Jetson Nano Board setup:

Two ways to work with Jetson nano board

1. With an external monitor, keyboard, and mouse:

- ➔ The first step is to flash the SD with a Jetson iso image and insert it into the board's SD card slot which provides the main storage for the board.
- ➔ Power supply the board with a USB cable. Connect to the internet through an Ethernet cable.
- ➔ Connect the board with an external monitor through the HDMI output port, as well as a mouse and keyboard onto respective ports.
- ➔ Now power on the board to boot. It blinks a green light.
- ➔ We will be able to see the operating system booting on the monitor, we need to set up the OS and log into it.

2. Headless mode using SSH terminal on the host machine (windows) using software like PuTTY:

- ➔ First, flash the SD card with the Jetson iso image inserted into the board, The SD card provides main storage and also holds the https://www.researchgate.net/publication/377130487_Implementation_of_AI_in_Traffic_Management_Need_Current_Techniques_and_Challengesoperating system in it.
- ➔ In this mode, we will access the board using the host machine's serial terminal application, for example, PuTTY, through a USB cable connection to the board and computer.
- ➔ Power board using only barrel socket only and jumper the J48 power header pins.
- ➔ Connect the board to the internet through an Ethernet cable or USB wifi.
- ➔ Open 'Device Manager' on Windows go to the COM port and note the port number.
- ➔ Open the PuTTY application go to configurations select 'Session ' Enter the port number and speed as 115200 and click enter.
- ➔ We get the login interface in the host machine and are ready to use the board.
- ➔ After successful setup in the PuTTY, we can access the Jetson board directly through VS code using the SSH remote access extension.

3 . Dataset from Kaggle: Indian traffic dataset
https://www.kaggle.com/datasets/ashfakyeafi/road-vehicle-images-dataset/data

4 . Reference Research papers:

1. https://www.researchgate.net/publication/377130487_Implementation_of_AI_in_Traffic_Management_Need_Current_Techniques_and_Challenges

2. https://ieeexplore.ieee.org/abstract/document/9331256

- Paper 2 explains in detail the setup and workflow of the model and all the necessary algorithms, and plugins used in the development process.
- Application of the model based on the above paper goes like arranging 4 Raspberry Pi cameras on signal light poles at the four directions of the four-road junction and Jetson nano board as the processing unit.
- The main task is to find a more congested path (which has more numbers of vehicles) for the traffic and it should display a green signal on that way.
- Algorithms
  We should create an algorithm for traffic light signals based on threshold.
  We should create an algorithm for 7 7-segment display counter.

5. Jeston-py emulator:
   https://pypi.org/project/jetson-emulator/

In the Jetson emulator, we performed image classification, object detection, and image segmentation.
However, in the emulator, there are restrictions in object detection and image segmentation. It can detect up to a maximum of 3 objects and it can only work with virtual live cams because of this we cannot use the traffic data set from the Kaggle.

