Computer Vision: Detection of dangerous objects in baggage to automate control at airports

Project Report

Léna AIX	Akshay SHASTRI	Nhat Mai NGUYEN	Maxime MULOT	Vanshika SHARMA
ESSEC & CentraleSupélec	ESSEC & CentraleSupélec	ESSEC & CentraleSupélec	ESSEC & CentraleSupélec	ESSEC & CentraleSupélec
lena.aix @essec.edu	akshay.shastri @essec.edu	nhatmai.nguyen @essec.edu	maxime.mulot @essec.edu	vanshika.sharma @essec.edu

ABSTRACT

The increased security threats associated with air travel necessitate efficient and accurate methods for detecting dangerous objects in baggage. Manual screening of luggage is time-consuming, error-prone, and requires significant human resources. To address these challenges, this corporate research project aims to develop an automated pipeline for detecting dangerous objects in baggage using the YOLOv5 object detection algorithm. Additionally, we integrate CodeCarbon, a carbon emissions calculator, to assess the environmental impact of the developed system. Our pipeline achieves 89% precision and integrates both safety and sustainability aspects in airport operations, providing valuable insights for future airport security systems.

KEYWORDS

Object Detection, Dangerous Objects, X-ray Images, YOLOv5, Deep Learning

1. Introduction

Recently, the growing concern over security in public places has become a prominent issue. With the increase in terrorism and mass shootings, there is a pressing need to enhance security measures to safeguard people from potential threats. In response, Deloitte has initiated a project to augment these security measures for their clients in high-security areas. They plan to achieve this by utilizing computer vision technology to detect dangerous objects.

The primary objective of this project is not to replace human personnel but to enhance their efficiency. By employing computer vision, the system can identify objects that may go unnoticed by humans, thereby reducing human error and bias. To transform this idea into a comprehensive product, Deloitte has decided to focus on airport security as an initial example and subsequently extend it to other security clients. Currently, several airports in the US, such as Miami, Detroit, and LA, have already begun implementing computer vision for screening purposes. This adoption aims to mitigate human error and minimize passenger wait times. However, most airports in the EU still rely on manual methods. Thus, Deloitte sees Europe as a potential market for this technology.

Deloitte faces certain challenges with this project. Firstly, it is their first venture in this domain, lacking prior experience, which may make it difficult to attract clients for the product. Secondly, the unavailability of real-world data poses another obstacle. To overcome these challenges and convince clients to collaborate on high-security issues, Deloitte can adopt several strategies. They can focus on developing a minimum viable product (MVP) that showcases the core functionality and benefits of the computer vision system. By demonstrating the potential impact and effectiveness of the technology, Deloitte can generate interest and gain clients' trust. In terms of data, relying solely on publicly available data may not provide sufficient accuracy for the system. Therefore, employing data augmentation techniques to enhance the model's performance is essential. Data augmentation involves creating synthetic data by applying various transformations to existing data. This approach can help increase the diversity and volume of the training data, leading to improved accuracy. To ensure the project's scalability and generalizability to other security areas, a modular and adaptable approach is needed. The system should be designed so that it is easily configurable and customizable to suit the specific requirements of different security clients. By considering the unique needs and constraints of various security environments, Deloitte can create a solution that can be seamlessly extended beyond airport security.

Overall, by focusing on building an MVP, utilizing data augmentation techniques, and designing a flexible system architecture, Deloitte can develop a compelling project that attracts clients and addresses high-security concerns effectively.

2. Problem definition

Our team's objective is to assist Deloitte in building a MVP for their computer vision-based security solution, which aims to enhance security measures in high-security areas, initially focusing on airport security. The goal is to develop an MVP that effectively showcases the core functionality and benefits of the system, demonstrating its potential impact and generating interest from potential clients.

Challenges to be addressed during the collaboration with Deloitte include:

- Limited resources and expertise: constraints in terms of access to GPU resources for training and inference, as well as limited expertise in developing computer vision systems for security applications.

- Complex requirements: the solution must accurately detect dangerous objects, accounting for variations in object shape, size, and angle. It should be capable of differentiating between dangerous objects and harmless items, minimizing false positives to avoid unnecessary disruptions and delays.

