Elbit Systems

Abraham Jose
Advisor: Dr. Abhijit Mahalanobis
Center for Research in Computer Vision, UCF

1.Abstract

An end to end system was built aided by the motion cues to reduce the error in the detection and tracking of moving target objects of interest in a multi-scene dataset from Elbit systems, from 5 different multiple scenes. The primary outcome of the pipeline is to reduce the false positives from the detector and to refine the detector output. Various experiments were carried out using different super class setups and introducing various pipelines to the systems that reduces the false positives from a given detector. The pipeline included a detector, consolidation module, a motion cue based filter and a tracker. The pipelines were developed giving importance to modular data-flow pipelines which allows the modular deployment. The entire system was able to reduce the detections by a factor of 50%. It was generally able to use the motion cues to achieve the results. The detector used for the testing purpose is a resnet-50 backbone fasterRCNN network. A consolidation based on IoU constraints were used to reduce duplicate detections followed by a motion cue filter with a tracker associated with it. The consolidation process was able to reduce the duplicate entries by a factor of ~1/3, followed by motion filter with another ~1/3 and tracker that was able to reduce the result by a factor of 56%.

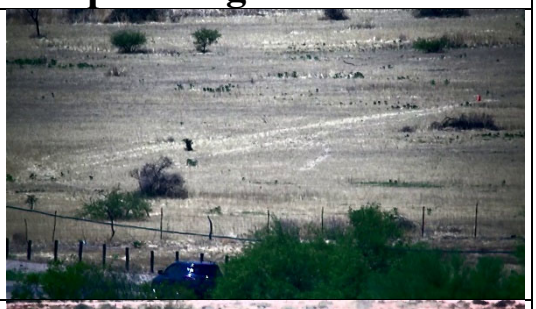
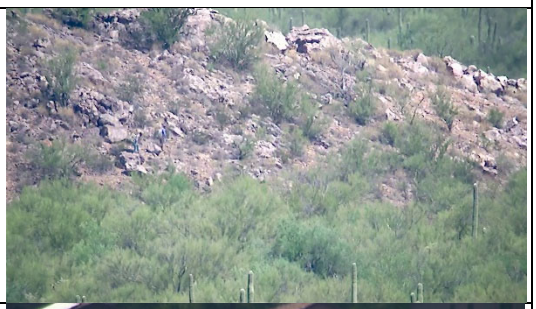
2) Dataset

2.1) Elbit Dataset : RGB

The dataset from Elbit is a multi-scene color dataset with annotations with specific YOLO format. There are 16 Color Videos available in the Elbit dataset, across 5 multiple scenes with annotations. The training dataset is classified into training and testing dataset based on 5 multiple scenes. The annotations available for the dataset has The annotation instances available across the 15 videos has 9 vehicle classes and a human/person class. The ground truth dataset has different classes of activities each human entity is doing. However, for the detection and tracking, all the activity classes were converted to the corresponding object class, ie, human. Kindly note that the instances available in are not unique instances in the entire video sample. For example, in the class tractor, the instances are taken from consecutive frames of the same object from a video sample from scene 4. The videos were categorized into train and test sets based to balance the number of instances. The test samples however lacks 3 classes, train. Tractor and motorbike as these samples are limited to a single video.

The available number of instances with the train and test dataset are as follows across all the five scenes, followed by the scene-wise samples and videos available in a given scene.

Objects	Total	Train	Test
truck	13121	5654	7467
car	33851	18297	15554
pickup	44646	30799	13847
bus	1454	135	1319
van	4025	2034	1991
tank	751	432	319
human	146667	105664	41003
train	2483	2483	0
tractor	7853	7853	0
motorbike	272	272	0

Scene	Video	Sample Image
1	vlc-record-2019-08-20-12h30m07s-rtsp_10.248.32.69_RTP-Multicast_pMediaProfile1-.mp4 vlc-record-2019-08-20-13h01m08s-rtsp_10.248.32.69_RTP-Multicast_pMediaProfile1-.mp4	
2	vlc-record-2019-08-20-13h17m35s-rtsp_10.248.32.69_RTP-Multicast_pMediaProfile1-.mp4 vlc-record-2019-08-20-13h54m06s-rtsp_10.248.32.69_RTP-Multicast_pMediaProfile1-.mp4	
3	vlc-record-2019-08-20-14h54m28s-rtsp_10.248.32.69_RTP-Multicast_pMediaProfile1-.mp4 vlc-record-2019-08-20-15h11m22s-rtsp_10.248.32.69_RTP-Multicast_pMediaProfile1-.mp4 vlc-record-2019-08-20-15h17m47s-rtsp_10.248.32.69_RTP-Multicast_pMediaProfile1-.mp4 vlc-record-2019-08-20-15h25m54s-rtsp_10.248.32.69_RTP-Multicast_pMediaProfile1-.mp4 vlc-record-2019-08-20-15h47m27s-rtsp_10.248.32.69_RTP-Multicast_pMediaProfile1-.mp4	
4	vlc-record-2019-08-21-09h35m40s-rtsp_10.248.32.69_RTP-Multicast_pMediaProfile1-.mp4 vlc-record-2019-08-21-10h20m47s-rtsp_10.248.32.69_RTP-Multicast_pMediaProfile1-.mp4 vlc-record-2019-08-21-10h29m27s-rtsp_10.248.32.69_RTP-Multicast_pMediaProfile1-.mp4	
5	vlc-record-2019-08-21-16h36m39s-rtsp_10.248.32.69_RTP-Multicast_pMediaProfile1-.mp4 vlc-record-2019-08-21-16h41m00s-rtsp_10.248.32.69_RTP-Multicast_pMediaProfile1-.mp4 vlc-record-2019-08-21-17h08m53s-rtsp_10.248.32.69_RTP-Multicast_pMediaProfile1-.mp4 vlc-record-2019-08-21-19h02m49s-rtsp_10.248.32.69_RTP-Multicast_pMediaProfile1-.mp4	

The dataset was classified modified into 4 different versions for the experiments, by creating superclass from the available class labels. The first version is the All Categories w.r.t. original Elbit dataset as described above and the rest of the categories of datasets are:

2.1.2) COCO classes from Elbit dataset

The objective of this category is to infer the performance of the detections using the COCO trained models without finetuning. The detections were run on both training and test samples available to infer the results and performance in comparison with the available dataset.

