Explainable Video Analytics - A wide area multi-camera tracking perspective

Vivekraj Sekhar[1], Liu Wenkai[1], Adriel Kuek[2]
[1]NUS High School of Math and Science, 20 Clementi Ave 1, Singapore 129957
[2]DSO National Laboratories, 12 Science Park Dr, Singapore 118225
31 Dec 2022

Abstract

With over 90,000 security cameras in Singapore alone, these devices have become an essential tool for tracking terrorists and identifying threats to the public quickly. However, with the large number of security cameras, a large amount of manpower is needed to complete this task. Thus, allowing for the criminal to escape and potentially pose a threat to the public. This project aims to utilise machine learning on closed-circuit television (CCTV) footage to help track criminals across various CCTVs. In the experiment, videos collected from DSO's very own CCTVs were run through our assembled pipeline to try and track a person/vehicle based on the imputed attributes. Firstly, the frames are extracted and run through our pipeline. A compiled set of videos and a JSON file with text descriptions will be outputted. We manage to create a model that offers quite good results, managing to successfully track a target 70% of the time in a given testing video.
Keywords: security cameras, attributes, JSON

Introduction

In order to catch criminals, police officers comb through hours and hours of CCTV footage to try and craft the path of the suspect moving away from the crime scene to try and gain information related to the whereabouts of this person. This process is not only manpower intensive but also time consuming. However, with the help of Artificial intelligence this process can be streamlined and can significantly reduce the man hours needed. Hence, this research will build and assess the feasibility of a multi-camera tracking pipeline that can access multiple video streams and track a target using a custom pipeline. These include Resnet50, SlowFast, and DeepSORT. This pipeline will be able to track a target regardless if they try to disguise themselves or flee in vehicles. This will reduce the over-reliance on manpower for tracking criminals. Hence, this study may serve as an effective measure for tracking persons of interest involving time-sensitive cases.

The project will mainly focus on identifying persons using human related attributes, identifying vehicles using vehicle related attributes, interactions and tracking between these two entities. For person identifying, we aim to identify them based on their face and a list of set

attributes. For vehicle identification, we aim to identify them based on licence plate, car model and colour.

Due to the high variety of cars on Singapore's roads, a large amount of data is needed to train our complex models that possess a large number of parameters. Furthermore, when identifying actions such as getting into a car, there may be certain conditions that may hinder the successful detection of the action. For example, there may be occlusions in the frame that may prevent certain aspects from being captured. Other issues include shadowing and low resolution. As such we have to compensate for this by having our model train on data with these occlusions present.

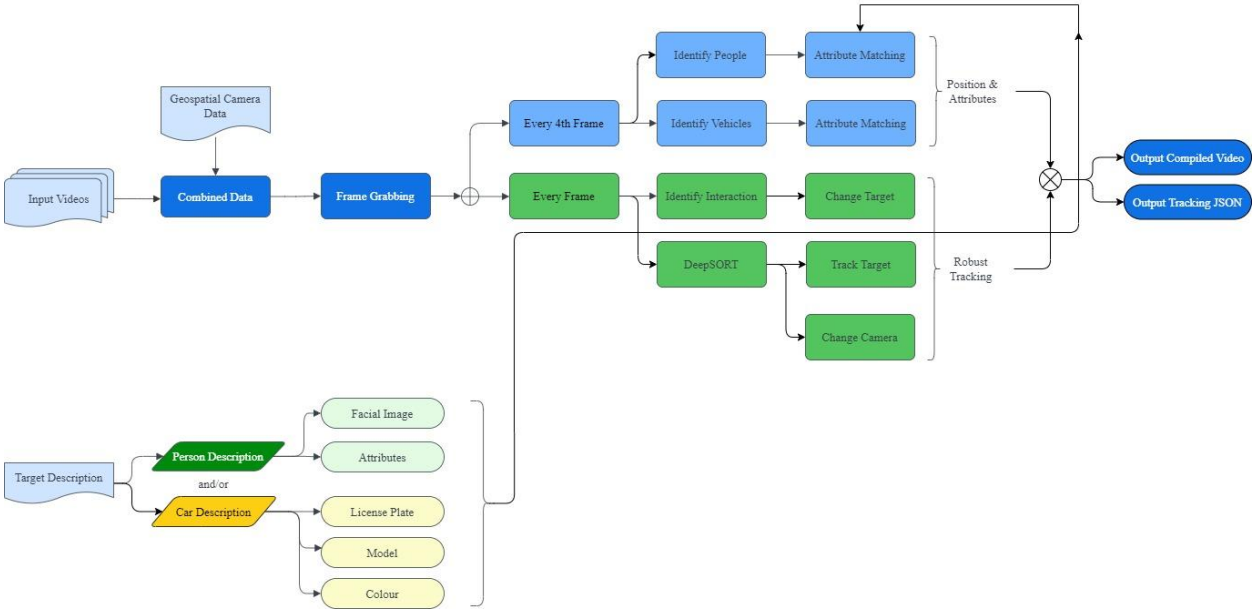


Figure 1: Final pipeline with many models

Literature Review

Slowfast model

A slowfast action recognition pretrained model provided by pytorchvideo by Facebook Research was also used [1]. The incorporation of the pretrained model through ensemble learning also aided our model in violence detection. We trained our model on the tinyVIRAT dataset[2].

The slowfast action recognition pretrained model provided by pytorchvideo uses two pathways. The first pathway focuses on processing spatial appearances such as colours and objects that can be viewed at low frame rates. The other pathway looks for rapidly changing motions such as clapping or waving that are more easily recognized in video shown at higher frame rates.

