Overhead Person Detection on Mobile Platforms

Tong Siew Wen
School of Engineering
Monash University Malaysia
46150, Selangor, Malaysia
ston0005@student.monash.edu

Dr. Soon Foo Chong
School of Engineering
Monash University Malaysia
46150, Selangor, Malaysia
soon.foochong@monash.edu

Dr. Shahnewaz Chowdhury
Distributions Products Department
ELID Sdn. Bhd.
46100, Selangor, Malaysia
shahnewaz@elid.com

***Abstract*— Person detection is the core element in visual surveillance. This research focuses on the training of an existing object detection model, MobileNetV2 SSD, to perform detection overhead person on mobile platforms in unconstrained environments. The model is trained using a self-collected dataset that undergoes data augmentations via traditional augmentation methods and via StyleGAN2-ADA model. Results show that traditional augmentation techniques help in improving the model performance, but not StyleGAN2-ADA although it is able to generate realistic images. Transfer learning is also applied by freezing the pretrained model layers during training, and it is proved to be able to increase the training speed without affecting the model performance. A precision of over 90% and recall of almost 70% is achieved when tested in the target environment at a deployment speed greater than 30 frame per second.**

Keywords*— person detection; overhead view person; mobile platforms; real time systems; data augmentations; Generative Adversesial Network; transfer learning.

I. Introduction

The COVID19 pandemic has brought a huge impact to humans' daily life. To slow down the spread of the virus, the government has enforced several standard operating procedures (SOP), one of it being the restriction in the human capacity in premises [1]. A cost-effective solution to replace manpower is to utilize artificial intelligence to perform people counting. Other than aiding companies and organizations in adhering the SOP [2], people counting could also help to detect tailgating. Tailgating is a loophole in access control, allowing unauthorized people to enter a restricted area. Physical access systems that solely depend on sensors aren't sufficient, as they do not monitor the number of people entering and leaving the facilities [3]. Therefore, a real-time video surveillance system is designed to tackle the problems mentioned above. The video surveillance system involves person detection, which is the core element, followed by person tracking. This study focuses on the person detection of video surveillance system from an overhead view. The reason overhead view is chosen because it solves the occlusion problem, thus prevents missed detections which cause inaccurate people counts and failure in detecting tailgating. Compared to frontal view, there are lesser features in the overhead view which needs to be tackled [4].

Since the application of person detection is gaining more attention especially in video surveillance and access control, a great amount of research is done. However, there are still challenges exist such as a large variety of human features due to change in pose, illumination, occlusion, etc. [5]. The common input used in person detection approaches are depth images and RGB images. The authors in [5, 6] utilized the depth images. The architecture of DPD Net is presented in [5], i.e., two fully convolutional encoder-decoder blocks built with residual layers, where the encoder proposes a pixel-wise confidence map to be refined by the decoder. DPD Net achieved an outstanding performance of 99% accuracy. However, for more complicated datasets, the model requires fine-tuning using 80% of that dataset, before it is able to achieve the same accuracy [5]. The authors in [6] proposed a two-stage method, where the first stage compares the geometric property of a human head with the depth images to generate detection proposals, then the second stage utilizes a convolutional network and a SVM classifier to extract features and validate the proposed detections. This approach achieved an average miss rate of 0.046 and false-positive-per-image of 20.48 which is not satisfactory [6]. This shows that handcrafted approaches are not as effective as deep learning methods for overhead person detection. As the aforementioned surveillance system will perform person tracking after person detection, RGB images will be preferable so that each person can be identified uniquely to ease the tracking process. Authors in [4] proposed to use Aggregated Channel Features with Adaboost Classifier that takes RGB overhead human images as input. To enable real-time performance, the detection scale and camera resolution are optimized, while utilizing asynchronous multithreaded design for the deployment of the model coupled with reuse of memory. In good lighting and static background, the model achieved an accuracy of 78%, but only achieves 50% accuracy otherwise. Also, the inference time grows with the number of people in the scene [4]. This is not desirable as a good model should have its inference time independent from the number of detections [5].

II. Overview of Proposed Approach

In this project, Single Shot Detector (SSD) is chosen for its convenience as the whole pipeline from training to compiling to Edge Tensor Processing Unit (TPU) file is supported in TensorFlow Object Detection API [7]. Different from the model "You Only Look Once" (YOLO) that handcrafted its convolutional neural network (CNN), SSD uses VGG16 [8], an existing CNN, as its 'backbone' with the last layer stripped off and replaced with additional convolutional layers. Inspired by anchor boxes in Faster R-CNN, SSD not only separates the image into grids, but also introduces default boxes to perform prediction. This effectively produces tight bounding boxes that better match the shape of the object. Even better, SSD suggests performing predictions at multiple layers so that the bounding box will have a variety of sizes and aspect ratios to better fit the object. SSD outperforms YOLO in terms of accuracy but runs slightly slower. On VOC2007 test, SSD achieved 74.3% mean Average Precision (mAP) at 46 frame per second (FPS) [9]. As this application is designed as an offline standalone device, the model requires a light-weight backbone CNN model for SSD. Hence, MobileNetV2 is chosen as it requires much lesser memory and computational power with satisfying accuracy [10]. MobileNetV2 achieves this by introducing the idea of bottleneck layers blocks which consists of pointwise convolution as expansion layers, depthwise convolution as feature extractors, and another pointwise convolution as compressor. Different from residual blocks in Resnet, the shortcuts are placed in between bottleneck layers to tackle the vanishing gradient problem. With bottleneck layers blocks, the model parameters and the memory usage are efficiently reduced without affecting its performance [10]. The architecture of MobileNetV2 SSD is illustrated in Fig. 1. In the COCO test-dev2015 detection, MobileNetV2 SSD achieved a similar performance as the original SSD, while being able to reduce the number of parameters by a factor of 20. Google Coral, which is an Edge TPU device will be used to accelerate the deployment speed to enable the model to run in real time.

