Multimedia Tools and Applications (2021) 80:5495–5515
https://doi.org/10.1007/s11042-020-09964-6

# A deep learning approach to building an intelligent video surveillance system

**Jie Xu**[1]

Received: 15 May 2020 / Revised: 15 September 2020 / Accepted: 22 September 2020 /
Published online: 7 October 2020


**Abstract**
Recent advances in the field of object detection and face recognition have made it possible to develop practical video surveillance systems with embedded object detection and face recognition functionalities that are accurate and fast enough for commercial uses. In this paper, we compare some of the latest approaches to object detection and face recognition and provide reasons why they may or may not be amongst the best to be used in video surveillance applications in terms of both accuracy and speed. It is discovered that Faster R-CNN with Inception ResNet V2 is able to achieve some of the best accuracies while maintaining real-time rates. Single Shot Detector (SSD) with MobileNet, on the other hand, is incredibly fast and still accurate enough for most applications. As for face recognition, FaceNet with Multi-task Cascaded Convolutional Networks (MTCNN) achieves higher accuracy than advances such as DeepFace and DeepID2+ while being faster. An end-to-end video surveillance system is also proposed which could be used as a starting point for more complex systems. Various experiments have also been attempted on trained models with observations explained in detail. We finish by discussing video object detection and video salient object detection approaches which could potentially be used as future improvements to the proposed system.

**Keywords** Deep learning · Video surveillance · Machine learning · Object detection · Face recognition

## 1 Introduction

In the past few decades, surveillance cameras, also known as Closed-circuit television (CCTV), have had a rapid growth in number around the world. Take Great Britain as an example. In England and Wales, the number of surveillance cameras rose from 100 in 1990 to about 4.2 million in 2007, which means there are over 7 surveillance cameras for every

✉ Jie Xu
jie.xu-4@student.manchester.ac.uk

[1] Department of Computer Science, The University of Manchester, Manchester, UK

100 people [8]. With such a huge number of surveillance cameras deployed, it would require enormous effort for outputs from all these cameras to be monitored by human.

Therefore, in this paper, a deep learning approach to automatically process images captured by surveillance cameras is presented, which focuses on automated object detection and face recognition. The aim is to explore a feasible way to integrate both object detection and face recognition methods into commercial video surveillance systems by evaluating and experimenting state-of-the-art algorithms.

We have chosen to use deep-learning methods for both object detection and face recognition tasks for a number of reasons. First of all, deep-learning methods are easier to deploy and possess better scalability than conventional machine-learning methods thanks to their ability to process data in their raw form [21]. Without deep learning, we would have to first transform the input data, in our case pixel values into a certain representation that our models were able to process, which required significantly more effort than deep-learning methods. Moreover, deep-learning methods are capable of learning more complex features by taking advantage of multiple levels of representation [21]. Again, these features are learned systematically rather than designed manually by humans.

It is also important to show that automated object detection and face recognition are indeed useful to modern surveillance camera systems. As an example, around 39% undergraduates and 17% graduate students have admitted that they have cheated on tests [28]. Sometimes, it is nearly impossible for a small number of exam invigilators to catch everyone who is trying to cheat in a large exam room, and it would be such a huge waste of manpower to have many exam invigilators walking around for hours just in case someone was planning to cheat. However, with surveillance cameras equipped with automated object detection, things that are not allowed in a certain exam, such as smartphones or sometimes calculators can be easily spotted.

On the other hand, despite the 4% annual decrease, there have been over 380,000 burglaries committed in England and Wales in the year ending September 2019 [6]. Likewise, face recognition can help us prevent burglaries by detecting and recognising faces in a given area. If a face is both detected and recognised, then we can safely ignore it as it suggests that it's from someone we know. However, if a face is detected but not recognised, it is possible that an unwelcome stranger has broken into the secured area, and this can be immediately reported to someone in charge.

In this paper, we compare the performance of some of the latest object detection and face recognition approaches in terms of both accuracy and speed. By evaluating the performance of trained models, we conclude that Faster R-CNN [32] with Inception ResNet V2 [37] is suitable for achieving incredibly high accuracy on the assumption that the available hardware is powerful enough to handle the amount of computation required so as to maintain a reasonable frame rate. Otherwise, if the hardware at one's disposal is not advanced enough to achieve real-time rates with the previous approach, SSD [24] with MobileNet [14] could be considered which is much less computationally expensive and still provides impressive accuracy that should suffice to meet requirements of most video surveillance applications.

As for face recognition, FaceNet [34] with Multi-task Cascaded Convolutional Networks (MTCNN) [44] produces incredibly high accuracy without being too hardware consuming. Our evaluation results in Section 4 show that there is no significant difference between training the model using only static images or image sequences. Although the best accuracy is still achieved by training on a mixture of as many images of both types as possible, we can still reach promising accuracy with only a single type of training images. It is, however,

crucial to train the model on high-quality images if the model is going to be applied on high-quality image sequences for performing face recognition.

With regard to the rest of the paper, in Section 3 we present an end-to-end video surveillance system which is capable of performing both object detection and face recognition tasks. The structure of the proposed system is also illustrated and explained in detail. Finally in Section 5 we describe a few video object detection approaches which could be considered as future improvements to the proposed system.

## 2 Related work

### 2.1 Algorithms

In this section, we compare the accuracy and speed of some state-of-the-art object detection and face recognition approaches. This is especially crucial to video surveillance applications since the tolerance to false positives and negatives is extremely low for safety reasons. Furthermore, a reasonably high frame rate has to be maintained consistently in order to avoid missing important frames. It is therefore of vital importance for us to achieve an optimal balance between accuracy and speed without sacrificing one for the other to such an extent that the overall error rate becomes intolerable. Here we strive to make our comparisons as fair as possible by collecting evaluation results tested on the same set of datasets. In the case of object detection, most meta-architectures are also tested with the same feature extractor. Note that this was partially done in [23] with some information missing, which we are going to present later in this section.

#### 2.1.1 Object detection

Obviously, there are quite a few existing object detection algorithms with promising accuracies. Spatial pyramid pooling networks (SPPnets) [12], for example, are able to achieve up to 59.2% mAP on PASCAL VOC 2007 [11] for detection tasks, which is slightly better than the 58.5% mAP produced by region-based convolutional neural networks (R-CNNs) [10, 12]. More importantly, SPPnet was invented to achieve better efficiency over R-CNN by sharing computation. As a result, SPPnet can outperform R-CNN by 10 to 100× in terms of speed while still producing accuracy of similar level [12].

However, when using the same pre-trained VGG-16 [35] network as a starting point and bounding-box regression, Fast R-CNN achieves 68.1% mAP, which is a fine improvement over the 63.1% mAP obtained by SPPnet, both trained using PASCAL VOC 2007 without "difficult" examples [9, 12]. Not only does Fast R-CNN produces higher accuracy, it is also notably faster. When performing detection using VGG-16, Fast R-CNN can be 3× faster than SPPnet and 9× faster than R-CNN [9]. It is also worth mentioning that Fast R-CNN is faster to train when compared to R-CNN and SPPnet. When training VGG-16, training time is reduced by 2.7× and 9× when compared to SPPnet and R-CNN, respectively [9]. Interestingly, as stated in [9], R-CNN yields 66.0% mAP under nearly identical test conditions, which differs from what's reported in [12].

An improvement over Fast R-CNN [9] known as Faster R-CNN [32] is later proposed with the addition of Region Proposal Networks (RPNs). Instead of utilising an external proposal generator as in the case of R-CNN and Fast R-CNN, Faster R-CNN generates proposals using convolutional neural networks and was amongst the first papers to introduce the so called anchors methodology. The RPN introduced in Faster R-CNN replaces the region

proposal method known as Selective Search (SS) [40] used by Fast R-CNN, which is generally considered as a computational bottleneck. In comparison with Selective Search, RPN costs much less time to generate region proposals by sharing most computation with the object detection network. The result of which is that Faster R-CNN with RPN can perform detection 10× faster than Fast R-CNN with SS [32]. When tested on PASCAL VOC 2007, Faster R-CNN is able to achieve 73.2% mAP, which makes it comparatively more accurate than Fast R-CNN which produces 68.1% mAP [32]. This makes Faster R-CNN both faster and more accurate than Fast R-CNN and SPPnet, which is why we decide to train our first object detection model using Faster R-CNN as its meta-architecture. A comparison of mAP and speed for the methods mentioned above is shown in Table 1.

