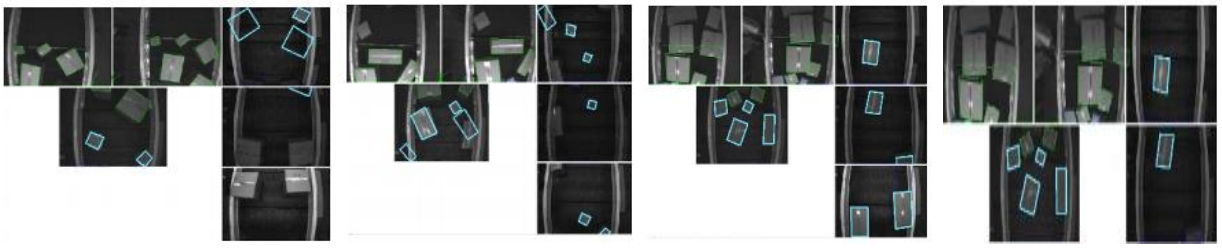
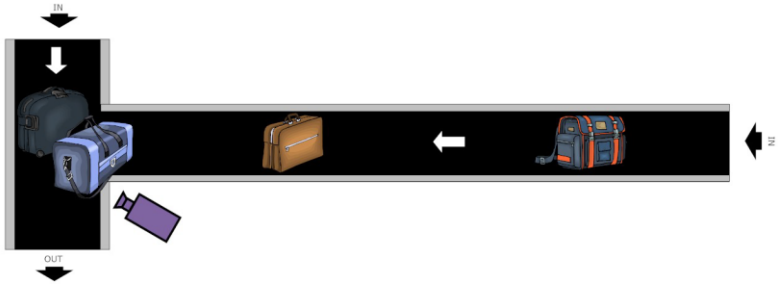


Figure 39 – a) Using LKT alone b) Using feature+LKT c) Using LKT+ EdgeMaaping d)Using Edge Mapping alone [114]

In [115] authors have employed simple background subtraction techniques to detect the jams in Brussels airport's baggage handling systems. The authors have developed a full-fledged website to check the baggage jams in the conveyor system (when two conveyor belts are installed to form a "T" section)and avoid mistracking of baggage shown in Figure 40 (a) and Figure 41 (a).

The latter problem is very common in the airport baggage handling section and the number of PLCs are used in this airport to detect such jams. Authors observe that often reason for such a jam would be unknown and since cameras were placed in such junctions, often it was cumbersome to analyze whole video segments to understand the cause of such jams. Hence, they have implemented a logic where jam videos were created by splitting the hours of surveillance recording into a watchable size to analyze the cause of jams as shown in Figure 40.



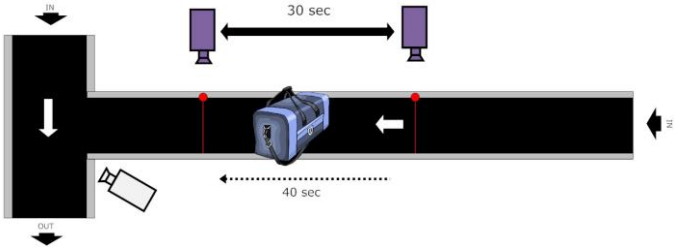
a)

b)

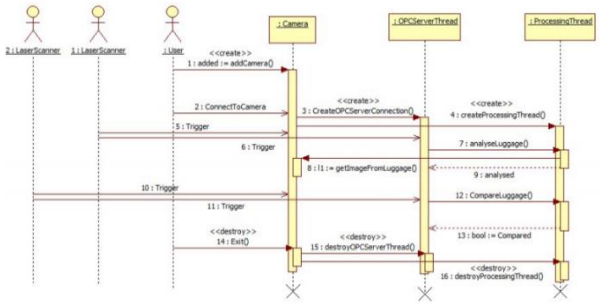
Figure 40 – a) Possible jam scenario b) Possible jam videos split from continuous surveillance videos [115]

The mistracking problem was solved using a rather simple vision-based technique, background subtraction. Tracking of baggage in the conveyor based system was done using laser lines attached to the belts. Baggage was tracked by calculating the time taken by baggage between each laser line. If baggage would not appear on time at a certain laser line, the system was unsure If the baggage being tracked was the same one or new baggage had arrived. The former case is illegal and it needed to be checked manually due to security reasons which would bring down the system capacity due to frequent halts.

Authors used background subtraction, shadow removal techniques like Normalized Cross Compilation (NCC), edge detection algorithms like LOG and blob extraction using OpenCV functions. They have developed a tool where when baggage arrived at one laser line, they applied segmentation, shadow removal, edge detectors and were muxed together to be sure that the frame contained baggage. An extraction algorithm was applied in the later stage and the best match was found by a scoring algorithm. This scoring algorithm calculated the area of blobs and considered the best position of all baggage on the belt (center of the belt) for recognition. A recognition software compared to the images of baggage obtained from camera 1 associated with laser scanner 1 with the camera 2 associated with laser scanner 2.



a) Mistracking



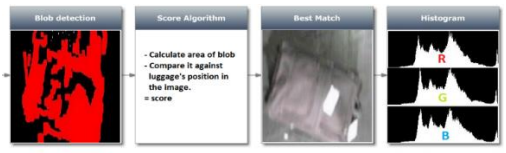


Figure 41 – b) UML diagram of the developed solution **c)Scoring algorithm for matching bags from cam1,2 [116]**

The cameras were placed in a top-down viewing angle. The results that they discuss were far from a test drive development and were not on an active baggage handling system as they mention it due to security reasons in the airport. But according to authors, the developed solution could detect most of the mistracking leading to an improvement in the tracking compared to using standalone laser line trackers. Software like .NET, SQL database, QT and OPC server-client frameworks were used to develop the functioning web application.

In this work, we believe that the cameras installed, had controlled lighting conditions and hence their background subtraction worked well which clearly is not the case in our environment. Also, the authors do not discuss the scenario where the pieces of luggage could be close to each other (few mm away In our case) and using background subtraction could they identify two separate pieces of luggage correctly. In our experiments with background subtraction with a custom viewing angle, we have found it hard to detect two TSUs as two, when they were really close to each other.

In [116] have proposed a multi-object tracker adapted to the conveyor system and have proposed methods to handle occlusions. They exploit the ordering of objects on the conveyor to re-establish the association of an object's identity during occlusion. The authors have followed the on-line tracking-by-detection paradigm, discussed in 2.1.4 (DBT) using background subtraction for object detection. They discuss solutions for complex occlusion management and the detection of stationary objects on the conveyor.

Complex occlusion management was solved by using an ordering scheme different from the traditional appearance scheme where the objects are re-identified using an object's appearance model such as color and shape. This scheme was proposed because the appearance of objects in the logistics scenario has a similar color and shape.

Stationery objects were detected by their novel approach of a feedback loop from tracking to the detection module. This was employed based on the fact that the traditional background subtraction methods tend to absorb the objects which become motionless. The motionless scenario is generally seen in logistic hubs where the objects are stopped for certain periods of time for routing purposes.

Since this is the DBT framework, the detection was performed first in the pipeline followed by tracking. Here, the authors used the background subtraction technique which extracts blobs from the background by comparing the input frame with the static background model. The camera was placed in a top-down viewing angle or bird-view angle. Morphological operations were performed to remove noise and holes present in the extracted blob (foreground mask). Erosion, dilation, and connected component analysis were performed to remove noisy pixels, fill holes and to label the connected pixels respectively. Smaller sized blobs were discarded and features were extracted from the set of selected blobs. Each blob was represented by its center coordinates, bounding box, and color histogram.

The tracking module is split into two blocks, data association block and a group of other modules performing various tasks such as group creation, split handling, updating tracks, detecting new objects to track and a tracking management module. The data association block detected the association between the detected blobs and tracked objects (Figure 42). Based on the overlapping criteria between objects and blobs, complex associations were detected. Objects and blobs which were not associated in the complex association were further matched using the Hungarian algorithm based on their centroid. The tracking management block made series of checks like block creation, suppression, association, recovery of objects after a split or merge of objects into groups based on the detection associations (complex or matching) between the blobs and objects.

They have defined the complex association based on certain observations explained below and can be seen in Figure 42.

- **Merge blob** – This corresponds to a blob with a larger bounding box overlapping simultaneously with the smaller bounding box of several objects.
- **Split object** – This corresponds to an object with a larger bounding box overlapping simultaneously with the smaller bounding box of several blobs.

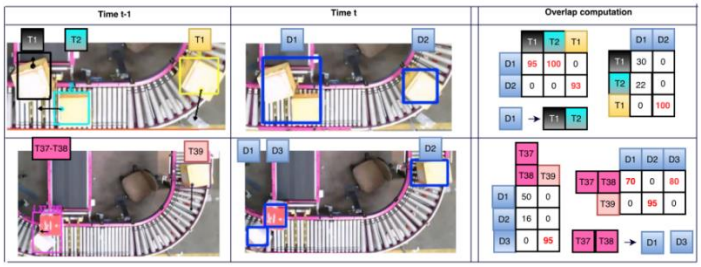


Figure 42 – Hungarian algorithm for merge and split [116]

Those detected objects and blobs that do not involve in the above procedure, blobs are assigned separately to visible objects based on spatial proximity. Using Euclidean distance, they have defined a cost matrix to define the cost of assigning centroids of a blob to an object. They also define a threshold to allow maximum displacement of objects between consecutive frames. A Hungarian algorithm is used to obtain the optimal assignment.

The merge-split approach was proposed by the authors to handle occlusions. During the beginning of occlusion, individual objects were grouped together containing objects and this group is tracked as another object. When the parcels were split (going into another belt), the re-identification of objects from the group was established using the ordering of objects during routing. This ordering relation is derived from the conveyor model. A simple polyline was fit on the pathways of the conveyor to order the conveyed objects**,** which could be either defined manually or during the training phase which was unclear from the paper.

Modified controlled GMM was used noticing the drawbacks of pure blind and conservative update schemes followed in background subtraction methods. In general, the GMM used a learning rate that incorporates pixel samples and it was tuned to zero for the stationary object pixels. They compared this strategy with a conservative update strategy model called ViBe which uses a spatial update scheme to incorporate background

information. They control the parameters of this strategy to avoid absorbing the stationary pixels. The stationary objects were detected using speed estimation by Kalman filter associated with each tracked object compared with a fixed speed threshold. These parameters were adapted via a feed-back loop to update the background models.

In order to measure the improvement from each contribution (GMM and ViBe) different settings were evaluated: enabled and disabled for ViBe and GMM in the tracking pipeline. They have evaluated the performance of the tracker using the standard metrics such as Multi-Object Tracking Accuracy (MOTA), MOT Precision, FP, Missed objects and ID switches. These were tested with the reference to the occlusion handling strategies like the traditional appearance modeling and proposed order modeling. With experiments, they observed that the GMM was more sensitive to the stationary objects and by their feedback loop approach showed significant improvements lowering the missed objects. Using ViBe and with the proposed ordering relation further lowered the missed objects and id switches respectively compared to the appearance modeling method.

The proposed method has been tried out in a controlled setting where the number of parcels in the field of view is really less (5). Also, the camera is placed only to see a portion of the conveyor as seen from the Figures above. It is also not clear from the paper about its real-time performance when the number of parcels In the scene increased.

[117] was the only deep learning-based method to detect and track the parcels in the distribution hubs. The developed solution was to help the human operators to recover the lost parcels in such hubs. The authors argued that traditional methods like background subtraction used for detection and other tracking methods would not be efficient in the parcel hubs due to their heterogeneous environment. Hence, they have used a CNN-based solution, specifically Mask R-CNN, a semantic segmentation-based object detector for accurate localization.

Mask R-CNN is Faster R-CNN with additional segmentation head as seen from Figure 42**.** The reason behind is this choice is unclear in the discussion. As discussed in the above sections, Mask R-CNN uses RoI Align instead of the traditional RoI Pooling proposed in the Faster R-CNN. They also proposed a novel approach for tracking based on [118], a Siamese network, also a CNN based network to measure the similarity between two features vectors to improvise the tracking performance. They have added the feature improver network as a head to the detection pipeline (Figure 43) which could be trained. The authors used pre-trained models trained on the COCO dataset and fine-tuned the Mask R-CNN network with their custom dataset which was annotated manually. Additional training was done for the proposed feature improving network. Also, the performance analysis of the proposed tracker is made with the OpenCV's inbuilt object trackers and their methodology had an upper hand.

4 cameras were positioned in an overlapping and non-overlapping viewing angle fashion. In order to track the parcels within these viewing angles (inter and intra), authors have used the ArUco marker to calibrate the cameras. ArUco markers were placed on each side of a rectangular box, known as calibration parcel, which provided at least one calibration target from each viewing direction. They have estimated the surface of the belts with the piecewise planar segments. The authors argued that the surface knowledge of belts was important to ignore the parcels beside them. As mentioned before, the Mask R-CNN was used as an object detector.

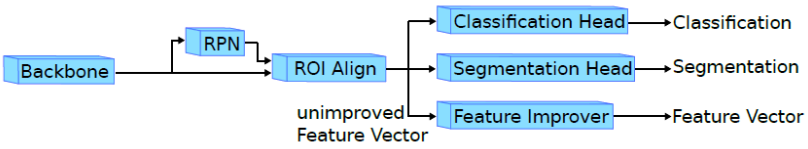


Figure 43 – Modified version of Mask R-CNN by Clausen et al [117]

Inter-camera and intra-camera tracking were performed by the authors, wherein the intra-camera assignment was seen as a weighted bipartite matching problem. Cost matrix was constructed for all the possible matchings of detections from one frame to the next frame. The optical flow algorithm[118] was used to improve the matching. The similarity score was then calculated using the distance between the predicted and detected positions. Occlusion handling was developed to handle missed detections. They have included the unassigned contours as additional matching candidates which allowed them to track parcels even when it was not tracked in some frames.

Since in the inter-camera viewing angle, the parcels could have very different poses, they have used the extrinsic calibration of each camera to project the parcels from one camera to the other. To obtain the information of height or 3D shape of objects, they have used the knowledge of the conveyor belt surface as an intermediate step for projection.

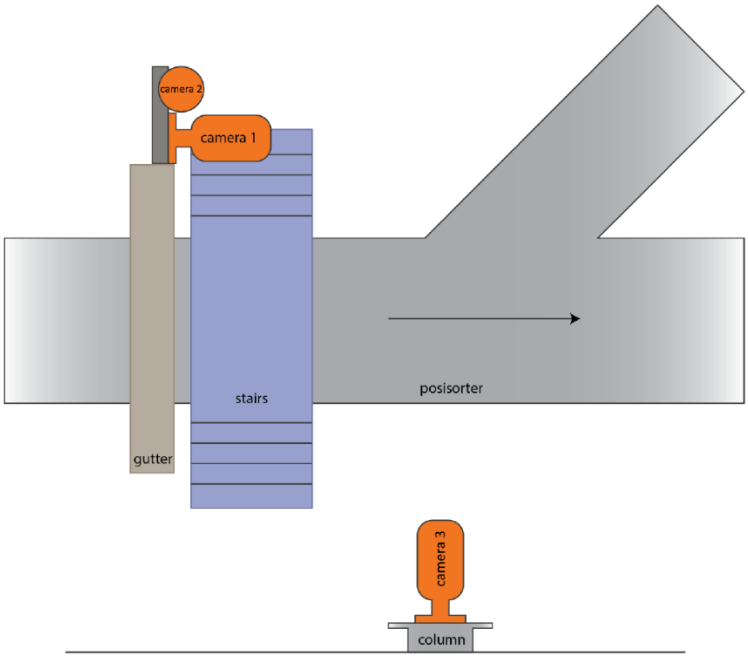
A 4-step process is used to track the parcels when an operator initializes the tracking process. In the first step, corner points from the upward surface were selected located in the opposite direction of the parcel. The bottom part of the corner points of the detected contour was in-plane with the belt surface. In the second step,

estimating the parcel's base, they have fit a parallelogram. As a third step, this parallelogram was projected onto the surface of the belt. And lastly, the quadrangle was projected from the belt surface into the image plane of the other camera. In this way, relevant contours from one camera were projected to the next camera and these contours were matched with the detected ones. Defining the cost matrix and solving using the Hungarian algorithm was done.