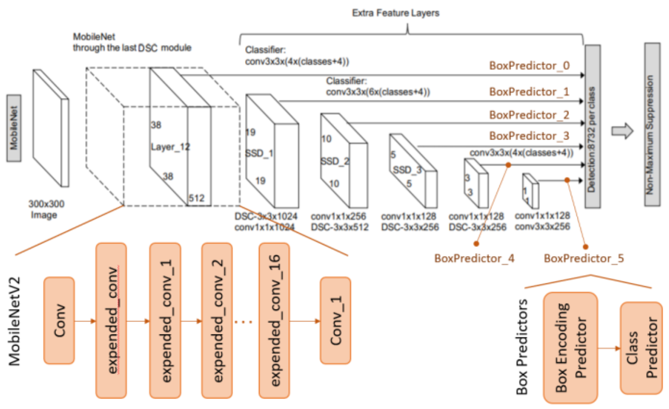

Fig. 1. *Architecture of MobileNetV2 SSD. Expended_conv layers are the bottleneck layer blocks of the MobileNetV2 model. Modified from [11].*

To have a good-performing deep learning model, it is important to have high-quality training that is sufficiently large and varied [12]. At the initial stage of this project, it is realized that the size and variation of the training dataset are not sufficient, causing a bottleneck in the performance. Hence, image augmentations will be performed on the training data to tackle this problem. As traditional augmentation methods have limited invariances, Generative Adversarial Networks (GAN) are also implemented to introduce more variations to the training dataset [13]. GAN has been a very active topic, as it is unsupervised and implicitly learns an underlying distribution. Many GAN models are presented to perform image augmentations, such as Data Augmentation GAN (DAGAN) and StyleGAN2-ADA. DAGAN differentiates itself from Conditional GANs as class information is not given. The generator is a combination UNet and ResNet which takes in low dimensional information encoded from the original image concatenated with a noise vector. The discriminator follows the architecture of DenseNet but replaced batch normalization with layer normalization. The discriminator receives two real images. This is to ensure that DAGAN generates images that are related but different from their original images, and able to generalize to all classes [13]. DAGAN is proved able to generate realistic images that are different from the original real images and still be in the same class [13]. On the other hand, StyleGAN2-ADA is an improved version of StyleGAN, and aims to generate realistic images despite having a small training dataset [14]. StyleGAN2-ADA is an extension to the progressive GAN architecture, but nearest neighbor sampling is replaced with bilinear sampling. Instead of using a point from a latent space as input, StyleGAN2-ADA uses two sources of randomness to generate images, which are the standalone mapping network and noise vectors. The mapping network encodes the input latent vector into an intermediate latent space, while the noise vectors are just a single-channel picture with uncorrelated Gaussian noise to induce stochastic details without affecting the overall perception of the image [15]. To solve the overfitting issue of the discriminator for small training dataset size, Stochastic Discriminator Augmentation and two plausible overfitting heuristics are introduced [14]. StyleGAN2-ADA is proved to generate realistic images in data-limited scenarios with controllable variation and style transfer [14, 15]. Hence, DAGAN and StyleGAN2-ADA are attempted to perform image augmentations on the training data.

It is also desired to have the object detection model to be deployed as soon as possible on site. Thus, this project also aims to implement transfer learning so as to minimize the training duration. Transfer Learning transfers knowledge from a task that is related and learned by the model. It is believed that transfer learning is most effective when the features learned are general so that the knowledge can be transferred and improved to solve a niche problem [16]. This assumption is normally for the first few layers, which act like Gabor filters or color globs [16]. For this project, the pre-trained MobileNetV2 SSD model is trained on the COCO dataset, a generic dataset with 2.5 million labeled instances in 328k images over 91 objects types, hence transfer learning is suitable to be implemented. An easy and commonly used method to implement transfer learning is to freeze the initial layers of the pre-trained model and train the model with the dataset of the target task. This method is proven to be able to shorten the training duration while achieving a high accuracy in Alzheimer’s neurological disorder detection [17], thoraco-abdominal lymph node (LN) detection and interstitial lung disease (ILD) detection [18].

The contribution of this project are (i) presenting a pipeline from data processing to deployment to train a reliable overhead person detection model for mobile platforms in unconstrained environments, (ii) study the effect of data distribution on model performance, (iii) study image augmentation via GAN models, (iv) presenting the effects performing data augmentation via traditional techniques and GAN models on the model performance, and (v) incorporating transfer learning to shorten the training duration without affecting the model performance.

III. Methodology

A. *Data preparation for object detection model*

Overhead person images can be self-collected and gathered through online datasets by other researchers. An overhead camera is programmed to capture images whenever it detects motion. This can be done via background segmentation for every frame and capture an image if the foreground threshold is exceeded. Then, bounding boxes are labeled via the LabelImg application. For some online datasets, the annotations are processed and converted to the required format. For self-collected images, images that are collected with the same background will be grouped to be a dataset. The images are then split into training and validation dataset. To ensure fair comparison, the EDOB is chosen to be target environment for the model to be deployed, hence the validation images will be from the EDOB dataset for all experiments.