Once the meta-architecture is settled, we then need to decide on the feature extractor. According to the evaluation results presented in [16], Faster R-CNN achieves the best accuracy when using Inception ResNet V2 [37] as the feature extractor. In fact, Faster R-CNN together with Inception ResNet V2 forms the most accurate model tested in [16]. Therefore, we decide to use Faster R-CNN and Inception ResNet V2 to train the first object detection model which aims at achieving highest accuracy possible without sacrificing too much efficiency. However, [16] also suggests that both Faster R-CNN and Inception ResNet V2 can be computationally expensive. This leads to the decision of training a second object detection model whose goal is to make it possible to perform detection on a wider range of machines.

Unlike Faster R-CNN [32], You Only Look Once (YOLO) [31] utilises a single neural network that performs bounding box regression and classification at the same time from full images. This makes YOLO 90× faster than Faster R-CNN with VGG-16 when trained on PASCAL VOC 2007 [11] with a promising 63.4% mAP [31]. However, YOLO also has certain limitations such as grid cells with fixed aspect ratios.

Single Shot Detector (SSD) [24] overcomes some of those limitations by allowing different aspect ratios and scales. SSD also uses default boxes similar to anchors used in the RPN of Faster R-CNN. The result of which is that SSD generates boxes that better match the scales and shapes of objects detected. Furthermore, SSD with VGG-16 achieves 74.3% mAP with 59 FPS on PASCAL VOC 2007, both of which are more superior than results published in [32] for Faster R-CNN. However, according to [16], when using other feature extractors with better average accuracies such as ResNet-101 [13] and Inception ResNet V2 [37], Faster R-CNN still makes the most accurate model.

Region-based Fully Convolutional Networks (R-FCN) [5], with the addition of position-sensitive score maps and position-sensitive Region-of-Interest (RoI) pooling layer, yields even better results using ResNet-101 [13], reaching 83.6% mAP with only 0.17 second spent on each image. In contrast, Faster R-CNN with ResNet-101 achieves slightly better 85.6% mAP at the cost of being nearly 20× slower [5]. Again, it is stated in [16] that Faster R-CNN is able to achieve slightly better accuracy when using Inception ResNet V2 [37]

**Table 1** Object detection results on Pascal VOC 2007 test set collected from [9, 10, 12, 32]. All methods use VGG-16. Test speedups are calculated in terms of runtime FPS

| Method | mAP (%) | Test speedup |
|---|---|---|
| R-CNN | 58.5 − 66.0 | 1× |
| SPPnet | 63.1 | 20× |
| Fast R-CNN | 63.1 − 66.9 | 98 × −146× |
| Faster R-CNN | 73.2 | 90× |

instead of ResNet-101 as the feature extractor. R-FCN, however, gives the best performance with ResNet-101 and does not perform as well with all other feature extractors tested.

Based on the test data reported in [16], although it is possible for us to train an R-FCN model that's more accurate than an SSD one without sacrificing too much speed, the fastest models are still SSD models with MobileNet [14] and Inception V2 [17] feature extractors. It is also worth mentioning that MobileNet is approximately twice as fast as Inception V2 when ignoring post-processing time [16]. This makes it obvious for us to choose SSD with MobileNet for training our second object detection model since its purpose is to be as computationally cheap as possible while maintaining sensible accuracy. Table 2 provides a detailed comparison for some of the methods we have considered for training our second object detection model.

As mentioned above, both Faster R-CNN and SSD follow the so-called anchors methodology, which means they share similar training processes. A fixed number of anchors, also known as default boxes, of different shapes and aspect ratios are first placed on each image at different locations. A model is then trained to predict the class of the anchor and an offset by which the anchor has to be shifted to match the groundtruth bounding box.

With regard to training parameters for our object detection models, we use a minimum dimension of 600 pixels and maximum dimension of 1024 pixels for the Faster R-CNN model and a fixed size of 300 $\times$ 300 for the SSD model. This means when training a Faster R-CNN model, the shorter edge of an image is always scaled to 600 pixels and the longer edge is scaled down to 1024 pixels if it's original length is longer than the maximum dimension specified.

Since TensorFlow [1] supports the meta-architectures and feature extractors chosen, we decide to use TensorFlow to train and evaluate our object detection models. In order to save time from training brand new models from scratch, we obtain both models by performing transfer learning on models provided on TensorFlow detection model zoo [1].

### 2.1.2 Face recognition

For our face recognition model, our goal is to discover the optimal algorithm from recent advances such as DeepFace [38], DeepID2+ [36] and FaceNet [34]. As demonstrated in Table 3, FaceNet achieves 95.12% accuracy on YouTube Faces DB [43] and an impressive 99.63% accuracy on Labeled Faces in the Wild (LFW) [15] dataset. In contrast, DeepFace and DeepID2+ are reported to achieve 91.4% and 93.2% accuracies on YouTube Faces DB, respectively [36, 38]. This results in FaceNet reducing the error rate of DeepFace by

**Table 2** Object detection results on Pascal VOC 2007 and COCO for resolution=300 collected from [5, 24, 31]. Test speedups are calculated in terms of runtime FPS

| Method | mAP (%) | Test speedup |
|---|---|---|
| YOLO | 63.4 (PASCAL VOC 2007) | 2.1× |
| YOLO (VGG-16) | 66.4 (PASCAL VOC 2007) | 1× |
| SSD (VGG-16) | 74.3 (PASCAL VOC 2007) | 2.2× |
| SSD (MobileNet) | 20.0 (COCO) | 6.8× |
| SSD (Inception V2) | 22.0 (COCO) | 6.1× |
| R-FCN (ResNet-101) | 83.6 (PASCAL VOC 2007) | 5.3× |
| R-FCN (MobileNet) | 83.6 (COCO) | 5.1× |

**Table 3** Face recognition results on YouTube faces DB and LFW collected from [34, 36, 38]

| Method | Accuracy on YouTube faces DB (%) | Accuracy on labeled faces in the wild (%) |
|---|---|---|
| DeepFace | 91.4 | 97.5 |
| DeepID2+ | 93.2 | 98.70 |
| FaceNet | 95.12 | 99.63 |

nearly 50% and of DeepID2+ by 30% when tested on YouTube Faces DB. Similarly, when evaluated on LFW, FaceNet reduces the error rate of DeepID2+ by 30% and accomplishes error rate more than 7× lower than that of DeepFace [34].

In addition to its incredible accuracy, as later illustrated in Section 4, face recognition with FaceNet is in our case a much less computationally intensive task when compared to object detection with Faster R-CNN [32]. This is mainly due to the fact that with FaceNet, face recognition can be solved by simply applying k-NN classification once a Euclidean embedding is learnt using a deep convolutional network [34]. Therefore, FaceNet provides us with satisfying accuracy while being fast enough for real-time applications, which is why we decide to only train a single face recognition model using FaceNet instead of two as for object detection.

In terms of training parameters, we use an embedding dimension of 128 to train our FaceNet models. This embedding size was also chosen in the original FaceNet paper for nearly all experiments as it could take more time to achieve the same accuracy with larger embeddings [34].

Similar to object detection, we mainly use the Tensorflow implementation of FaceNet [34] and pre-trained models published on [33] for training our face recognition models. The pre-trained model we use for our experiments is an Inception ResNet V1 [37] model trained using softmax loss on VGGFace2 [4] with Multi-task Cascaded Convolutional Networks (MTCNN) [44] used for face detection and alignment. Note that triplet loss was used instead of softmax loss in [34].

### 2.2 Datasets

Choosing the right datasets on which the models are trained is another important decision we have to make. For practical applications, this probably depends significantly on the purposes of the trained models. Since the goal of this paper is to evaluate model performance in general video surveillance cases, we try to find datasets that are suitable for as many potential applications as possible.

#### 2.2.1 Object detection

For training object detection models, PASCAL VOC [11], Microsoft COCO [22], and Open Images Dataset (OID) [20] are some of the most popular options. Although PASCAL VOC 2007 and PASCAL VOC 2012 are perhaps the most commonly used datasets for evaluating the performance of object detection models, they only contain images of 20 object classes [11]. The lack of object categories makes PASCAL VOC datasets unsuitable for many commercial applications, which in turn makes it inappropriate for our evaluation. This is improved by COCO which comes with 80 object categories and 1.5 million object instances, and even further by OID with 600 object classes and 16 million bounding

boxes [20, 22]. The primary benefit of training on a dataset which covers more object categories is that it is less likely that we miss any object classes important to many potential video surveillance applications. In other words, the evaluation results obtained from models trained on datasets with more object categories are going to be more informative and representative in our case even though the models are not trained on all available categories. For this reason, we decide to train our object detection models on OID with 32 selected object classes so as to allow the evaluation results to be fine-tuned for video surveillance applications.