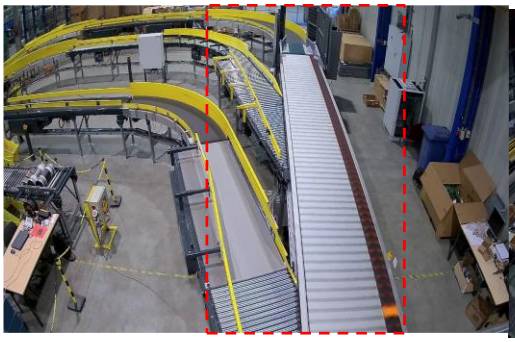
ResNet-101 was used as the backbone (without FPN). The authors proposed their own metrics to analyze the detector's performance, deviating from the traditional COCO matrices. They argue that the AP obtained using the COCO metric would lead to misleading because the AP would reach 1 if the objects were detected with higher confidence than the false positives. This AP score would not be affected by any low-confidence false-positive detections. Hence, they count the true positives, false positives and the ground truth annotation for all validation images and they have calculated one global precision, recall and F1 scores which allowed them to select the best training result from their tracking algorithm. Results from ResNet-50 were compared with the reference ResNet-101 and they have observed that the deep network had a slightly better performance but with an increase in inference time from 150ms to 170ms approximately. The performance of the proposed tracker was analyzed using MOTA scores (a combination of true miss rate and true false-positive rate) with and without optical flow predictions for inter-camera and combined (inter and intra camera). They observed that with the proposed improved feature vector and with optical flow prediction the percentage of tracked parcels showed slight improvements in case of inter-camera tracking while significant improvements were seen in the combined scenario. Mismatches were also less with their solution. Further, they compared their results with the baseline OpenCV trackers Boosting, Kernelized Correlation Filters, MedianFlow, Tracking Learning Detection (TLD), Minimum Output Sum of Squared Error (MOSSE), Channel and Spatial Reliability tracker (CSRT) and GOTURN. They observed that these trackers performed well when target parcels did not change the direction but when they did, only CSRT managed to track it until 5 frames and lost it eventually.

5. CAMERA PLACEMENT

Camera placements and their respective viewing angles can be seen from Figure 44.



a)



b)

c)

d)

Figure 44 a) Schematic of camera positions b) Camera-1 view c) Camera-2 view d) Camera-3 view

All three cameras are high-resolution Bosch security cameras with different technical aspects. Camera 1 and 3 records at 30 FPS while camera 2 with 60 FPS. In the current project, we had no control over either the technical aspects of the positions of the cameras placed. Since it was difficult to annotate the TSUs from camera 3 viewing angle, we had to choose between camera 1 and 2. The ultimate choice of using camera 1 was purely random. But we hypothesize that having trained an object detection model with camera 1 viewing angle, the model would have generalized well to detect TSUs with similar viewing angles (E.g. camera 2).

In our further discussions, all analysis is done on the footages from Camera 1 with certain Region of Interest (marked in red) as seen from Figure 44(b) which will only be processed.

6. DISCUSSIONS

As discussed in the very earlier sections, the main aim of the project is to detect the jams caused in the sorting system. Extracting each TSUs location and further tracking them could potentially solve this problem. Hence, we have employed the famous tracking-by-detection (TBD) paradigm which is followed widely in the literature. Since we are using the TBD framework, this problem can be viewed as the unification of subproblems like object detection problems and tracking problems. While the main focus of the former problem is to efficiently locate TSUs by classification and localization, the latter deals with assigning and maintaining the identity of the detected TSUs. Further sections discuss the various vision-based and deep learning-based object detection techniques followed by the tracking techniques employed in this project.

6.1. Object detection - Computer Vision techniques

Initial experiments carried out within Vanderlande [123] to detect the TSUs using traditional computer vision techniques such as Background Subtraction and contour extraction proved to be inefficient for detecting individual TSUs. This was due to the different lighting conditions, poor background modeling and poor segmentation extraction of TSU contours. This was also because the contour extraction depended on the FG-BG estimation and the area of pixels to be extracted as the required FG varied due to the camera perspective. This can be seen in Figure 44 where the train of TSUs has seen two chunks of blobs.

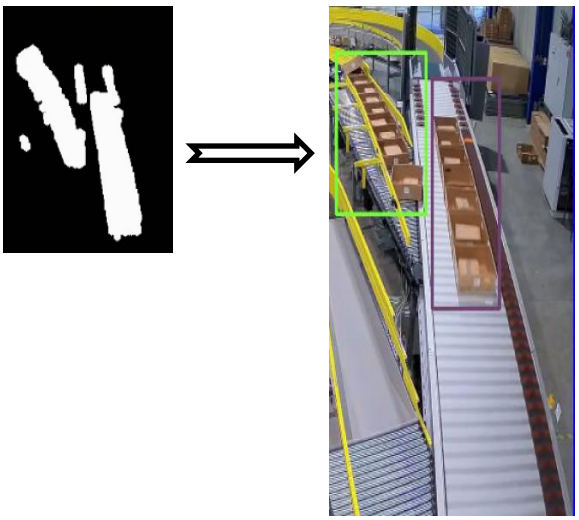


Figure 45– Foreground extraction using Background subtraction [123]

As a next step, the template matching technique experimented with a single template with a single image with 5 TSUs on the sorter. Only three out of 6 methods produced the bounding box (Figure 46, bottom-most figures) with only one method providing only one bounding box instead of 5. This is expected behavior since the template matching technique needs to have a one-to-one correspondence between the templates and the given image to get good performance. Since establishing such correspondence needs more manual effort in designing a database of every possible template in every scenario is cumbersome. The performance also degrades if there is any slight deviation in the new images from the templates in the database. Figure 46 shows the various methods used in template matching provided by OpenCV, (Bradski, 2000). Using different mathematical models, the matching results are obtained which indicates the region of matching with the bright spots in the queried image.

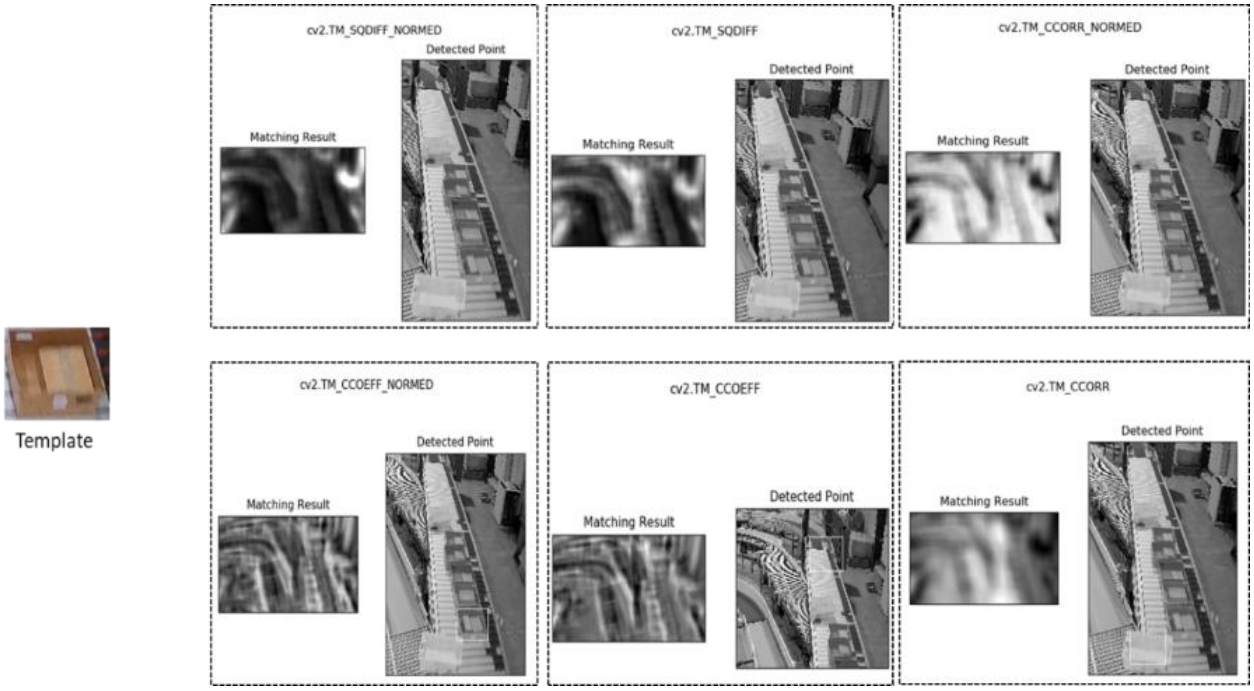


Figure 46 – Template matching using 6 different methods provided in OpenCV

Further, experiments were carried out to detect the TSUs using feature extractors like SIFT, SURF, Harris corner detectors as discussed in section 2.4. Using these features extractors, keypoint and their respective descriptors were obtained from a reference image. Using FLANN based matching technique, these features were matched against the query image which resulted in Figure 47. In Figure 47, the top left corner is the reference image and the image to its right Is the query image. It can be observed that most of the features extracted from the reference image have matched with the different unimportant features in the queried image. Though such outliers could be eliminated using background subtraction, certain features provided by such algorithms were unreliable in this situation where the features are almost similar Hence, establishing a clear difference between two close-by or occluded TSUs needs detailed analysis and research. This is also similar to the template matching technique wherein the careful design of the database of reference images needs to be provided. These techniques also fail to handle shadows, abrupt motions causing blur, illumination changes and are not translation and rotation invariant.

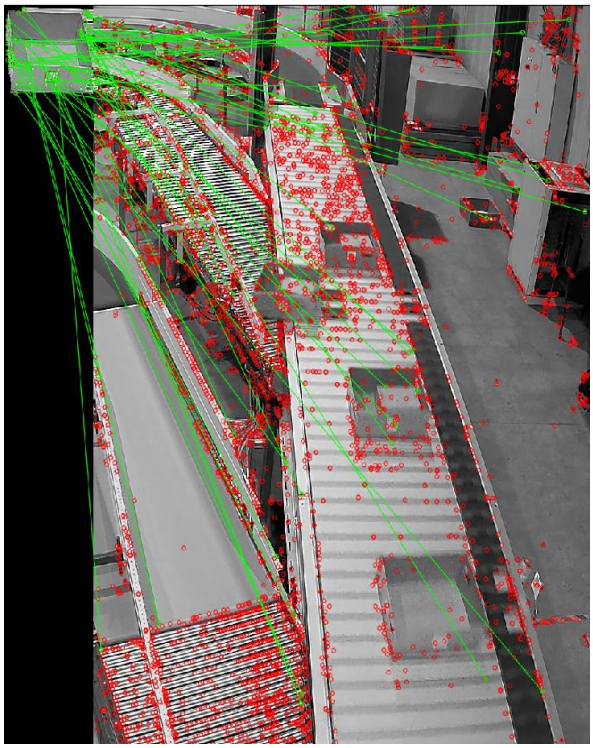


Figure 47- SIFT feature extractor with FLANN matching algorithm

Observing that the HoG with linear SVMs needed large datasets to perform detection, this technique was dropped from further analysis. Moreover, since the features of the TUSs remain almost the same in the scene, we hypothesize that the detector would produce false positives and more overlapping bounding boxes. The former problem could be solved using a good NMS algorithm but might reduce the detection rate, considering two TSUs as one for example. Also, though HoG could handle occlusions and illumination changes to some extent, it is not scale-invariant.

6.2. Deep learning-based object detection- Design choices

- **Choice of the object detector**

It is seen from the sections 3.5.1 and 3.5.2 that there are many object detectors under single-stage and two-stage detectors at our disposal to be used in the project. We already know from the literature that the single-stage detectors are prone to make errors since they struggle to localize the objects under the hood making them fast and ready to be employed in real-time. While two-stage detectors have stronger localization due to the RPN, making them accurate models but are slower than their single-stage cousins, due to the RoI pooling operation which requires large input size. Yet most of the recent object detectors are two-stage detectors based on Faster R-CNN which are easier to optimize. Though the single-stage detectors are quite fast they produce inferior results and need more data and supervised data augmentation techniques to produce better results compared to the two-stage detectors [124]. This hypothesis has also been explored which will be discussed in later sections.

Since the performance of the tracker strictly depends on the performance quality of the object detector, we need our detector to produce high-quality detections. This hypothesis is proved in [125] where the authors have observed an improvement up to 18.9% In the tracking performance by changing the detector. Hence, observing all the advantages, ignoring the real-time requirement, to this end Faster R-CNN is used as the object detector in this project. Figure 48 shows the bottom-up pipeline used in this project. We could also observe from [100] that this model could be made to work in real-time by carefully tuning the hyperparameters and other options which will be discussed in the later sections.

Since there are myriad CNN architectures to construct the Faster R-CNN, the goal is to choose such an architecture and parameters that take lesser training time which in turn means that the model would converge well and produces satisfactory accuracy for the given dataset. Hence further sub-sections deal with the experiments performed to squeeze the performance of the model from the available dataset.

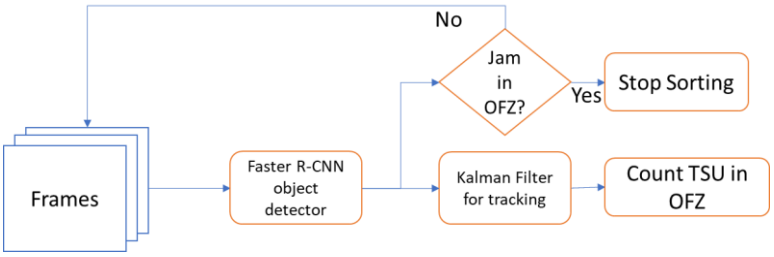


Figure 48 – Jam detection pipeline, a bottom-up approach

- **Choice of the backbone**

As seen from section 3.7, there are many backbone architectures that can be used for extracting the features from the images. In this project, we utilize the widely used ResNet-50 with the FPN (Figure 25). There are two important aspects of this consideration. Firstly, the ResNet-50 network converges well in less time compared to the shallower networks (VGG 16/19)as discussed in section 3.7. Secondly, these are thinner and deep network meaning they have less channel depth and uses lesser GPU FLOPs per convolution calculations. In fact, ResNet-50 with 49 layers deep uses approximately 3.8 Billion FLOPs while the shallower VGG-16/19 uses 15.3 and 19.6 billion FLOPs, respectively. Also, ResNets trains faster as the gradients flow smoothly during backpropagation due to the identity mappings/shortcut connections, Batch normalization layers, in turn, converging in less time with fewer errors. Also, this architecture solves the vanishing and exploding gradient problems with deep network design.

The FPN[67] is used alongside to boost the detection of small objects in the image without any overheads. It is to be noted that the FPN based Faster R-CNN model is faster than the standalone ResNet based model. This is because the FPN model uses two fully connected (*fc*) Multi-Layer-Perceptron (MLP) of 1,024-d each as the network head in the Faster R-CNN pipeline, which is lighter weight and faster. While the standalone ResNet based Faster R-CNN model uses the conv5 layers which are 9 layers deep subnetwork as the head on top of the conv4 feature maps which acts as *fc* layer followed by the class-specific classification and bounding box

regression network (please refer to Figure 8 in the appendix, conv5_x has the dimension of 2,048-d for ResNet-50).

- **Choice of deep learning framework**

Post these design choices, choosing the right deep learning platform (Pytorch, TensorFlow, Caffe, etc.) to implement this also plays an important role. The author in [126] provides a detailed benchmark of accuracies of various CNN architectures implemented in Keras and Pytorch. These comparisons serve as a first-hand guide to the researchers to choose the best platform to start with new projects. According to the author, ResNet models are better implemented in Pytorch and Inception models perform better using Keras. Hence the accuracies of them are better in the respective framework. The author also provides some key insights to work with Keras when using pre-trained models. More information about transfer learning (using pre-trained models) will be discussed in the upcoming sections. Inspired by the observation from [126], the Faster R-CNN implementation in Pytorch is considered in this project open-sourced by Facebook[127].