The following conversion is used to convert from Elbit to COCO labels. The effective number of samples in train and test is modified as following. The COCO converted dataset are solely for testing the performance of the detection model without any finetuning, trained on COCO dataset. The testing on Elbit COCO converted dataset ensured that the model is able to extract relevant features from the dataset, without finetuning the model.

Elbit Dataset Label	COCO Label
Pickup	Car
Van	Car
Tank	Truck
Tractor	Truck
Motorbike	Motorcycle

Objects	Total	Train	Test
truck	21725	13939	7786
car	82522	51130	31392
bus	1454	135	1319
human	146667	105664	41003
train	2483	2483	0
motorcycle	272	272	0

2.1.3) Elbit Vehicle superclass Type 1 and Type 2

The dataset is versioned into Type1 and Type2 dataset by combining the vehicle categories. In Type1 all vehicle categories are combined to a single class object. This allows the detector to increase the accuracy and to be able to generalize most of the object as vehicle. In Type2, the vehicle categories are split into 2 classes Vehicle-Small(has Pickup, Van and Car) and Vehicle-Large(having Truck. Bus, Tank and Tractor), The Typ2 is meant to distinguish between the bigger and smaller vehicles accurately than before. Some of the labels, van and pickup are confusing for the detector for the finetuned model. The Type2 was meant to reduce the confusion in the labels as much as possible.

The classes and the corresponding categories in the Type 1 and Type 2 versions of the datasets are as followed in the table below.

Elbit Label	Type 1	Type 2
pickup	Vehicle	Vehicle Small
van	Vehicle	Vehicle Small
car	Vehicle	Vehicle Small
truck	Vehicle	Vehicle Large
bus	Vehicle	Vehicle Large
tank	Vehicle	Vehicle Large
tractor	Vehicle	Vehicle Large
train	train	train
human	human	human
motorbike	motorbike	motorbike

Type 1 and Type 2 class wise distribution of the training and the testing dataset available are as following in the tables attached.

Objects	Total	Train	Test
vehicle	105,701	65,204	7786
human	146667	105664	41003
train	2483	2483	0
motorbike	272	272	0

Type 1 distribution

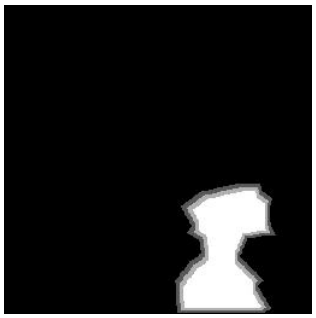
Type 2 distribution

Objects	Total	Train	Test
vehicle small	82522	51130	31392
vehicle large	23,179	14,074	9,105
human	146667	105664	41003
train	2483	2483	0
motorbike	272	272	0

2.2) Simpson Dataset

The Simpson dataset is a toy dataset that was used for training the autoencoder model used for testing purpose to test the hypothesis of model capable of suppressing the noise during training process. The dataset had 5 different classes and 404 images in training dataset with 5956 images for testing, however with no ground truth. Hence the dataset was limited to 404 images with ground truth on the segmentation. The dataset split across the classes and the sample images are as follows. The dataset is further processed to introduce noise to the decoded output using a series of multiple erosion and dilution process as shown in the sample process. A kernel size of (6,6) is used to generate the multiple erosions of the images as in the following image sample.

Class	No of instances
Bart	81
Homer	81
Lisa	80
Maggie	81
Marge	81

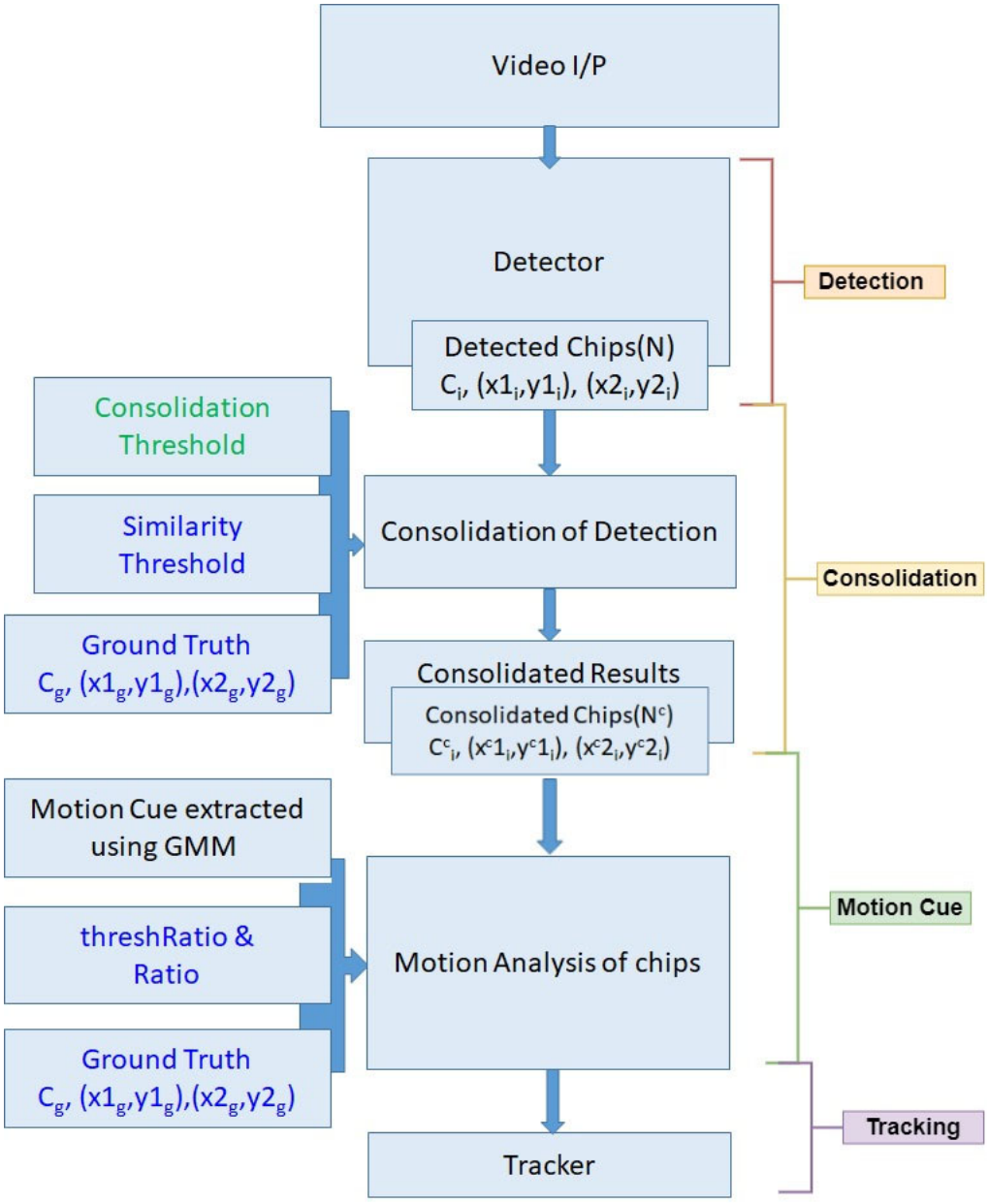

a) Input image 2) generated mask 3) region on interest

From the dataset a segmentation specific dataset was also built with same erosion and dilution methods to generate class wise segmentations as well for the autoencoder model with segmentation. Following is sample classwise dataset used for segmentation.