#### 2.2.2 Face recognition

We will see later in Section 4 that in order for us to produce comprehensive evaluation results on the performance of our face recognition models, datasets consisting of both static images and image sequences are needed. Therefore, we decide to train our face recognition models on two separate datasets, one with static images and the other with image sequences. Obviously, these two datasets will have face images of two different groups of people, and for our evaluation, it has to be the case that a subset of people is shared by both datasets.

With that in mind, we discover that VGGFace2 [4] together with YouTube Faces DB [43] makes a promising combination. VGGFace2 contains over 3.3 million face images of over 9000 people with an average of 362 images per subject [4]. Meanwhile, YouTube Faces DB provides 3,425 videos of 1,595 different subjects with an average of 2.15 videos for each subject [43]. Although only a small subset of subjects is shared by VGGFace2 and YouTube Faces DB, it is enough for carrying out our evaluation.

In addition to YouTube Faces DB, we also collect a few extra videos from YouTube for each of the subjects used for our evaluation. This is mainly because most of the frame images provided by YouTube Faces DB are of fairly low resolution. On the other hand, the static images from VGGFace2 are, in most cases, of much higher quality. Furthermore, the number of videos per person ranges from 1 to 6 for YouTube Faces DB, which means that it only has a single video for some subjects. If we test our classifiers on image sequences from YouTube Faces DB only, we then have to use images from the same video for both training and evaluation for subjects with one video only. It is later shown in Section 4 that, the evaluation results are much less meaningful even when completely different subsets of frame images from the same video are used for both training and evaluation.

## 3 Proposed approach

This section aims to describe our proposed structure of an end-to-end video surveillance system with built-in object detection and face recognition capabilities. The proposed system can be divided into two separate stages, with the first stage focusing on model implementation and the second on the workflow of the front-end application.

### 3.1 Model implementation

Since our system is designed to be able to perform both object detection and face recognition, and the two object detection models of our system have different structures as well, three different sets of approaches are presented in Fig. 1.

As far as object detection is concerned, we have mostly adopted original implementations of Faster R-CNN [32] and SSD [24] with a few changes made to adapt our system design

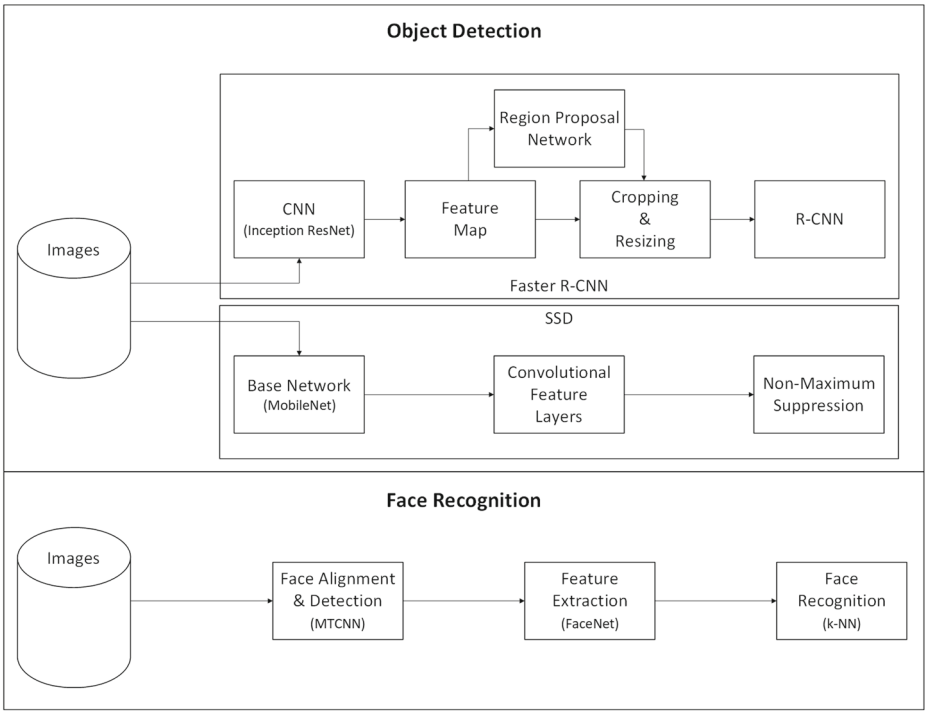


**Fig. 1** Implementations of object detection and face recognition models of the proposed system

to fit video surveillance purposes. For instance, we use Inception ResNet [37] as the feature extractor for training our Faster R-CNN model instead of VGG-16 [35] as in the original paper. Likewise, we replace the VGG-16 network in the original implementation of SSD with a MobileNet [14] network as the base network. The reason for both of these structural modifications is so that we can achieve higher accuracies with the trained models, which is particularly important for security systems. Moreover, Region of Interest (ROI) pooling is replaced by TensorFlow's 'crop_and_resize' operation for training Faster R-CNN models.

With regard to our implementation of Faster R-CNN [32], we first take an image and pass it to a convolutional neural network (CNN) which extracts features from and produces a feature map for that image. We then apply a Regional Proposal Network (RPN) [32] on that feature map and obtain a set of object proposals along with their respective objectness scores. Proposals from the previous step are then resized to a fixed dimension, and after that, passed to a region-based convolutional neural network (R-CNN) [10] which classifies the resized proposals and refines bounding boxes.

As for SSD [24], we start with the base network which extracts feature maps of a given image. Detection predictions are then made by the added convolutional feature layers using a set of convolutional filters. Default bounding boxes are also used to speed up the training process. At last, final detection results are generated by removing duplicate predictions from the previous step using non-maximum suppression.

The implementation of face recognition is mostly borrowed from the project [33] developed by David Sandberg, which implements the face recogntion system described in [34]. It also utilises ideas from the paper [30] and is hugely inspired by the project [2].

In order to perform face recognition, we first apply face detection and alignment on an image using MTCNN [44]. Faces detected and aligned are then fed to a trained FaceNet model which creates embeddings for the faces. Once the embeddings are created, face recognition can be achieved using k-NN technique. This is possible because FaceNet transforms face images to an Euclidean space in such a way that distances directly correspond to similarity. Hence, the similarity between two face images can be evaluated simply by finding the distance between the two points representing the faces, which is defined as:

$$\begin{aligned} d(\mathbf{p}, \mathbf{q}) &= d(\mathbf{q}, \mathbf{p}) \\ &= \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + ... + (q_n - p_n)^2} \\ &= \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2} \end{aligned} \tag{1}$$

### 3.2 Application

Upon completion of model training, we are then ready to apply the trained models onto a real video surveillance system. Here, we describe the structure of our application, which could be used as the base model from which more advanced systems could be built. The workflow of the proposed application is illustrated in Fig. 2. Note that this application is designed to process images from a single remote camera and has a linear structure with a loop to process images one at a time.

