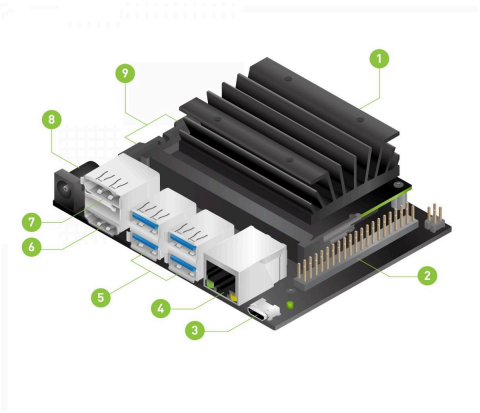
Hardware Setup:

Equipments:

- NVIDIA Jetson Nano Developer Kit
- Webcam / Waveshare IMC219-160 Camera
- Coral USB Accelerator
- Computer Monitor / EVICIV 7 inch (Raspberry Pi Monitor)
- microSD card (32GB minimum)
- USB Keyboard and Mouse
- Micro-USB Power Supply
- Ethernet Cable / Wifi USB Adapter



NVIDIA Jetson Nano

Webcam

Coral USB Accelerator

EVICIV Monitor

microSD Card

USB Keyboard and Mouse

PowerBank (Micro-USB Cable)

Wifi Adapter

Initial Setup:

At first we need a computer (pc or laptop) with an Internet connection and the capacity to read and write SD cards, either through an adapter or a built-in slot are prerequisites for setting up the microSD card. Then we need to download the SD Card Formatter (Formatter) to format the SD Card, Etcher (balenaEtcher) and SD Card Image (Jetson Nano SD Card Image) and to write into the microSD Card.

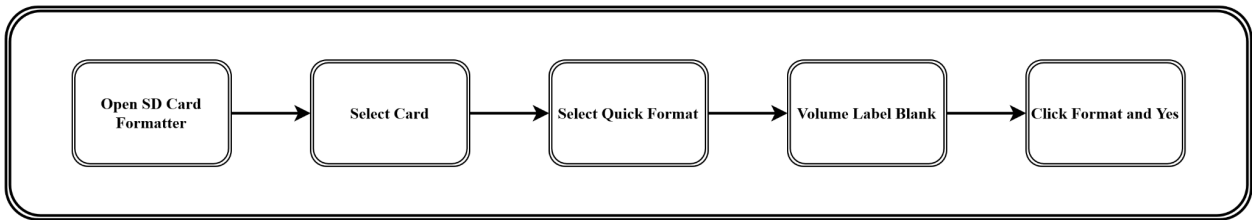


Figure: Format microSD Card using SD Card Formatter

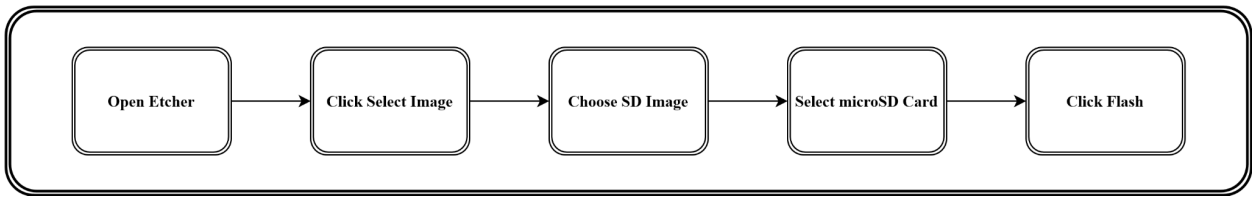


Figure: Flash microSD Card with SD Image using Etcher

Connections Setup:

First, put the flashed microSD card into the Jetson Nano. Now turn on the display using the PowerBank and link the Jetson Nano to the HDMI connector. Next, link the USB ports on the Jetson Nano to the mouse, keyboard, wifi, and webcam adapters. Next, turn on the Jetson Nano by connecting it to a micro USB connection on the PowerBank. Linux will launch immediately. Lastly, go through a few simple Linux system configurations to get the device started.