Marge Homer Maggie Lisa Bart

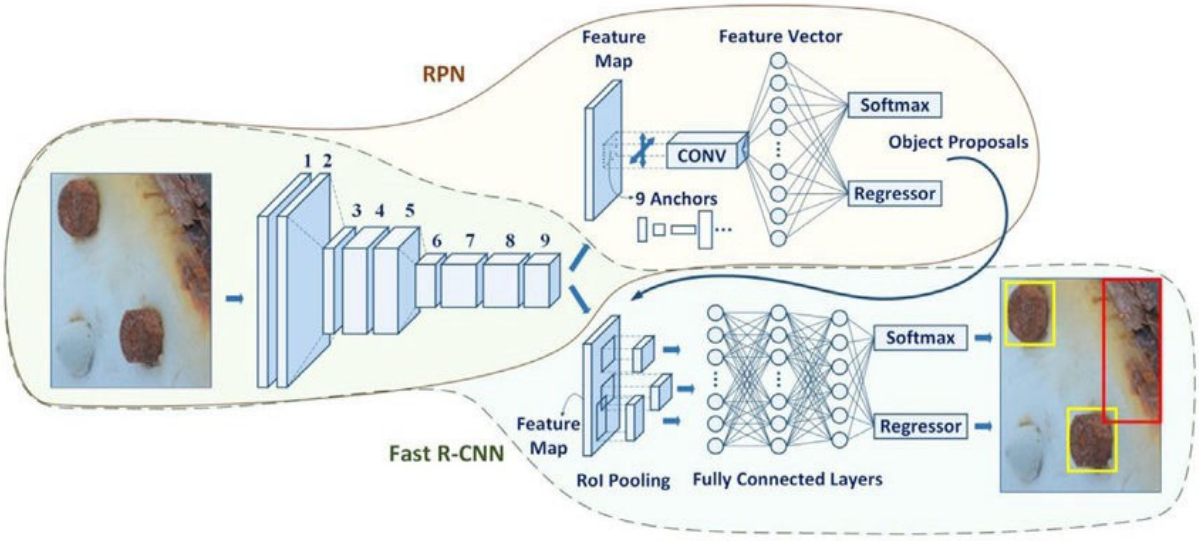
3) Architecture and Algorithms

The overall architecture of the final proposed system is follows. The systems has a detection module followed by a consolidation process, motion cue based filter and a tracker.



3.1) Detector

The detection algorithm uses a fasterRCNN based model with resent-50 feature extractor trained in COCO dataset. The pretrained model has a box AP of 40.2%, a train memory of 3 GB. The detector has relatively one of the best throughputs. Each object detected are denoted as class C_i , and bounding box, $[(x1_i,y1_i), (x2_i,y2_i)]$ for the i-th detected object in N number of detections.



3.2) Consolidation

The challenge with the dataset is that there are multiple detection for the same output objects which results in mirroring of many detection results. Inorder to reduce the duplicates from the detector, an aggressive single degree consolidation process constrained by class label and IoU of available detections.

Consider that there are N detections after the detector each object detected denoted as class C_i , and bounding box, $[(x1_i,y1_i), (x2_i,y2_i)]$. After the consolidation, assume that we have N^c detections, class labels as $C^c i$, and bounding boxes $[(x1^c_i,y1^c_i), (x2^c_i,y2^c_i)]$. During the consolidation process the number of detections dropped from N to N^c. During the consolidation process the consolidated results are compared with the ground truth data to generate a parallel between the detections and groundtruth.

The following algorithm is used to consolidate the detections from the detector.

for c_i,box_i in range(detections[:N-1]:
for c_j,box_j in range(detections[i+1:]:
find AoU of (box_i,box_j)
if AoU > consolidation_threshold &c_i == c_j:
store AoU,c and index i,j
consolidate(box_i,box_j) for the best AoU, continue
compared with Ground Truth annotation and matches using similarity_threshold

1) Detection output

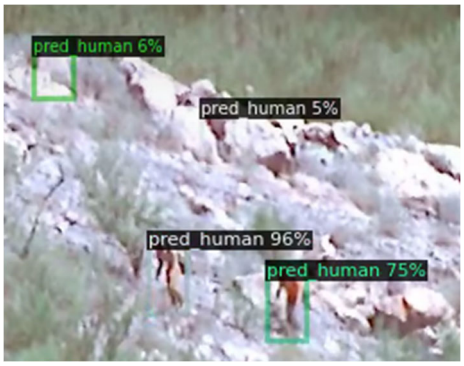
2) Consolidated output

3.3) Motion Cue extraction

A foreground extracted mask that denotes the motion in the frame w.r.t. previous frames are generated using the motion cue extractor. Following the motion cue extraction, the region for the consolidated detections is extracted from the current frame.

The following algorithm is used to consolidate the detections from the detector. The consolidated frame is defined as C^m and the bounding box *b[0],b[1],b[2],b[3]*. *'motion'* is the extracted greyscale of a detected region with motion cue extracted from the image. Using the threshold ratio, threshold actual is calculated. This helps us to define the threshold value dynamically, which helps with consistent performance across multiple videos. The threshold actual is value we will be using to threshold the motion region in the detected bounding box. Then the number of pixels having a value greater than threshold actual is calculated and compared with the (1-ratio)*instance area((b[2]-b[0]) * (b[3]-b[1])). If the number of pixels in detected region is greater than the defined number, we will consider it as detection with motion cue. The algorithm is as follows.