- Generalization to other security areas: the MVP should be designed and implemented in a manner that allows for easy adaptation and generalization to suit the specific requirements of different security clients beyond airport security.

- Limited data availability: scarcity of real-world training data, which can hinder the development of an accurate and robust computer vision model for detecting dangerous objects.

3. Related works

When building a MVP for computer vision-based security solution, it's important to understand the existing research in this area. Prominent related works are:

- "Object Detection in Deep Surveillance" by Thakur, N. et al. (2021) discusses the importance of object detection in computer vision and surveillance applications, particularly focusing on pedestrian detection. It highlights the increasing number of techniques developed for identifying pedestrians in images, including deep neural network-based frameworks for object detection. The research aims to address the need for a powerful object detector that not only accurately positions and labels bounding boxes but also provides relative positions of the objects. To achieve this, the paper evaluates several state-of-the-art algorithms, namely Faster-RCNN, SSD, and YOLO, using the MOT20 dataset and a custom dataset recorded using an Unmanned Aerial Vehicle (UAV) in the organization's premises. The experimental analysis reveals that the Yolov5 model outperforms the other models, achieving a precision of 61% and an F-measure value of 44%. This finding suggests that Yolov5 is a promising choice for object detection in deep surveillance applications. [1]

- In paper "Weapon Detection Using YOLOv3 for Smart Surveillance System" by Sanam Narejo et al. (2021), a computer-based automated system is developed to identify handguns and rifles, aiming to address the issue of gun-related violence. The researchers utilize the YOLOv3 object detection model and train it on a customized dataset. The results demonstrate that YOLOv3 outperforms YOLOv2 and traditional CNNs, while also requiring fewer computational resources due to the use of transfer learning. By integrating this model into surveillance systems, the goal is to enhance public safety, reduce manslaughter rates, and prevent mass killings. The proposed system can also be implemented in high-end surveillance and security robots to detect weapons and mitigate potential risks to human life. The study emphasizes the need for improved surveillance capabilities and highlights the potential of smart surveillance systems and digital monitoring systems, including robots, to replace existing infrastructure as technology advances. [2]

- In "Comparison of Object Detection Algorithms for Street-level Objects", Martinus Grady Naftali et al. (2022) focuses on comparing different one-stage detector algorithms for street-level object detection in real-time images. The algorithms compared include SSD MobileNetv2 FPN-lite 320x320, YOLOv3, YOLOv4, YOLOv5l, and YOLOv5s. The evaluation is conducted on a modified Udacity Self Driving Car Dataset, with preprocessing and augmentation techniques applied. The algorithms are trained and evaluated based on precision, recall, F1-Score, Mean Average Precision (mAP), and inference time. The results indicate that all algorithms perform well in terms of inference time and accuracy. YOLOv5l achieves the highest accuracy with an mAP@.5 of 0.593, while MobileNetv2 FPN-lite has the fastest inference time of 3.20ms. YOLOv5s is found to be efficient, offering a balance between accuracy and speed comparable to MobileNetv2 FPN-lite. These findings suggest that the evaluated algorithms are suitable for street-level object detection, making them viable for applications such as self-driving cars. [3]

- According to Marko.Horvat3, Ljudevit.Jelecevic et Gordan.Gledec (2022), the main advantages of YOLOv5 lie in its integration of advances from other areas of computer vision research, enhancing object recognition performance, and its availability in the PyTorch framework for easy integration in modern application development environments. The algorithm offers a range of architectures, allowing for various applications beyond accuracy-driven tasks. YOLOv5 improves the application of trained neural networks in real-time video processing applications, such as people counting or intelligent decision support systems. It is a versatile algorithm that can be trained for practical applications, including those demonstrated in the paper. YOLOv5 simplifies the application of trained neural networks on

weaker devices, which was previously challenging. However, to improve object recognition performance without changing the algorithm, a better understanding of the scene is advised, such as using ontologies and formal knowledge representation methods. Additionally, in practical AI applications, reliability and trust are crucial, and Explainable AI (XAI) approaches should be considered to provide users with understandable explanations for the system's decisions. Utilizing YOLO algorithms effectively requires comprehensive explanations for classifications or decisions made by the system. [4]

In light of Deloitte's aspirations to implement real-time video processing applications, YOLOv5 stands out as the epitome of suitability for their needs and this project.