- **Images used**

Two sets of image resolutions are used (Table 9) in the project. Totally 566 images are used with 464 images as the training set and 102 images are validating set. The dataset also includes the images with different lighting conditions, with jam, without jam, and with background only to have a different distribution in the dataset as seen from the Figures in 48.

(WxH)
498x719
827x1080

a)

Training set	464
Validation set	102

b)

Table 9- a) image resolutions in the dataset b) Dataset split

a) b) c) d)

Figure 49- a) No jam with default lighting condition b) Different lighting condition c) with Jam d) background

- **Number of classes and annotation format**

TSUs form a single class in this project and since we are interested in only bounding boxes, [127] expects it to be in COCO format. Approximately 253 images were annotated manually using [129] and trained the model. Also, as the number of TSUs increased in an image, annotation time per image increased linearly. In order to increase the dataset further, a semi-automated annotation tool was developed on top of [129]. By using this tool 313 more images were annotated with little manual effort comparatively which drastically reduced the annotation timing per image irrespective of the number of TSUs in the image. It took approximately 3 hours in total to annotate the 250 images while with the semi-automated annotations of the remaining 313 were completed (including manual correction if needed and annotating missed TSUs) within approximately 20-30 minutes.

- **Training and Hyperparameter details**

Because we have used pre-trained models for faster convergence with reduced training time, default values are used which are as follows. Unless specified otherwise, these are values are used in this project.

Parameters	Values
Base learning rate	0.0025
Weight Decay	0.0001
Linear warm-up factor	1/3
Optimizer	Stochastic Gradient Descent (SGD)
Momentum	0.9
Images per batch	4
Iterations	2000
Steps	500
(Min, Max) Image Size	(800,1333)
Layers fine-tuned	13 out of 49
Classes	2 (TSU and background)

Hyper- parameters

Table 10 – Default Hyper parameter values

Empirically it has been observed that training deep neural networks with a constant learning rate resulted in problems like over-fitting the model, lesser accurate models and mainly the models would not converge to the global minima. It was also observed that with a lower learning rate, the training was more reliable and stable but with the problems in optimization, which took a lot of time due to the steps towards the minimum loss function were tiny. A contrary to this, with higher learning rates the model could not converge or diverge because of rapid changes in the weights of the network which made the optimizers to overshoot the minima leading to worst loses. This can be seen in Figure 49 for the convex-shaped loss landscape.

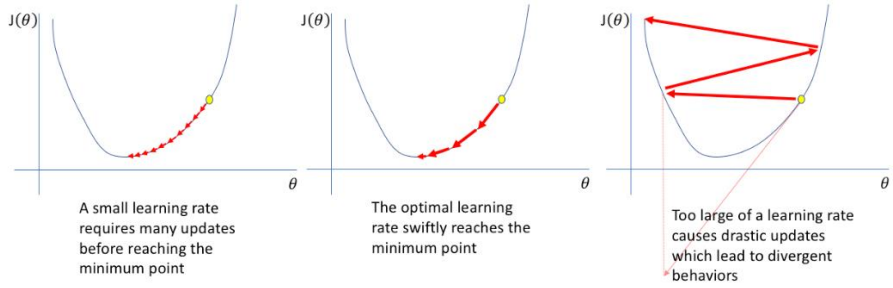


Figure 50- Variation of random cost function J(θ) w.r.t to θ [144]

The linear warm-up factor indicates the variation of base learning rate(LR) to the target learning with respect to the number of *steps* indicated. A warmup iteration of 500 with a base learning rate of 0.0025 with a target learning rate of 0.00025 (weight decay= 0.0001) indicates that the learning rate must linearly increase to 0.0025 until 500 iterations and not instantaneously. Further, the learning rate could still be reduced by indicating the number of steps or iterations after which it has to reduce using the weight decay factor (lr*1/10 after each given *step*). Such a technique is used to avoid the early overfitting I.e., to reduce the training instability in the deeper layers. Also, there are many other techniques like Stochastic Gradient Descent with Warmup restarts (SGDR) and cycling learning rates but discussion of such techniques is outside the scope of these projects. These methods are widely termed as learning rate annealing.

Unlike the 4-step alternating training employed in [66], we have trained the model in an approximate joint training fashion (end-to-end) which is the recent trend employed by the research community. This is because of the ease of training the model and sometimes it has provided similar or greater accuracies than the model trained in the 4-step process.

The shorter edge of the input image is scaled to 800 pixels maximum image input is set to 1333 unless specified otherwise during both testing and training. Each mini-batch involves 4 images on a single GPU with sampling 512 ROIs with 256 anchors per image for training(losses are calculated). During test time only one image is used though the batch inference is possible. 5 scales are used unless specified otherwise both during testing and training as seen in [67]. As there are 4 levels in our ResNet-50 FPN (P_2, P_3, P_4 & P_5) we fix the top 2000 proposals during training and 100 for testing *per level* before applying NMS. Also, post-NMS we retain the top 2000 and 100 *per level*. Finally, from all the FPN levels we sample top 2000 proposals during training and 100 during testing. Shortly we will discuss the ablation studies performed by varying these parameters. An anchor is considered as a positive sample if the minimum IoU with the ground truth (gt) is greater than or equal to 0.7 and is considered negative if its maximum overlap between gt is less than the threshold of 0.3. The rest of the anchors are considered as neural and are discarded from the training meaning losses are not calculated for those anchors. In order for the samples to be not biased with either FG only or BG only anchors,

a hyperparameter is set to 0.5 to balance the FG-BG ratio of drawn samples or anchors per FPN level. Since RoI pooling is attached to the heads of the class-specific classifier, if the IoU threshold of an RoI is greater than or equal to 0.5 it is considered as FG and less than that is considered as BG. COCO evaluation metric is used entirely discussed in the section 3.6.2

- **Hardware used**

Training of the model is done using Microsoft's Linux based Deep Learning Virtual Machine (DLVM) environment available in Microsoft's Azure cloud solution. We have used one GPU, an Nvidia Tesla K80 with 56 GB RAM known as the NC6 instance equipped with the Intel E5-2690v3 processor.

- **Pre-trained models or network initialization**

Because the deep learning models need more data to generalize well, it is very rare to train a model from scratch (initializes the network's weights with random distribution during training) because it is relatively rare to have datasets of sufficient size. Also, initially modern CNN took weeks to train across multiple GPUs on ImageNet until the introduction of batch normalization by Ioffe and Szegedy [130] which was used by Goyal et al., [131] who trained a model using ImageNet in an hour with 256 GPUs with different other learning strategies. Later in the year He et al.,[132] made a deeper comparative analysis on the accuracy and convergence rate while training a model on ImageNet with training from scratch (random initialization) and training by using pre-trained model. They concluded that using such pre-trained models would only speed up the convergence early in the training but would not aid for any accuracy improvements. A revelation observation was also made when the authors trained a model from scratch, i.e. the model generalized well and were robust even when only 10% of the training data was used for deeper and wider networks on multiple tasks and metrics.

Pre-trained models are the models that would be trained on a large dataset like ImageNet. It is empirically proven that such weights are good enough to be used as initial weights for the new tasks based on the similarity of a dataset to the models that were pre-trained on. It is a trend that is seen in the research community to use existing pre-trained models to train on new tasks and fine-tune the newly trained model further. Fine-tuning is referred to as a technique where the last classification layers of a pre-trained model are replaced with the custom classification layer based on the number of classes in the new dataset the model would be trained on. This is necessary because the ImageNet pre-trained models have 81 classes which would be replaced for the custom classes in the new dataset. Also, during fine-tuning only class-specific layers are trained which are the further most layers in CNN because the earlier layers are more generic and have rudimentary information of edges, blobs, color, etc. There is a set of strategies on fine-tuning layers based on the available dataset which is shown in Figure 50.

Hence, pre-trained models trained on ImageNet-1k and fine-tuned on COCO 2017 dataset provided by [133].

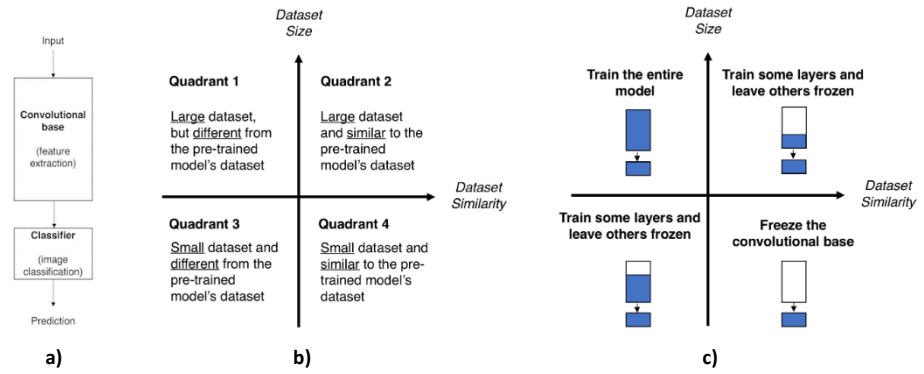


Figure 51- a) simple CNN based image classifier b) Similarity matrix for decision map c)decision map for fine-tuning pre-trained models [145]

- **Batch Normalization (BN)**

Since training deeper networks is complicated due to constant change in the distribution of weights in the various layers during training. As a result, it would take more training time and needs careful network initialization to avoid the gradients to explode. Authors from [130] identify this problem and refer it as an internal covariance shift. By the introduction of BN layers, a deep network gains more stability with added advantages of using higher learning rates, eliminating the need for other regularization techniques like the dropouts. All the recent SOTA deep networks have made this a part of their architecture (ResNet, Inception). The correct placement of the BN layers is still being debated since the authors in [130] have placed them after

the activation unit. It is suggested in recent research that using the BN layers before the activation units has more advantages and is logical. Hence architectures like ResNets place them before the non-linear activations. In simpler terms, this works because BN calculates the mean and standard deviations (std) of the weights and normalizes the inputs of each layer and while training they de-normalizes them using the trainable mean and std. Hence while using the pre-trained models the BN layers are frozen and while fine-tuning with the new smaller dataset the distribution must not tamper much.

Since we are using pre-trained models in our project, the statistics of the BN layers are unchanged in ResNets. This is necessary and, in this implementation, the running mean and variance are already merged into the bias and weight (learnable parameters) variables in the BN formula given below. The reason behind freezing the BN layer's weights and bias variables (mean and variance) is also to aid the training from multiple GPUs because of the unavailability of the library in Pytorch yet, that can accumulate the BN statistics from different GPUs to calculate a global mean and variance. This is because, usually when training a deep network, images are fed into different GPUs in a mini-batch (group) fashion and hence need to calculate the global mean and variance. Figure 51 shows the formulae used to calculate the mean and variance of layers using mini-batch of images (usually order of 2^n mini-batches where n=1,2,m) during training where the parameter γ and β are learned. And during test time, batches of images are not used and only one image is used and hence a rough estimate of values of mean and variance are used. These are termed as running mean and variance or exotically weighted mean and variance values.

Research on the effect of different mini-batches is analyzed by the authors in [131] while training a neural net on the ImageNet dataset and have observed no loss when used a mini-batch size of 8192 on 256 GPUs.

The statistics in the BN layers change only if the network is trained from scratch.

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma^2_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma \widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

Figure 52 – Formula to calculate mean and variance using mini-batches for training

- **Data Augmentation**

Unless specified otherwise, data augmentation techniques like horizontal-flipping, shuffling of the dataset before loading images to the model for either training or testing is performed.

- **Effect of the pre-trained models vs training from scratch**

As discussed earlier, we have used two pre-trained models. One with only the feature extractor (ResNet-50) by [127] and the one with the end-to-end pre-trained Faster R-CNN model provided by [133]. Using pre-trained models, as a baseline model, only the last 13 layers were fine-tuned and earlier layers were frozen and the rest of the layers in the model were initialized with the He initialization [134] rather than random initialization. The initialization is the current trend in deep learning for initializing the networks while using ReLu as the activation functions and it proved to eliminate the vanishing and exploding gradient problems that were seen during training deep networks. This is a replacement to the Xavier initialization which can still be seen in the research but used only when the sigmoid activation function is used in the network.

Pre-trained model	Iteration	Training time(hh:mm)	$AP_{0.5:0.05:0.95}$ (AP)	$AP_{0.5}$	$AP_{0.75}$	AP_s	AP_m	AP_l
Backbone only	2000	1:30	0.64	0.90	0.76	0.41	0.68	0.78
End-to-end	2000	1:30	0.68	0.94	0.82	0.47	0.71	0.80
(from scratch)	2000	1:41	0.58	0.91	0.66	0.43	0.62	0.69

Table 11- Validation accuracy results from two pre-trained models and a model trained from scratch

Clearly, it can be seen from Table 11, that using pre-trained models the accuracy obtained was more for the considered training time and iteration because of the well-initialized weights which served as starting values for the layers being trained on the new dataset. It can be noticed that training the model from scratch produced accuracies closely to that of the accuracy of the pre-trained model of the backbone alone. Yet the former performs well, indicated by the AP due to the better learning rates and initializations. While in the former case, more experiments need to be done for finding the optimal learning rate of the network which is again time-consuming and cumbersome. Nevertheless, more experiments were performed to achieve better accuracies.

- **Effect of Group Normalization (GN)**

Group Normalization[135] is a recent technique where the training is independent of the number of mini-batch of images used during training. One of the other reasons for the introduction of this technique was inspired by the fact that for training a model using large mini-batches needs to have large memory which is also a constraint in modern GPUs. The concept was inspired by the traditional methods like HOG/SIFT where the normalization (having 0 mean and unit variance) is calculated across the channel dimensions of the feature vectors than the batch dimension like in BN. Since the available pre-trained models were of the ResNet-50 FPN (backbone), we have used the same and results are shown in Table 12. Because of the way the GN is implemented it takes slightly more training time due to the additional calculations.

Pre-trained model	Iteration	Training time(hh:mm)	$AP_{0.5:0.05:0.95}$	$AP_{0.5}$	$AP_{0.75}$	AP_s	AP_m	AP_l
Backbone	2000	1:52	0.62	0.88	0.74	0.37	0.67	0.78

Table 12- Accuracy results from a pre-trained model using GN with G=32

- **Training from scratch**

Further, training the model from scratch was continued to another 2000 iterations (4000 iterations in total) with a change in the top pre-NMS proposals set to 8000. No significant improvements were observed though the proposal numbers were increased. This indicates that while training from scratch one must carefully experiment with different learning rates and different step sizes also termed as *babysitting* the model.

Parameters	Values
Iterations	4000
Post NMS proposals	8000
Steps	1000 and 3500

Table a)

Training time(hh:mm)	$AP_{0.5:0.05:0.95}$	$AP_{0.5}$	$AP_{0.75}$	AP_s	AP_m	AP_l
3:23	0.58	0.90	0.66	0.43	0.62	0.69

Table b)

Table 13 a)Changes in the hyper-parameter values for training the model from scratch b) Validation accuracy results of the model trained from scratch

- **Fine-tuning the backbone pre-trained model**

Since the backbone achieved slightly better performance than the model trained from scratch, further experiments to improve the accuracy were performed and the results are shown in Table 14 (b). No further improvements were observed with these values indicating that more tuning has to be performed with the decay and learning rates to arrive at optimized or more generalized values which are again similar to training from scratch.

Parameters	Values
Iterations	3000
Weight Decay	0.00001
Steps	1000 and 2000

Table a)

Training time(hh:mm)	$AP_{0.5:0.05:0.95}$	$AP_{0.5}$	$AP_{0.75}$	AP_s	AP_m	AP_l
2:15	0.66	0.92	0.78	0.46	0.70	0.82

Table b)

Table 14 a)Changes in the hyper-parameters for fine-tuned pre-trained backbone only b) Validation accuracy results

- **Fine-tuning the end-to-end pre-trained model**

Since the end-to-end pre-trained model produced good accuracy initially, fine-tuning it was performed which includes unfreezing the earlier layers, changing the step size, etc. Information is provided in Table 15.