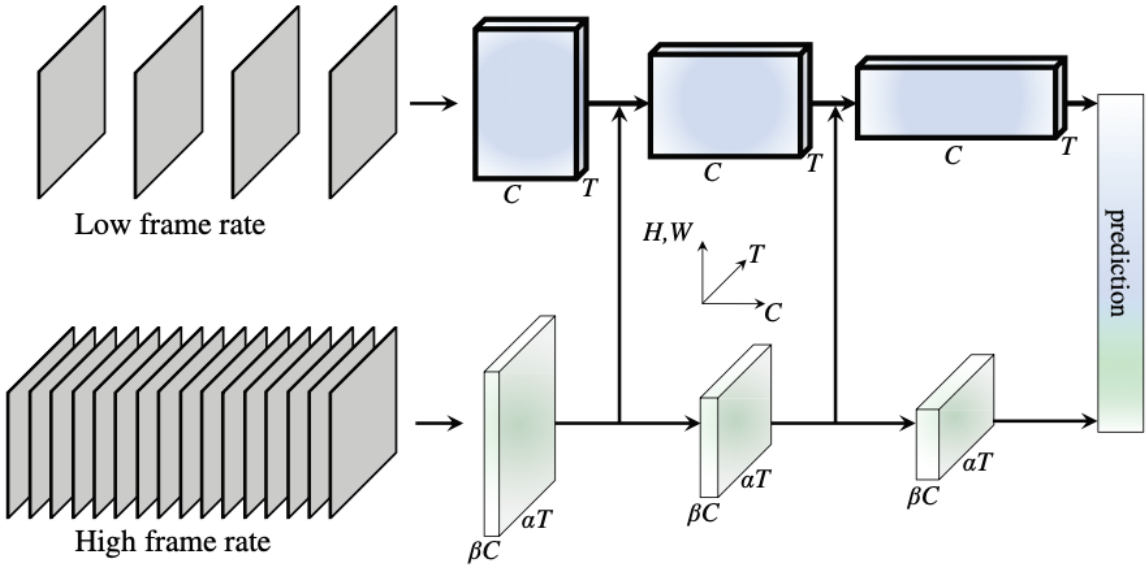


Figure 2: SlowFast network architecture [1]

Resnet50 model

ResNet-50 is a convolutional neural network that is 50 layers deep. You can load a pretrained version of the network trained on more than a million images from the ImageNet database [3]. The pretrained network can classify images into 1000 object categories, such as keyboard, mouse, pencil, and many animals.

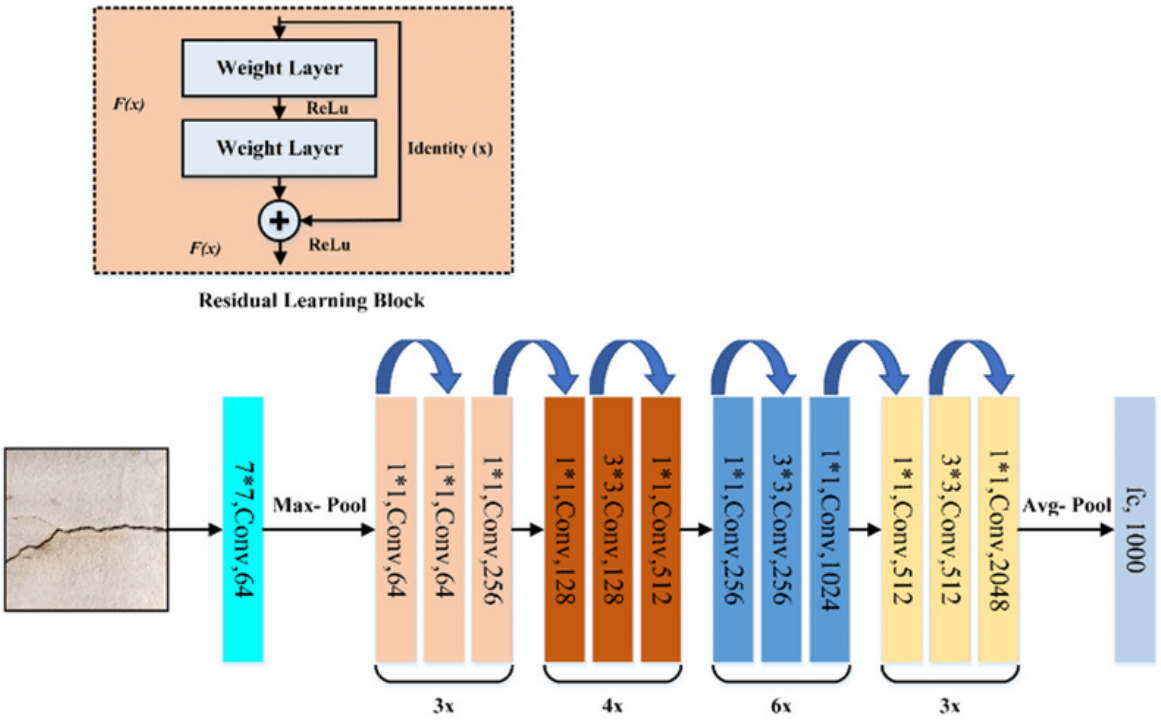


Figure 3: Resnet50 architecture [3]

Deepsort Model

DeepSort is a real-time object tracking algorithm that combines a deep neural network and the Hungarian algorithm for assignment[6]. The deep neural network is used to extract visual features from video frames and generate affinity scores between pairs of bounding boxes. These affinity scores are then used by the Hungarian algorithm to assign the bounding boxes to the correctly tracked objects. The combination of the deep neural network and the Hungarian algorithm allows DeepSort to accurately track multiple objects in real-time even in crowded and cluttered scenes.