4. Methodology

The project was implemented over 4 phases. Tasks in each phase were decided by the documentation provided by Deloitte, and issue tree which decided the flow of the project. Phase 1 was exploratory, phase 2 was modeling, phase 3 was carbon footprint analysis and phase 4 delivery.

4.1 Data labeling and splitting

For the project, we used the GDXray [5] dataset for training and testing our model. A brief description of the Baggage dataset is as followed:

Series	Images	kpixels	Description
B0001–B0006	9–14	722.5-5935.1	Pen case with several objects
B0007	20	129.6	Razor blade for training
B0008	361	745.8	Rotation of a knife in 1^0
B0009–B0043	4–19	276.6	Backpack with handgun and objects
B0044	178	5935.1	Backpack with handgun
B0045	90	1287.0	Pen case in 90 positions
B0046–B0048	200	5412.0–5844.0	Backpack with handguns, shuriken, razor blades
B0049–B0051	100–200	165.6–759.5	Handguns, shuriken, and blades for training
B0052–B0054	144	741.3	Shuriken with 6,7,8 points for training
B0055–B0057	800–1600	16.9–18.1	200 4, 6, 8-image sequences of single objects
B0058	64	196.6	Crops of clips, springs, razor blades and others
B0059	64	196.6	Binary ideal segmentation of images of B0058
B0060	2	5935.1	Images for dual-energy experiments
B0061–B0073	17–25	2856.1–3656.8	Razor blade in cases
B0074	37	2856.1	Rotation of a door key in 10^0
B0075–B0077	576	1581.8–5935.1	Knife in 576 positions

We started with data exploration and labeling the dataset for using YOLOv5. Given that many images are similar (slightly variate in the angle), we decided to manually select and label certain images to save time and write Python code to enrich the dataset later. We used a tool called MakeSense.AI to label the dataset with YOLO Bounded boxes. Then, we stored the annotated images and labels in separate folders, 'images' and 'labels'. We maintained consistent naming between the images and their corresponding annotations as a requirement of YOLOv5.

The dataset is then split into test, train, and validation datasets. Due to the similarity in the images, we opted for a manual split for this project in order to avoid data spillage. This allowed us to ensure that images used in the test set were not replicated in the training and validation set.

4.2 Data augmentation

Augmentation is essential to increase the diversity and variability of the training data, improving the model's ability to generalize and perform well on unseen images. Therefore, the next stage involved generation of augmented images to improve our training results. We did two types of augmentations - rotate and crop/zoom the training images and created two separate Python Notebooks.

In the first notebook, we demonstrate how to rotate images and their corresponding bounding boxes using OpenCV and NumPy. We define a class called yoloRotatebbox and two functions yoloFormattocv() and cvFormattoYolo(). These functions and classes are responsible for converting bounding box coordinates between YOLO and OpenCV formats during the rotation process. It further provides a function generate_rotated_image(), which reads all images in a given directory, rotates each image by a random degree within a given range, and also rotates the bounding box annotations in YOLO format. The rotated images and bounding boxes are then saved to the given directories.

In the second notebook, we perform cropping of the original images along with cropping of vertical rotated images and vertical + horizontal rotated images using Albumentations library. We also define functions to plot the bounded boxes on the image and demonstrate two test images for each type of augmentation. Thereafter, we save the transformed images and bounding boxes with updated names to the given directories.

We tested 14 different combinations of original and augmented images, achieving an improvement in the precision by 14% and the recall by 5% for detecting dangerous objects.