Before any image processing could be done, we have to first set up the working environment which involves importing necessary packages and declaring variables that will be used in later stages. We are then ready to start processing input images after loading our trained object detection and face recognition models into the program. Once an image from the remote camera is received by our program, we may wish to preform some pre-processing before applying our models on it. A variety of different image processing techniques may

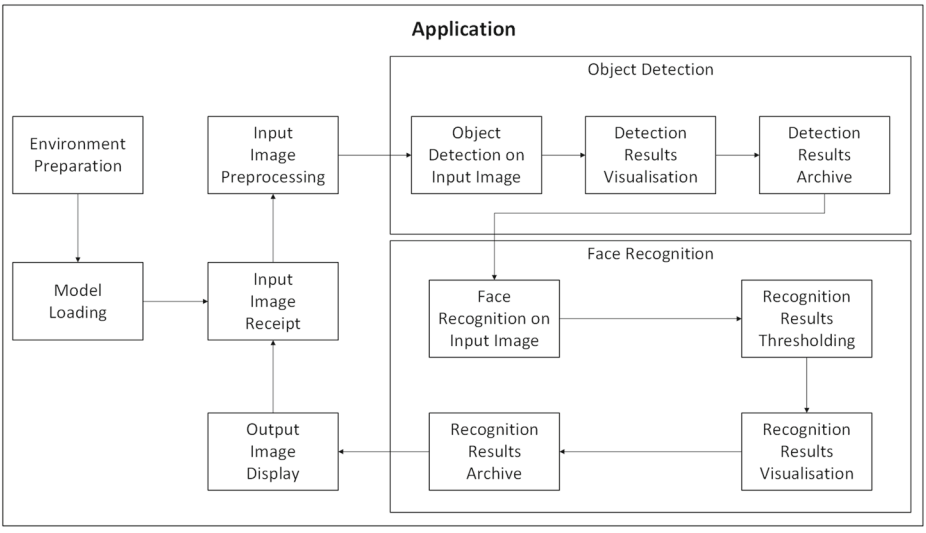


**Fig. 2** Workflow of the proposed video surveillance application

be used depending on the specific user case. For example, if images captured by the remote camera are of a resolution that is too high to be processed real-time by the computer that is used to run this application, then image downsampling could be applied in order to achieve real-time rates. Note that once an image is downsampled in this stage, visualisation results from later stages, such as coordinates for detection boxes have to be scaled back before drawing them onto the image to be displayed.

Object detection and face recognition in our case have similar workflows with the exception of an extra stage for face recognition, namely, the thresholding stage. The first stage, which is shared by both tasks aims to apply the specific model on the input image and collect various data as the result. For object detection, this includes category, confidence score and box coordinates for each detected object. Similarly, applying a face recognition model on an input image could yield information such as class, confidence score and box coordinates for each detected face. It is worth mentioning that we are interested in the recognition results for every face detected, not just the ones that are accurately recognised. This is because faces detected may be considered as being from someone suspicious if their recognition scores are lower than a predefined threshold value, which is the job of the next stage.

The goal of the second stage for face recognition is to define the identity of each detected face. This stage is exclusive to face recognition because it requires us to perform recognition on detection results, which is not part of the workflow for the object detection task. Here, we use thresholding techniques to classify each detected face based on its corresponding recognition score. If the recognition confidence score of a face falls below the threshold value defined by the user, then this face will be classified as not being recognised. The implication of such a result could be an alert from the system indicating the discovery of an unknown person in the area under surveillance. On the other hand, a face will be classified as a known person if the recognition score is not lower than the threshold value, and therefore no further action is needed.

Naturally, a higher threshold value means more false negatives and a lower one yields more false positives. It is completely up to the user to decide whether a particular system is more tolerable to false positives or negatives. If safety is of utmost importance, then it will be sensible to have a reasonably high threshold value so that the chance of missing anyone suspicious is minimised. Otherwise, decreasing the threshold value is going to be a better choice as it reduces the number of wrongly classified cases to be filtered out manually, which requires more ‘human effort’.

The next stage, which is again shared by both tasks, is to visualise the results so that it is easier for direct visual inspections to be carried out. This may involve drawing boxes around detected objects and faces, showing their categories/classes and displaying their corresponding confidence scores. Depending on the user story, more data could be displayed to accommodate the user with extra information if necessary.

According to the design of our application, the last stage for both object detection and face recognition is where we archive results from previous stages. This is beneficial to the user in many ways. For example, the user no longer has to worry about missing something important from the visual feedback which is a vital improvement to the system as far as safety is concerned. In terms of the information to be archived, this may include timestamp, category and confidence score for each detected object. In the case of face recognition, since we are only interested in faces from unknown people, only the timestamp and recognition score need to be archived for each unrecognised face as they all belong to the same class, namely, the ‘unknown’ class.

Once we have completed all processing for both object detection and face recognition tasks, the last stage before the iteration of the next image begins is the one where we display the image we have been processing in the current iteration. The image we display, of course, contains various visual effects added from previous stages to help the user identify objects and faces that are worth more attention. After displaying the image, we then proceed to the next iteration starting from the receipt of the next image from the remote camera.

## 4 Experiments

In our experiments, we evaluate all of our trained models comprehensively for both object detection and face recognition. Both object detection models are evaluated on the validation subset of Open Images Dataset (OID) [20]. As we use the training subset of OID for training our object detection models, the validation subset contains all the object categories that our models are trained on, and thus is sufficient for carrying out evaluation.

Choosing the right dataset for evaluating our face recognition classifiers is trickier, especially as we are trying to evaluate the performance of our classifiers on image sequences. It is later shown that the evaluation results are much less reliable even when completely different subsets of image sequences from the same video are used for both training and evaluation.

### 4.1 Speed

Figure 3 shows the average FPS of the proposed application when running different models. We first approximate the average FPS of the application without running any object detection or face recognition models. This is needed because we have to first be aware of the number of frames we receive from the input camera and the amount of time taken to process and render the raw images to make it clearer how much slower the system becomes after extra processing being taken into account. Here, the system itself can process 11.3 images per second without running object detection or face recognition on its inputs. The number

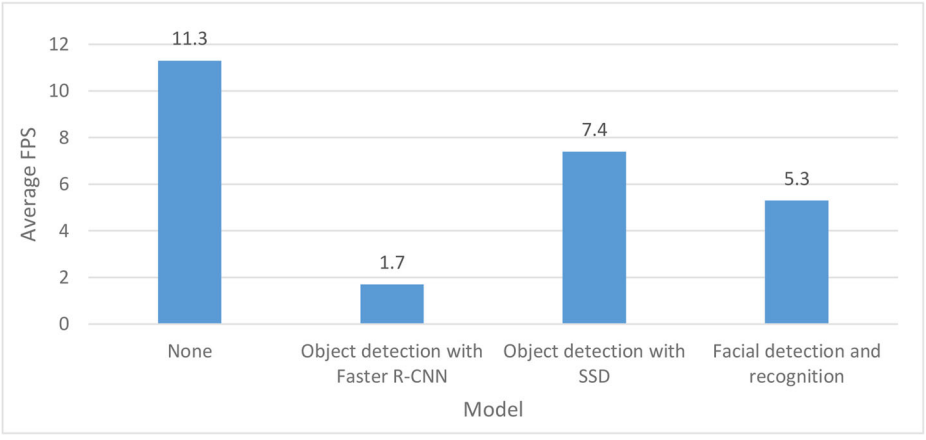


**Fig. 3** Average FPS on a GTX 1060 GPU with none or each of the trained models running. Tested on a remotely connected camera

of frames per second that the system can achieve ranges from 1.7 to 7.4 when either object detection or face recognition is enabled. We observe that the Faster R-CNN [32] model is indeed much slower than the SSD [24] model, which justifies our decision of training an SSD model for better efficiency.

Since the results shown in Fig. 3 are obtained by testing on a GTX 1060 GPU, it can be concluded that SSD models are more suitable than Faster R-CNN ones when the available hardware for running the application offers similar amount of mid-range computational power. With only 1.7 frames per second, it is very likely that we will miss the few frames containing valuable information that we are looking for, hence making Faster R-CNN models nearly useless in such situations.

Furthermore, we can see from Fig. 3 that face recognition in our case is also less computationally expensive than object detection with Faster R-CNN [32]. However, it is still slightly slower than running object detection with SSD [24] since two separate tasks, namely face detection and face recognition have to be performed instead of just one as in the case of object detection.

## 4.2 Accuracy

### 4.2.1 Object detection

Figure 4 shows the overall mAP of the original pre-trained models with 600 object classes as well as the ones we re-train for 32 object classes using transfer learning. Unsurprisingly, Faster R-CNN [32] models yield substantially higher mAP than SSD [24] models, with the re-trained Faster R-CNN model producing an mAP of 53% as opposed to the 40% mAP achieved by the re-trained SSD model. Of course, the higher overall mAP comes at the cost of speed as illustrated in Fig. 3 . Both re-trained models also possess similar accuracies compared with their pre-trained counterparts, which is to be expected.