If the GAN models can generate realistic images that are different from the original set of training images, the generated human instances will be incorporated into the original training images. In other words, the original image will have one or more original human instances, and an additional synthetic human generated by the GAN model. The bounding boxes annotations of the synthetic human instances need to be added. Then, the training images are augmented with traditional augmentation techniques, which includes geometric distortions such as rotation, flip and changing the image aspect ratio, color distortions such as changing image hue, saturation, brightness and contrast, and adding disturbance to the image such as salt and pepper noise, Gaussian blur, Random Erase and grid masks. One or more of these traditional techniques are grouped to be augmented combinations to be applied to an image. Each augmentation combination creates a unique augmented image, hence applying k augmentation combinations to one image effectively increase the dataset size by a factor of k. After that, both training and validation dataset with their corresponding images and bounding box annotations are converted to tfrecords to be compatible for training in TensorFlow API.

B. *Image Augmentation via Generative Adversarial Networks*

Before using GAN to augment images, the GAN model needs to be trained. Referring to previous research, the target is always in the middle and occupies most of the image, which is not the case for the images collected for object detection. Hence, the human instances are cropped out using the bounding box annotations. To have some background information, the cropped images are a little wider than the bounding boxes. Then, images are filtered to keep only those that are not occluded and resized to 256x256 to ease the learning by the GAN model. As DAGAN is coded for multiclass problems, code for data preparation is modified as this project is a single class problem. Following the guideline in their GitHub page [19], the number of generator inner layers is chosen to be 3, the number of discriminator inner layers is chosen to be 5, the z dimension is chosen to be 100 and the dropout rate is chosen to be 0.5. On the other hand, StyleGAN2-ADA does not require any modification of the code. Following the guideline in their GitHub [20], mirroring the images is enabled in the x-direction, while the discriminator augmentation pipeline is chosen to be pixel blitting, geometry, and color transformation. The StyleGAN model is trained from the pre-trained model on the FFHQ dataset. The quality of the images created by the models are visually monitored, and the training is halted once the images generated are satisfactory.

C. *Training and validation of the object detection model*

After setting up a docker environment that consists of TensorFlow Object Detection API, the checkpoint and config file of the pre-trained MobileNetV2 SSD model, the training can start using the tfrecord files that was prepared in the previous step. To perform transfer learning, the layers to be frozen need to be indicated in the config file. Other than adjusting the batch size to suit the GPU capability, other model hyperparameters are remained to be the same as the pre-trained MobileNetV2 SSD model. During training, the performance of the model can be monitored via Tensorboard, which shows the mean Average Precision (mAP), Average Recall (AR), training losses, and the visualization of the model performance on the validation images. The precision is measured using mean Average Precision for IoU greater than 0.5 (mAP@IOU50 because 0.5 IoU is accurate enough to be used for the subsequent stage, i.e. tracking of each detected individual). On the other hand, recall is measured using Average Recall given 10 detections per image (AR10) as from the images collected, the number of human instances is from 1 to 20, hence AR10 is the most suitable choice compared to AR1 and AR100.

If the trained model is not satisfactory, then the model will need to be retrained, after changing corresponding model hyperparameters, increasing or augmenting the training images. This process of preparation, training and testing will need to be repeated many times until the performance of the model is satisfactory i.e., able to give have accuracy and recall of 80% on unseen environments. If the trained model is able to achieve the required performance, it is also desired to minimize the number of training images required to achieve similar performance. This is because collecting and labelling images needs to be performed at every site where the model will be deployed, which requires great effort, hence minimizing the number of training images required will also minimize the time and effort required before the model can be deployed.

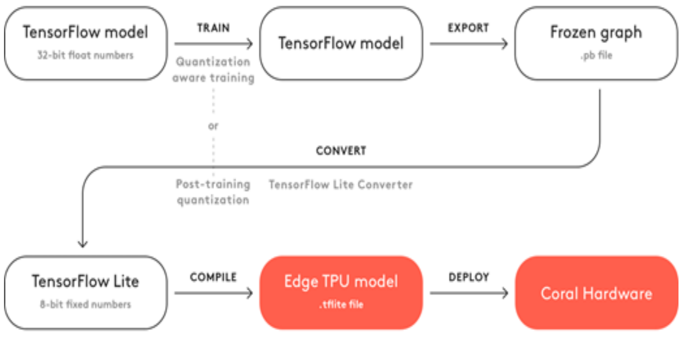

Fig. 2. *The process to compiling a Edge TPU model. Adapted from* [21].

Once the model is successfully optimized in terms of performance and time required for the model to be ready to be deployed, it needs to be compiled to an Edge TPU model following the procedure illustrated in Fig. 2. First, the trained model will need to be converted to a frozen graph, quantized to a TensorFlow Lite model with an 8-bit fixed number, then compiled into an Edge TPU model. Fortunately, this process is automated in TensorFlow API, hence all that needs to be done is to run the necessary scripts. The Edge TPU model will then need to be deployed on the Google Coral and tested real-time on the Raspberry Pi overhead camera, to ensure that the real-time performance of the model is satisfactory.

IV. Results and Discussion

A. Datasets collected

A total of 7984 images with 14245 annotations are collected and labeled. Overhead cameras are located at different locations at ELID to collect images. The image resolutions of the self-collected images are 640x480 and 1280x960 where the former is two-third of total image collected. The camera gain is set to 100, camera exposure is set to 6 and the image saturation is increased to 80, while the other settings are remained as their default values. All images collected in ELID are also self-labeled and require great amount of time and effort, hence the number of self-collected images is also limited. The number of images and annotations of self-collected datasets are summarized in Table I.