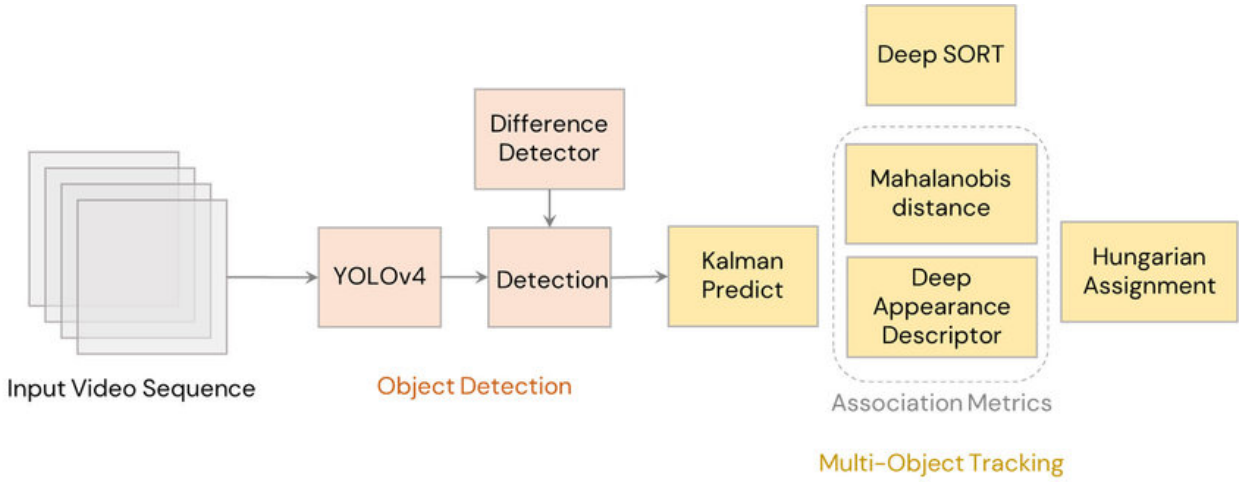


Figure 4: Deepsort architecture [4]

Method

Proposed method

In this section, we look at our proposed method in detail. Firstly, a JSON file containing the target's description will be passed into the model. The JSON file will contain a list of differing attributes depending on whether the target is a person or a car. Every frame, the model will try and identify interactions and try to change the target accordingly. The DeepSORT object tracker will also be used to track the target and toggle the cameras if the target moves from one camera to the next. At the same time, every 4th frame the model will try to identify people and vehicles via attribute matching. By limiting these computationally intensive processes to every 4th frame, the entire pipeline is more efficient and expedient. Finally the model will output both the compiled video of the target's movement and a JSON file with the updated tracking data.

Subsystems

1. **Vehicle Identification & Attributes**

For the vehicle identification and vehicle attributes subsystem, we used the DVM-Car dataset[5]. This dataset consists of background-removed images of cars, at many different angles and viewpoints, and with many different lighting conditions and shadows. The cars are labelled by their brand, model, year of manufacture, and colour, among others. The dataset contains a huge number of images, totalling 14 million, across 899 different models of cars.

Figure 5: Snapshots of cars from the DVM car dataset

Dataset Preparation

Because training on 14 million images would take an exorbitant amount of time, we used techniques to reduce the size of the dataset, and make it more specific to our purposes, to detect common car models in Singapore. Our techniques are described below:

1. **Discard Unneeded Data**. Using LTA's car registration data, we narrowed down the number of models that we were looking at, by restricting our dataset to brands with more than 500 cars registered during the period of 2005 to 2021. In total, **38** brands met this criteria, out of the original **84** in the dataset.

2. **Normalise Classes**. To normalise the classes in the dataset, we sorted all of the cars by brand, model and colour. We created two datasets from this information, one for the training of model identification, and one for the training of colour. For all resulting classes with less than **1,000** images, they were thrown away; the rest of the classes were normalised to have exactly **1,000** images total, randomly picked from the possible images.

3. **Splitting the Dataset**. After the **1,000** images were chosen, they were split among a training set, a testing set, and a validation set, with a split of **70% - 15% - 15%**. Thus, there were **700** images per class in training, and **150** in testing and validation each.

4. **Normalisation**. The images were normalised, in order to have a mean of 0, and a standard deviation of 1. This makes the model more consistent for the images.

5. **Resizing**. The images were all resized to a set scale of **128x128**, in order to ensure homogeneity of the training data.

6. **Augmentation**. Random perspectives, rotations and Gaussian blurs were added to the dataset randomly every epoch, in order to help the model generalise and make better predictions for any scenario.

After this process, **272** classes for car models and **17** colour classes remained, making up a total of **289,000** images.

Model Architecture

The Vehicle Identification model and Colour models are based on the **ResNet-50** CNN architecture. These **ResNet-50** models were pre-trained on the ImageNet dataset, and transfer learning was utilised to adapt them to the specific use-case. These models take an input image, and output a prediction as to what model or what colour the car in the image is. Originally, the two models were separate, and trained independently of each other. However, since the tasks are similar, we also made a single model, that split a single ResNet-50 in two branches, one that predicts model, and one that predicts colour. We will refer to this model as the **Split ResNet**.

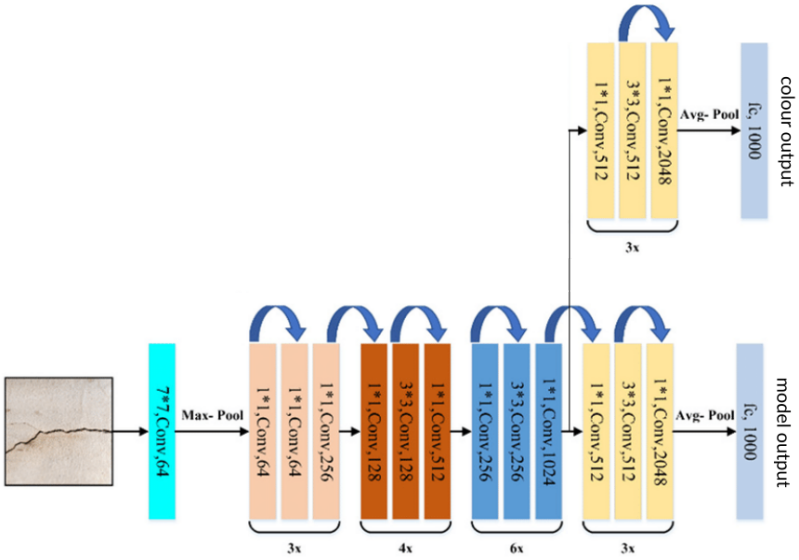


Figure 6: Split ResNet architecture, split at layer 4

Model Training

The models were both trained in **PyTorch**, using the **SGD optimiser**, as it offers slower convergence, but better accuracy than Adam. **Cross Entropy** was used as the loss function, as with most vision classification tasks. A Learning Rate scheduler was used that reduced LR when the training loss stopped decreasing. **Automatic Mixed Precision** (AMP) to speed up the training process, by using half tensors whenever possible.