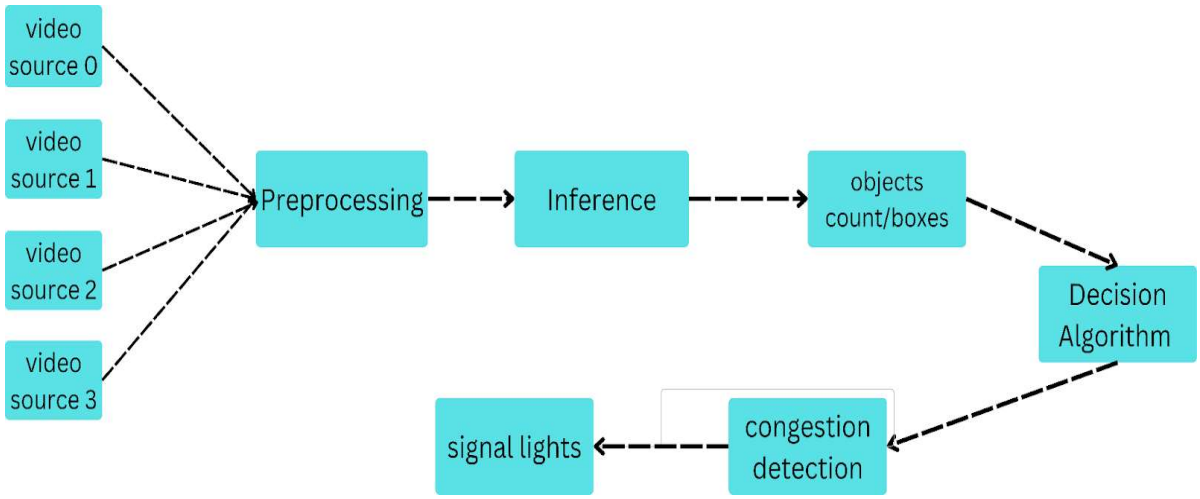
Framework and pipeline 1st Oct

Nvidia Deepstream SDK:

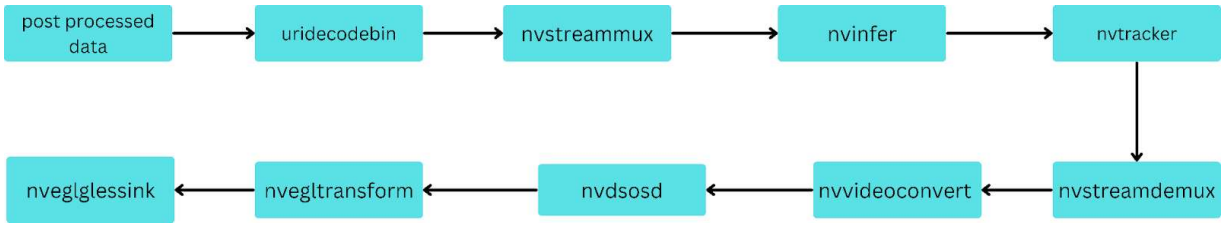
DeepStream, NVIDIA's AI inferencing SDK, is specifically designed for Jetson devices and GPUs to deliver high throughput and real-time performance. It provides development-ready inference models, plugins, and tools for easily creating custom pipelines that support multi-sensor input processing, all optimized with GPU acceleration. Combined with TensorRT, DeepStream enables efficient AI inferencing with enhanced performance and scalability.

Neural Network Architecture - ResNet18 (exploring more about the model for conclusion) for vehicles and multi-object detection.

Flow chart of architecture as follows:
The model should be trained on Indian traffic data.



1) Video streams from multiple cameras are collected and preprocessed.
2) Preprocessing: Decoding, Batching, Cropping and Scaling.
3) Inferencing: Multi-input multiple object-detection pipeline as follows.
Several plugins are crucial in the object detection pipeline.



- uridecodebin: decodes video from the input.
- nvstreammux: combines multiple video inputs into a single batch for optimal processing.
- nvinfer: This is the core of object detection where the ResNet18 model is used.
- nvtracker: track objects across frames using the KTR tracker algorithm (Kanade-Luca-Tomasi). This plugin handles batched input and ensures separate results for each stream.
- nvstreamdemux: Splits the batch back into separate individual streams for separate post-processing of each video stream.
- nvvideoconvert: converts the video format after the detection to prepare for visualization.
- nvdsosd: The On-screen-display (OSD) plugin displays bounding boxes and labels.
- nvegltransform: Prepare the video data for rendering using EGL.
- nveglglesssink: renders the processed video to a window.

4) Display Bounding boxes:

Results of the above pipeline show bounding boxes and maintaining count of objects. Let’s B is the bounding box (b-box) set, Ci set, is the confidence of the ith b-box, the expression is,

5) Decision Algorithm: This is the core part, that decides which side of the traffic is more congested. The more congested side is given priority and triggers traffic lights dynamically adjusting the timer.