TABLE I. Self-collected datasets

Dataset Name	Total images	Total annotations
ELID Cafeteria (EC)	1310	2408
ELID DPD Entrance (EDE)	2225	2921
ELID DPD Indoor 1 (EDI1)	90	148
ELID DPD Indoor 2 (EDI2)	109	282
ELID DPD Indoor 3 (EDI3)	323	1052
ELID DPD Outdoor Bright (EDOB)	1049	1536
ELID DPD Outdoor Dark (EDOD)	754	1227

The data collection via online resources is harder than expected. This is because most of the online datasets for person detection are of frontal views, such as Caltech Pedestrian Dataset [22] and INRIA Person dataset [23]. Online datasets that are collected are the Top View Person Reidentification dataset (TVPR) [24] and Top View Multi-Person Cafeteria (TVMPC) [25]. The TVPR dataset, consists of 640x480 RGB-D top view human images collected using Asus Xtion Pro Live, while the TVPMC consists of 1280x960 RGB images. Since the TVPR dataset is meant for person reidentification, there is no bounding box annotations. The Top View Multiple Person Cafeteria (TVMPC) dataset [25] is also not for person detection, but for person tracking purposes. Although the bounding boxes are labeled, the bounding boxes are not accurate and there are some missed human instances that are not labeled. Hence, bounding boxes are added via LabelImg application for both datasets. As the images from the online datasets are taken at a much higher FPS, images are filtered to prevent duplication of similar images, causing the model to overfit to these images. The number of images and annotations of self-collected datasets are summarized below in Table II.

TABLE II. Online datasets collected

Dataset Name	Total images	Total annotations
TVPR [24]	1893	1893
TVMPC [25]	231	2778

B. Image augmentation via Generative Adversarial Networks

The EDE dataset and the EDOB dataset are used to train the GAN models, where the details are shown in Table III.

TABLE III. Details of the Training data for the GAN models

Origin dataset	Number of human instances
EDE dataset	1724
EDOB dataset	609

1) Data Augmentation GAN (DAGAN)

The authors included the creation of visual outputs of the DAGAN model during training. In the visual output images, the first column is the original image, and the second column onwards are the augmented images that are generated by the DAGAN model that given the original image in the same row is inputted. A DAGAN model is trained using cropped images from the EDE dataset for continuously two days on the GPU using the same settings described in the paper. A portion of the visual output of the DAGAN model trained on cropped images from the EDE dataset is as shown in Fig. 3.

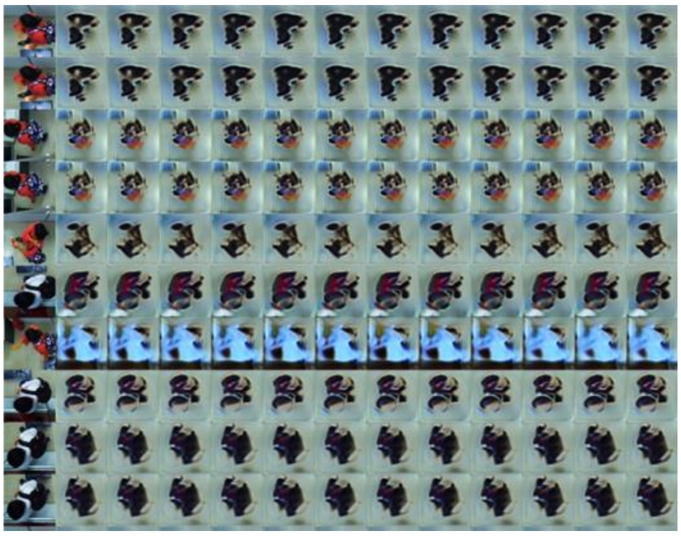

Fig. 3. Portion of the visual output of the DAGAN model

The results are barely acceptable, as the generated images do not resemble overhead humans and do not relates with the original image in any way. Also, it can be observed that mode collapse [26] has occurred as the generated images is the same regardless of the noice vector inputted to the model as shown across the column in the visual output in Fig. 3. The deduction made for this failure is that overhead person images are much complicated than the datasets used in the research paper, which includes the Omniglot dataset, the EMNIST dataset, and the VGG-Face dataset. For instance, the VGG-Face dataset has the face features at fixed locations, e.g., noses at the middle of the image and mouth below it. In contrast, the overhead human from the EDE dataset images have different orientations, e.g., the head is not always at the middle, the human is facing at different directions, the hand positions are varied across images. Hence, it is much more challenging for DAGAN to learn the data distribution and interpolate from the learned distribution to generate images. Based on this deduction, more convolutional layers are added to the DAGAN model, in order to help the DAGAN model to learn the complicated features and improve its performance.

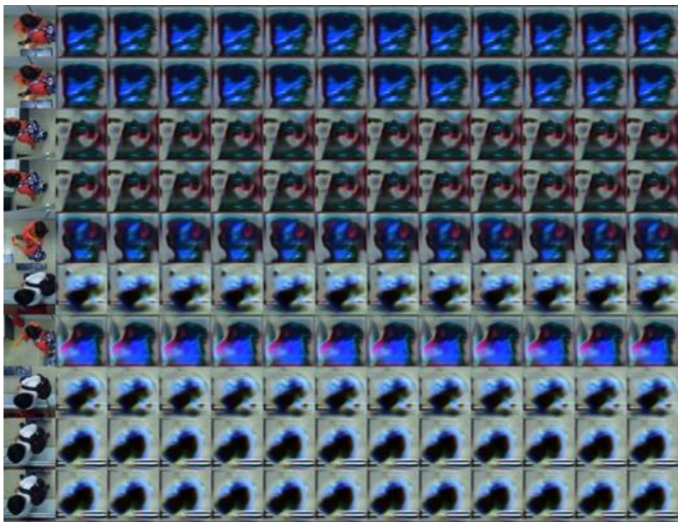
Fig. 4. Portion of the visual output of the DAGAN model with more convolutional layers.