- **Effect of different mini-batch (images per GPU)**

Because the mini-batch size was decreased from 4 to 2, the number of iterations was increased and so was the number of weight decay steps which provided bad accuracy compared to the baseline results from Table 11. This proves that increasing the number of iterations not necessarily increases the accuracy of the model. Also, no significant accuracy improvements can be seen from the baseline model from Table 11 for the end-to-end pre-trained model. Hence, a mini-batch of 4 was fixed in further experiments.

Parameters	Values
Iterations	4000
Mini-batch (*)	2
Mini-batch (#)	4
Steps	1000, 2000 and 3000

Table a)

Training time(hh:mm)	$AP_{0.5:0.05:0.95}$	$AP_{0.5}$	$AP_{0.75}$	AP_s	AP_m	AP_l
(*) 1:40	0.62	0.91	0.74	0.37	0.67	0.77
(#) 3:26	0.69	0.95	0.84	0.51	0.72	0.81

Table b)

Table 15 a)Changes in the hyper-parameters with different number of mini-batches per GPU b) Validation accuracy results for mini-batch=2(*) and 4(#) respectively

- **Unfreezing earlier layers**

The model (#) obtained from the above experiment was further fine-tuned by unfreezing the earlier layers (3) and training those layers in addition to the earlier trained layers. This makes the top layers to train more. Totally 16 last layers were trained including the already trained 13 layers. This saw a good +2 points improvement in the overall AP with the same configuration as the baseline model from Table 11 but with increased training and tuning the weight steps accordingly.

Parameters	Values
Iterations	+4000
Mini-batch	4
Steps	1000, 2000 and 3000

Table a)

Training time(hh:mm)	$AP_{0.5:0.05:0.95}$	$AP_{0.5}$	$AP_{0.75}$	AP_s	AP_m	AP_l
2:30	0.70	0.96	0.83	0.52	0.72	0.82

Table b)

Table 16 a)Changes in the hyper-parameters by unfreezing more layer of ResNet-50 b) Validation accuracy results when training more number of layers than the baseline model

- **Effect of changing proposal number during training**

It can be observed from the above experiments that in order to increase the accuracy of the model using a pre-trained model, one of the ways was to carefully fine-tune the model by unfreezing the models with the different number of steps and more training. Further experiments answers if there is any necessity of unfreezing the earlier layers since they would have learned generic features such as edges, blobs, etc. from the large COCO dataset. The main goal is to get the model converged with fewer iterations with the fixed proposals, implying less training time on a single GPU.

Since the number of proposals from each FPN level was set to 2000 in all the experiments, this experiment focused on the effect of increasing the number of proposals of each FPN level with the default total proposals of 2000 during training and 100 during testing. Also observing that the number of iterations is not a vital parameter to be set, in this experiment and in all the other experiments it

is set to a lower number of 2000 owing for the lesser training time and quick convergence of the model. It can be seen from Table 17 (b) that indeed by just increasing the number of proposals during training we can obtain a model that was trained with 8000 and 12000 proposals to obtain almost the same results (Table 16 (b)).

Parameters	Values
Iterations	2000
Mini-batch	4
Steps	1000
Pre NMS top proposals (train)	4000(*) , 8000 (#), 12000 (^)

Table a)

Training time(hh:mm)	$AP_{0.5:0.05:0.95}$	$AP_{0.5}$	$AP_{0.75}$	AP_s	AP_m	AP_l
(*)1:36	0.69	0.95	0.83	0.50	0.72	0.81
(#)1:42	0.69	0.95	0.83	0.51	0.72	0.81
(^)1:38	0.69	0.95	0.84	0.50	0.72	0.81

Table b)

Table 17 a)Parameters used during training end-to-end pre-trained model with the different number of proposals before applying NMS b) Validation accuracy results with different region proposals used during training the model.

The model # will be considered as the new baseline model for further investigations when specified.

- **Effect on accuracy and inference time changing the resolution during training**

Since we are using FPN, it needs high-resolution images to provide good accuracy. But choosing the right resolution for the model is also important. Table 18 shows the effect of decreasing the model's resolution during training. Now the model would resize the shorter side of an input image to 600 pixels and the maximum size was set to 1000 pixels. Because no significant difference in the accuracy was observed from the above experiment the proposal number was set to 8000 and rest were the default hyperparameters. As expected, there is a dip in the accuracy of 2 points from the above model with the default model size of (800,1333). This was also the observation in [100] wherein reducing the resolution reduced the accuracy on the COCO dataset. Overall accuracy has decreased by 3 points compared to Table 17 (b) values though the training time has decreased.

Parameters	Values
Iterations	2000
Mini-batch	4
Steps	1000
Pre NMS top proposals (train)	8000
Size (min, max)	(600,1000), (600,600)

Table a)

Training time(hh:mm)	$AP_{0.5:0.05:0.95}$	$AP_{0.5}$	$AP_{0.75}$	AP_s	AP_m	AP_l	Inference (s)
1:00	0.68	0.95	0.82	0.48	0.72	0.80	0.208
0:40	0.65	0.92	0.77	0.41	0.70	0.80	0.1227

Table b)

Table 18 a)Hyper-parameter values changed when training the model with the different model sizes or the resolution of images. b) Validation accuracy results for (600,1000) and (600,600) resolution

- **Effect of data augmentation**

Data augmentation is a technique to increase the datasets. In our project, only horizontal flipping of images was used. This experiment explores the effect of data augmentation techniques on accuracy such as vertical flipping and controlled changes in the brightness of the images in the training dataset. Since this is being fine-tuned on model *#,* all hyperparameters follow its values except two values shown in Table 19 (a). Indeed, the model has gained some more accuracy due to the increased variation in the distribution of data points.

Parameters	Values
Iterations	+2000
Vertical flipping	0.5
Brightness	0.5
Steps	1000

Table a)

Training time(hh:mm)	$AP_{0.5:0.05:0.95}$	$AP_{0.5}$	$AP_{0.75}$	AP_s	AP_m	AP_l
1:39	0.69	0.96	0.83	0.51	0.72	0.81

Table b)

Table 19 a)hyper-parameters changed while training the model using different data augmentation technique to boost the performance of it b) Validation accuracy results

- **Effect of using region proposals**

Using the above model as the new base model, the region proposals during test time were changed as given in Table 20. All the above experiments were conducted by fixing the test time region proposals to 100 from each FPN level post-NMS and the top 100 proposals were sampled from overall FPN levels and accuracy was obtained. We already know that increasing the number of proposals during training does not necessarily boost the model's accuracy. This experiment answers the variation of the model's accuracy and inference time with the changes in the proposals during test time with fixed resolution as seen in Table 10. Inference time is defined as the time taken by the model during one forward pass. In this implementation inference time is fully image-to-detections including the proposal generation with NMS which runs on the CPU. We show the average inference time on 102 images in the validation dataset. During test time no data augmentation techniques are used. These timings are reported using a batch size of one inconsistent with [100].

Parameters	Values
Top pre NMS (TpreNMS)	10,30,50,80,100,300,500,1000
Top post NMS (TpostNMS)	10,30,50,80,100,300,500,1000
Total proposals (TPr)	10,30,50,80,100,300,500,1000

Table a)

(TpreNms,TpostNMS, TPr)	$AP_{0.5:0.05:0.95}$	$AP_{0.5}$	$AP_{0.75}$	AP_s	AP_m	AP_l	Inference (s)
(10,10,10)	0.47	0.64	0.56	0.25	0.50	0.68	0.3018
(30,30,30)	0.63	0.87	0.78	0.42	0.68	0.78	0.3019
(50,50,50)	0.68	0.93	0.82	0.48	0.71	0.81	0.3022
(80,80,80)	0.69	0.95	0.82	0.51	0.72	0.81	0.3054
(100,100,100)	0.69	0.96	0.83	0.52	0.72	0.81	0.3063
(300,300,300)	0.70	0.98	0.83	0.55	0.73	0.81	0.3155
(500,500,500)	0.70	0.99	0.84	0.55	0.73	0.81	0.3197
(1000,1000,1000)	0.70	0.98	0.84	0.55	0.73	0.81	0.3132

Table b)

Table 20 a)Changes in the pre and post NMS proposal hyper-parameters values b) Variation of validation accuracy and inference time of the model tested using different proposal values

- **Effect of decreasing resolution during test-time**

As seen from the above Table, no significant inference time variations are observed but the accuracy of the model is indeed highly dependent on the number of quality proposals generated by the model. This experiment mainly focuses on the variation of accuracy and inference time of the model trained on high resolution (800,1333), while fixing the test time resolution to MxM.

Parameters	Values
Top pre-NMS	300
Top post-NMS	300
Total proposals	300
Size (M)	600,800

Table a)

$AP_{0.5:0.05:0.95}$	$AP_{0.5}$	$AP_{0.75}$	AP_s	AP_m	AP_l	Inference (s)
0.45	0.76	0.48	0.11	0.48	0.75	0.1228
0.60	0.94	0.66	0.34	0.63	0.79	0.1907

Table b)

Table 21 a)Changes in the hyper-parameters b) Validation accuracy results

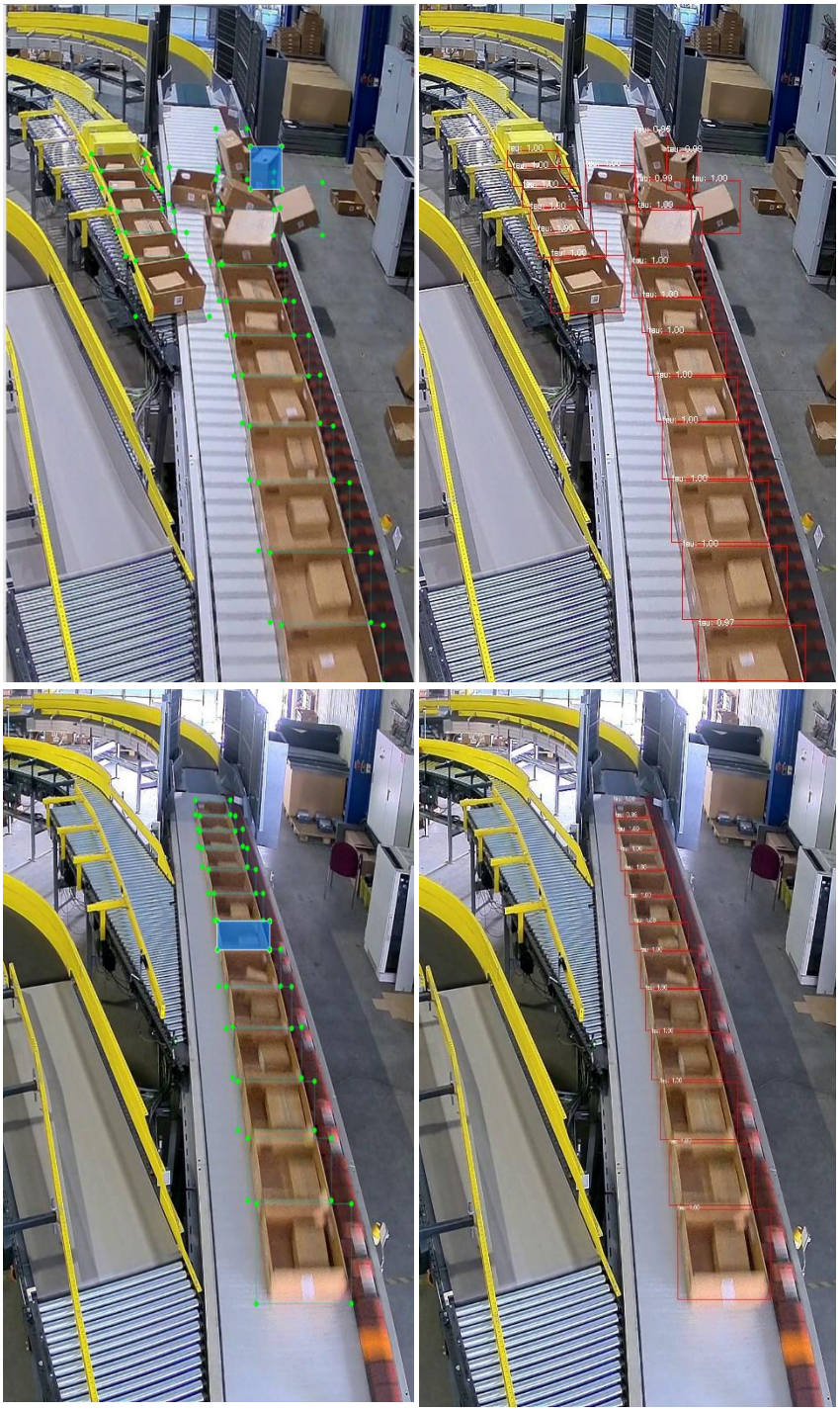
Experiments	Table no.
Validation accuracy from two different pre-trained models and the model trained from scratch	11
Accuracy results using pre-trained models with Group Normalization	12
Accuracy results and hyper-parameter changes to the model trained from scratch with different proposals	13
Summarizes the accuracy results when backbone pre-trained was fine-tuned with different weight decay parameters	14
Summarizes the variation of accuracy when an end-to-end pre-trained model was fined tuned with different mini-batches	15
Results of the validation accuracy of end-to-end pre-trained model when more layers were unfrozen	16
Model's accuracy variation upon changing proposal number during training	17
Accuracy variation when trained on different resolution	18
Changes in the accuracy with training the model using the data augmentation	19
Inference and accuracy variation when region proposals were changed during test time	20
Variation in the model's inference timing and accuracy with different test-time resolution when the model was trained on high resolution	21

Table 22 -Summary Table of the different experiments performed and their corresponding Table to refer to.

- **Results**

Final results from the optimized model are shown in Figures 53 with the confidence score set to 0.9 with a number of proposals set to 500.

a)



b)

Figure 53 – a)& b) Results from the jam and no jam footages with different lighting conditions with ground truth (left) and corresponding prediction (right)

Though our area of focus is the OFZ, we have tried to detect the TSUs in the overflow zone also for future work, in case jams need to be detected in the overflow zone also.

Efforts towards training the SSD model with the same backbone, ResNet 50-FPN was carried out. In the initial experiments, it was observed that with the small dataset used in the project it was difficult to train the model with the chosen hyper-parameters. The model could not converge quickly without any data augmentation techniques in fewer iterations. This also indicates that the model needs more data and carefully chosen data augmentation techniques to converge quickly taking less training time. Such an observation is also made by the authors in [124]. It was also observed that though it was able to detect large objects it almost failed to detect the smaller objects which are in line with the observations from [100]. These experiments were performed using Tensorflow from [100].

As discussed in the earlier sections, the intrinsic problem of the SSD is the class imbalance that RetinaNet solves. Because [127] has released RetinaNet pre-trained models, only one experiment was performed to check its performance with our dataset. Brightness and verticle flipping along with horizontal flipping was used as the

data augmentation technique. Proposals were fixed as default values used In Faster R-CNN. Indeed the model surprisingly generalized well with similar accuracies, inference time and training time and competed closely with Faster R-CNN on larger objects while performing less accurately on the smaller objects. This goes to prove that the models which perform well on smaller objects imply that they perform well on larger objects but the vice-versa is not true [100].

The above experiments could be automated using automation tools such as HyperDrive instead of babysitting our model. These are advanced APIs provided by Microsoft to work with their cloud solution Azure. This is built on searching techniques such as random sampling, grid sampling and Bayesian sampling for hyperparameters searching based on the range of values of the specified hyper-parameters. With this automation, optimization could be made easy, eliminating the need for manually experimenting with a myriad of hyper-parameter values. Based on the sampling technique chosen, the tool would ultimately return the best hyperparameter values of the model, which could take days to arrive at a certain value. This also depends on the amount of dataset being used and the number of iterations being targetted. Some efforts were indeed made towards integrating this library into our pipeline but ultimately failed due to a)advanced API to be used b)lack of expertise in this field. But currently, research (Google's AutoML and Pytorch's HyperSearch)is being done towards such automation techniques to remove the cumbersome manual efforts needed to arrive at an optimized model.