Additionally, detailed information regarding the performance of our re-trained Faster R-CNN model is shown in Figs. 5 and 6. The performance details for the re-trained SSD model

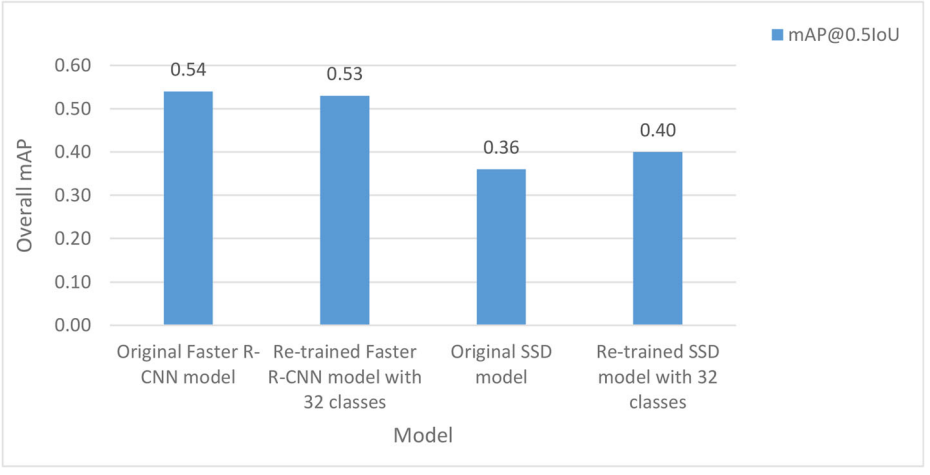


**Fig. 4** Overall mAP of object detection models, including both the original ones with 600 object classes from which we perform transfer learning, and the ones that we re-train for 32 object classes

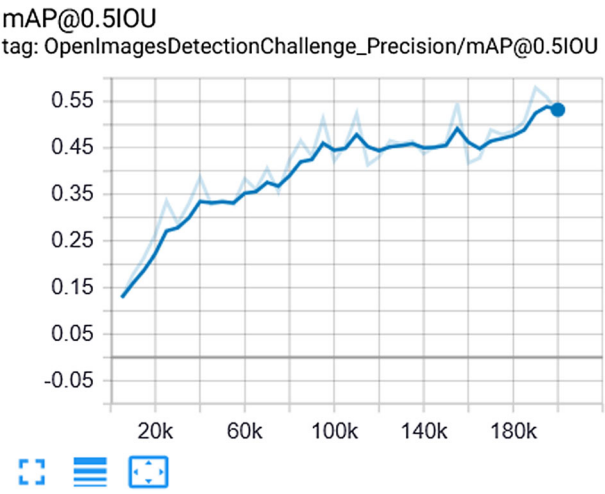


**Fig. 5** Accuracy of the re-trained Faster R-CNN model with 200k iterations, using a momentum optimizer with manual step learning rate

are presented in Figs. 7 and 8 instead. The Faster R-CNN model was re-trained with more iterations mainly because each iteration took a few more times to complete when re-training the SSD model.

It can be seen from Figs. 3 and 4 that despite being slower, Faster R-CNN models are indeed more accurate than SSD models. Although the extra accuracy may not be worth the additional computation required when it comes to applications with greater tolerance to false positives and negatives, some systems may require extremely high accuracy. We then have to rely on Faster R-CNN if the accuracy achievable by an SSD model does not meet the minimum requirement of a particular system. Of course, in such cases, hardware that is more powerful than the GPU used in our experiments has to be deployed in order to be able to achieve real-time rates with Faster R-CNN so as to minimise the error rate.

Since the proposed system is designed for video surveillance purposes, it is also important to evaluate the performance of the chosen object detection approaches on video-based datasets. The evaluation results presented on [25] show that it is possible to achieve an accuracy of up to 70% mAP with a pre-trained SSD MobileNet v1 model on COCO dataset and fine-tuning the last layers on the CORe50 [26, 27] dataset. Based on our previous results illustrated in Figs. 5 and 7, it is to be expected that an even higher accuracy could be achieved by a Faster R-CNN model under similar conditions. Thus, the evaluation results on CORe50 is another proof that the selected object detection approaches are capable of giving solid performance when used in video surveillance applications.

### 4.2.2 Face recognition

As mentioned in Section 2, we evaluate the accuracy of face recognition models comprehensively by training and testing on static images alone, image sequences alone, and different combinations of the two. The static images used are taken from VGGFace2 [4], whereas image sequences are from two different sources, YouTube Faces DB [43] and one with images of higher resolution built by ourselves. We take the average of the results produced

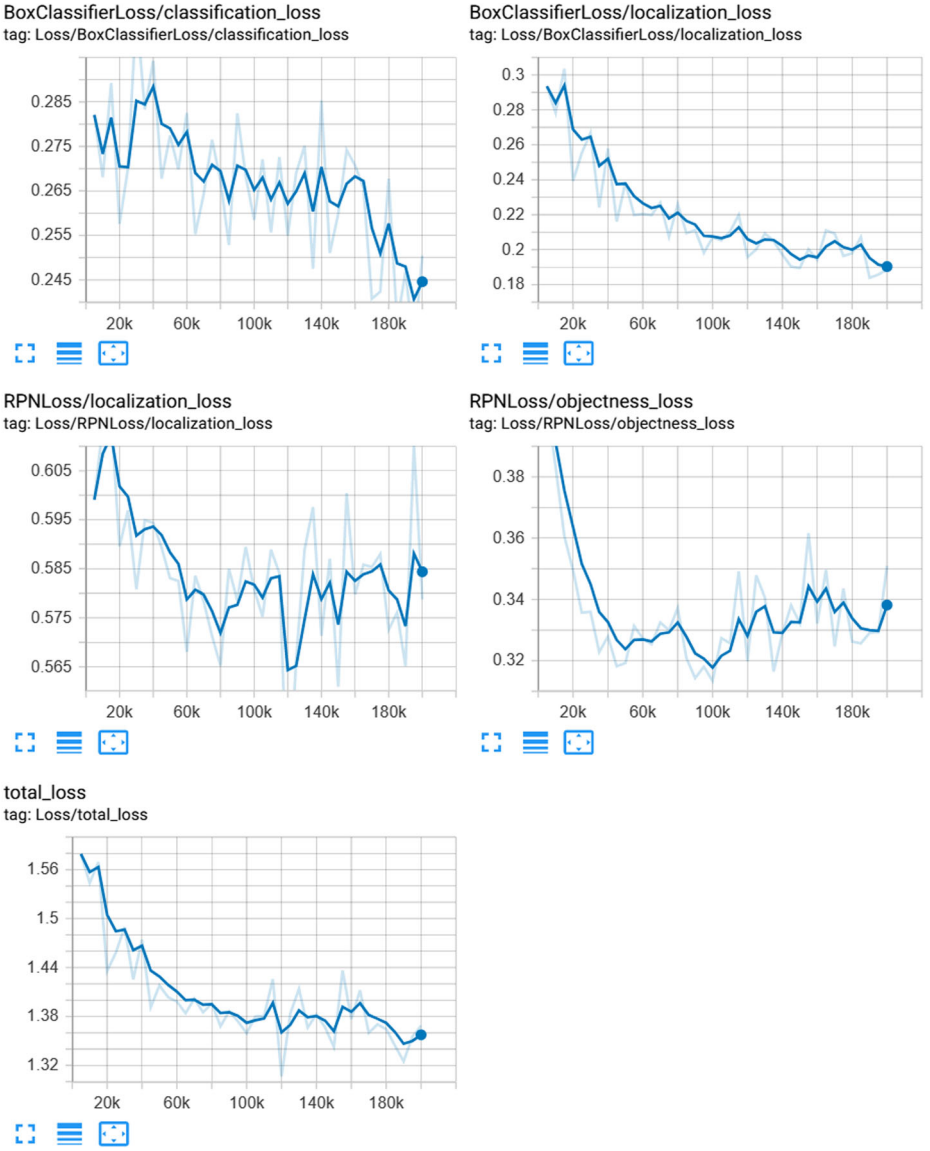


**Fig. 6** Losses of the re-trained Faster R-CNN model with 200k iterations, using a momentum optimizer with manual step learning rate

by four random subjects chosen at the very beginning of the evaluation process to generate the final results. These subjects are carefully chosen so that there are enough images for them from both VGGFace2 and YouTube Faces DB.

We first present the evaluation results produced by training on static images from VGGFace2 and image sequences from YouTube Faces DB in Figs. 9 and 10, which include models trained on static images only, image sequences only, same number of static images and image sequences, and all available images, respectively.

