**Traffic Monitoring for Urban Hospitals**

# Project Overview:

The primary goal of this project is to create a real-time system that can count and track vehicles in a hospital's parking lot. By using the video footage from CCTV cameras placed in both outdoor and basement areas, the system will gather important data on parking space occupancy, location, and vehicle arrival times. This will help improve the parking experience for visitors to the hospital.

# Technical Approach:

## Image Processing:

The system is implemented entirely in Python, leveraging powerful computer vision libraries like OpenCV, Supervision, and Ultralytics. These frameworks excel at image processing tasks like video capture, segmentation, annotation, filtering, and other pre-processing steps. The pre-processed images are then fed into a state-of-the-art, pre-trained neural network for object classification and tracking.

## Deep Learning for Object Detection and Tracking:

The chosen model, You Only Look Once (YOLO), is a Convolutional Neural Network (CNN) specifically designed for real-time object detection. CNNs mimic the human visual system by and analysing visual data through multiple layers of neurons.

These layers consist of:

* **Convolutional Layers:** These layers apply filters to the input image, detecting various features like edges, textures, and shapes. They perform convolutions, operations preserving spatial relationships while learning image features through small input data sections.
* **Pooling Layers:** These layers reduce the data's spatial dimensions while retaining essential features. This reduction in complexity prevents overfitting and improves computational efficiency
* **Fully Connected Layers:** These layers integrate the extracted features to classify the input into various categories (e.g., car, truck).

YOLO was chosen for its simplicity, superior speed, and accuracy compared to other methods like Haar cascades.

## Text Extraction and Data Collection:

To enrich the data collected by the object detection system, an additional Python program was developed for license plate information extraction. This program utilizes the data collected by the object detection pipeline and processes it with the help of the EasyOCR library.

### Optical Character Recognition (OCR)

EasyOCR is an open-source Python wrapper for the Tesseract OCR engine. EasyOCR is a highly regarded OCR tool capable of recognizing text within images. This integration allows the system to extract license plate information from the segmented vehicle images.

### Extracting Textual Data with Image Processing

OpenCV's image processing capabilities are harnessed to enhance the license plate image before feeding it to EasyOCR. Specific filters are applied to extract edges and contours within the image. These extracted features are crucial for EasyOCR to recognize the text characters on the license plate.

### Storing Valuable Data

The license plate information is stored in a CSV file along with the vehicle's entry timestamp. This structured format enables easy data storage, retrieval, and potential integration with other systems for further analysis.

# Challenges:

* **Detection Accuracy:** Ensuring reliable vehicle detection in varying conditions (lighting, weather) and crowded scenes. Fine-tuning models is essential to reduce errors.
* **Real-Time Processing:** Maintaining real-time performance with high-resolution video. Optimizing the computational pipeline is key.
* **Handling Occlusions:** Tracking partially obscured or overlapping vehicles requires advanced algorithms and temporal data analysis.
* **License Plate Recognition:** Accurate extraction of license plate data despite challenges like obscured or dirty plates, using high-resolution imaging and OCR tools.
* **Scalability:** Ensuring the system scales effectively as the number of vehicles and areas monitored increases, with modular and expandable architecture.
* **Environmental Factors:** Adapting to changing environmental conditions (e.g., light, weather) to maintain detection accuracy and reliability.

# References:

* <https://supervision.roboflow.com/latest/>
* <https://docs.ultralytics.com/>
* <https://pjreddie.com/darknet/yolo/>
* <https://github.com/raspberry-pi-maker/NVIDIA-Jetson/blob/master/XavierNX-YOLOv8/highway_traffic.mp4>
* <https://pypi.org/project/opencv-python/>
* <https://github.com/JaidedAI/EasyOCR>