6.3. Tracking

Because the object detection quality is the key aspect for tracking [138,117], the idea is to perform detection is once in N frames and in the remaining frames having the tracker to estimate the positions of the bounding boxes without having any information about the speed of the sorter.

Simple Online Real-time Tracking (SORT) [138] is chosen as the tracker in this project due to its simplicity and ease to integrate into our pipeline. This employs a simple linear Kalman filter to predict the states and corrects them with the detections and associates IDs using the Hungarian Algorithm and IoU matching technique with a single hypothesis.

New IDs would be assigned to the newly detected objects in the $(N+1)^{th}$ frame. If the tracker could not associate itself to any detections in the next frame, new IDs would be assigned again to avoid the waiting for data associated with the corresponding detection in future frames. Since the interest is only to calculate the TSUs in the OFZ, tracking analysis was performed on the TSUs present on the sorter and in the OFZ. Analysis such as associating the IDs post occlusion seen while TSUs gets transferred from the sorter on to the overflow zone was ignored due to more failure detection rate in that area, due to fewer data samples the model was trained on. This implies that with more data, the detection accuracy of the TSUs in the overflow zone could be increased which in turn increases the tracking performance.

Applying detection once in the N^{th} frame failed to produce good tracking results with N=5 because of the wrong estimation heuristics by the filter. Hence detection is performed in every frame assigning the IDs. Results are shown the Figures 52. Counting the number of TSUs entering and exiting OFZ were performed using the simple euclidean distance between the centroids of the TSU and the fixed points (red dots) on the virtual lines (green) drawn at the entrance and at a certain distance In the OFZ. These points vary as the resolution of the video varies. The jam detection is performed based on the detections. IoU between the $(N-1)^{th}$ frame detections and N^{th} frame detections were calculated and certain thresholds were empirically set for two different resolutions given in Table 22. Table 23 provides the context of true positives and false positives. In this project, the videos that were provided had two different frame rates, 30 FPS and 15FPS. The reason is quite unknown but it has worked out to be in our favor due to the fact that in 30FPS, with high-speed sorting, no major displacement of the TSUs was seen from one frame to another. Hence we could implement our jam detection just by employing the IoU matching heuristics. This also implies that we could employ even lower FPS cameras for counting and detecting jams.

Since we need to have better detection in every frame for detecting the jams, we measure the performance of the detector with respect to the number of frames where all the TSUs would be detected. This metric is defined as the *total number of frames the detection fails*. Lesser the number of this metric, the better are the detections. *A total number of TSU/s missed* is a metric that shows the overall count of TSU/s missed during failed detections. Lower the number better the tracking. In order to track the performance of the tracker, two metrics are defined as a) number of ID switches b)number of IDs lost in presence of detection. ID switches indicate the number of times the reported identity associated with TSU/s in the past frame changed its association with a different TSU in the future frame. This necessarily does not have any impact on jam detection but there might be wrong TSU count while it is exiting the OFZ. In our video analysis, there were a couple of times where the exit count of TSU/s in the OFZ mismatched with the actual number of TSU/s exiting it by a very small margin during high capacity. Hence detailed analyses could be made if more videos were available at our disposal. We think that by carefully tuning the Kalman Filter's parameters could solve the ID switching issue.

The metric *Number of times IDs lost with its associated detection* is a metric that shows the capability of the tracker to maintain its association with the detections over the total frames. Lesser the IDs lost, better is the tracker's performance. Again, as mentioned earlier, this metric's results would not pose any problem for the jam detection but might have a negative effect on the TSU counts in the OFZ if this metric has large numbers. False positives count indicate the total number of times, in the total frames, either jam was not detected though there was no jam or the no jam case was indicated as jam. Lower the count, better it is and this depends on the detection rate across the frames. In case if a TSU is jammed and if the model could not detect that TSU In a certain number of consecutive frames, it would harm the system's performance. Hence the detection of TSUs in every frame is of utmost importance. This FP number could be related to the metric *# of TSUs missed* but cannot infer anything about the jams. For example, for the metric *# of TSUs missed* is 9, the corresponding *FP count* is 2 which indicates that the model could detect the TSU causing the jam in almost every frame.

Separate analyses have been made on the effect of reducing the resolution on the above-mentioned metrics in Tables 24 and 25. It is to be noted that though the *# of TSUs missed* and *#frames detection failed* has higher numbers for a 600x600 had the least effect on the Jam detection (*#FP count*). Since the available jam videos were less in number due to the difficulty in creating such jam scenarios in the facility, thorough testing needs to performed and this research would aid to continue this project even further.

Resolution	IoU threshold
498*719 (Jam Videos)	0.8
827*1080 (No Jam Videos)	0.95

Table 22- Different IoU threshold values for jam and no jam videos with different resolution

	Jam	No Jam
Jam	TP	FP
No Jam	FP	TP

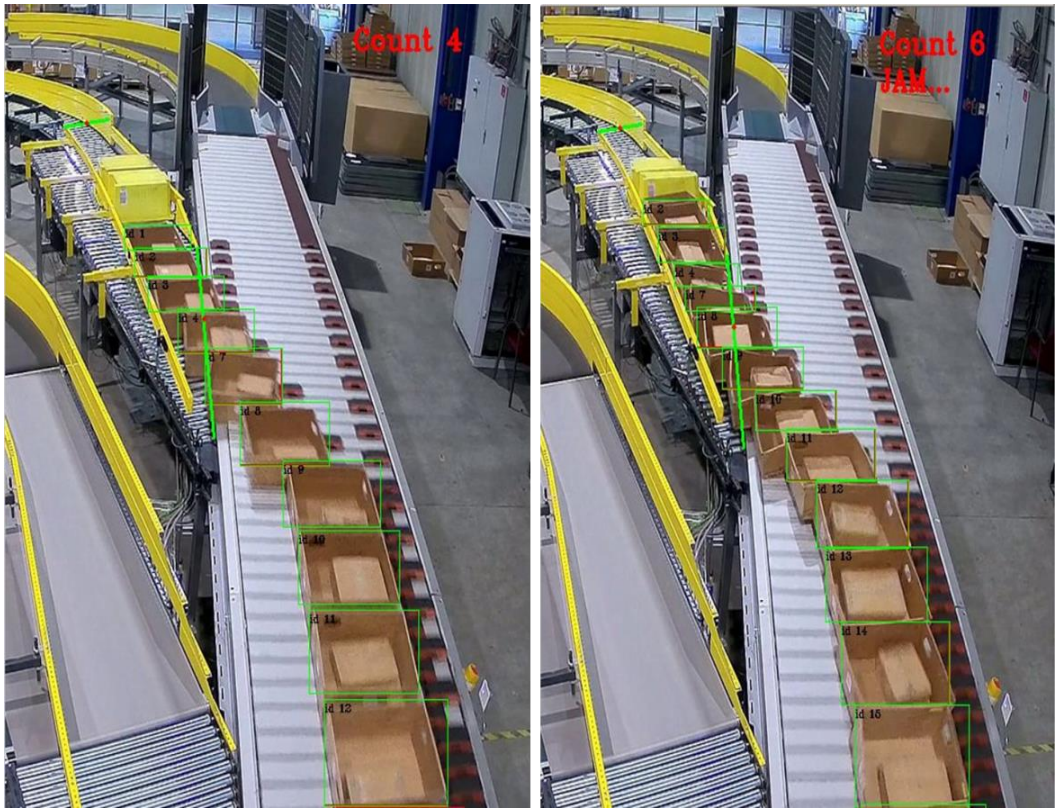
Table 23- Context of True positive (TP) and False Positive (FP) in this project

Video	Total frames	Total TSU/s	# frames detection failed	# TSUs missed	Id switches count	#Id lost with detection	FP count
Jam	110	32	1	2	1	3	1
Jam	135	17	2	3	1	2	2
No Jam	71	16	1	1	1	0	1
No Jam	50	33	1	2	0	1	1

Table 24- Detection and tracker performance with model size of (800,1333) with 500 proposals for jam and no jam videos with different resolutions. Lower the numbers, better the performance. (# = Total number of)

Video	Total frames	Total TSU/s	# frames detection failed	# of TSUs missed	# Id switches	# Id lost with detection	# FP count
Jam	110	32	19	9	2	6	2
Jam	135	17	2	2	1	2	0
No Jam	71	16	36	6	2	3	1
No Jam	50	33	20	5	1	2	1

Table 25- Performance of detector and tracker. The model size of the detector is (600,600) with 500 proposals. Lower the scores, better are the performances of detector and tracker. (# = Total number of)



a) b)

Figure 54- Jam video a)Before the jam, in Nth frame 4 TSUs are counted in the OFZ b) In N+4 frame jam is detected. IDs are still maintained for some time from the Nth frame in the OFZ.

a) b)

Figure 55- No Jam videos a) Tracking in the Nth frame and the corresponding count in the OFZ b) Tracking in the N+1 frame and corresponding count in the OFZ where one TSU (ID=1)has moved away from the exit line of the OFZ. The track IDs are sustained (1,3,5,7).

7. CONCLUSION

Using various methods to increase the detection accuracy of the deep learning-based Faster R-CNN model, we were able to generalize it well including under different lighting conditions, it could detect all the TSUs on the sorter and up to a certain known OFZ length using high-resolution images. Though the TSUs in the overflow zone were detected using the same model, for tracking analysis it was discarded because of too many missed detection during high capacity. Increasing the dataset with more examples from the overflow zone, carefully designing the anchor sizes and Installing one more camera near the overflow could solve the problem.

For tracking Simple Kalman filter to predict and correct the generated trajectories with Hungarian algorithm and IoU matching for data association proved to be effective and performed well with very few ID switches. These ID switches are observed more near the entrance of the TSU. We hypothesize that the co-variances of the Kalman filter during the initial prediction and updating of the measurements might be a cause for such switches. Also, since the SORT algorithm is highly dependent on the object detector, with a 100% detection rate in a given video, it was observed the tracker maintained the correct IDs throughout the video sequences apart from very few ID switches.

Handling the occlusions (transition from sorter to overflow) and data re-association during failed detection of TSUs are recommended for the future work and are not handled in this project here due to the above explanation.

There are effectively only three ways to decrease the inference time a) decreasing the number of layers in the model b)decreasing the image resolution c)Using a high-speed GPU like Nvidia's Titan X (currently used a lot in the research community). Since our problem is detecting a single class, the TSU, using a higher-end GPU would not be cost-effective. Hence we need to make a trade-off between a) and b). We already know that decreasing the resolution is bound to hurt the model's performance from our experiments and hence the only option is to reduce the number of layers. But new theories indicate that using lesser layers reduces the accuracy (ResNet-34 performs inferior to ResNet-50) which can also be seen in the paper [100] and hence we suggest not to reduce the layers but to prune the layers for dead neurons or less activated neurons using various model pruning techniques such as weight pruning, filter pruning, etc. Model pruning and model compression is the current research trend where the researchers are trying to reduce the inference time by removing the unwanted layers resulting in not hurting the model's performance significantly and decreases the model's inference time significantly. This is necessary because models like ResNets cannot be used in mobile or embedded devices. Also, with more ground-truth, one could also use the single stage detectors like the SSDs or Tiny YOLO (very recently Nano-YOLO) which are superior in terms of speed.

Essentially, with this research, the company could ideally track the TSUs (with different load capability) and can theoretically be able to predict the jams. This could be done by tracking the angle of each TSU with some reference point in the image plane because every TSU aligns on the sorter in a straight line and when a TSU is pushed into the OFZ, it does so in certain known angle. Significant deviations from the known angle could lead to a potential jam.

Finally, with this research, we have proven that by using a simple IP RGB camera, we could count the number of TSU irrespective of its load and irrespective of the speed of the sorter. With this, we could a) safely eliminate the dedicated PECs in the OFZ b) a significant reduction in the testing time solely for jam detection c) avoids searching the *sweet spot* for the placing the dedicated PECs in the OFZ.

8. FUTURE WORK:

1. Using a dedicated object detection backbone rather than a classification backbone. One such architecture is Detnet-59 which could be built on top of the existing Pytorch architecture.[2]
2. Use of light heads networks [3]
3. Using weight pruning techniques to increase the inference time and to reduce the model size or memory footprint using different techniques like L1-norm based channel pruning, Network slimming, sparse structure selection, soft filter pruning, and unstructured weight-level pruning.
4. Automation for selecting the best hyperparameter for the model. Hyperdrive (Microsoft Azure), AutoML, Tune, etc. support frameworks like TensorFlow, Pytorch models based on the Random search and Bayesian optimization techniques.
5. Use of network of cameras and calibrating them for data association to gain more precise tracking information using optical flow.[1]
6. Using Siamese style networks (using one-shot learning) for classifying jams.
7. Increasing dataset and labeling by generating them synthetically using 3D models of the parcels.
8. Training a Faster R-CNN model with an additional class (OFZ_TSU) to count the number of TSUs in the OFZ. This would drop the necessity to add trackers in the pipeline.

9. RECENT RESEARCH IN DEEP LEARNING:

1. More research towards back-propagation algorithms like [140] where instead of using the chain rule they have used Hilbert Schmidt Independence Criterion (HSIC) method which has convincing results on the CIFAR-10 dataset to eliminating vanishing and exploding gradient problem.

2. Also, the recent introduction of Adaptive Adam optimizer like [141] has got quite a lot of attention.

3. Facebook's model compression technique where the authors have reduced the ResNet-50 model from 300 MB to a mere 5MB maintaining a similar accuracy from the benchmarks on the ImageNet [142] where CPU could be used to get the inference.

10. REFERENCES:

[1] Yuksel, S.E., Dubroca, T., Hummel, R.E. and Gader, P.D., 2012, May. An automatic detection software for differential reflection spectroscopy. In *Algorithms and Technologies for Multispectral, Hyperspectral, and Ultraspectral Imagery XVIII* (Vol. 8390, p. 83900B). International Society for Optics and Photonics.

[2] Li, Z., Peng, C., Yu, G., Zhang, X., Deng, Y. and Sun, J., 2018. DetNet: A backbone network for object detection. *arXiv preprint arXiv:1804.06215*.

[3] Li, Z., Peng, C., Yu, G., Zhang, X., Deng, Y. and Sun, J., 2017. Light-head r-cnn: In defense of two-stage object detector. *arXiv preprint arXiv:1711.07264*.

[4] Luo, W., Xing, J., Milan, A., Zhang, X., Liu, W., Zhao, X. and Kim, T.K., 2014. Multiple object tracking: A literature review. arXiv preprint arXiv:1409.7618.

[5] Ågren, S., 2017. Object tracking methods and their areas of application: A meta-analysis: A thorough review and summary of commonly used object tracking methods.

[6] Hu, W., Li, W., Zhang, X. and Maybank, S., 2014. Single and multiple object tracking using a multi-feature joint sparse representation. *IEEE transactions on pattern analysis and machine intelligence*, *37*(4), pp.816-833.

[7] Yang, H., Shao, L., Zheng, F., Wang, L. and Song, Z., 2011. Recent advances and trends in visual tracking: A review. Neurocomputing, 74(18), pp.3823-3831**.**

[8] Rout, R.K., 2013. A survey on object detection and tracking algorithms (Doctoral dissertation).

[9] Chau, D.P., Bremond, F. and Thonnat, M., 2013. Object tracking in videos: Approaches and issues. *arXiv preprint arXiv:1304.5212*.

[10] Fan, L., Wang, Z., Cail, B., Tao, C., Zhang, Z., Wang, Y., Li, S., Huang, F., Fu, S. and Zhang, F., 2016, August. A survey on multiple object tracking algorithm. In 2016 IEEE International Conference on Information and Automation (ICIA) (pp. 1855-1862). IEEE.