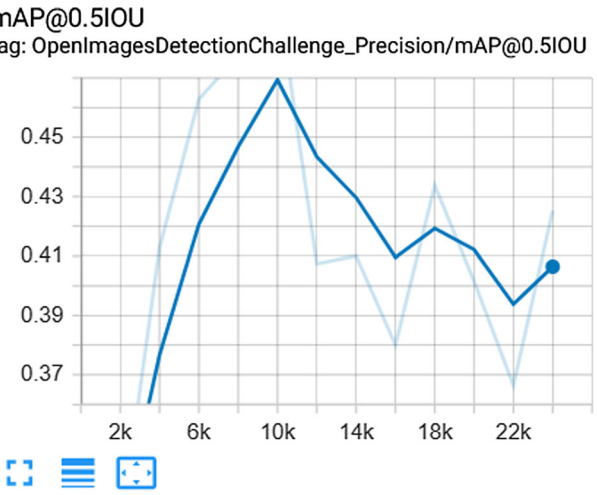


**Fig. 7** Accuracy of the re-trained SSD model with 20k+ iterations, using an RMSprop optimizer with exponential decay learning rate

Perhaps unsurprisingly, it is shown in Fig. 9 that the highest accuracy on static images is achieved by models trained on static images only and all available images. On the other hand, the model trained on image sequences alone yields a much lower accuracy, which could be because images provided by YouTube Faces DB are generally of low resolution and full of noises. In addition to image quality, YouTube Faces DB contains 1 to 6 videos per subject, which means that many images are taken from the same video. The result of which is that many images of the same subject appear to be somewhat similar, and thus could be redundant. However, this is not a problem when it comes to static images as most of them for the same subject are taken under various different conditions. The accuracy of the model trained on equal number of static images and image sequences in this case is somewhere in between, which is to be expected as we have slightly more static images than image sequences for this part of the evaluation. Therefore, by having the same number of both types of images, we have to remove some static images from the training dataset, which causes the accuracy to drop.

The evaluation results shown in Fig. 10 are likely to offer more insights into video surveillance scenarios since data captured by most surveillance cameras comes in the form of image sequences, not static images. This time, the model trained on image sequences outperforms the one trained on static images which could be due to various reasons. For example, although different images are used for training and testing, here we use images from the same set of videos for both purposes. Therefore, an image in the training dataset might be nearly identical to another image in the testing dataset without being exactly the same image, which leads to the improvement in accuracy for the image sequences model. Again, the model trained on all images achieves a promising accuracy, which is further improved by the one trained on equal number of both types of images for similar reasons as in the case of evaluation on static images.

As mentioned above, YouTube Faces DB has quite a few limitations which could introduce negative impact on our evaluation results. For example, YouTube Faces DB only provides a single video for some subjects, which is primarily why we use different image sequences from the same set of videos for both training and testing. Furthermore, image

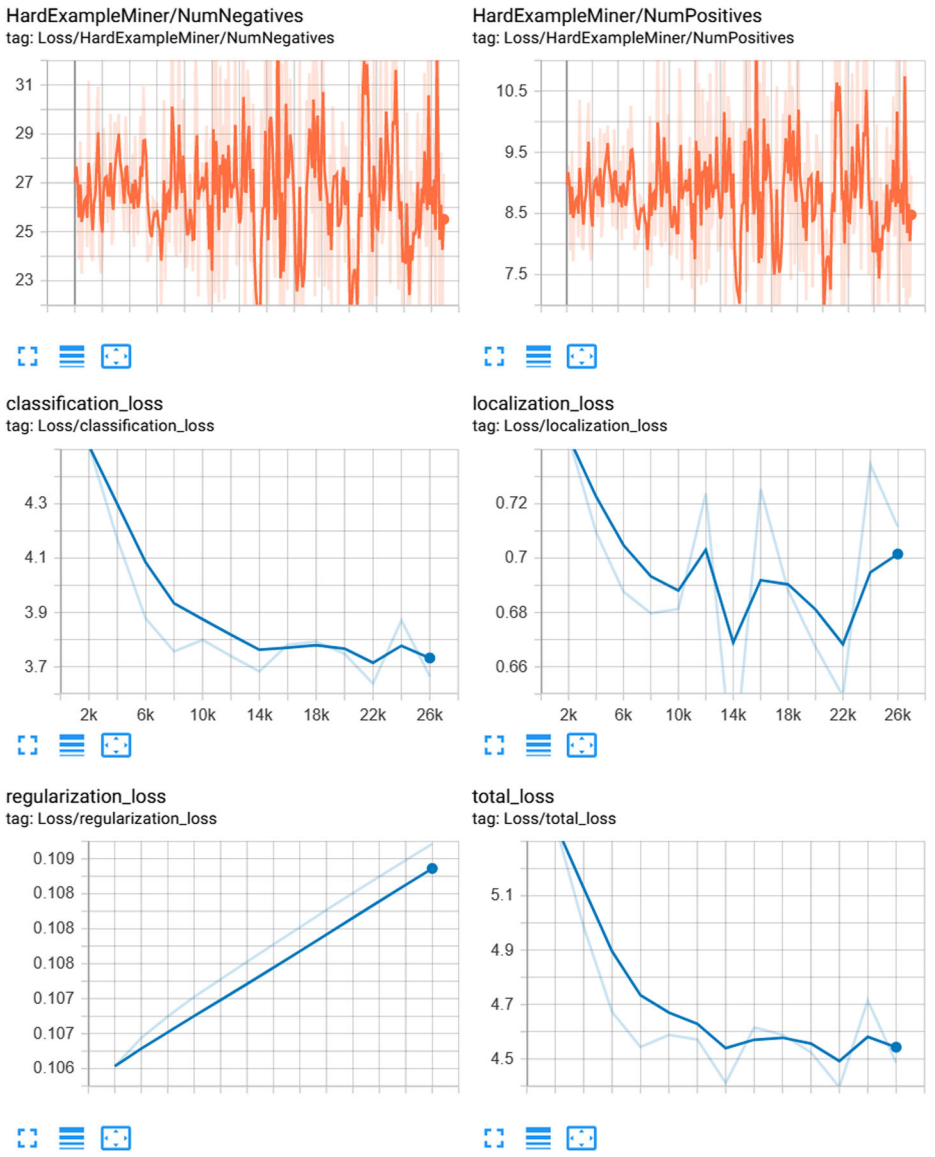


**Fig. 8** Losses of the re-trained SSD model with 20k+ iterations, using an RMSprop optimizer with exponential decay learning rate

sequences provided by YouTube Faces DB are generally of low quality, which makes it extremely difficult for models trained on high-quality static images to achieve satisfactory accuracies. Although images captured by surveillance cameras are in most cases of lower quality than static images, most modern surveillance cameras could arguably produce images of higher quality than the ones provided by YouTube Faces DB.

Therefore, we build another testing dataset ourselves with image sequences of higher quality taken from videos downloaded from YouTube. A new training dataset is also built

using a collection of completely different YouTube videos from the testing dataset. Consequently, we can see in Fig. 11 that the accuracy of the model trained on low-quality image sequences from YouTube Faces DB drops steeply from 88.9% to 37.5% compared to the one obtained by testing on YouTube Faces DB as well. Conversely, the model trained on static images from VGGFace2 yields a slightly higher accuracy, which could be mainly due to the improvement in image quality. Since image sequences from different videos are used for training and testing, the accuracy of the model trained on our own dataset is only leading by a very narrow margin compared to the one trained on static images. Again, we get slightly higher accuracy this time by training our model on the same number of static images and image sequences, and even higher accuracy with all available images. However, we have to bear in mind that the small improvement in accuracy is only made possible with more data and computation at the same time.

Hence, from this part of our experiments, we can conclude that when developing a practical video surveillance system, the best accuracy can be achieved by training the face recognition model on a mixture of static images and image sequences. Additionally, the ratio of images can usually be ignored, meaning that it is normally preferred to have more images in the training dataset without worrying if there are more static images than image sequences, or vice versa. However, if computational resources are at a premium, or it is the case that only static images or image sequences are available for training, it is generally acceptable to train the model on a single type of images without sacrificing too much accuracy. Note that this is only true when image sequences used for training are of relatively high quality.