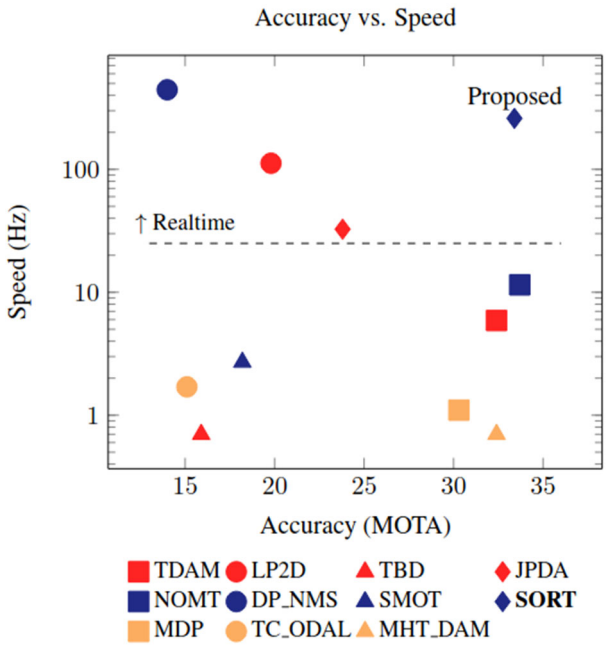
motion is the extracted greyscale motion for generated for current frame.
for $C^m{}_i, b_i$ *in range(consolidated_detections):*
instance_area = $(b_j[0]*b_j[2]) * (b_j[1]*b_j[3])$
threshActual = average(motion$[b_j[1]:b_j[3], b_j[0]:b_j[2]]*threshRatio)$
if count_nonzero((motion$[b_j[1]:b_j[3], b_j[0]:b_j[2]]>threshActual)) > (1-ratio)*instance_area):$
motion of chip detected and appends $[C^m, b_j]$ *to motion detection array.*



1) Detection output

2) FG Motion Cue filter mask

3.4) Tracker



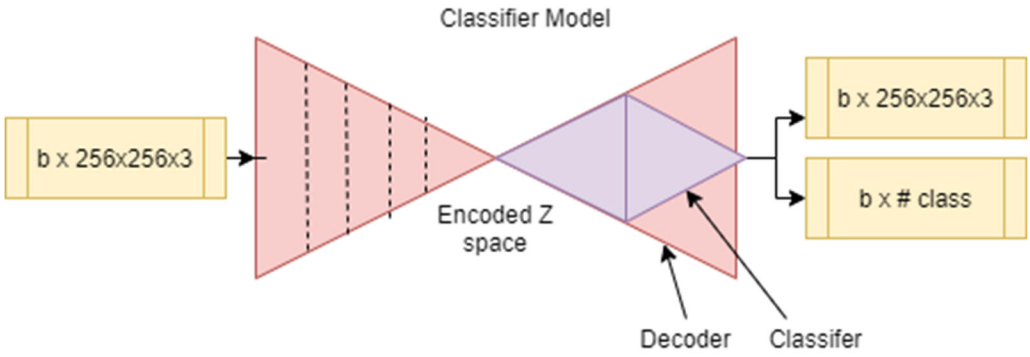
A SORT(Simple Online Realtime Tracker) tracker is used to track the multiple entries. The sort tracker was initially introduced as a multi-object tracker with a detector to improve performance significantly using Kalman filter and Hungarian Algorithm for the optimization and made use of the IoU and distance between multiple detections as a criteria to track a object. The estimation model using the Kalman filter uses the class ID and bounding box and assumes a constant aspect ratio for the detected object and uses a linear velocity model to estimate the next location of the object that is being tracked. Inorder to optimize the data association, Hungarian algorithm is used to associate the IoU distances from the target to detections.

3.5) Autoencoder Model

An autoencoder model was trained to reduce the influence of the background noise and the background objects. The model was built on the premise that the model will be able to reduce the noise by compressing the data into a z space during the encoding and minimize the image noise that might affect the decisions that the network is making. 2 Autoencoder models were tested. The results were not promising on COCO dataset, hence the model was dropped from the final presentation. The dataset used for the experiment was a toy dataset from kaggle which has segmentation data along with class information for each images to test the hypothesis. The dataset can be found here https://www.kaggle.com/mlwhiz/simpsons-main-characters. The entire Autoencoder model is available at

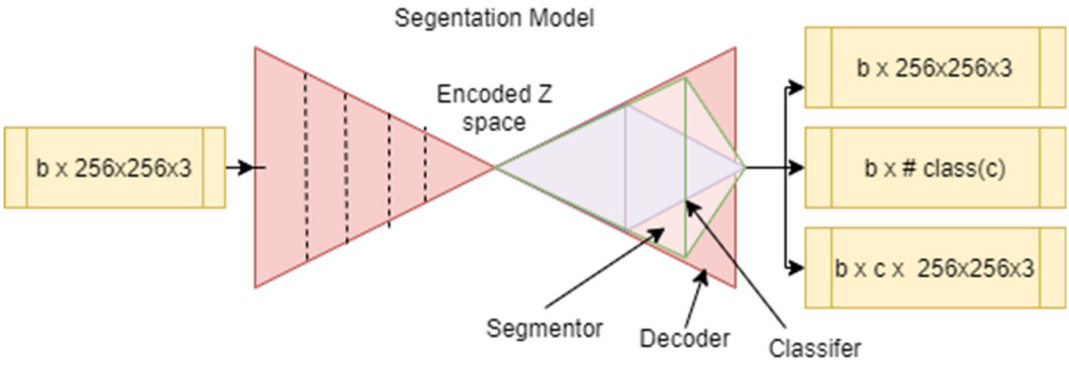
https://github.com/abramjos/classSegRGB.

3.5.1) Classifier model



The model has a encoder decoder network with 5 layers in both encoder and decoder network respectively. The decoder was trained using the segmented dataset from the dataset and a classifier is attached to the encoded space trained along with the training of the decoder module. This allowed the learning of the classifier and the decoder model simultaneous and ensured that the featured extracted is used for classification and the features representing the background of the object that is under question is suppressed by the decoder module from the encoded space. This model was trained with the decoder loss as MSELoss and the classifier loss as CrossEntropyLoss.

3.5.2) Segmentation model



The Segmentation model was created to ensure that the model was able to distinguish between the classes based on features that it was trained on, while having the classifier model attached to the main encoder decoder pipeline, to improve the performance of the classifier model. The updated model had Classifier, Decoder block and the segmentation block as well. The decoder block was trained by the MSELoss, the classifier is trained using CrossEntropyLoss/Multi-class BCE loss and the Segmentation was trained using MSELoss and the Dice and BCE loss.