[11] [76] M. Rodriguez, S. Ali, and T. Kanade, "Tracking in unstructured crowded scenes," in Proc. IEEE Int. Conf. Comput. Vis., 2009, pp. 1389–1396.

[12] H. Izadinia, I. Saleemi, W. Li, and M. Shah, "(MP)2T: Multiple people multiple parts tracker," in Proc. Eur. Conf. Comput. Vis.,2012, pp. 100–114

[13] S. Walk, N. Majer, K. Schindler, and B. Schiele, "New features and insights for pedestrian detection," in Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit., 2010, pp. 1030–1037.

[14] W. Choi, "Near-online multi-target tracking with aggregated local flow descriptor," in Proc. IEEE Int. Conf. Comput. Vis., 2015, pp. 3029–3037.

[15] J. Xing, H. Ai, and S. Lao, "Multi-object tracking through occlusions by local tracklets filtering and global tracklets association with detection responses," in Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit., 2009, pp. 1200–1207

[16] M. D. Breitenstein, F. Reichlin, B. Leibe, E. Koller-Meier, and L. Van Gool, "Robust tracking-by-detection using a detector confidence particle filter," in Proc. IEEE Int. Conf. Comput. Vis. 2009, pp. 1515–1522.

[17] A. Milan, S. Roth, and K. Schindler, "Continuous energy minimization for multitarget tracking," IEEE Trans. Pattern Analysis .Mach. Intel., vol. 36, no. 1, pp. 58–72, Jan. 2014**.**

[18] B. Yang and R. Nevatia, "An online learned CRF model for multitarget tracking," in Proc. IEEE Comput. Soc. Conf. Comput. Vis.Pattern Recognit., 2012, pp. 2034–2041**.**

[19] A. Milan, K. Schindler, and S. Roth, "Detection- and trajectory level exclusion in multiple object tracking," in Proc. IEEE Comput.Soc. Conf. Comput. Vis. Pattern Recognit., 2013, pp. 3682–3689

[20] A. Andriyenko and K. Schindler, "Multi-target tracking by continuous energy minimization," in Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit., 2011, pp. 1265–1272.

[21] Buch, N., Velastin, S.A. and Orwell, J., 2011. A review of computer vision techniques for the analysis of urban traffic. *IEEE Transactions on Intelligent Transportation Systems*, *12*(3), pp.920-939**.**

[22] Zhang, X., Hu, W., Luo, G. and Maybank, S., 2007, November. Kernel-bayesian framework for object tracking. In *Asian Conference on Computer Vision*(pp. 821-831). Springer, Berlin, Heidelberg**.**

[23]Michele Pace - Multi-target Tracking with PHD Filters.[online] Available at https://www.math.ubordeaux.fr/~mpace/PhdFiltering.html [Retrieved July 2019]

[24] J. Zhong, J. Tan, Y. Li, et al,"Multi-Targets Tracking Based On Bipartite Graph Matching," *Cybernetics & Information Technologies*, vol. 14,no. 5, pp.78-87,2014

[25] Jalal, A.S. and Singh, V., 2012. The state-of-the-art in visual object tracking. *Informatica*, *36*(3).

[26] Rout, R.K., 2013. *A survey on object detection and tracking algorithms* (Doctoral dissertation).

[27] Grandham. Sindhuja and Renuka. Devi. A survey on detection and tracking of objects in video sequence. International Journal of Engineering Research and General Science, 3(2), 2015

[28] Agarwal, S., Terrail, J.O.D. and Jurie, F., 2018. Recent advances in object detection in the age of deep convolutional neural networks. *arXiv preprint arXiv:1809.03193*.

[29] S.H Shaikh, K. Saeed, N Chaki, "Chapter-2, Moving Object Detection Approaches, Challenges and Object Tracking", Moving Object Detection using Background Subtraction, springerBriefs in Computer Science, , ISBN 978-3-319-07386-6, 2014

[30] Verma, R., 2017. A review of object detection and tracking methods. *Int. J. Adv. Eng. Res. Dev*, *4*(10), pp.569-578.

[31] Panchal, P., Prajapati, G., Patel, S., Shah, H. and Nasriwala, J., 2015. A review on object detection and tracking methods. International Journal for Research in Emerging Science and Technology, 2(1), pp.7-12.

[32] Parekh, H.S., Thakore, D.G. and Jaliya, U.K., 2014. A survey on object detection and tracking methods. *International Journal of Innovative Research in Computer and Communication Engineering*, *2*(2), pp.2970-2979**.**

[33] P. Bilinski, F. Bremond, and M. Kaaniche. Multiple object tracking with occlusions using HOG descriptors and multi resolution images. In The International Conference on Imaging for Crime Detection and Prevention (ICDP), London, UK, 2009.

[34] K. Robert. Night-Time Traffic surveillance: A robust framework for multi-vehicle detection, classification, and tracking. In The Sixth IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS), September 2009.

[35] Chau, D.P., Brémond, F., Thonnat, M. and Corvée, E., 2011. Robust mobile object tracking based on multiple feature similarity and trajectory filtering. *arXiv preprint arXiv:1106.2695*.

[36] Hai Tao, Harpreet S Sawhney, and Rakesh Kumar. Object tracking with Bayesian estimation of dynamic layer representations. IEEE transactions on pattern analysis and machine intelligence, 24(1):75–89, 2002

[37] Y. Zhou, B. Hu, and J. Zhang. Occlusion detection and Tracking Method based on Bayesian decision theory. In The Lecture Notes in Computer Science, Volume 4319/2006, pages 474–482, 2006

[38] L. Snidaro, I. Visentini, and G. Foresti. Dynamic models for people detection and tracking. In The 8th IEEE International Conference on Advanced Video and Signal- Based Surveillance (AVSS), pages 29–35, September 2008.

[39] E. Monari, J. Maerker, and K. Kroschel. A Robust and Efficient Approach for Human Tracking in Multi-Camera Systems. In The International Conference on Advanced Video and Signal-Based Surveillance (AVSS), 2009.

[40] A. Colombo, J. Orwell, and S. Velastin. Color Constancy Techniques for Re-Recognition of Pedestrians from Multiple Surveillance Cameras. In Workshop on Multicamera and Multi-modal Sensor Fusion Algorithms and Applications (M2SFA2), 2008**.**

[41] D. Comaniciu and P. Meer. Robust analysis of feature spaces: color image segmentation. In The IEEE conference on Computer Vision and Pattern Recognition (CVPR), pages 750–755, 1997

[42] J. Kang, I. Cohen, and G. Medioni. Object Reacquisition Using Invariant Appearance Model. In The International Conference on Pattern Recognition (ICPR), 2004.

[43] S. Torkan and A. Behrad. A New Contour Based Tracking Algorithm Using Improved Greedy Snake. In The 18th Iranian Conference on Electrical Engineering (ICEE), 2010.

[44] N. Martel-Brisson and A. Zaccarin, "Learning and removing cast shadows through a multidistribution approach," IEEE Trans. Pattern Anal. Mach. Intell., vol. 29, no. 7, pp. 1133–1146, Jul. 2007

[45] C. Stauffer andW. E. L. Grimson, "Adaptive background mixture models for real-time tracking," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recog.*, Jun. 1999, vol. 2, pp. 246–252.

[46] R. Cucchiara, M. Piccardi and A. Prati (2003). Detecting moving objects, ghosts, and shadows in video streams. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol.25, no.10, pp.1337-1342.

[47] D. Sugimura, K. M. Kitani, T. Okabe, Y. Sato, and A. Sugimoto, "Using individuality to track individuals: Clustering individual trajectories in crowds using local appearance and frequency trait," in Proc. IEEE Int. Conf. Comput. Vis., 2009, pp. 1467–1474.

[48] D. Mitzel, E. Horbert, A. Ess, and B. Leibe, "Multi-person tracking with sparse detection and continuous segmentation," in Proc. Eur. Conf. Comput. Vis., 2010, pp. 397–410.

[49] L. Zhang, Y. Li, and R. Nevatia, "Global data association for multi-object tracking using network flows," in Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit., 2008, pp. 1–8.

[50] A. Ess, K. Schindler, B. Leibe, L. van Gool, "Improved Multi-Person Tracking with Active Occlusion Handling," Proceedings of the IEEE ICRA Workshop on People Detection and Tracking,2009

[51] D. Acunzo, Y. Zhu, B. Xie, and G. Baratoff, "Context-adaptive approach for vehicle detection under varying lighting conditions," in Proc. IEEE ITSC, Sep. 30–Oct. 3, 2007, pp. 654–660

[52] B. Leibe, N. Cornelis, K. Cornelis, and L. Van Gool, "Dynamic 3-D scene analysis from a moving vehicle," in Proc. IEEE Conf. CVPR, Jun. 2007, pp. 1–8.

[53] R. B. Girshick, From rigid templates to grammars: Object detection with structured models. Citeseer, 2012\

[54] Zou, Z., Shi, Z., Guo, Y. and Ye, J., 2019. Object Detection in 20 Years: A Survey. *arXiv preprint arXiv:1905.05055***.**

[55] Krizhevsky A., Sutskever I., Hinton G. (2012) ImageNet classification with deep convolutional neural networks. In: NIPS, pp. 1097–1105

[56] Liu, L., Ouyang, W., Wang, X., Fieguth, P., Chen, J., Liu, X. and Pietikäinen, M., 2018. Deep learning for generic object detection: A survey. *arXiv preprint arXiv:1809.02165*.
[57] Garcia-Garcia, A., Orts-Escolano, S., Oprea, S., Villena-Martinez, V. and Garcia-Rodriguez, J., 2017. A review on deep learning techniques applied to semantic segmentation. arXiv preprint arXiv:1704.06857.
[58] Zhang, S., Wei, Z., Nie, J., Huang, L., Wang, S. and Li, Z., 2017. A review on human activity recognition using vision-based method. *Journal of healthcare engineering*, *2017*.
[59] Sambit Mahapatra - Why Deep Learning over Traditional Machine Learning [online]. Available at https://towardsdatascience.com/why-deep-learning-is-needed-over-traditional-machine-learning-1b6a99177063 [Retrieved July 2019]
[60] Camilleri, D. and Prescott, T., 2017, July. Analysing the limitations of deep learning for developmental robotics. In Conference on Biomimetic and Biohybrid Systems (pp. 86-94). Springer, Cham**.**
[61] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.Y. and Berg, A.C., 2016, October. Ssd: Single shot multibox detector. In European conference on computer vision (pp. 21-37). Springer, Cham.
[62] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in CVPR, 2016
[63] Lin, T.Y., Goyal, P., Girshick, R., He, K. and Dollár, P., 2017. Focal loss for dense object detection. In Proceedings of the IEEE international conference on computer vision (pp. 2980-2988)
[64] He, K., Zhang, X., Ren, S. and Sun, J., 2015. Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE transactions on pattern analysis and machine intelligence*, *37*(9), pp.1904-1916.
[65] Girshick, R., 2015. Fast r-cnn. In Proceedings of the IEEE international conference on computer vision (pp. 1440-1448).
[66] Ren, S., He, K., Girshick, R. and Sun, J., 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. In Advances in neural information processing systems (pp. 91-99).
[67] Lin, T.Y., Dollár, P., Girshick, R., He, K., Hariharan, B. and Belongie, S., 2017. Feature pyramid networks for object detection. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 2117-2125).
[68] Dai, J., Li, Y., He, K. and Sun, J., 2016. R-fcn: Object detection via region-based fully convolutional networks. In Advances in neural information processing systems (pp. 379-387).
[69] DetNetLi, Z., Peng, C., Yu, G., Zhang, X., Deng, Y. and Sun, J., 2018. Detnet: A backbone network for object detection. arXiv preprint arXiv:1804.06215.
[70] He, K., Gkioxari, G., Dollár, P. and Girshick, R., 2017. Mask r-cnn. In Proceedings of the IEEE international conference on computer vision (pp. 2961-2969).
[71] Ronneberger, O., Fischer, P. and Brox, T., 2015, October. U-net: Convolutional networks for biomedical image segmentation. In International Conference on Medical image computing and computer-assisted intervention (pp. 234-241). Springer, Cham.
[72] Badrinarayanan, V., Kendall, A. and Cipolla, R., 2017. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. IEEE transactions on pattern analysis and machine intelligence, 39(12), pp.2481-2495.
[73] Chen, L.C., Papandreou, G., Kokkinos, I., Murphy, K. and Yuille, A.L., 2017. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. IEEE transactions on pattern analysis and machine intelligence, 40(4), pp.834-848.
[74] Liu, S., Qi, L., Qin, H., Shi, J. and Jia, J., 2018. Path aggregation network for instance segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 8759-8768).
[75] Long, J., Shelhamer, E. and Darrell, T., 2015. Fully convolutional networks for semantic segmentation. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 3431-3440).
[76] Liu, W., Rabinovich, A. and Berg, A.C., 2015. Parsenet: Looking wider to see better. arXiv preprint arXiv:1506.04579.
[77] Noh, H., Hong, S. and Han, B., 2015. Learning deconvolution network for semantic segmentation. In Proceedings of the IEEE international conference on computer vision (pp. 1520-1528).
[78] Zhao, Z.Q., Zheng, P., Xu, S.T. and Wu, X., 2019. Object detection with deep learning: A review. *IEEE transactions on neural networks and learning systems*.
[79] Weng, L. (2019). *Object Detection for Dummies Part 3: R-CNN Family*. [online] Lil'Log. Available at: https://lilianweng.github.io/lil-log/2017/12/31/object-recognition-for-dummies-part-3.html
[80] Girshick, R., Donahue, J., Darrell, T. and Malik, J., 2014. Rich feature hierarchies for accurate object detection and semantic segmentation. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 580-587)
[81] Sik-Ho Tsang- *Review: SPPNet —1st Runner Up (Object Detection), 2nd Runner Up (Image Classification) in ILSVRC 2014*. [online] Available at: https://medium.com/coinmonks/review-sppnet-1st-runner-up-object-detection-2nd-runner-up-image-classification-in-ilsvrc-906da3753679 [Retrieved June 2019]