Hyperparameters were tuned for this training. The learning rate was found to be optimal at around **0.02**, and momentum was set to **0.9**. The training ran for **25** epochs, as we empirically found it to be enough for the transfer learning. The model with the least validation loss was chosen as the final model.

Due to the presence of the split, the training process was different for the split architecture, as compared to the completely separate architecture used originally. To train, we first ran through a batch of images from the model dataset, followed immediately by a batch of images from the colour dataset. This trains them both in every epoch. We looked only at car model validation loss in order to determine the best model.

Pipeline Usage

The DVM-Car dataset uses background-removed cars, but our input will be a whole scene, rather than a nice background-removed car like in the dataset. Thus, some techniques are used to prepare the objects for feeding into the model.

1. **YOLOv5**. First, we use the YOLOv5 object detector to detect anything that looks like a car in the scene. YOLOv5 is another model trained on ImageNet, suited to output bounding boxes around objects detected on a scene. By looping through the found regions, we split the scene into possible cars, which we can then detect with our model.

2. **U2Net**. For each individual image, we use U2Net to get a mask of background and foreground, and then use this mask to remove the background of the image. This allows the real data that we feed in to be as close as possible to the DVM-Car dataset.

These changes allow the model to do well on data that it has never seen before.

2. Person Attributes

For the Person Attributes subsystem, the PA-100k dataset was used, a dataset of low-resolution photos of people, with 26 binary attributes to be classified. A picture can have any number of the attributes, from none of the attributes, or all of the attributes.

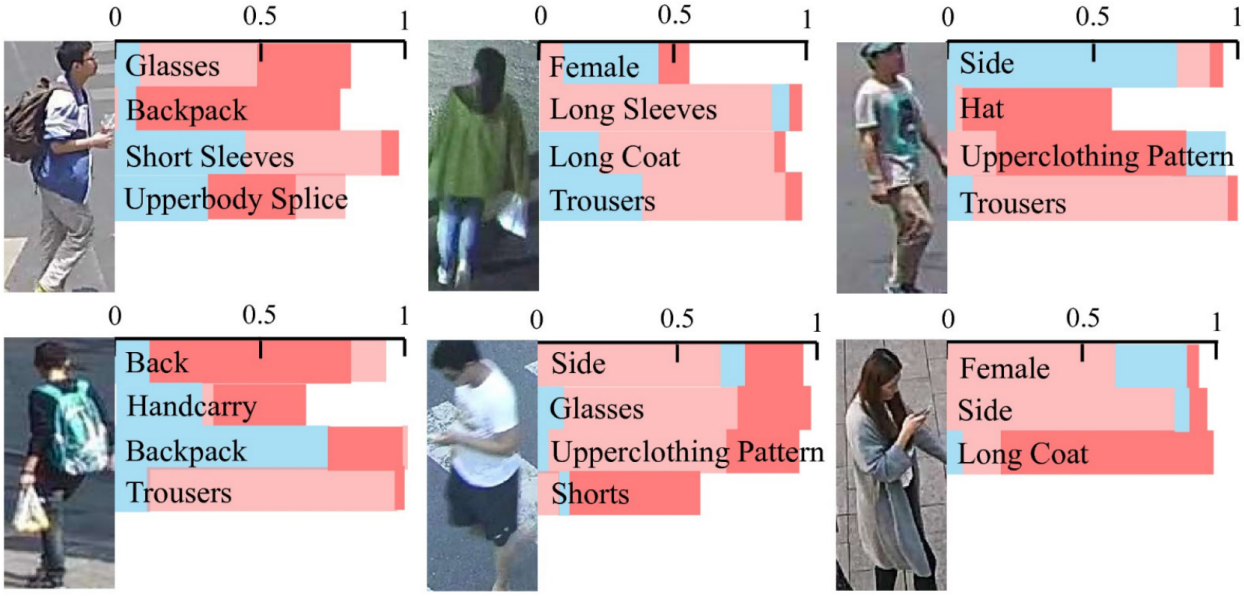


Figure 7: Example attributes from the dataset

Dataset Preparation

The 100,000 pictures were first randomly sorted, then split into training, testing and validation classes, at a ratio of **70% - 15% - 15%**. Thus, there were 70,000 training images, and 15,000

images each in the testing and validation sets. Each image was resized to a set size of **128x128**, before being fed into the models.

Model Architecture

We used a **ResNet-50** model for this purpose, acting as a 26-class binary classifier. A picture from the dataset was fed to the net, which outputted a 26-element long array, each containing the result for one binary attribute. By passing the array to a sigmoid function, we get percentage chances for each attribute to be present. For example, say that attribute 4, i.e. “backpack”, has an output of 0.94. That means that the model believes there is a 94% chance that the input pedestrian is currently wearing a backpack.

Model Training

The models were both trained in **PyTorch**, using the **SGD optimiser**. **Binary Cross Entropy with Logits** was used as the loss function. A Learning Rate scheduler was used that reduced LR when the training loss stopped decreasing. **Automatic Mixed Precision** (AMP) to speed up the training process, by using half tensors whenever possible.

Hyperparameters were tuned for this training. The learning rate was found to be optimal at around **0.01**, and momentum was set to **0.9**. The training ran for **25** epochs, as we empirically found it to be enough for the transfer learning. The model with the least validation loss was chosen as the final model.

The loss used for this model was special. Since there are a lot more absent attributes than present attributes, we weighted correct guesses for present attributes a lot more strongly than correct guesses for negative attributes. This is to prevent the model from simply predicting that every attribute was absent for every image; this would afford high accuracy in the dataset, but would be completely useless for our purposes.

Pipeline Usage

The usage of this model is quite similar to that of the vehicle attributes model. YOLOv5 was used to first isolate instances of pedestrians, and we then run them through the attribute classifier. This gives us a profile for every pedestrian, and we can then match it to one we have seen previously in another camera, or to the target that we are tracing.

3. **Entity Interaction**

For Entity Interaction, we used the TinyVIRAT dataset. This dataset consists of labelled, low-resolution actions, like people walking, cars turning and so on.