4) Experiments

Multiple experiments in each pipeline were carried out in order to achieve the best possible results. The detector was set as a constant inorder to validate the results in the pipeline with consistency w.r.t to the dataset. All the experiments were carried out with different dataset version containing the superclasses. The experiments carried out in each pipeline are as follows.

4.1) Detector

The detector was trained in COCO dataset and was finetuned in different versions of the dataset including the Elbit ground truth with combined human activity class, the Type1 and Type2 datasets with combined vehicle categories. Also the results from the untrained model also has been used.

4.1.1) Trained on COCO dataset, No Finetuning in Elbit dataset

The detection results without the finetuning are as follows. Please note that even though the results seems inferior, for certain classes, human and cars, the visual inference detection results are good as the groundtruth doesn't have tight groundtruths.

Label	**Train data(5 Videos)-AP@5**	**Test data(5 Videos)-AP@5**
Human	4.004	5.704
Truck	3.954	7.862
Car	29.542	21.086
Bus	2.654	2.533
Train	12.692	n/a
motorcycle	0	n/a

Sample output of COCO v/s ground truth results. The ground truth is loosely annotated while the COCO dataset is tightly annotated and hence when the mAP is calculated the results looks less promising. The Average Precision and Recall in the IoU range 0.5:0.95 are 0.093 and 0.169 respectively.

4.1.2) Trained on COCO dataset, Finetuned in Elbit dataset

The experiment was carried out using the available groundtruth train dataset split to finetune the model using the trained model using COCO dataset. The activity classes from the human activities are combined to the entity human. The Precision and Recall in the IoU range 0.5:0.95 was 0.313 and 0.381 respectively.

Label	**Test data(5 Videos)-AP@50**
Human	41.774
Car	56.487
Truck	45.829
Bus	0
Pickup	56.635
Van	18.689
Tank	0

4.1.3) Trained on COCO dataset, Finetuned in Elbit Type1 & Type2 dataset.

The Type1 is the superclass dataset that combine all the vehicle classes into a single category and two category, vehicle. The Type2 is the superclass dataset that combines all the available classes into 2 different classes based on the size of the vehicle. By combining the class we were able to reduce the confusion across class labels and better generalization of vehicle category, given the lack of diversity in unique samples instances available in vehicle category .

The results of the finetuned models on Type1 and Type2 are as follows:

Label	**Type1**
Human	41.774
Vehicle	62.670
Average Precision @[IoU=0.50:0.9	**0.528**
Average Recall @[IoU=0.50:0.95]	**0.199**

Label	**Type1**
Human	41.774
Vehicle Big	36.636
Vehicle Small	63.115
Average Precision @[IoU=0.50:0.9	**0.476**
Average Recall @[IoU=0.50:0.95]	**0.298**

The training datasets available with the Car, Pickup, Van accounts for 51130 training instances as opposed to the Truck, Tank, Bus, Tractor having 6221 image instances, causing data imbalance in tune of 8x.

4.1.4) Benchmarking Detection Module.

The detection module was benchmarked using a V100 system and the training and inference benchmark are as follows. For the training and batch inference, a batch of 16 image sets are used. Single image inference time was estimated to be 0.094 seconds per image, with a ~10 image/second computation capacity.

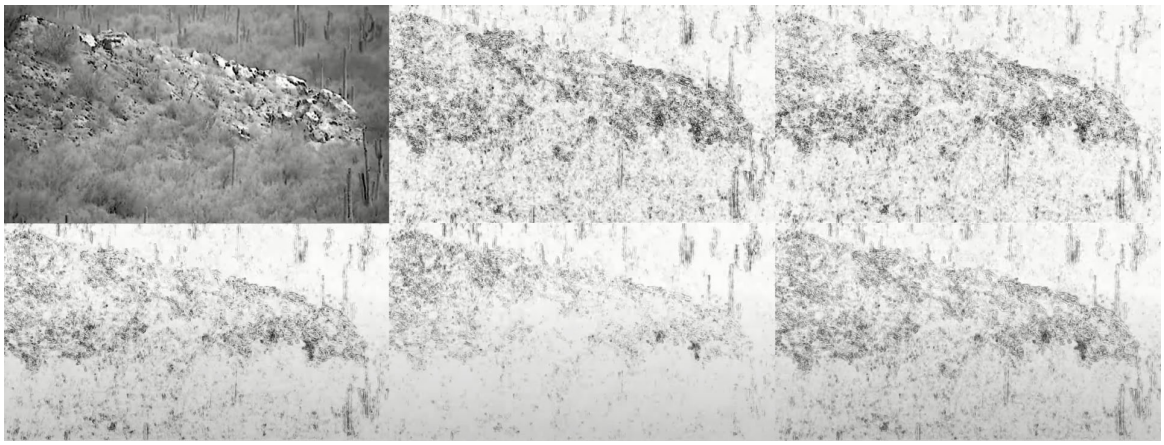
Training (sec/iteration)	Batch Inference(sec/image)	Single Image Inference(sec/image)
0.209	0.038	0.094

4.2) Motion Cue extraction

The following methods were used to extract the motion cues from the videos. Consecutive images are used to generate the image foregrounds/moving targets in the image.