## 5 Future work

In this paper, we have only discussed object detection methods designed to process static images. This is mainly because video object detection is a more challenging problem and existing methods for such a problem still have a lot of room for improvement in comparison. With that said, recent advances in video object detection such as [29, 41, 46] have shown that such methods are capable of achieving higher detection accuracies at real-time rates

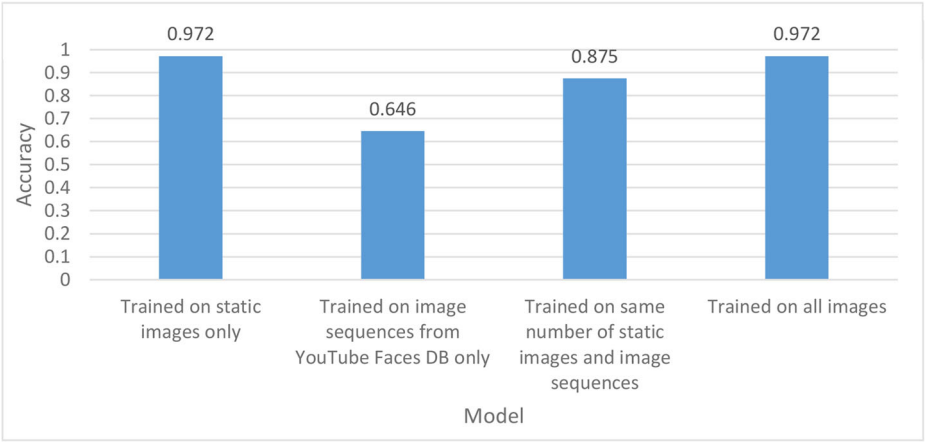


**Fig. 9** Average accuracy of various face recognition models tested on static images from VGGFace2

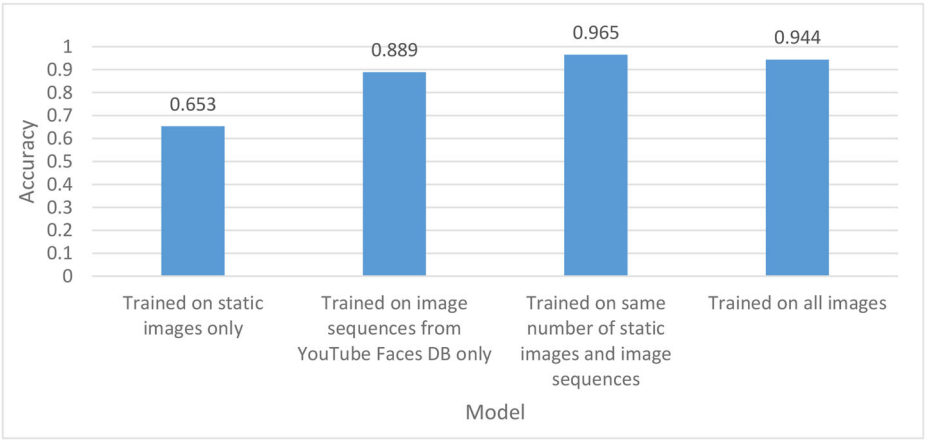


**Fig. 10** Average accuracy of various face recognition models tested on low-quality image sequences from YouTube Faces DB

and may be more robust to difficulties such as occlusions. Therefore, video object detection is suitable to be considered as part of the future improvements for the proposed video surveillance system.

Furthermore, it is discussed in papers including [3, 7, 42, 45] that it is possible to achieve impressive accuracies with video salient object detection (VSOD) based on the visual attention mechanism. Saliency models have been proved to be generally faster than traditional object detection models and require less annotation work to be done. However, it was also stated in [3] that we still lack a reliable and robust salient object detection algorithm which is capable of producing promising results in nearly all cases. This makes VSOD a possible improvement to the proposed system once better algorithms have been invented.

In addition to video object detection and video salient object detection, video processing frameworks such as Convolutional 3D (C3D) [18, 19, 39] could be adopted to extract

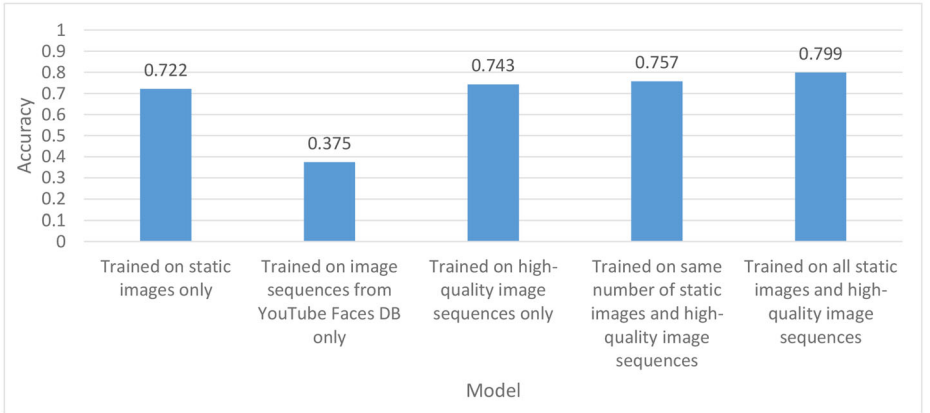


**Fig. 11** Average accuracy of various face recognition models tested on high-quality image sequences from videos downloaded from YouTube

spatiotemporal features from videos in an accurate and efficient way. According to experiment results reported in [39], C3D outperforms state-of-the-art methods on 4 different benchmarks while being easy to train and deploy. The fact that C3D is capable of using 3-dimensional convolutional networks (3D ConvNets) to perform accurate object and action recognition while being efficient and simple to use makes it a promising future improvement to the proposed system.

## 6 Conclusion

We have presented a comprehensive overview of some of the existing object detection and face recognition algorithms and reasons why they may or may not be preferable to video surveillance applications. Detailed comparisons have been made amongst some of the latest algorithms and these comparisons have been kept as fair as possible by making use of control variables such as training datasets and feature extractors. Steps for preparing the necessary datasets and training the models required are also illustrated. From the proposed end-to-end video surveillance system, we can see that it is possible to build such a system with both object detection and face recognition capabilities using deep learning methods.

By comparing accuracy and speed of the latest algorithms, we conclude that Faster R-CNN [32] with Inception ResNet V2 [37] is amongst the best options for achieving the highest object detection accuracy without paying too much attention to speed. Otherwise, if speed is of utmost importance, SSD [24] with MobileNet [14] makes a much faster model without sacrificing too much accuracy.

In terms of face recognition, FaceNet [34] achieves state-of-the-art accuracy whilst being fast enough for most real-time applications with MTCNN [44] used for face detection and alignment. Additionally, from our experiments, we can see that although the best face recognition accuracy is achieved by training on a mixture of as many static images and image sequences as possible, the accuracy would probably not suffer too much if only a single type of images are available for training. It is crucial, however, to train the model on high-quality image sequences, especially when images captured by the surveillance camera are of high quality as well.



## References

1. Abadi M, Agarwal A, Barham P, Brevdo E, Chen Z, Citro C, Corrado GS, Davis A, Dean J, Devin M, Ghemawat S, Goodfellow IJ, Harp A, Irving G, Isard M, Jia Y, Józefowicz R, Kaiser L, Kudlur M, Levenberg J, Mané D, Monga R, Moore S, Murray DG, Olah C, Schuster M, Shlens J, Steiner B, Sutskever I, Talwar K, Tucker PA, Vanhoucke V, Vasudevan V, Viégas FB, Vinyals O, Warden P, Wattenberg M, Wicke M, Yu Y, Zheng X (2016) Tensorflow: Large-scale machine learning on heterogeneous distributed systems. arXiv:1603.04467