	Before (545 training images)	After (7,130 training images)
Precision	71%	85%
Recall	43%	48%

Note: for dangerous objects only

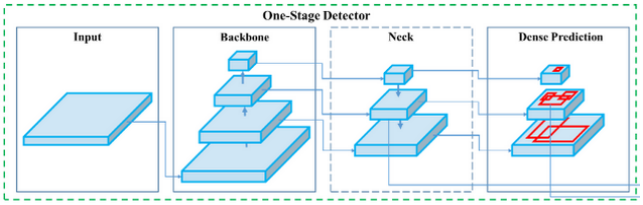
4.3 Modeling

After having chosen the global YOLOv5 model architecture, several variants of the model can be used, depending on the ressources (GPU size, etc.), the objectives (e.g. applied to videos) and also the characteristics of the images (e.g. image size, size of the objects to be detected). Ultralytics (the community proposing different pre-trained YOLO architectures) proposes different generations of YOLO architectures and different sizes of it, that you can choose. After trying different solutions, we chose the

YOLOv5m6 architecture [6] pre-trained on the COCO dataset. Here is a quick comparison of the YOLOv5m6 model with another one:

Architectures comparison (in general)	YOLOv5m versus Resnet 50 FPN
Precision	Similar
Inference speed	8 times faster
Training time per epoch	8 times faster

YOLOv5 is an object detection architecture that follows a one-stage, anchor-free approach. YOLOv5 uses a convolutional neural network (CNN) backbone, typically based on the EfficientNet architecture, to extract features from input images. These features are then processed by a series of convolutional layers, followed by a prediction head that predicts bounding boxes and class probabilities for detected objects. YOLOv5m6 is a medium size YOLOv5 but with a P6 output layer added. This model is composed of 3 parts: the backbone, the neck and the head. The backbone and the neck use pre-trained layers: CSP-Darknet53 as a backbone, SPP and PANet in the model neck.

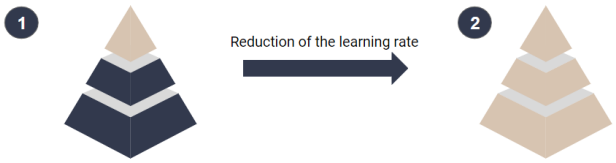


4.4 Training & hyperparameter tuning

As per different strategies we and the previous users tried, the way of training that was the most effective for our case is as follow:

1. Freeze the backbone and neck and only train the head with a given learning rate
2. Finetune the result model of step 1 by training all the layers with a much smaller learning rate

- **2 steps training:**



We provided a Jupyter Notebook as well as hyperparameters configuration files used for the training. Several parameters can also be modified and must be adjusted regarding the dataset in use. After trying several different values for the main hyperparameters, our final parameters are the following:

Parameters	Values
Learning Rate for Head training	0.001
Learning Rate for Global training	0.0001
IoU	0.5
Batch Size	32
Epochs	100
Optimizer	Adam

4.5 Prediction

The previous sections of this report deal with model training. In most cases, this training, which is a time-consuming task, will only need to be carried out once in order to obtain a trained model capable of responding to a particular task. In our case, the model is capable of detecting dangerous objects from an image. The main goal of our project being to allow one to detect dangerous objects on images, we needed to provide Deloitte with a tool to do so once our model has been trained, and its parameters are saved as a ".pt" file.

Due to project time constraints, we were unable to develop a comprehensive web interface to demonstrate the model's functionality when users upload images. Instead, we have provided Deloitte with a notebook called "predict.ipynb". To detect objects on images, simply enter the path to the ".pt" file:

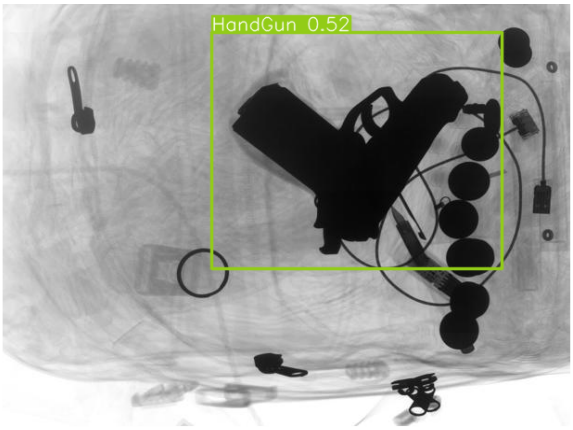
```
# Load custom model
model = torch.hub.load('ultralytics/yolov5', 'custom', path='/content/drive/My
```