4.2.1) Image Differencing

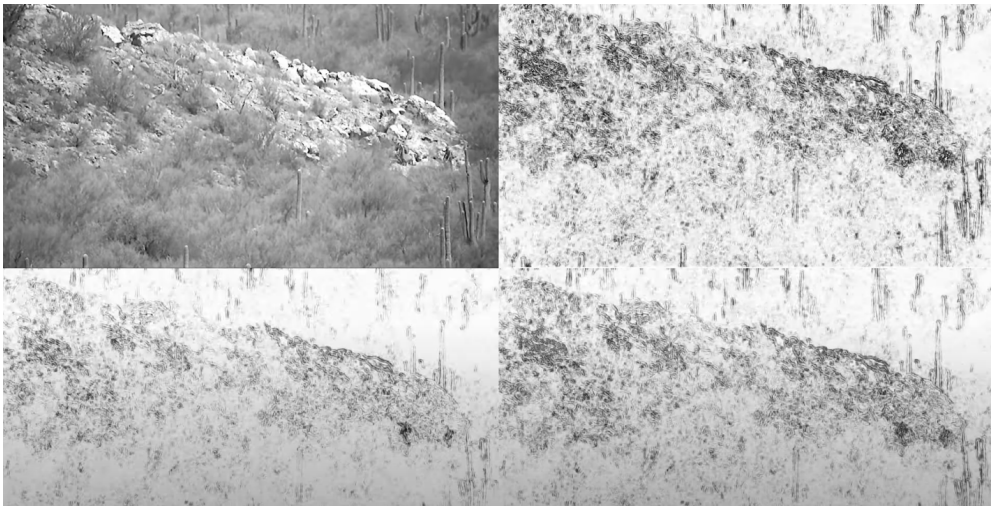
A simple image differencing is used at multiple stages. For example, the first image is the grayscale image after gaussian blur and following the blur, consecutive images are differenced to get the first-degree difference(Δ) of the image and consecutively collecting the second(Δ^2) and third(Δ^3) degree difference of the images. 9 consecutive images as a sliding window is used to generate the results upto the image difference Δ^5. The process can highlight the moving objects however with a lot of noise. Sample output is available at https://youtu.be/yMi55m5aHas.



4.2.2) Image Similarity based Motion Cue Extraction

In this technique, the structural similarity metrics between consecutive frames are used to create a map of the difference from one image to the next. The Structural Similarity Metric compares the contrast, luminance and the structure of the given image with the next one and combines the results to get the similarity measure. The similarity measure w.r.t. consecutive frames are used to generate first degree(Δ) difference images and the

differences of difference images are used to generate the second degree (Δ^2) difference image respectively. Four degrees of such differences are calculated. The method is computationally expensive and the difference images has a lot of noise as the difference images propagates the noise in every image difference.

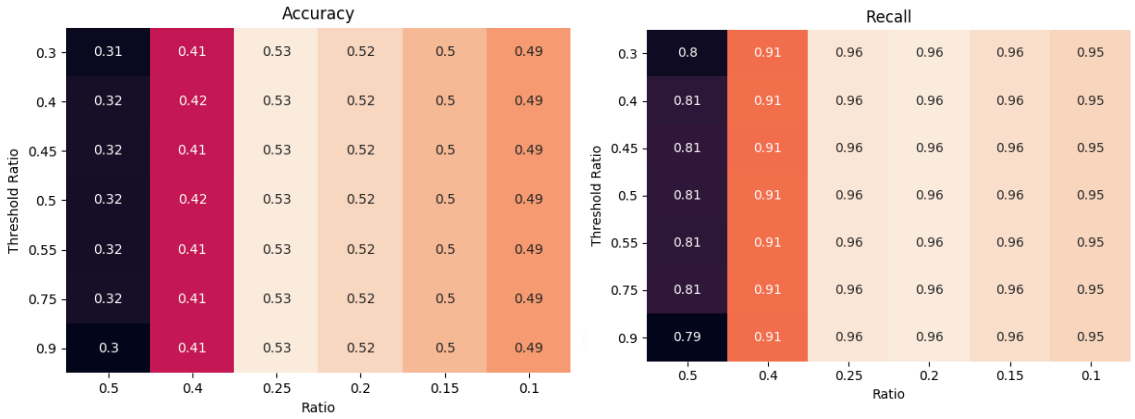


4.2.3) Improved Adaptive Gaussian Mixture Model for Background Subtraction

A Gaussian Mixture Model(GMM) with fixed number of components used and the GMM models decides if a pixel belongs to foreground or background. Number of components for the GMM is automatically selected using the Maximum Likelihood(ML) and Dirichlet Prior. The process is CPU depended and helps to reduce the overload on GPU. The method gives superior results with least noise profile in the extracted foreground. The following is the profile is a negative representation of the process.

4.2.4) Motion Cue Dependency on the Thresholding Parameters.

Foreground motion is extracted from the video and is used to suppress false detections. There are two key variables in the motion cue extraction process. One of the variable is **Ratio**(the threshold for Area moving in any given time/ total area of an instant in detection). The Ratio is varied from [0.1-0.5]. Another variable is **threshold_Ratio,** the ratio used for thresholding the value w.r.t average value, which is varied between [0.3-0.9] respectively and the precision and recall for each process are calculated to estimate best setpoints for **threshold_Ratio** and **Ratio**.

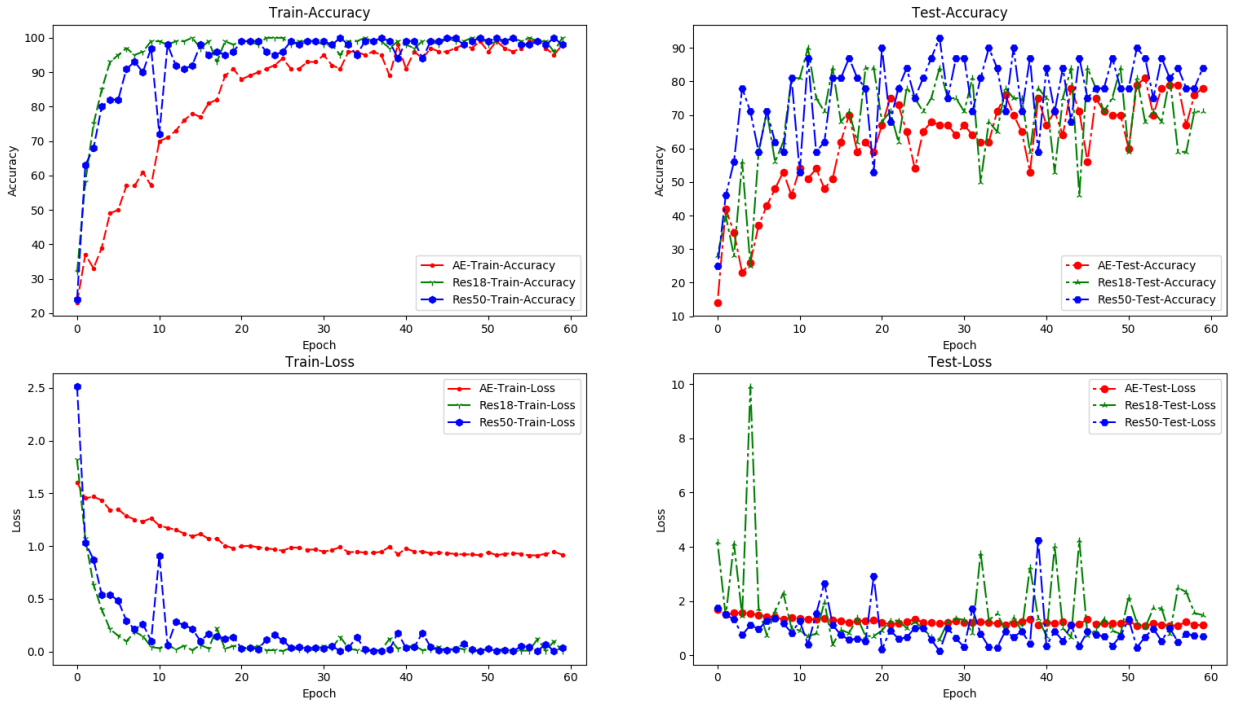


The values that are set to Ratio is set in the range of 0.25[0.3-0.2] and Threshold ratio is set in the range of 0.5 range respectively for the best results.