Figure 8: Snapshots of a person opening a car door from the TinyVirat dataset[2]

Data Preparation

Of the many classes in the model, we picked out a few videos and split them into two classes: those that showed people entering and exiting their vehicles, and other miscellaneous videos of people walking by cars. The goal of the model is to differentiate between people simply walking past cars, versus people actually entering or exiting the cars.

The following steps were taken to prepare the data for training:

1. **Random Temporal Subsampling**. For both the cases where there was interaction, and for the cases where there are not, **32** frames were randomly chosen from the data in order to add complexity, and to prevent overfitting.

2. **Resizing**. The videos were all resized to a set scale of **192x192**, in order to ensure homogeneity of the training data.

3. **Normalisation**. The videos were first divided by 255, to make each pixel to be between 0 and 1, and then normalised to a mean of 0 and a standard deviation of 1. This ensures a more even dataset.

Model Architecture

As the interaction can be quite short, and a ResNet-50 would be quite slow for this purpose, we decided on testing two models, a **ResNet-18** and a **SlowFast** model for the same task. The models were both fed the 32-frame video from the datasets, and outputted a single value, which was put through a sigmoid function. The output represents a probability of the video having an interaction, with 0% meaning no interaction, and 100% meaning there was one.

Transfer Learning was utilised in this model too. Both models were pretrained on the Kinetics-400 dataset, a dataset used for action recognition, and very close to our intended purpose for this model. Thus, we decided to use transfer learning to drastically speed up training times.

Model Training

The models were both trained in **PyTorch**, using the **SGD optimiser**. **Binary Cross Entropy with Logits** was used as the loss function. A Learning Rate scheduler was used that reduced LR when the training loss stopped decreasing. **Automatic Mixed Precision** (AMP) to speed up the training process, by using half tensors whenever possible.

Hyperparameters were tuned for this training. The learning rate was found to be optimal at around **0.001**, and momentum was set to **0.9**. The training ran for **25** epochs, as we empirically found it to be enough for the transfer learning. The model with the least validation loss was chosen as the final model.

For both models, the same process was used. We first ran them through the training set, then through the validation set in evaluation mode. We then use the validation loss to step the LR scheduler, and then repeat the cycle for the next epoch. At the end, we run the model once through the test set, which then gives us a final accuracy reading.

Pipeline Usage

In the pipeline, we first use the object detector YOLOv5 in order to determine positions of people and vehicles. If their bounding boxes are intersecting, we first zoom in onto the pair, before feeding the 32 frames around that point into the Entity Interaction model. If the result is "yes", the two objects are marked as linked; if not, we ignore the interaction. This is an efficient way to run this model without taking up too much computational resources.

Results

The accuracy of all our manually-trained models can be found in the table below.

Subsystem	Model	Accuracy	Comments
Vehicle Attributes	2 ResNet-50	Model - **89.0%** Colour - **69.0%**	
	Split ResNet (Linear Layer)	Model - **85.5%** Colour - **64.7%**	
	Split ResNet (Layer 4)	Model - **88.6%** Colour - **69.8%**	*Chosen for pipeline*
Human Attributes	ResNet-50 (Unweighted Loss)	**83.6%**	*Heavy bias towards predicting no attributes*
	ResNet-50 (Weighted Loss)	**74.4%**	*Chosen for pipeline*
Entity Interaction	ResNet-18	**65.7%**	
	SlowFast	**72.4%**	*Chosen for pipeline*

Analysis for Chosen Models

For the models we chose to be in the final pipeline, we did further testing, and the metrics and results are presented below.

Model Metrics

The following evaluation metrics will be used to determine how well a model performs.

Confusion Matrix

A confusion matrix provides a comparison between the model's classifications and the actual classes of the videos. It includes the number of videos that are True Positive (TP), False Positive (FP), False Negative (FN), and True Negative (TN). It is organised as follows:

Predicted / Actual	Positive	Negative
Positive	TP	FN
Negative	FP	TN

For multi-class classification purposes, the confusion matrix is modified to be a two-dimensional heatmap, with each class on the x and y axis. The y axis is the labels; the x axis is the model's prediction.

Accuracy, Precision, Recall

Accuracy, Precision and Recall are all statistics for a multi-class classification model.

- Accuracy measures the number of correct predictions, as a percentage of all possible predictions.
- Precision, for a class in a multi-class classifier, is the ratio between the number of true positives, and the number of false positives; i.e. for a class the model predicts, how many turn out to actually be of that class.
- Recall is the opposite of Precision; it is the ratio between the number of true positives and the number of items in the class; i.e. for a class that is in the dataset, how many are predicted correctly by the model.

These three measures provide a multifaceted view of how accurate a given machine-learning model is at its classification task.

F1 Score

The F1 score involves using the harmonic mean of a model's precision and recall to provide a fair measure of a model's accuracy. The equation for F1 score is given by:

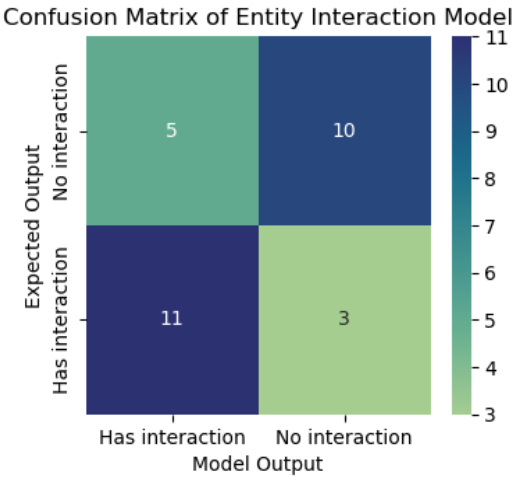
$$F_1 = \frac{TP}{TP+0.5*(FP+FN)}$$

We will focus on discussing the results from a few key subcomponents that are integral to the final pipeline.

Entity Interaction

The Entity Interaction model focuses on identifying useful events linking a person and a vehicle. This includes entering and exiting a vehicle, allowing us to associate previously unknown vehicles to a target that is currently being tracked. We can thus gain additional information about a target's personal mode of transport, and associate them with it.