[82] Ski-Ho Tsang. (2019). *Review: Fast R-CNN (Object Detection)*. [online] Available at: https://medium.com/coinmonks/review-fast-r-cnn-object-detection-a82e172e87ba [Retrieved June 2019]
[83] Fei-Fei Li & Justin Johnson & Serena Yeung, Topic: Lecture 11:"Detection and Segmentation", Stanford University, May 10,2017
[84] KHAZRI, A. - *Faster RCNN Object detection*. [online] Medium. Available at: https://towardsdatascience.com/faster-rcnn-object-detection-f865e5ed7fc4 [Retrieved August 2019]
[85] Jonathan Hui - *Understanding Feature Pyramid Neworks for object detection (FPN) .* [online] Available at: https://medium.com/@jonathan_hui/understanding-feature-pyramid-networks-for-object-detection-fpn-45b227b9106c [Retrieved August 2019]
[86] Li, Z., Peng, C., Yu, G., Zhang, X., Deng, Y. and Sun, J., 2017. Light-head r-cnn: In defense of two-stage object detector. *arXiv preprint arXiv:1711.07264*.
[87] Law, H. and Deng, J., 2018. Cornernet: Detecting objects as paired keypoints. In *Proceedings of the European Conference on Computer Vision (ECCV)* (pp. 734-750).
[88] Szegedy, C., Reed, S., Erhan, D., Anguelov, D. and Ioffe, S., 2014. Scalable, high-quality object detection. arXiv preprint arXiv:1412.1441.
[89] Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R. and LeCun, Y., 2013. Overfeat: Integrated recognition, localization and detection using convolutional networks. arXiv preprint arXiv:1312.6229.
[90] Yoo, D., Park, S., Lee, J.Y., Paek, A.S. and So Kweon, I., 2015. Attentionnet: Aggregating weak directions for accurate object detection. In Proceedings of the IEEE International Conference on Computer Vision (pp. 2659-2667).
[91] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.Y. and Berg, A.C., 2016, October. Ssd: Single shot multibox detector. In European conference on computer vision (pp. 21-37). Springer, Cham.
[92] Najibi, M., Rastegari, M. and Davis, L.S., 2016. G-cnn: an iterative grid based object detector. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 2369-2377). **[93]** Medium- *Evolution of Object Detection and Localization Algorithms* .[online] Available at https://towardsdatascience.com/evolution-of-object-detection-and-localization-algorithms-e241021d8bad [Retrieved June 2019]
[94] J. Redmon and A. Farhadi, "Yolo9000: better, faster, stronger," arXiv:1612.08242, 2016.
[95] Redmon, J. and Farhadi, A., 2018. Yolov3: An incremental improvement. arXiv preprint arXiv:1804.02767.
[96] Weng,.L (2019). *Object Detection Part 4: Fast Detection Models.* [Online] Availabile at: https://lilianweng.github.io/lil-log/2018/12/27/object-detection-part-4.html [Retrieved May 2019]
[97] Law, H. and Deng, J., 2018. Cornernet: Detecting objects as paired keypoints. In *Proceedings of the European Conference on Computer Vision (ECCV)*(pp. 734-750).
[98] Newell, A., Huang, Z., and Deng, J. (2017). Associative embedding: End-to-end learning for joint detection and grouping. In Advances in Neural Information Processing Systems, pages 2274{2284.
[99] Newell, A., Yang, K., and Deng, J. (2016). Stacked hourglass networks for human pose estimation. In European Conference on Computer Vision, pages 483-499. Springer.
[100] Huang, J., Rathod, V., Sun, C., Zhu, M., Korattikara, A., Fathi, A., Fischer, I., Wojna, Z., Song, Y., Guadarrama, S. and Murphy, K., 2017. Speed/accuracy trade-offs for modern convolutional object detectors. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 7310-7311).
[101] Canziani, A., Paszke, A. and Culurciello, E., 2016. An analysis of deep neural network models for practical applications. arXiv preprint arXiv:1605.07678.
[102] LeCun, Y., Bottou, L., Bengio, Y. and Haffner, P., 1998. Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11), pp.2278-2324.
[103] Krizhevsky, A., Sutskever, I. and Hinton, G.E., 2012. Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems (pp. 1097-1105).
[104] Zeiler, M.D. and Fergus, R., 2014, September. Visualizing and understanding convolutional networks. In European conference on computer vision (pp. 818-833). Springer, Cham.
[105] Simonyan, K. and Zisserman, A., 2014. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.
[106] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V. and Rabinovich, A., 2015. Going deeper with convolutions. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1-9).
[107] He, K., Zhang, X., Ren, S. and Sun, J., 2016. Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).
[108] He, K., Zhang, X., Ren, S. and Sun, J., 2016, October. Identity mappings in deep residual networks. In European conference on computer vision (pp. 630-645). Springer, Cham.
[109] Xie, S., Girshick, R., Dollár, P., Tu, Z. and He, K., 2017. Aggregated residual transformations for deep neural networks. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1492-1500).

[110] J. Liu, S. Zhang, S. Wang, and D. N. Metaxas, "Multispectral deep neural networks for pedestrian detection," arXiv:1611.02644, 2016.
[111] S. S. Farfade, M. J. Saberian, and L.-J. Li, "Multi-view face detection using deep convolutional neural networks," in ICMR, 2015
[112] L. Huang, Y. Yang, Y. Deng, and Y. Yu, "Densebox: Unifying landmark localization with end to end object detection," arXiv:1509.04874, 2015
[113] Lundervold, A.S. and Lundervold, A., 2019. An overview of deep learning in medical imaging focusing on MRI. Zeitschrift für Medizinische Physik, 29(2), pp.102-127.
[114] Karaca, H.N. and Akınlar, C., 2005, October. A multi-camera vision system for real-time tracking of parcels moving on a conveyor belt. In International Symposium on Computer and Information Sciences (pp. 708-717). Springer, Berlin, Heidelberg.
[115] Boudewijn, C. and Bart, C. (2019). Computer Vision at the luggage handling system of Brussels airport. Post Graduate.
[116] Benamara, A., Miguet, S. and Scuturici, M., 2016, December. Real-time Multi-Object Tracking with Occlusion and Stationary Objects Handling for Conveying Systems. In International Symposium on Visual Computing (pp. 136-145). Springer, Cham.
[117] Clausen, S., Zelenka, C., Schwede, T. and Koch, R., 2018, October. Parcel Tracking by Detection in Large Camera Networks. In German Conference on Pattern Recognition (pp. 89-104). Springer, Cham.
[118] Leal-Taixe, L., Canton-Ferrer, C., Schindler, K.: Learning by tracking: Siamese CNN for robust target association. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition Workshops. pp. 418{425 (2016). https://doi.org/10.1109/CVPRW.2016.59
[119] Zoph, B., Vasudevan, V., Shlens, J. and Le, Q.V., 2018. Learning transferable architectures for scalable image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 8697-8710).
[120] Huang, G., Liu, Z., Van Der Maaten, L. and Weinberger, K.Q., 2017. Densely connected convolutional networks. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 4700-4708).
[121] Szegedy, C., Ioffe, S., Vanhoucke, V. and Alemi, A.A., 2017, February. Inception-v4, inception-resnet and the impact of residual connections on learning. In Thirty-First AAAI Conference on Artificial Intelligence.
[122] Hu, J., Shen, L. and Sun, G., 2018. Squeeze-and-excitation networks. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 7132-7141).
[123] Kevin Schäfer, "Stauerkennung von Behältern auf einem Hochgeschwindigkeits-Gleitschuhsortierer durch den Einsatz von Computer Vision", Bachelor Thesis, Vanderlande [Retrieved August].
[124] Cheng, B., Wei, Y., Shi, H., Feris, R., Xiong, J. and Huang, T., 2018. Revisiting rcnn: On awakening the classification power of faster rcnn. In Proceedings of the European Conference on Computer Vision (ECCV) (pp. 453-468).
[125] Bewley, A., Ge, Z., Ott, L., Ramos, F. and Upcroft, B., 2016, September. Simple online and realtime tracking. In 2016 IEEE International Conference on Image Processing (ICIP) (pp. 3464-3468). IEEE.
[126] Northcutt, C. (2019). Towards Reproducibility: Benchmarking Keras and PyTorch. [online] L7.curtisnorthcutt.com. Available at: https://l7.curtisnorthcutt.com/towards-reproducibility-benchmarking-keras-pytorch [Retrieved 1 June. 2019].
[127] Massa, F., and Girshick. R., 2018. *massa2018mrcnn.* s.l.:https://github.com/facebookresearch/maskrcnn-benchmark.[Retrieved 1 June 2019]
[128] Indra, B (19 December 2017). Battle of the Deep learning frameworks part I, *Medium.com*
Retrieved 5 June 2019, from https://towardsdatascience.com/battle-of-the-deep-learning-frameworks-part-i-cff0e3841750
[129] Tzutalin. LabelImg. https://github.com/tzutalin/labelImg [Retrieved 10 April 2019]
[130] Ioffe, S. and Szegedy, C., 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167.
[131] Goyal, P., Dollár, P., Girshick, R., Noordhuis, P., Wesolowski, L., Kyrola, A., Tulloch, A., Jia, Y. and He, K., 2017. Accurate, large minibatch sgd: Training imagenet in 1 hour. arXiv preprint arXiv:1706.02677.
[132] He, K., Girshick, R. and Dollár, P., 2018. Rethinking imagenet pre-training. arXiv preprint arXiv:1811.08883.
[133]Girshick R., Radosavovic I., Gkioxari G., Doll P ., and He K. Detectron,
https://github.com/facebookresearch/detectron [Retrieved 1 June 2019]
[134] He, K., Zhang, X., Ren, S. and Sun, J., 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In Proceedings of the IEEE international conference on computer vision (pp. 1026-1034).
[135] Wu, Y. and He, K., 2018. Group normalization. In Proceedings of the European Conference on Computer Vision (ECCV) (pp. 3-19).
[138] Wojke, N., Bewley, A., and Paulus, D., 2017, September. Simple online and realtime tracking with a deep association metric. In 2017 IEEE International Conference on Image Processing (ICIP) (pp. 3645-3649). IEEE.

[140] Synced (14 August 2019), Does Deep Learning still need backpropagation. *Medium.com.* Retrieved 5 July 2019, from https://medium.com/syncedreview/does-deep-learning-still-need-backpropagation-fb32bf636a80
[141]] Less Wright (15 August 2019), New State of the Art AI Optimizer: Rectified Adam (RAdam). Improve your AI accuracy instantly versus Adam, and why it works. *Medium.com.* Retrieved 15 August 2019, from https://medium.com/@lessw/new-state-of-the-art-ai-optimizer-rectified-adam-radam-5d854730807b
[142] Stock, P., Joulin, A., Gribonval, R., Graham, B. and Jégou, H., 2019. And the Bit Goes Down: Revisiting the Quantization of Neural Networks. arXiv preprint arXiv:1907.05686.
[143] Wu, X., Sahoo, D., and Hoi, S.C., 2019. Recent Advances in Deep Learning for Object Detection. arXiv preprint arXiv:1908.03673.
[144] Jeremy Jordan (1 March 2018), Setting the learning rate of your neural network. *Jeremyjordan.me*. Retrieved 10 July 2019, from https://www.jeremyjordan.me/nn-learning-rate/.
[145] Pedro Marcelino (28 October 2018), Transfer learning from pre-trained models. *Towardsdatascience.com* Retrieved 10 July 2019 from https://towardsdatascience.com/transfer-learning-from-pre-trained-models-f2393f124751