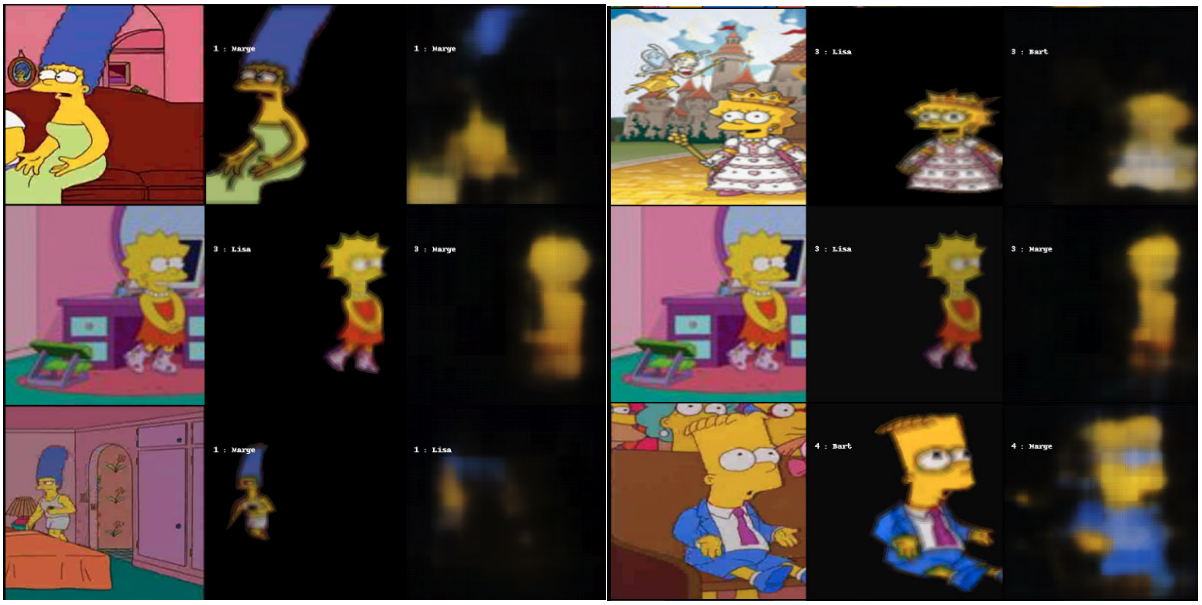
4.3) Autoencoder Model

4.3.1) Autoencoder Model – Classifier

The Classifier and the decoder network were trained in multiple passes during the back propagation of the error in the dataset. The training and test datasets were split by an 80/20 and the training setup was ensured to provide that the data used for training was never seen by the model before by ensuring that the datasets are mutually exclusive. The accuracy achieved after 500 epochs on the dataset on training and testing dataset are 98% and 85% respectively. The test accuracies were consistent, and model was compared with resnet-18 and resnet-50 models to compare the result during the training and testing process. The following chart shows the learning trend were we can see that the learning curve for the autoencoder model was quiet steep and it took 200 epochs to reach a considerable accuracy for encoder-decoder network to perform as a classifier.



The sample decoded output is follows for the testing images. We can see that the finer features were not clearly identified across the testing images. The first column corresponds to the input image, second column to ground truth for the decoder and the 3rd column for the decoded output from the trained model.



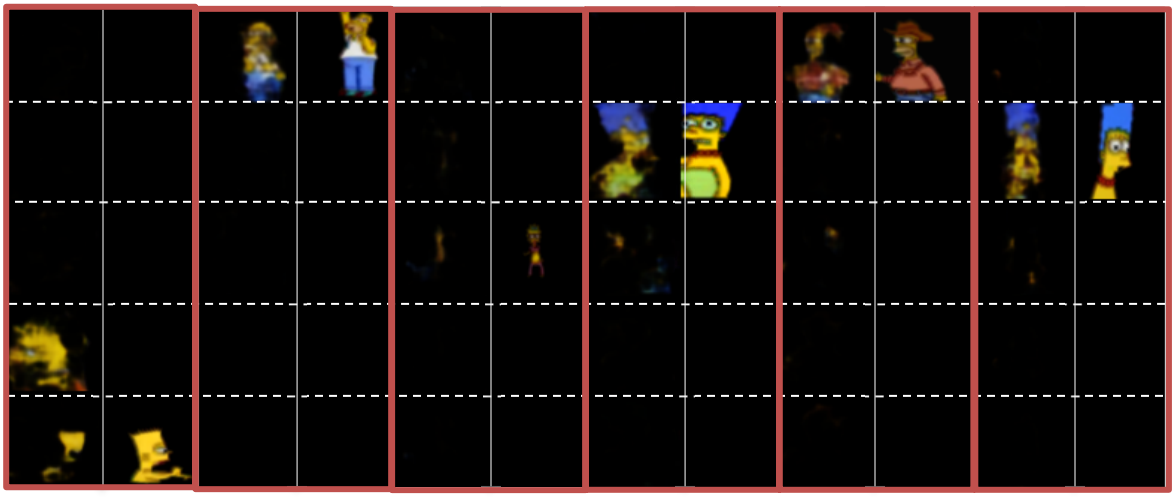
a) results after 200 epochs

b) results after 500 epoch

4.3.2) Autoencoder Model – Segmentation

The segmentation model was used to guide the training process of the encoder to specifically improve the ability to distinguish the features in the images in any given class and to improve the classification process. However, the introduction of the segmentation to the network made the process albeit more difficult than before with less agreement in the spatial domain. Although the feature details were finer than before.

Input image, ground truth and the output from the decoder network.