As shown on the visual output after retraining of the DAGAN model with additional convolutional layers in Fig. 4, adding convolutional layers did not help the DAGAN model to learn the features better. This might reflect that DAGAN doesn't perform well in single class applications. Another possible reason for the poor performance is that the dataset size is too small for the DAGAN model to be able to learn and interpolate the data distribution. Compared to Omniglot dataset, the EMNIST dataset, and the VGG-Face dataset that has 32 thousand to 2.6 billion images, the overhead human dataset prepared from EDE dataset has only 1724 images. Hence, it can be foreseen that the performance of the DAGAN model on EDOB dataset will not achieve satisfactory performance. Due to the underperforming of the DAGAN model in augmenting overhead human instances, the generated human instances will be of low quality which will affect the performance of the object detection model. Thus, it is decided not to use in augmenting the human instances and to include them in the training of the object detection model.

2) *StyleGAN2-ADA*

Similar to DAGAN, the authors of StyleGAN2-ADA also included the creation of visual outputs of the DAGAN model in their code. The visual output is a photo college of 1024 generated images via interpolation of the learned data distribution. A StyleGAN2-ADA model is trained using cropped images from the EDE dataset for continuously four hours on the GPU using the same settings described in the paper. As shown in the visual output of the model is as shown in Fig. 5, although the training time of the StyleGAN2-ADA model is shorter than the training time of the DAGAN model, the performance of the StyleGAN2-ADA model is much better. A great contribution goes to the model initialization using the weights of the pre-trained StyleGAN2-ADA model trained on the FFHQ dataset. If the StyleGAN2-ADA model is trained from scratch with randomly initialized weights, the training duration required to achieve a similar performance will be much longer. Aside from shorter training time, there is also no sight of mode collapse, as all the generated images are different from each other. The generated human instances also look realistic, and when compared to the original images, it is observed that only small details like the hand or leg postures and hairstyles are changed. This result is exactly what is desired, as it performed augmentations that are not possible via traditional augmentation techniques.

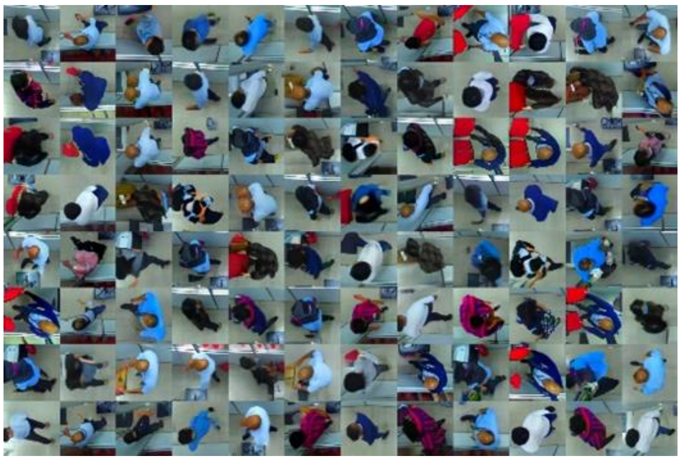
Fig. 5. Portion of the generated images by the StyleGAN2-ADA model trained on cropped images from EDE dataset.

However, there are some generated images that are imperfect, e.g., having no head or two heads, impossible head and limbs posture, disjointed limbs, etc. Some examples can also be found in Fig. 5. Among all defects, the most common defect is having two heads. This is out of expectation, as all human instances in the training images only have one head. The actual reason for this flaw is not known, but a possible reason could be the varied head positions in the training images which might have caused confusion to the model. As the StyleGAN2-ADA model is able to generate realistic images of overhead human instances, it will be used to augment human instances in the training images of the object detection model as described in the methodology.

C. *Performance of the object detection model*

1) *Effect of data distribution*

Each dataset represents a different data distribution due to different background contexts of the training images. Table IV shows the model performance, i.e., the precision (mAP) and the recall (AR) of the model when the model is trained using training datasets with different data distributions.

Initially, since the EDOB dataset is chosen to act as the target environment, it is used as the validation dataset while all remaining images collected are used as training images, as shown in Experiment 1. Augmentation combinations are applied ten times on the training set to increase the training set size and variation by a factor of 10. However, the performance is not satisfactory, especially the Average Recall which is only around 55% which indicates only half of the human instances will be detected by the model. This is because the data distribution of the training dataset is different from the data distribution of the validation dataset, due to not having a large and varied enough training set that can represent a more general data distribution.

TABLE IV. MODEL PERFORMANCE WHEN TRAINED WITH DATASETS OF DIFFERENT DATA DISTRIBUTIONS.