The SlowFast model used for this task was trained using transfer learning on the TinyVIRAT dataset, using a small collection of videos of positive and negative interactions, as we can differentiate between entering and exiting by simply using object detectors.



Accuracy	**72.4%**
Precision	**0.688**
Recall	**0.786**
F1 Score	**0.733**

Figure 7: Confusion Matrix and other metrics of Entity Interaction model

From the statistics, we can see that the model has a good Precision and Recall, both being around 70%. The F1 Score of 0.733 is also indicative of its good accuracy. This shows that the model is good at the task, and can generalise well.

Vehicle Identification and Attributes

The vehicle identification and attributes subsystem is meant to look at frames of detected cars and detect their model and colour. This allows us to match to any given vehicular target provided, or to help identify a car that is used by the target.

Transfer learning is used to achieve this goal. The architecture of the model is unique. Since the attributes of model and colour are similar, to capitalise on this overlap, we use a single ResNet-50, split at layer 4 into two subnets, which are independently trained.

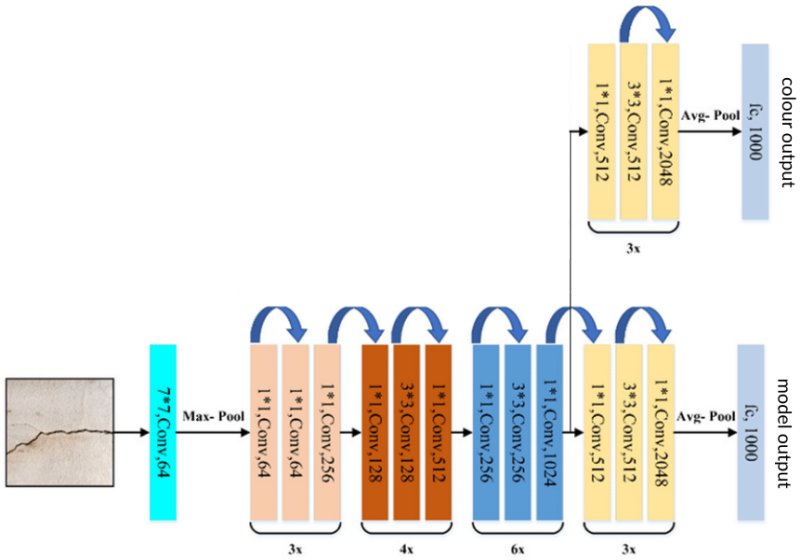


Figure 8: Modified ResNet50 model for vehicles

This worked quite well for the classification task, as can be seen by the metrics below.

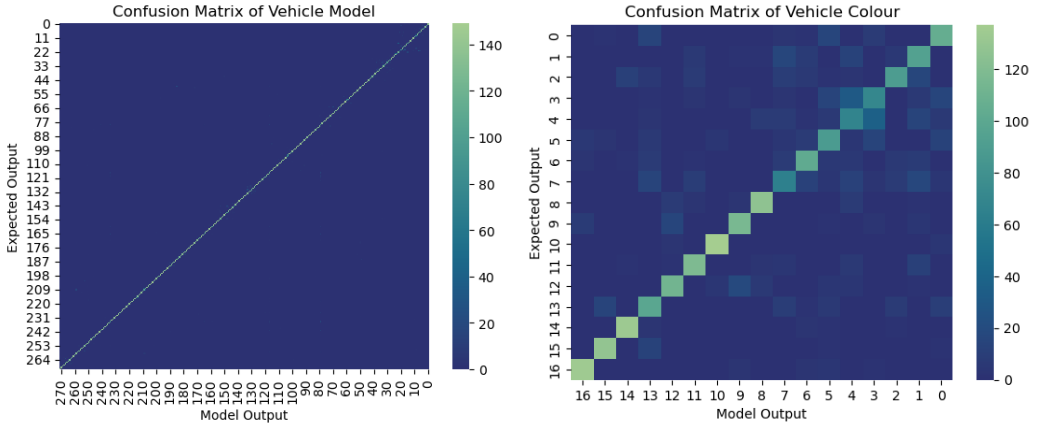


Figure 9: Confusion matrix for vehicle attributes and identification

This model achieves very good accuracy for both vehicular models and colour, though is slightly lacking in colour. This can be attributed to multiple similar colours, like Bronze and Beige, which look very similar under certain lighting conditions. This is a limitation of the task itself.

Person Attributes

The person attributes subsystem is meant to identify useful attributes of targets, useful to match them across cameras, and to match with the given data about the target. This is done with a ResNet50 model, which outputs 26 binary features, being "Attribute Present" or "Attribute Absent". For example, feature 4 is whether a target has a backpack or not. When it is 1, the target is predicted to be wearing a backpack; 0 means they are not.

Accuracy was measured by a mean of how many of the 26 predictions matched the actual labels. For example, getting all 26 predictions correct is 100%, and getting 13 of 26 would be 50%.

The model achieved an accuracy of **83.6%** overall on the dataset. This means that, on average, it gets 22 of the 26 features right, for any given sample case. This is very usable for our purposes.

Node-to-Node

When a target exits from a camera, to determine where they will next show up, we use a system of points. Each camera has a set of points, linking them to another camera. When a target starts exiting the current camera, the closest point to the last known location to the target is found, and we monitor the cameras that this point links to, in order to wait for the target's reappearance in any of those cameras.