The segmentation output and ground truth, each red box corresponds to an image entry and the columns corresponds to output and ground truth from the segmentation model with rows corresponding to each class.

5) Results

5.1) Metrics

Assuming that we have actual ground truth and the predicted results for the testing dataset that is under examination, we can classify the dataset into True Positive(TP), True Negative(TN), False Positive(FP) and False Negative(FN) depending on what the real positive, real negative and predicted positive and predicted negative values are.

The metrics that are calculated for the training process are the following:

Accuracy

The accuracy is the proportion of correct predictions (both true positives and true negatives) among the total number of instances examined w.r.t. the ground truth data. It is defined as

$$\text{Accuracy} = TP+TN/(TP+TN + FP+FN)$$

Recall

Recall also known as sensitivity is the number of relevant documents retrieved by a search divided by the total number of existing relevant documents, while precision is the number of relevant documents retrieved by a search divided by the total number of documents retrieved by that search. It can be calculated by:

$$\text{Recall} = TP/(TP+FN)$$

Precision

Precision is a measure of quality of the detections, and it means that the algorithm returns more relevant results than irrelevant ones.

$$\text{Precision} = TP/(TP+FP)$$

F1 Score

F1 Score is the harmonic mean of the precision and the recall and it is used as the metric to measure the accuracy of the system w.r.t. the relevant retrieved data, in our case it is the detections that corresponds to the ground truth s available to the Elbit dataset.

$$\textbf{F1 Score} = 2*\text{Precision}*\text{Recall}/(\text{Precision}+\text{Recall})$$

Suppression

Suppression is defined as the ratio of the detections that were dropped in a given pipeline stage during the process to the total number of detections available before the stage.

$$\textbf{Suppression} = \text{total number of dropped detections}/\text{Total number of detections}$$

Inference Time

Inference time is the average time taken by the process to process a single iteration of the image through the entire pipeline of the system.

5.2.1) Final Benchmark – Dataset Wise

The benchmarking has been done on all different versions of the dataset, the Elbit dataset, the Type 1 dataset and the Type 2 dataset respectively. All metrics had significant improvement from the detection model alone compared with the entire pipeline. The Type 1 dataset had better performance compared to the Type 2 and All Vehicle dataset as the confusion across the classes was significantly reduced by combining all the vehicle classes. The Type 2 Big Vehicle category has only 6221 images against the 51130 samples in the Type 2 Small Vehicle category. On average there has been almost 50% of suppression of the detections, ie, 1 out of 2 detections from the model has been dropped through out the entire pipeline.

Metrics	All Veh(Finetuned)	Type 1	Type 2
Accuracy	0.732	0.86	0.807
Precision	0.596	0.89	0.822
Recall	0.681	0.691	0.693
F1score	0.636	0.814	0.752
Supression	0.56	0.639	0.591
inference time	0.09073	0.08961	0.09064
inference freq	11.02171277	11.15946881	11.03265666

Following is the results precision and recall with and without the pipelines. There has been significant improvement in the F1 Score, accuracy and precision. The improvements has been able to significantly improve the performance across the board as the pipeline was able to significantly remove a lot of the false detections from the detected bounding boxes.

Metrics	**All Vehicle**		**Type 1**		**Type 2**	
	w/o pipeline	w pipeline	w/o pipeline	w pipeline	w/o pipeline	w pipeline
F1-Score	**0.351**	**0.636**	**0.289**	**0.814**	**0.183**	**0.752**
Precision	**0.346**	**0.596**	**0.528**	**0.89**	**0.476**	**0.822**
Recall	**0.358**	**0.681**	**0.199**	**0.691**	**0.298**	**0.693**

5.2.2) Final Benchmark – Pipeline Wise

The following results are based on the results obtained from each pipeline while using the testing dataset through

the entire pipeline. We can find that the motion cue guided filtering process has been able to improve the precision of the system significantly. Also the process was able to suppress the detections that was not aided by motion by a factor of 1/3, ie, almost 1/3rd of the detections after the segmentation are being dropped during the motion cue filtering process.

The accuracy for Type1 is significantly more than that of the Type2 and All Vehicle category because the models doesn't have to distinguish between the vehicle classes in Type1. This helps us to overcome the class imbalance and reduce the confusion across the vehicle classes as we are not distinguishing them. The drop in the accuracy for Type1 from Consolidation pipeline to the Motion pipeline is because the Type1 reduced the confusion among the same vehicle category. The benchmarks performed revealed that the lower bounds of the average inference time for the consolidation and Motion Cue filtering process across 5 Video scene is 0.0008 ms/frame and 0.005 seconds respectively.

Metrics	All Vehicle cat.			Type 1(One Vehicle cat.)			Type 2(Two Vehicle cat.)		
	Consolid.	Motion	Tracked	Consolid.	Motion	Tracked	Consolid.	Motion	Tracked
Accuracy	0.613	0.603	0.852	0.843	0.696	0.773	0.642	0.786	0.8
Precision	0.565	0.686	0.752	0.76	0.801	0.741	0.738	0.694	0.746
Recall	0.773	0.675	0.935	0.981	0.677	0.677	0.677	0.979	0.784
F1score	0.636	0.583	0.72	0.85	0.656	0.642	0.611	0.802	0.677
Supression	0.315	0.335	0.345	0.337	0.371	0.294	0.375	0.287	0.325
inference time	0.00081	0.0057	-	0.00068	0.0046	-	0.00071	0.0047	-
inference freq	1234.568	175.4386	260	1470.588	217.3913	260	1408.451	212.766	260

6) Conclusion

The entire pipeline has been able to reduce a lot of false and duplicate detections in the system and was able to utilize the modularity of the system to check the performance of the system during each pipeline. The unique Motion Cue filtering algorithm process has been able to reduce the false detections in the given test video dataset aided by motion in a given detected ROI. The problem has been quiet challenging with the dataset that has very small object foot print and significant clutter in the scene and in the background. However, along with state of the art detection model trained on COCO dataset and finetuned on the Elbit dataset along with the improvements in the data pipeline was able to improve the throughput significantly by removing the detections that are redundant. The consolidation network following the detection was able to reduce the number of detections significantly and reduce the computation required at the motion cue filtering process. The entire process was able to reduce the detections by a factor of 56%. The Autoencoder was not able to provide any improvements in the results in larger dataset pool as compared to the toy Simpson dataset. A lot more improvements to the autoencoder is required to encode the feature space to suppress the noise content in the dataset.