2. Amos B, Ludwiczuk B, Satyanarayanan M (2016) Openface: A general-purpose face recognition library with mobile applications. Tech. rep., CMU-CS-16-118 CMU School of Computer Science
3. Borji A, Cheng MM, Hou Q, Jiang H, Li J (2019) Salient object detection: a survey. In: Computational visual media, vol 5, pp 117—-150
4. Cao Q, Shen L, Xie W, Parkhi OM, Zisserman A (2018) VGGFAce2: A dataset for recognising faces across pose and age. In: International conference on automatic face and gesture recognition
5. Dai J, Li Y, He K, Sun J (2016) R-FCN: Object detection via region-based fully convolutional networks. In: Lee DD, Sugiyama M, Luxburg UV, Guyon I, Garnett R (eds) Advances in neural information processing systems 29. Curran Associates, Inc., pp 379–387. http://papers.nips.cc/paper/6465-r-fcn-object-detection-via-region-based-fully-convolutional-networks.pdf
6. Elkin M (2020) Crime in England and wales: year ending September 2019. Office for National Statistics, Newport UK
7. Fan DP, Wang W, Cheng MM, Shen J (2019) Shifting more attention to video salient object detection. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)
8. Farrington DP, Gill M, Waples SJ, Argomaniz J (2007) The effects of closed-circuit television on crime: meta-analysis of an English national quasi-experimental multi-site evaluation. J Exp Criminol 3:21–38. https://doi.org/10.1007/s11292-007-9024-2
9. Girshick R (2015) Fast r-CNN. In: The IEEE international conference on computer vision (ICCV)
10. Girshick R, Donahue J, Darrell T, Malik J (2014) Rich feature hierarchies for accurate object detection and semantic segmentation. In: The IEEE conference on computer vision and pattern recognition (CVPR)
11. Gool LV, Williams CKI, John Winn AZ (2010) The pascal visual object classes (VOC) challenge. Int J Comput Vis 88:303–338
12. He K, Zhang X, Ren S, Sun J (2015) Spatial pyramid pooling in deep convolutional networks for visual recognition. IEEE Trans Pattern Anal Mach Intel 37(9):1904–1916
13. He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. In: The IEEE conference on computer vision and pattern recognition (CVPR)
14. Howard AG, Zhu M, Chen B, Kalenichenko D, Wang W, Weyand T, Andreetto M, Adam H (2017) MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. arXiv:1704.04861
15. Huang GB, Ramesh M, Berg T, Learned-Miller E (2007) Labeled faces in the wild: a database for studying face recognition in unconstrained environments. Tech. Rep. 07–49, university of massachusetts amherst
16. Huang J, Rathod V, Sun C, Zhu M, Korattikara A, Fathi A, Fischer I, Wojna Z, Song Y, Guadarrama S, Murphy K (2017) Speed/Accuracy Trade-Offs for modern convolutional object detectors. In: The IEEE conference on computer vision and pattern recognition (CVPR)
17. Ioffe S, Szegedy C (2015) Batch normalization: accelerating deep network training by reducing internal covariate shift. arXiv:1502.03167
18. Jia Y, Shelhamer E, Donahue J, Karayev S, Long J, Girshick R, Guadarrama S, Darrell T (2014) Caffe: Convolutional Architecture for Fast Feature Embedding. arXiv:1408.5093
19. Karpathy A, Toderici G, Shetty S, Leung T, Sukthankar R, Fei-Fei L (2014) Large-Scale Video classification with convolutional neural networks. In: 2014 IEEE Conference on computer vision and pattern recognition, pp 1725–1732
20. Kuznetsova A, Rom H, Alldrin N, Uijlings J, Krasin I, Pont-Tuset J, Kamali S, Popov S, Malloci M, Duerig T, Ferrari V (2018) The Open Images Dataset V4:, Unified image classification, object detection, and visual relationship detection at scale. arXiv:1811.00982
21. LeCun Y, Bengio Y, Hinton G (2015) Deep learning. In: Nature, vol 521, pp 436–444
22. Lin TY, Maire M, Belongie S, Bourdev L, Girshick R, Hays J, Perona P, Ramanan D, Zitnick CL, Dollá P. (2014) Microsoft COCO: common objects in context. In: Fleet D, Pajdla T, Schiele B, Tuytelaars T (eds) Computer vision – ECCV 2014. Springer International Publishing, Cham, pp 740–755
23. Liu L, Ouyang W, Wang X, Chen J, Liu X, Pietikäinen M (2020) Deep learning for generic object detection: a survey. In: International journal of computer vision, vol 128, pp 261–318
24. Liu W, Anguelov D, Erhan D, Szegedy C, Reed S, Fu CY, Berg AC (2016) SSD: Single shot MultiBox detector. In: Leibe B, Matas J, Sebe N, Welling M (eds) Computer vision – ECCV 2016. Springer International Publishing, Cham, pp 21–37
25. Lomonaco V CORe50. https://vlomonaco.github.io/core50/#differences (2019). Accessed: 15-09-2020
26. Lomonaco V, Maltoni D (2017) CORE50: A new dataset and benchmark for continuous object recognition. In: Levine S, Vanhoucke V, Goldberg K (eds) Proceedings of machine learning research, vol 78, pp 17–26

27. Lomonaco V, Maltoni D, Pellegrini L (2019) Fine-Grained Continual Learning. Computing Research Repository. arXiv:1907.03799
28. McCabe DL, Butterfield KD, Treviño LK (2017) Cheating in college: why students do it and what educators can do about it. The Johns Hopkins University Press, Baltimore
29. Meinhardt PBT, Leal-Taixe L (2019) Tracking without bells and whistles. In: Proceedings of the IEEE/CVF international conference on computer vision (ICCV)
30. Parkhi OM, Vedaldi A, Zisserman A (2015) Deep face recognition. In: Xie X, Jones MW, Tam GKL (eds) Proceedings of the British machine vision conference (BMVC). BMVA Press, pp 41.1–41.12. https://doi.org/10.5244/C.29.41
31. Redmon J, Divvala S, Girshick R, Farhadi A (2016) You only look once: unified, Real-Time object detection. In: The IEEE conference on computer vision and pattern recognition (CVPR)
32. Ren S, He K, Girshick R, Sun J (2015) Faster R-CNN: Towards real-time object detection with region proposal networks. In: Cortes C, Lawrence ND, Lee DD, Sugiyama M, Garnett R (eds) Advances in neural information processing systems 28. Curran Associates, Inc., pp 91–99. http://papers.nips.cc/paper/5638-faster-r-cnn-towards-real-time-object-detection-with-region-proposal-networks.pdf
33. Sandberg D Face Recognition using Tensorflow. https://github.com/davidsandberg/facenet (2018). Accessed: 12-04-2020
34. Schroff F, Dmitry Kalenichenko JP (2015) Facenet: A Unified Embedding for Face Recognition and Clustering. In: The IEEE conference on computer vision and pattern recognition (CVPR)
35. Simonyan K, Zisserman A (2015) Very deep convolutional networks for Large-Scale image recognition. In: International conference on learning representations (ICLR)
36. Sun Y, Xiaogang Wang XT (2015) Deeply learned face representations are sparse, selective, and robust. In: The IEEE conference on computer vision and pattern recognition (CVPR)
37. Szegedy C, Ioffe S, Vanhoucke V, Alemi A (2017) Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning. https://www.aaai.org/ocs/index.php/AAAI/AAAI17/paper/view/14806/14311
38. Taigman Y, Yang M, Ranzato M, Wolf L (2014) Deepface: Closing the gap to human-level performance in face verification. In: The IEEE conference on computer vision and pattern recognition (CVPR)
39. Tran D, Bourdev L, Fergus R, Torresani L, Paluri M (2015) Learning spatiotemporal features with 3D convolutional networks. In: 2015 IEEE International conference on computer vision (ICCV), pp 4489–4497
40. Uijlings JRR, van de Sande KEA, Gevers T, Smeulders AWM (2013) Selective search for object recognition. Int J Comput Vis 104:154–171
41. Wang S, Zhou Y, Yan J, Deng Z (2018) Fully Motion-Aware network for video object detection. In: Proceedings of the European conference on computer vision (ECCV)
42. Wang W, Zhao S, Shen J, Hoi SCH, Borji A (2019) Salient object detection with pyramid attention and salient edges. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)
43. Wolf L, Hassner T, Maoz I (2011) Face recognition in unconstrained videos with matched background similarity. In: The IEEE conference on computer vision and pattern recognition (CVPR)
44. Zhang K, Zhang Z, Li Z, Qiao Y (2016) Joint face detection and alignment using multitask cascaded convolutional networks. IEEE Signal Process Lett 23(10):1499–1503. https://doi.org/10.1109/LSP.2016.2603342
45. Zhao JX, Liu JJ, Fan DP, Cao Y, Yang JF, Cheng MM (2019) EGNEt: Edge guidance network for salient object detection. In: Proceedings of the IEEE/CVF international conference on computer vision (ICCV)
46. Zhu X, Dai J, Yuan L, Wei Y (2018) Towards high performance video object detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)