No.	Experiment settings		Model Performance
	Training set	Validation set	
1	All datasets except the EDOB dataset (87311 images, 144572 annotations)	EDOB dataset (1049 images, 1536 annotations)	mAP: 78.2% AR: 54.87%
2	All datasets with 500 images from the EDOB dataset added (87361 images, 145560 annotations)	EDOB dataset (849 images, 1297 annotations)	mAP: 75.3% AR: 53.73%
3	500 images from the EDOB dataset (738 annotations)	EDOB dataset (1049 images, 1536 annotations)	mAP: 97.8% AR: 74.29%

Since it is not easy to gather a great number of images for overhead images due to massive labelling effort, it is decided to include some images from the target environment in the training set in Experiment 2, so that the training set has a greater overlap in the data distribution with the target environment. The training set is applied with ten augmentation combinations, then 500 images that are randomly selected from the EDOB dataset (representing the target environment) are added to the training set. However, no significant improvement is observed. A possible reason is that 500 is a relatively small number compared to the training set, hence it will not affect the performance by a great amount.

In order to determine whether having a small portion of images from the target environment in the training dataset is the reason for the insignificant improvement in Experiment 2, Experiment 3 is conducted. The training set consists of only the 500 images from the EDOB dataset (representing the target environment), and the model is validated using the remaining images from the EDOB dataset. As expected, since the training and validation set are of the same environment, the performance is much better as compared to Experiment 1 and Experiment 2, with a huge increment of around 20% in both precision and recall.

2) *Effect of data augmentation*

Although the model trained in Experiment 3 is able to achieve high precision and recall, it is not robust to changes in the target environment e.g., changes in the floor color/pattern or addition of new objects to the background. Therefore, data augmentation should be heavily applied to the training set to prevent the model from overfitting to the background context and hence ensuring the robustness of the model towards changes in the background during deployment. Another reason to study the effect of augmentation is to find the optimum augmentation setting that minimizes the number of training images required while still achieving similar performance as the model in Experiment 3 in Table IV.

The training images from EDOB dataset are augmented using traditional augmentation techniques as well as using StyleGAN2_ADA in different settings (see Table V). To validate the selected setting is optimal to any datasets, the experiment is repeated using the EDE dataset in Table VI. The validation set for the experiments in Table V consists of 849 images from the EDOB dataset (1242 annotations), while the validation set for the experiments in Table VI consists of 445 images from the EDE dataset (597 annotations).

TABLE V. MODEL PERFORMANCE WHEN TRAINED WITH DATASETS ON DIFFERENT AUGMENTATION SETTINGS ON EDOB DATASET.

No.	Raw images	Augmentation settings	Model Performance
4	100	• 5 augmentation combinations per image	mAP: 86.8% AR: 56.15%
5	100	• 40 augmentation combinations per image	mAP: 92.3% AR: 60.56%
6	200	• 40 augmentation combinations per image	mAP: 97.1% AR: 69.75%
7	200	• 40 augmentation combinations per image • Synthetic instances via StyleGAN2-ADA added	mAP: 96.1% AR: 44.64%

TABLE VI. MODEL PERFORMANCE WHEN TRAINED WITH DATASETS ON DIFFERENT AUGMENTATION SETTINGS ON EDE DATASET.

No.	Raw images	Augmentation settings	Model Performance
9	1780	• No augmentation applied	mAP: 98.8% AR: 82.45%
10	200	• 40 augmentation combinations per image	mAP: 93.9% AR: 70.26%
11	200	• 40 augmentation combinations per image • Synthetic instances via StyleGAN2-ADA added	mAP: 95.3% AR: 43.77%

Since augmentations can generate new samples that are different from their original image, it is believed lesser raw images need to be collected to achieve a similar performance as a model trained with more original images. Compared to Experiment 3, the number of raw images is reduced to 100 in Experiment 4. Each training image is applied five augmentation combinations, resulting in the same number of training images with Experiment 3 i.e., 500 training images. The performance drops 11% in precision and 18% in recall. This shows that the model is underfitting due to insufficient training data. Hence in Experiment 5, the number of augmentation combinations applied is increased from 5 to 40, so that the number of training images is increased. Although the number of augmentation combinations applied is increased by a factor of eight, the improvement is small, i.e., only an increment of around 5% for both precision and recall. This is because the changes made by traditional augmentation techniques are limited. In other words, the posture, clothing and hairstyle of the human instances in the augmented images remain the same as the raw images. The trained model has not learnt with enough human instances, causing it to not perform well in detecting human instances with unseen human features. Thus, the number of raw images is increased to 200 while maintaining the number of augmentation combinations applied in Experiment 5. The performance of the model is improved, where the precision increased by 5% while the recall increased by 9%.

Although traditional augmentation techniques improve the model performance and reduce the number of raw images required for training, augmentation via StyleGAN2-ADA did not achieve similar results. Instead, incorporating generated human instances by StyleGAN2-ADA into the raw images caused a huge drop in recall. This might be due to the method of these synthetic human instances are added, as there are obvious boundaries between the synthetic human instances with the background image due to background mismatch between the raw image and the generated image. This might have caused the model to take the 'shortcut' by detecting humans from detecting sudden changes in the images which is the boundary of the pasted human instance. Another possible cause is that the synthetic human instances are very distinct from the real human instances in the perspective of the object detection model. It is found in previous researches that CNN models are biased towards texture compared to the shape when learning the features of an object [27]. This is proved as there are adversarial attacks which alters the image in such a way that it is undisguisable in human's eye, but could alter the output the CNN model in an unexpected way [28]. Thus, these synthetic human instances might be considered as hard examples to the model, causing the model trying hard to learn to detect them. These two possible reasons will cause the divergence of the model and the reduction of the model performance.