And the path to the folder containing the images to be predicted and the name of the images to be predicted:

```
# Set the path for test images and create a list of test image names
test_img_path = '/content/drive/MyDrive/CRP_Shared_Folder/CRP__Dataset_Repartition/
test_img = ['B0022_0002', 'B0050_0077', 'B0063_0002', 'B0075_0039']
```

Then, run the entire notebook. Once execution is complete (no more than a few seconds per image), the notebook returns the images with your model's predictions: the object's location, the name of the object detected, and the calculated probability as

demonstrated below:



Our code can be easily integrated with additional code to create a turnkey tool that is very easy to use and also adaptable to any other object detection project Deloitte may implement in the future.

.5. Evaluation

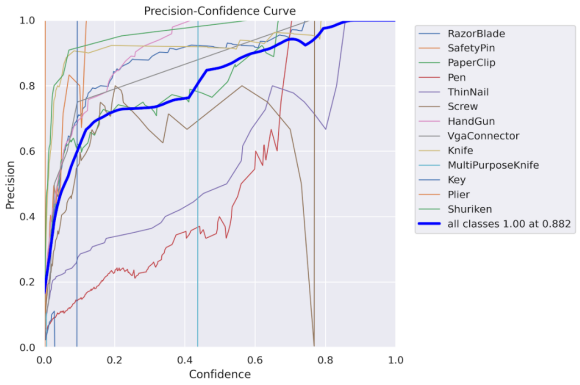
To evaluate and compare our different models, and particularly our final one, we used the following tab, called "Model summary", that is displayed in our notebook after each training:

```
Fusing layers...
Model summary: 276 layers, 35323956 parameters, 0 gradients, 49.1 GFLOPs
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100%
                   all        259        644      0.709      0.435      0.464      0.277
            RazorBlade        259        103      0.798      0.692      0.733      0.434
             SafetyPin        259         24          1     0.0706      0.297      0.127
             PaperClip        259        114      0.724      0.208      0.285      0.128
                   Pen        259        111      0.217       0.18      0.114     0.0324
              ThinNail        259         11      0.332      0.727      0.657      0.417
                 Screw        259        110      0.721      0.117      0.172     0.0506
               HandGun        259         55      0.781      0.909      0.903      0.525
         VgaConnector        259          3      0.781          1      0.995      0.588
                 Knife        259         48      0.918       0.75      0.815      0.504
      MultiPurposeKnife        259         29          0          0     0.0243     0.0194
                   Key        259         10          1          0     0.0396     0.0127
                 Plier        259          6          1          0          0          0
               Shuriken        259         20       0.94          1      0.995       0.76
Results saved to yolov5m6_airportIoU05/finetuning#
```

This tab gathers all the information we need to evaluate our model performances. The model also saves a lot of plots giving more information about our performances in the folder whose access path is specified below the value table.

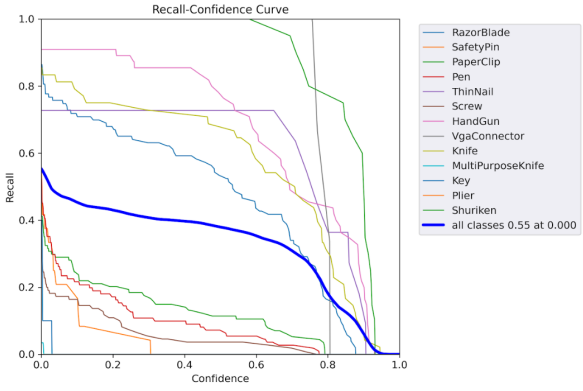
For each object (each class), it gives :

- The precision (column P): it is the number of true positives (i.e. the number of items correctly labeled as belonging to that class) divided by the total number of elements predicted as belonging to that class. It represents the percentage of the items predicted to belong to a certain class that are actually members of that class. The precision-confidence curve of our final model is the following:



We have a good overall precision, with an overall score of 71%; but this score is impacted by the objects for which we have very few images, such as pens or paper clips. If we only look at the objects that are important in our project, that is to say the **"dangerous" objects**, for which we have a lot of images, **we have an average precision of 89%.**