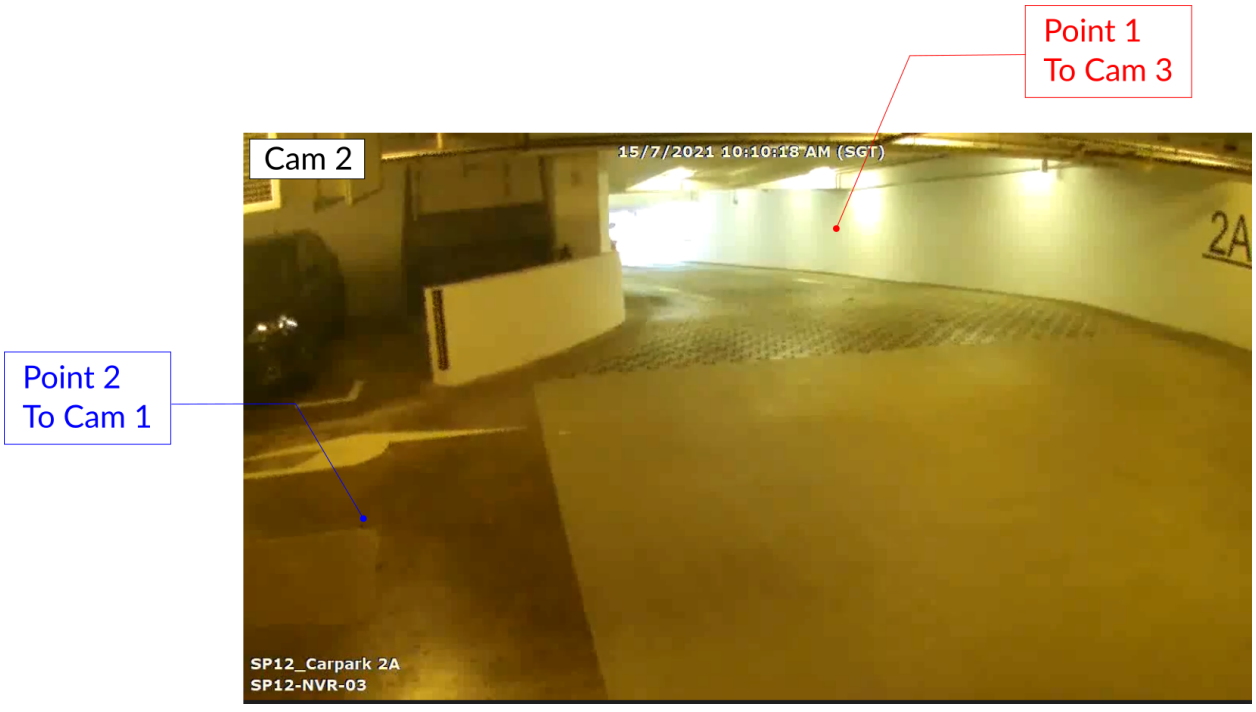


Figure 10: Example point-to-point system

For example, in the case above, a target exiting the scene by going down the ramp will be linked to Point 1. Camera 3 will then be monitored, to await the reappearance of the target in the other camera. In contrast, a target exiting the scene by going out from the left will be linked to Point 2, and Camera 1 will then be monitored for the object's reappearance.

For every object in the new camera, we try to match its attributes to the known person or vehicle that we are tracking. When a match is found, we link the two tracks together, and continue tracking from the new camera. This allows us to track a target between camera videos efficiently.

Testing the Combined System

To test the combined system, we used a video, where a target is walking around the DSO compound, gets into a car, and drives away. In order to score the model, we check, for every 4 frames, if the model knows where the target is. If it is correct, we add a point; if it is wrong, or does not know the target's position, we do not give points for that frame.