Besides, augmentation via StyleGAN2-ADA is also not very practical. This is because the StyleGAN2-ADA model needs to be trained every time the background context of the target environment changes. Two problems will arise such as the increment of training duration and insufficiency of training data. The training of the StyleGAN2-ADA took around 4 to 6 hours to be able to generate images that are realistic and usable, increasing the total duration required before the model can be deployed. Also, in the actual application, only around 200 raw images will be collected, indicating that the number of human instances that can be used to train the StyleGAN2-ADA model will reduce. This would lead to a longer training duration for StyleGAN2-ADA or/and reduced quality of the generated images, as StyleGAN2-ADA only guarantees good results for datasets with at least a few thousand training images [14]. More raw images need to be collected to ensure the quality of the generated images by StyleGAN2-ADA, indicating that the total required time will be increased further. Therefore, it is decided not to include image augmentation via StyleGAN2-ADA in the training pipeline.

The results for the trainings on the EDE dataset (Table VI) validates the discussion made above. With the application of traditional augmentation combinations, the raw images need to give the desired performance can be reduced. The number of raw images required would be around 200, but it should be adjust based on the scene complexity to achieve the best results. In contrast, image augmentation via StyleGAN2-ADA is not promising and will prolong the duration required before the deployment of the model, so it is not recommended to be included in the training pipeline of the MobileNetV2 SSD model in practice.

3) *Effect of transfer learning*

The desired optimum setting for transfer learning is to minimize the training time while maintaining the required performance in precision and recall. The pre-trained MobileNetV2 SSD model is trained using the overhead person dataset, while the number of frozen layers of the object detection model during training is varied to obtain the optimal setting. The training dataset is taken from Experiment 6 which consist of 8000 augmented images from the EDOB dataset with 11760 annotations, while the validation set consists of 849 images with 1242 annotations from the EDOB dataset. The model performance and the training duration per 100 iterations, t, is tabulated in Table VII.

TABLE VII. MODEL PERFORMANCE WHEN TRAINED WITH DATASETS ON DIFFERENT AUGMENTATION SETTINGS ON EDOB DATASET.

No.	Last layer frozen	Model Performance	t (s)
6	No layers are frozen	mAP: 97.1% AR: 69.75%	70
12	Box encoding predictors of the box predictors	mAP: 61.8% AR: 47.35%	15
13	Last convolutional layer of the feature extractor	mAP: 67.5% AR: 51.30%	18
14	The expended_conv_10 layer in the feature extractor	mAP: 94.4% AR: 63.91%	20
15	The expended_conv_7 layer in the feature extractor	mAP: 96.2% AR: 67.99%	15

It can be deduced that the more layers are frozen, the lesser the computation is required during training since parameter optimization and backpropagation of the frozen layers are not required, therefore the training will speed up. The rationale for freezing the earlier layers of the model is that given the low-level features learned in the previous task are the same as the target task, the earlier layers need not be trained anymore since their weights are already optimized. However, if the assumption that the features learned by the frozen layers are useful to the target task does not hold, then the performance will drop. In Experiment 12, only the class predictors of all box predictors in SSD are not frozen. This is a common setting that was done in transfer learning for classification problems, where only the classifier layers are changed and trained. However, the results show that this setting is not preferable for this project, as the model performance dropped significantly compared to the performance without transfer learning (Experiment 6). It is then concluded the optimum setting is to freeze the layers up to expended_conv_7 (the eighth bottleneck layer in the MobileNetV2 feature extractor), shown in Experiment 15. With this setting, the training duration is shortened by a factor of close to three, while only having a small compromisation in the precision and recall.

4) *Real time deployment speed of the model*

The model is deployed real time on raspberry pi, and the FPS of the first 500 frames plotted in Fig. 6. The deployment speed is always greater than 30 FPS as desired.

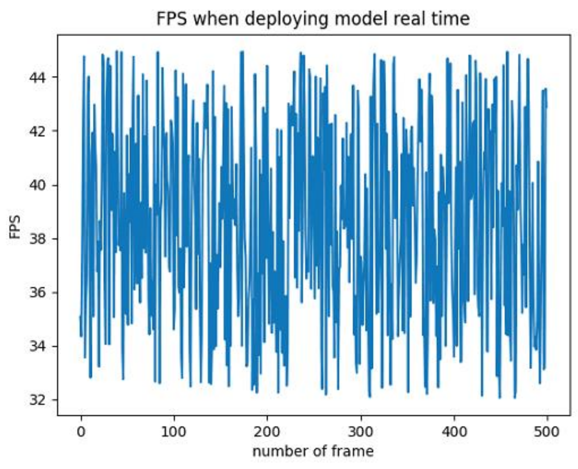

Fig. 6. Deployment speed of the model on Google Coral.

V. Conclusion and Future Work

Results show that training and validating the model in the same environment performs much better as compared to training in a few different environments and validating the model in an unseen environment. It is also concluded that replacing raw images with augmented images helps in improving the performance as it adds variance to the training set, and the application of transfer learning is proved to be able to increase the training speed while not affecting the model performance. A precision of over 90% and recall of almost 70% is achieved with a deployment speed greater than 30 FPS when the trained model is tested in the target environment. However, augmentation via StyleGAN2-ADA does not help in improving the model performance is not practical in actual application. Future improvements would include utilizing or modifying GAN networks that are able to take in background context and generate human instances on it, so that it only needs to be trained once on a generic overhead human dataset and prevent obvious boundaries in the image.

References

[1] "SOP MOVEMENT CONTROL ORDER (MCO)." Ministry of Health Malaysia. https://covid-19.moh.gov.my/faqsop/sop-perintah-kawalan-pergerakan-pkp (accessed 6th October, 2021).