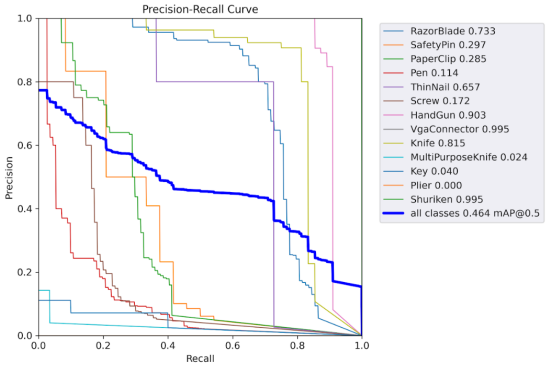
- The recall (column R): it is the number of true positives (i.e. the number of items correctly labeled as belonging to that class) divided by the total number of elements that actually belong to that class. This means that recall represents the percentage of the items of a class that our model detected. The precision-confidence curve of our final model is the following:



We can see that our recall isn't as good as our precision, we have an overall recall of 43%. But here again, we can clearly see that there are 2 categories of classes: one category for which our results are very poor: paper clips, safety pins, pens, screws, pliers,... And another category of classes for which we have better results: the dangerous objects (Razor blades, guns, shurikens, knives…). **For the dangerous objects, we have an average recall of 67%.**

- The mAP50 score: it is the average precision (AP) of the model with an Intersection over Union (IoU)

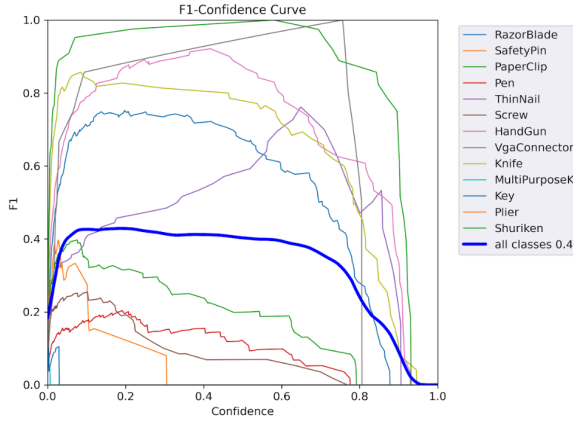
threshold = to 0.5. The IoU is the degree of overlap between the predicted bounding boxes and the ground truth bounding boxes for the prediction to be considered a true positive. The higher this threshold, the higher our precision, but the lower our recall. The AP is calculated as the area under the precision-recall curve, which shows how the precision and recall of the model vary at different thresholds.



The mAP50 score is the average of the AP values with a threshold of 0.5 across all classes. A higher mAP50 score indicates that the model is able to accurately detect and localize objects with a high degree of precision and recall. The mAP50-95 score is the average of the AP values across all classes and all IoU thresholds from 0.50 to 0.95 with increments of 0.05. Our final model has a mAP50-95 of 28% overall, and **45% for dangerous objects**.

- The F1 score curve: it displays the F1 scores over the confidence threshold, the F1 score is equal to 2 * Precision * Recall / (Precision + Recall). Here again, we see that we have an overall medium F1-score, due to very poor results on non-dangerous objects. The curves for the dangerous objects have much higher scores.

-



To conclude on the evaluation of our model, we are quite satisfied with our performance on dangerous objects. Our model's performance on non dangerous objects is rather poor, but we think it would be much better if we had access to more images focusing on these objects. We also note a trade-off between precision and recall. Indeed, before fine-tuning our model, we had, for dangerous objects, an accuracy of 91% and a recall of 58%. We were therefore able to increase our recall by almost 10% against a slight drop in accuracy.

6. Limitations

While we're convinced of the usefulness and performance of our model, we also note certain limitations and areas for improvement.

First of all, our performances are good enough for an MVP, but, in order to be used in industry, our model would require a more advanced finetuning stage; notably with a gridsearch or randomsearch, which we were unable to implement due to the limited capabilities of our IT tools.