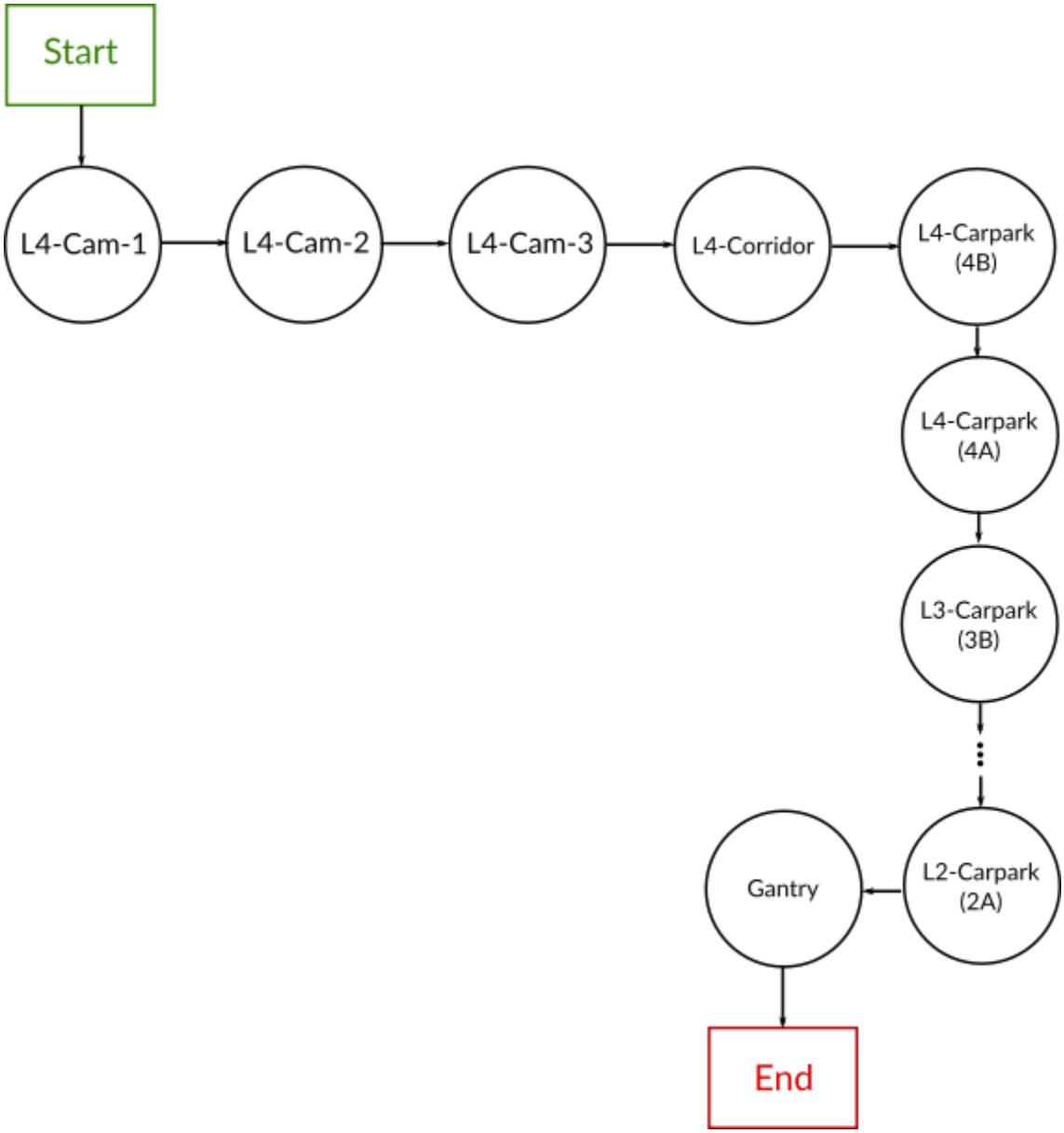


Figure 11: Path of target in video

The video was used for testing due to its large amount of camera changes and path occlusions. In total, there were 12 camera changes in total. Occlusions and shadows, especially in the car park, and the many cars were also meant to be confusing to the tracker and further increase the difficulty of the task at hand. There were also many angles where the car was quite far away from the camera, and compounded with the low resolution of the security cameras, this made it an extremely difficult challenge.

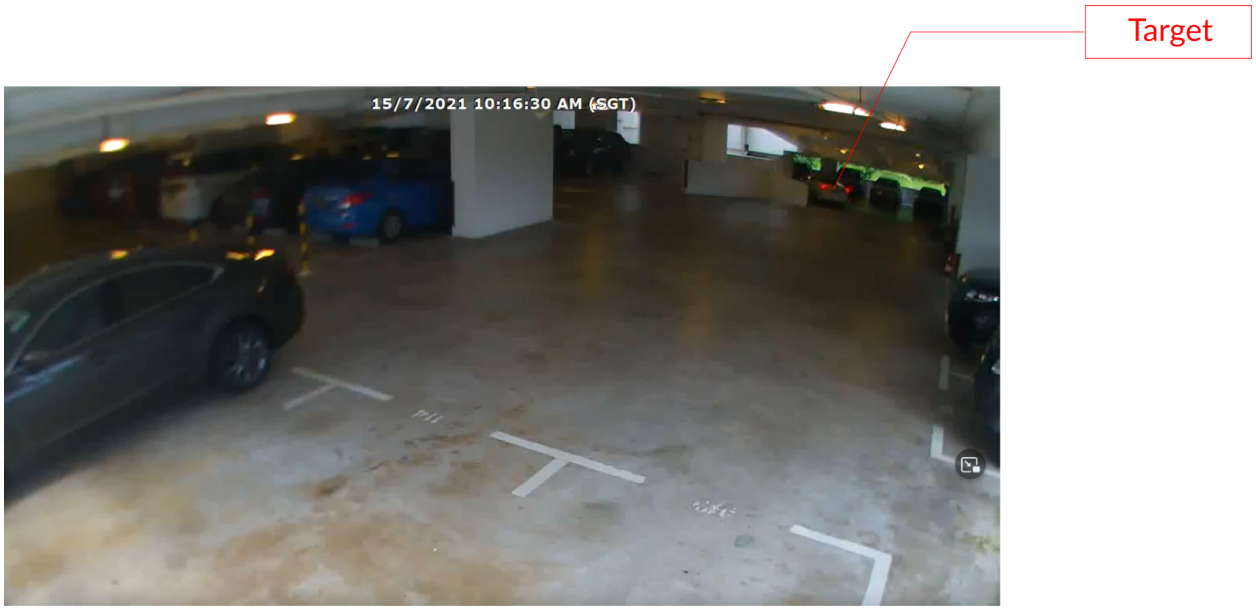


Figure 12: Example frame with tracking difficulties, including low lighting, occlusion, low resolution, etc.

Of the possible 327 points, our combined model scored 242 points, meaning that it found the target 74% of the time. Most of the losses are attributed to switching cameras and momentary tracking failures. All in all, it is quite a decent score for the complexity of the task at hand.

Discussion

In summary, our project aims to track persons of interest across multiple video streams. By utilising a pipeline involving Resnet50, Slowfast, and Deepsort. Our final models were able to achieve F1 scores 0.73 - 0.86. To bring the context even nearer to home, Singapore, we tested our model on local car park CCTV footage accessed at our research institution. We found that the model is able to track the target throughout a 10 min segment and even if the target disappears from CCTV coverage. This is due to the fact that the model will then rely on our node to node geospatial camera data and saved information in our JSON files to successfully reidentify the target. Possible further work may include detecting other items that pose an immediate threat to public security such as bags that are left alone using the same logic in our entity interaction model. We could also try rolling out this pipeline on a wider-scale CCTV network.

Acknowledgements

We thank Mr Adriel Kuek (DSO National Laboratories) and Ms Huey Ting for his continuous support and guidance throughout our research. This project was also supported by DSO National Laboratories under the R@YDSP program.

References

[1] Feichtenhofer, C., Fan, H., He, K., & Girshick, R. (2019). SlowFast Networks for Video Recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

[2] Kim, S., Xu, H., Zhang, H., Kahou, S. E., Liu, Z., Reid, I., Metaxas, D., Ramanan, D., & Zitnick, C. L. (2020). TinyVIRAT: A Tiny Video Understanding Research and Testbed. IEEE Transactions on Pattern Analysis and Machine Intelligence, 42(4), 892-906.

[3] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep Residual Learning for Image Recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

[4] Bewley, A., Ge, Z., Ott, L., Ramos, F., & Upcroft, B. (2017). Simple Online and Realtime Tracking with a Deep Association Metric. In Proceedings of the IEEE International Conference on Computer Vision (ICCV).

[5] Gong, Z., Wang, J., Tan, C., & Xu, C. (2021). Visualising the Structure of Neural Network for Multimodal Video Summary Generation. arXiv preprint arXiv:2109.00881.

[6] Ren, Z., Liao, S., Hu, Y., Sang, N., Liao, X., & Shi, H. (2018). PA-100K: A Large-Scale Pedestrian Attribute Dataset. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).