[2] S.-K. Jarraya, M.-H. Alotibi, and M.-S. Ali, "A Deep-CNN Crowd Counting Model for Enforcing Social Distancing during COVID19 Pandemic: Application to Saudi Arabia's Public Places," *Computers, Materials \& Continua*, vol. 66, no. 2, pp. 1315--1328, 2021. [Online]. Available: http://www.techscience.com/cmc/v66n2/40674.

[3] T. W. Chan, V. V. Yap, and C. S. Soh, "Embedded based tailgating/piggybacking detection security system," in *2012 IEEE Colloquium on Humanities, Science and Engineering (CHUSER)*, 3-4 Dec. 2012 2012, pp. 277-282, doi: 10.1109/CHUSER.2012.6504324.

[4] P. Maheshwari, D. Alex, S. Banerjee, S. Behera, and S. Panda, "Top View Person Detection and Counting for Low Compute Embedded Platforms," in *ACM International Conference Proceeding Series*, ed: ACM, 2018, pp. 35-43.

[5] D. Fuentes-Jimenez *et al.*, "DPDnet: A robust people detector using deep learning with an overhead depth camera," *Expert systems with applications,* vol. 146, p. 113168, 2020, doi: 10.1016/j.eswa.2019.113168.

[6] J. Zhao, G. Zhang, L. Tian, and Y. Q. Chen, "Real-time human detection with depth camera via a physical radius-depth detector and a CNN descriptor," in *2017 IEEE International Conference on Multimedia and Expo (ICME)*, 10-14 July 2017 2017, pp. 1536-1541, doi: 10.1109/ICME.2017.8019323.

[7] "TensorFlow Object Detection API - GitHub." https://github.com/tensorflow/models/tree/master/research/object_detection (accessed 5th April, 2021).

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

[9] W. Liu *et al.*, "Ssd: Single shot multibox detector," in *European conference on computer vision*, 2016: Springer, pp. 21-37.

[10] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L. Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 18-23 June 2018 2018, pp. 4510-4520, doi: 10.1109/CVPR.2018.00474.

[11] Y. Zhang, H. Peng, and P. Hu, "Cs341 final report: Towards real-time detection and camera triggering," *Standford, CA, USA, Tech. Rep. CS*, vol. 341, 2017.

[12] F. H. K. d. S. Tanaka and C. Aranha, "Data Augmentation Using GANs," *ArXiv*, vol. abs/1904.09135, 2019.

[13] A. Antoniou, A. Storkey, and H. Edwards, "Data Augmentation Generative Adversarial Networks," 11/12 2017.

[14] T. Karras, M. Aittala, J. Hellsten, S. Laine, J. Lehtinen, and T. Aila, "Training generative adversarial networks with limited data," *arXiv preprint arXiv:2006.06676*, 2020.

[15] T. Karras, S. Laine, and T. Aila, "A style-based generator architecture for generative adversarial networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4401-4410.

[16] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?," *arXiv preprint arXiv:1411.1792,* 2014.

[17] S. Naz, A. Ashraf, and A. Zaib, "Transfer learning using freeze features for Alzheimer neurological disorder detection using ADNI dataset," *Multimedia Systems*, 2021/05/04 2021, doi: 10.1007/s00530-021-00797-3.

[18] H. Shin *et al.*, "Deep Convolutional Neural Networks for Computer-Aided Detection: CNN Architectures, Dataset Characteristics and Transfer Learning," *IEEE Transactions on Medical Imaging,* vol. 35, no. 5, pp. 1285-1298, 2016, doi: 10.1109/TMI.2016.2528162.

[19] A. Antoniou. "Data Augmentation Generative Adversarial Networks (DAGAN)." https://github.com/AntreasAntoniou/DAGAN (accessed 1st August, 2021).

[20] NVlabs. "StyleGAN2-ada." https://github.com/NVlabs/stylegan2-ada (accessed 15th August, 2021).

[21] "TensorFlow models on the Edge TPU: Compatibility overview." https://coral.ai/docs/edgetpu/models-intro/#compatibility-overview (accessed 5th April 2021.

[22] P. Dollar, C. Wojek, B. Schiele, and P. Perona, "Pedestrian detection: A benchmark," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 20-25 June 2009 2009, pp. 304-311, doi: 10.1109/CVPR.2009.5206631.

[23] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, 20-25 June 2005 2005, vol. 1, pp. 886-893 vol. 1, doi: 10.1109/CVPR.2005.177.

[24] D. Liciotti, M. Paolanti, E. Frontoni, A. Mancini, and P. Zingaretti, "Person Re-identification Dataset with RGB-D Camera in a Top-View Configuration," Cham, 2017: Springer International Publishing, in Video Analytics. Face and Facial Expression Recognition and Audience Measurement, pp. 1-11.

[25] I. Prodaiko. "Person re-identification in a top-view multi-camera environment." https://github.com/ucuapps/top-view-multi-person-tracking (accessed 13th May, 2021).

[26] A. Creswell, T. White, V. Dumoulin, K. Arulkumaran, B. Sengupta, and A. A. Bharath, "Generative adversarial networks: An overview," *IEEE Signal Processing Magazine*, vol. 35, no. 1, pp. 53-65, 2018.

[27] R. Geirhos, P. Rubisch, C. Michaelis, M. Bethge, F. A. Wichmann, and W. Brendel, "ImageNet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness," *arXiv preprint arXiv:1811.12231,* 2018.

[28] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572,* 2014.