Similarly, more advanced versions of YOLO are now available, with more parameters and therefore more potential. Provided that sufficient computing power is available, their use would further boost the performance of this model.

On the other hand, while we have automated the model creation process as far as possible, certain steps, such as image labeling or the distribution of images into train/validation and test sets, remain essential and manual. We are well aware of how time-consuming these steps are, and for this reason we estimate that we can reduce pipeline creation time by only 50%. In order to avoid any mishandling that would render our pipeline unusable, we have provided Deloitte with a "Process Guide", describing step-by-step the process to be followed in order to use our pipeline, from labeling the images to evaluating the results.

Also, the performances of our pipeline are extremely dependent on the quality of the dataset used. In particular, we observe variable performance on our dataset depending on the number and quality of the images provided for each class of object. To ensure good performance on the detection of a particular object, we advise you to provide the model with many images of this object, both alone, from different angles, and partially hidden by other objects. In general, the bigger and more varied your data set, the better the performance.

Finally, for any deep learning model, our pipeline requires significant computational resources, which results in both significant energy consumption and long computation time. Moreover, the complexity of the model strongly limits its

explainability, and it is therefore very difficult at this stage to guarantee any robustness. We therefore advise in particular to use this model not as a replacement for a security guard, but rather as a tool to facilitate and speed up baggage checks.

7. Carbon Computations

The carbon footprint of the training has been calculated using the package CodeCarbon [7]. The total energy consumed for training is at the **same order of magnitude as a cycle of washing machine** (1kWh).

	Energy consumed (Wh)
Footprint for the head training (with frozen layers)	230
Footprint for fine tuning all layers	380
Total	**610**

Note that the energy consumption for data augmentation is negligible compared to that of the model training.

The carbon footprint in eqCO2 can be computed. However, the value depends a lot on the context. In addition to your setup, it is also important to take into account two main parameters to calculate the carbon footprint:

-the servers location

-the time at which the training is made (see the variation of the French time emission variations here [8]).

For instance, please find below the theoretical carbon footprint for different server locations:

Country	**Average carbon footprint of the training (g-eqCO2) (subject to small variations)**
France	21
USA	240
China	330

NB: Statistics from 2021 from Statista (China), EIA (USA) and RTE (France).

To reduce the carbon footprint of training as much as possible, it is therefore advisable to perform the calculations **in a country with a green electricity mix and during off-peak hours** (e.g. during the night).

8. Business recommendations

Based on our experiences and findings during this project, we have the following recommendations for Deloitte:

1. **Implementing Object Detection in Business Operations:** The project presented here serves as a Minimum Viable Product (MVP) with a pre-made and tested pipeline for Object Detection and Computer Vision. It's ready to be scaled and adapted to a variety of business operations. It can streamline and automate processes, enhancing efficiency and reducing manual labor requirements.
2. **Automotive Damage Detection:** The object detection model can be adapted to automatically detect and quantify damages on cars. This could revolutionize the way insurance claims are processed, leading to significant time and cost savings. It could also prove useful in quality control during the manufacturing process of cars.
3. **Inventory Management:** By training the model to identify different types of food items, we could automate the process of managing inventory in grocery stores, warehouses, and restaurants. The model could provide real-time data on stock levels, leading to better forecasting, efficient restocking, and a reduction in food waste.
4. **Queue Management:** The model could be utilized to monitor and manage queues in high traffic areas such as salons, hotels, and clubs. By identifying and tracking individuals in a queue, it could provide accurate wait times, enabling better resource allocation and enhancing customer experience.
5. **Security and Surveillance:** The model has the potential to significantly enhance security in public places. It could be trained to detect suspicious or abandoned objects, alerting security personnel to potential threats and improving public safety.
6. **Continuous Improvement:** This project represents the initial stages of incorporating object detection into business processes. As with all machine learning projects, continuous improvement and retraining of the model will be necessary. This will ensure the model stays accurate as new data and potential scenarios are encountered.