11. APPENDIX

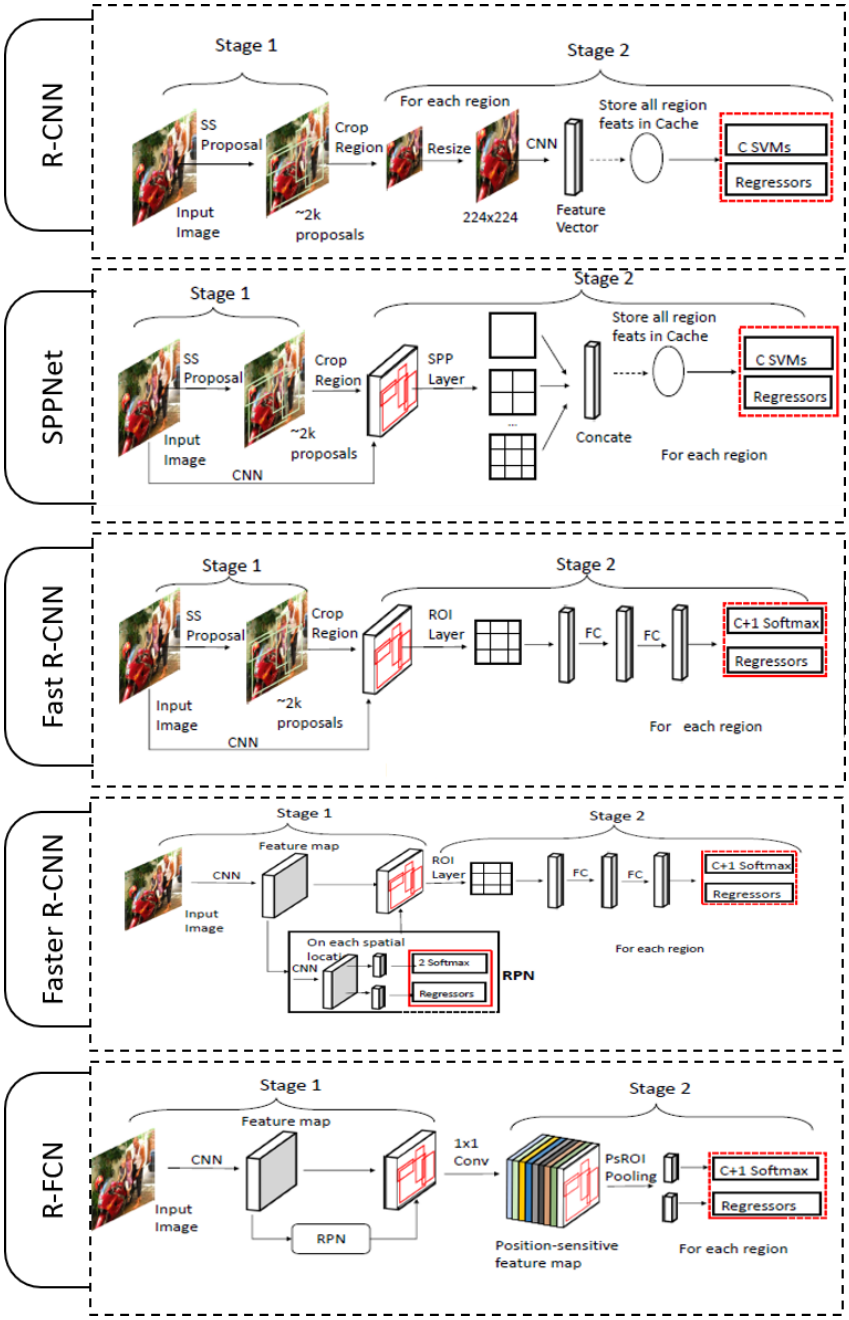


Figure 1 [143]- Two Stage Detectors

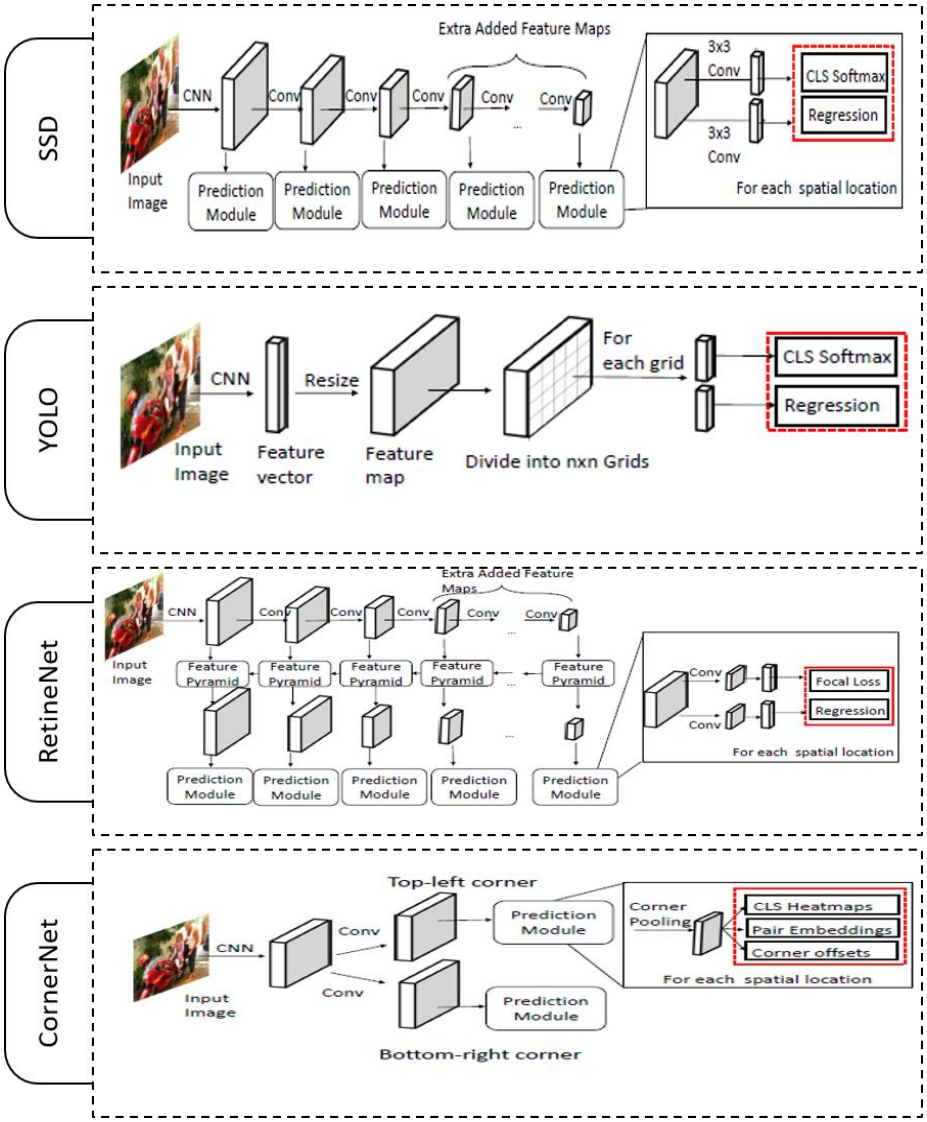


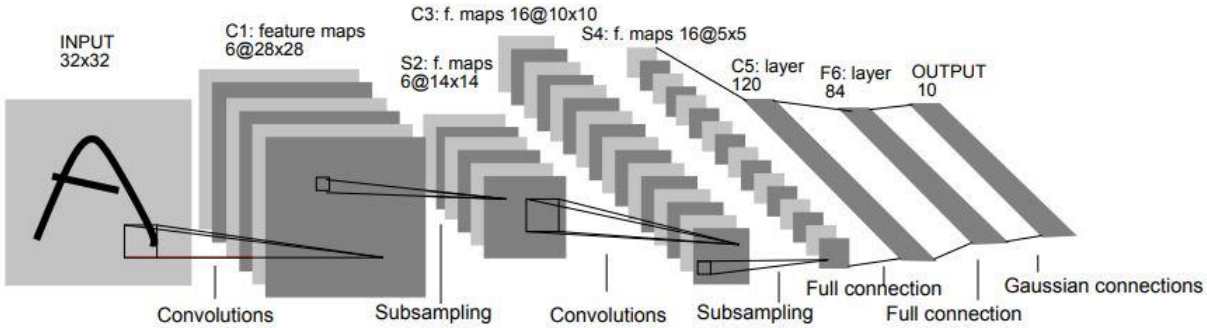
Figure 2 [143]- Single Stage Detector

Techniques	Advantages	Disadvantages
Background Subtraction	• Widely used and computation efficient • Low memory footprint • The model learns about BG-FG based on simple Otsu thresholding • Objects can become part of the background model without destroying the existing background model	• Inaccurate since they cannot deal with quickly changing scenes. • Initializing Gaussian parameters is important • Cannot handle shadows • False positives are more due to the non-availability of shadow handling techniques • Sensitive to illumination changes • Cannot handle multimodal background • Needs Intelligent selection of parameters.
Optical Flow	• Complete information about the motion of objects can be known • Subpixel accuracy	• Computation bottleneck • Sensitive to illumination such as flickering of light • Tends to be noisier around corners • Relies on strong assumptions, brightness constancy

Frame Differencing	• Works well with static background models • Easy to use and computation friendly varying from moderate to low depending on the application	• Needs a background with no objects in the scene to model the background • Sensitive to thresholding values.
Template Matching	• Good performance in a known static environment where objects do not change their appearance much.	• Cannot handle dynamic features in the scene • Brute force method • It can be employed only for one-to-one correspondence. • Cannot handle occlusions • Cannot handle objects which exit from the scene temporarily
Feature-based (HOG, SIFT, etc)	**HOG**	
	• It has a fixed-length feature vector enabling the features to be used by machine learning models. • Suitable for low-resolution images	• Not scale and rotation invariant • The sliding window technique is used for object detection which is very slow and inaccurate.
	SIFT or SURF	
	• Robust to occlusion • Distinct features • Number of features generated for small objects are huge • Rotation and scale-invariant	• Robust to affine transform to some extent • Both are highly sensitive to illumination changes • Do not work well with the motion blur
	LBP	
	• Robust to occlusion • Robust to illumination changes.	• High false positives • Generally less accurate than HOG

Table 1- Advantages and disadvantages of various vision based object detection techniques

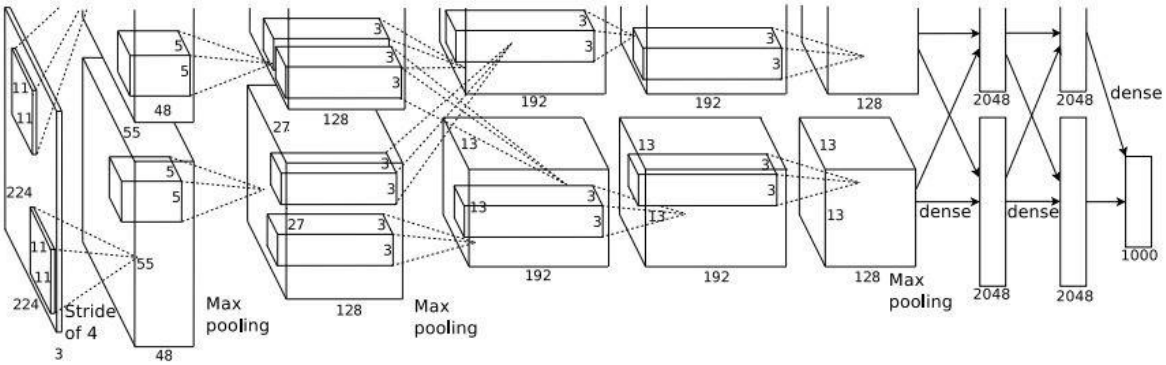
LeNet:



Layer	Feature Map	Size	Kernel Size	Stride
Image	1	32x32	-	-
Conv	6	28x28	5x5	1
Average Pooling	6	14x14	2x2	2
Conv	16	10x10	5x5	1
Average Pooling	16	5x5	2x2	2
Conv	120	1x1	5x5	1
FC	-	84	-	-
FC (output)	-	10	-	-

Figure 3 [102] – LeNet architecture

AlexNet:



Layer	Feature Map	Size	Kernel Size	Stride
Image	1	55x55x96		-
Conv	96	27x27x96	-	4
Max Pooling	96	27x27x256	11x11	2
Conv	256	13x13x256	3x3	1
Max Pooling	256	13x13x256	5x5	2
Conv	384	13x13x384	3x3	1
Conv	384	13x13x384	3x3	1
Conv	25	13x13x256	3x3	1
Max Pooling	256	6x6x256	3x3	2
FC	-	9216	-	
FC	-	4096	-	
FC	-	4096	-	
FC	-	1000	-	

Figure 4 [103]- AlexNet architecture

ZFNet:

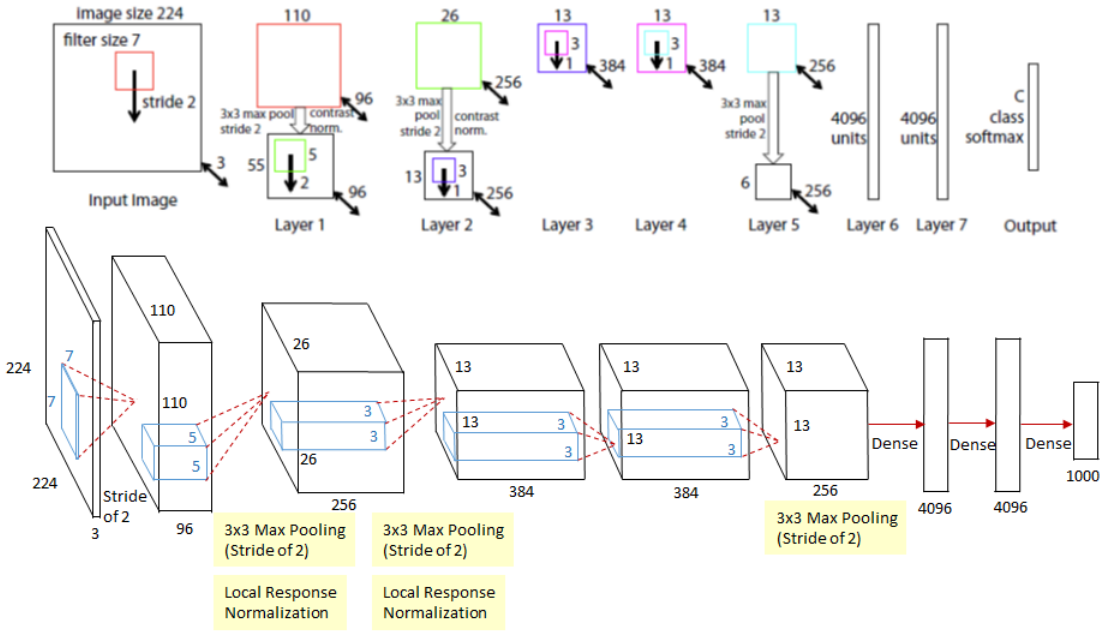
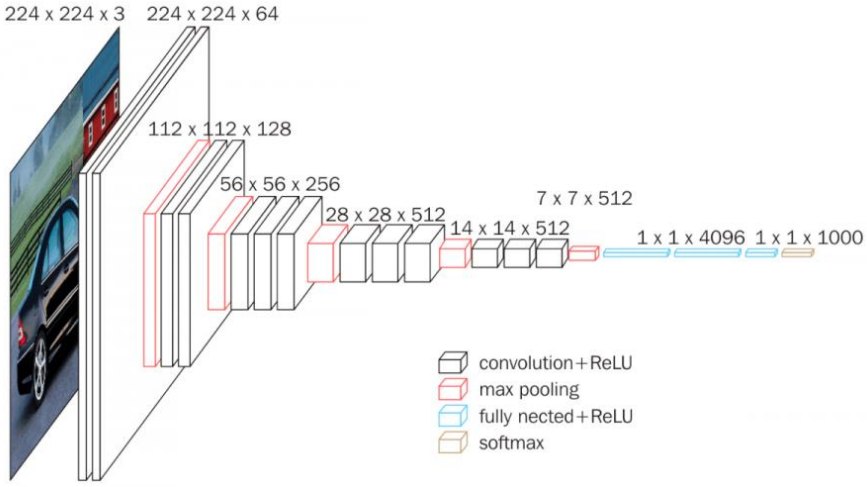


Figure 5 [104]- ZFNet architecture

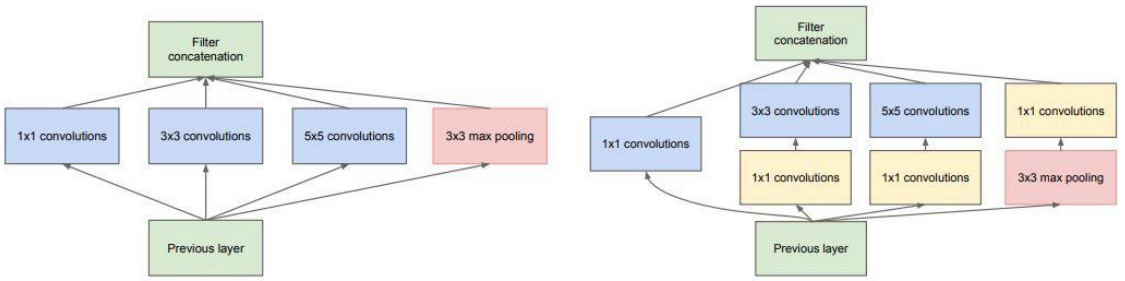
VGGNet:

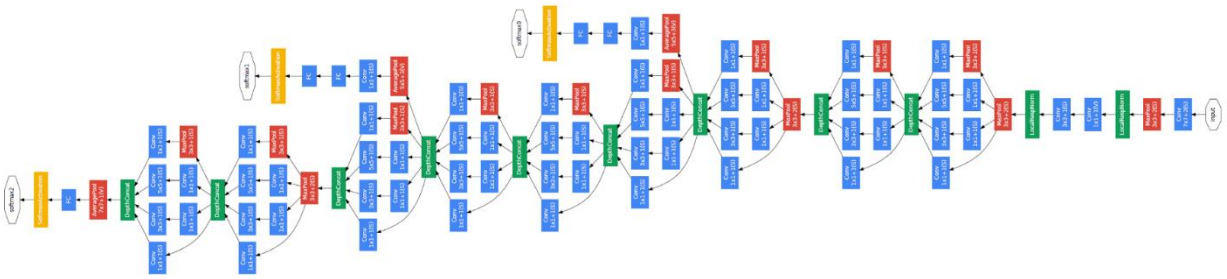


ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Figure 6 [105] VGGNet architecture

GoogLeNet:





type	patch size/ stride	output size	depth	#1×1	#3×3 reduce	#3×3	#5×5 reduce	#5×5	pool proj	params	ops
convolution	7×7/2	112×112×64	1							2.7K	34M
max pool	3×3/2	56×56×64	0								
convolution	3×3/1	56×56×192	2		64	192				112K	360M
max pool	3×3/2	28×28×192	0								
inception (3a)		28×28×256	2	64	96	128	16	32	32	159K	128M
inception (3b)		28×28×480	2	128	128	192	32	96	64	380K	304M
max pool	3×3/2	14×14×480	0								
inception (4a)		14×14×512	2	192	96	208	16	48	64	364K	73M
inception (4b)		14×14×512	2	160	112	224	24	64	64	437K	88M
inception (4c)		14×14×512	2	128	128	256	24	64	64	463K	100M
inception (4d)		14×14×528	2	112	144	288	32	64	64	580K	119M
inception (4e)		14×14×832	2	256	160	320	32	128	128	840K	170M
max pool	3×3/2	7×7×832	0								
inception (5a)		7×7×832	2	256	160	320	32	128	128	1072K	54M
inception (5b)		7×7×1024	2	384	192	384	48	128	128	1388K	71M
avg pool	7×7/1	1×1×1024	0								
dropout (40%)		1×1×1024	0								
linear		1×1×1000	1							1000K	1M
softmax		1×1×1000	0								

c)

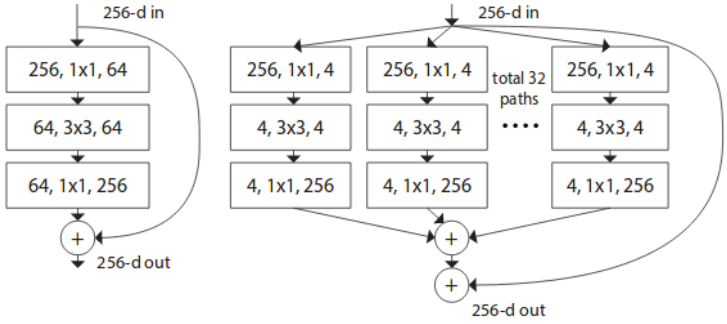
Figure 7[106]- a) Naïve version of inception module, b) Inception module with dimensionality reduction, c) GoogLeNet architecture

ResNet:

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Figure 8[107] – Different architectures of ResNet for classification

ResNeXt:



stage	output	ResNet-50	**ResNeXt-50 (32×4d)**
conv1	112×112	7×7, 64, stride 2	7×7, 64, stride 2
conv2	56×56	3×3 max pool, stride 2	3×3 max pool, stride 2
		[1×1, 64 3×3, 64 1×1, 256] ×3	[1×1, 128 3×3, 128, C=32 1×1, 256] ×3
conv3	28×28	[1×1, 128 3×3, 128 1×1, 512] ×4	[1×1, 256 3×3, 256, C=32 1×1, 512] ×4
conv4	14×14	[1×1, 256 3×3, 256 1×1, 1024] ×6	[1×1, 512 3×3, 512, C=32 1×1, 1024] ×6
conv5	7×7	[1×1, 512 3×3, 512 1×1, 2048] ×3	[1×1, 1024 3×3, 1024, C=32 1×1, 2048] ×3
	1×1	global average pool 1000-d fc, softmax	global average pool 1000-d fc, softmax
# params.		**25.5**×10^6	**25.0**×10^6
FLOPs		**4.1**×10^9	**4.2**×10^9

Figure 9 [109] – ResNeXt architecture compared with ResNet50 with cardinality C=32. Numbers outside the bracket (bottom image) are the layers stacked one above the other