Implementing these recommendations will allow Deloitte to leverage state-of-the-art AI technology, streamline processes, reduce costs, enhance customer experience, and potentially open up new lines of business or service offerings. More specifically, our deliverables will help Deloitte:

- Reduce Time to Create MVPs by 75-100%: expedite prototype creation & substantially decrease development costs.
- Gain Client Trust: showcase a workable solution to the prospective clients in a significantly shorter time.

- Deploy Client's Projects 50% Faster (save 1-1.5 month): use our tested pipeline to develop and deploy models for each client.
- Easily replicate the pipeline: efficiently mitigate potential implementation challenges using our user-friendly process guide to replicate the work

We strongly believe in the power of AI and machine learning to drive business innovation, and we see this project as a stepping stone towards that future.

9. Conclusion

In conclusion, this report presents a successful implementation of an automated pipeline for detecting dangerous objects in baggage at airports using the YOLOv5 object detection algorithm. The integration of CodeCarbon for calculating carbon emissions provides a valuable insight into the environmental impact of such systems. Carbon emissions can also be used during the training as an additional evaluation metric so that we can choose the right trade-off between emissions and performance (up to a certain limit). The developed pipeline demonstrates high precision in detecting dangerous objects, surpassing the limitations of manual screening methods. With better infrastructure, better annotations, larger dataset and larger model, this pipeline will work much better. However, it is important to remain sustainable while implementing these projects. By automating baggage control, airports can enhance security measures, improve efficiency, and contribute to a more sustainable aviation industry. Future work can focus on optimizing the performance, expanding the dataset, and exploring additional applications of YOLOv5 in airport security, including object detections in videos.

ACKNOWLEDGMENTS

We would like to express our sincere appreciation and gratitude to Kisebwe Hugo and Taleb Bendiab Ouissam from Deloitte for their valuable guidance and support throughout this project. Their expertise and insights have been instrumental in shaping the direction of our work and ensuring its success. Their dedication and commitment to the project have been truly commendable.

We would also like to extend our thanks to Marque Fabrice for his role as a business coach, providing valuable advice and guidance that has greatly contributed to the development and refinement of our project.

Additionally, we would like to express our gratitude to the technical advisor i.e. Christodoulidis Stergios for his expertise and technical insights, which have been invaluable in navigating the challenges of this project.

The contributions of these individuals have significantly enhanced the quality and impact of our work. Their guidance and support have been invaluable, and we are truly grateful for the opportunity to collaborate with them.

REFERENCES

[1] Thakur, N., Nagrath, P., Jain, R., Saini, D., Sharma, N., & Hemanth, J. (2021). Object Detection in Deep Surveillance. Research Article. Bharati Vidyapeeth's College of Engineering New Delhi. Karunya Institute of Technology and Sciences.

[2] Narejo, S., Pandey, B., Vargas, D. E., Rodriguez, C., & Anjum, M. R. (2021). Weapon Detection Using YOLO V3 for Smart Surveillance System. Research Article, Volume 2021, Article ID 9975700

[3] Martinus Grady Naftali, Jason Sebastian Sulistyawan, Kelvin Julian (2022). Comparison of Object Detection Algorithms for Street-level Objects.

[4] Marko.Horvat3, Ljudevit.Jelecevic, Gordan.Gledec (2022). A comparative study of YOLOv5 models performance for image localization and classification. Conference: 33rd Central European Conference on Information and Intelligent Systems (CECIIS) 2022At: Dubrovnik, Croatia.

[5] Mery, D.; Riffo, V.; Zscherpel, U.; Mondragón, G.; Lillo, I.; Zuccar, I.; Lobel, H.; Carrasco, M. (2015): GDXray: The database of X-ray images for nondestructive testing. Journal of Nondestructive Evaluation, 34.4:1-12.

[6] Ultralytics. (2023). YOLOv5: Object Detection with PyTorch. Retrieved from https://github.com/ultralytics/yolov5

[7] CodeCarbon. (2023). CodeCarbon: Measure and Reduce Your Code's Carbon Footprint. Retrieved from https://codecarbon.io/

[8] RTE France. (2023). Les émissions de CO2 par kWh produit en France. https://www.rte-france.com/eco2mix/les-emissions-de-co2-par-kwh-produit-en-france