Setup Environment and Dependencies:

Install Python 3 pip and virtualenv:

```
sudo apt-get install python3-pip
pip3 install virtualenv
```

Create a virtual environment with system site packages:

```
python3 -m virtualenv -p python3 env --system-site-packages
source env/bin/activate
```

Verify Python and OpenCV installation:

```
python -c 'import cv2; print(cv2.__version__)'
```

Install dependencies for OpenCV:

```
sudo sh -c "echo '/usr/local/cuda/lib64' >> /etc/ld.so.conf.d/nvidia-tegra.conf"
```

```
sudo ldconfig
sudo apt-get install build-essential cmake git unzip pkg-config
sudo apt-get install libjpeg-dev libpng-dev libtiff-dev
sudo apt-get install libavcodec-dev libavformat-dev libswscale-dev
sudo apt-get install libgtk2.0-dev libcanberra-gtk*
sudo apt-get install python3-dev python3-numpy python3-pip
sudo apt-get install libxvidcore-dev libx264-dev libgtk-3-dev
sudo apt-get install libtbb2 libtbb-dev libdc1394-22-dev
sudo apt-get install libv4l-dev v4l-utils
sudo apt-get install libgstreamer1.0-dev libgstreamer-plugins-base1.0-dev
sudo apt-get install libavresample-dev libvorbis-dev libxine2-dev
sudo apt-get install libfaac-dev libmp3lame-dev libtheora-dev
sudo apt-get install libopencore-amrnb-dev libopencore-amrwb-dev
sudo apt-get install libopenblas-dev libatlas-base-dev libblas-dev
sudo apt-get install liblapack-dev libeigen3-dev gfortran
sudo apt-get install libhdf5-dev protobuf-compiler
sudo apt-get install libprotobuf-dev libgoogle-glog-dev libgflags-dev
```

Download OpenCV and OpenCV contrib:

```
cd ~
wget -O opencv.zip https://github.com/opencv/opencv/archive/4.5.1.zip
wget -O opencv_contrib.zip https://github.com/opencv/opencv_contrib/archive/4.5.1.zip
unzip opencv.zip
unzip opencv_contrib.zip
```

Rename directories:

```
mv opencv-4.5.1 opencv
mv opencv_contrib-4.5.1 opencv_contrib
rm opencv.zip
rm opencv_contrib.zip
```

Build OpenCV:

```
cd ~/opencv
mkdir build
cd build
```

Run in a single block:

```
cmake -D CMAKE_BUILD_TYPE=RELEASE -D CMAKE_INSTALL_PREFIX=/usr -D
OPENCV_EXTRA_MODULES_PATH=~/opencv_contrib/modules -D
EIGEN_INCLUDE_PATH=/usr/include/eigen3 -D WITH_OPENCL=OFF -D WITH_CUDA=ON -D
CUDA_ARCH_BIN=5.3 -D CUDA_ARCH_PTX="" -D WITH_CUDNN=ON -D
WITH_CUBLAS=ON -D ENABLE_FAST_MATH=ON -D CUDA_FAST_MATH=ON -D
OPENCV_DNN_CUDA=ON -D ENABLE_NEON=ON -D WITH_QT=OFF -D
WITH_OPENMP=ON -D WITH_OPENGL=ON -D BUILD_TIFF=ON -D WITH_FFMPEG=ON -D
WITH_GSTREAMER=ON -D WITH_TBB=ON -D BUILD_TBB=ON -D BUILD_TESTS=OFF -D
```

WITH_EIGEN=ON -D WITH_V4L=ON -D WITH_LIBV4L=ON -D OPENCV_ENABLE_NONFREE=ON -D INSTALL_C_EXAMPLES=OFF -D INSTALL_PYTHON_EXAMPLES=OFF -D BUILD_NEW_PYTHON_SUPPORT=ON -D BUILD_opencv_python3=TRUE -D OPENCV_GENERATE_PKGCONFIG=ON -D BUILD_EXAMPLES=OFF ..

make -j4 (takes long time)

Install OpenCV:

cd ~
sudo rm -r /usr/include/opencv4/opencv2
cd ~/opencv/build
sudo make install
sudo ldconfig
make clean
sudo apt-get update

Install jtop (System Monitoring):

sudo -H pip3 install -U jetson-stats
sudo reboot
jtop

Check camera:

ls /dev/video0

Install Python 3.8:

sudo apt update
sudo apt upgrade
sudo apt install build-essential libssl-dev zlib1g-dev libncurses5-dev libncursesw5-dev libreadline-dev libsqlite3-dev libgdbm-dev libdb5.3-dev libbz2-dev libexpat1-dev liblzma-dev libffi-dev libc6-dev

wget https://www.python.org/ftp/python/3.8.12/Python-3.8.12.tar.xz

Extract the downloaded archive:

tar -xf Python-3.8.12.tar.xz
cd Python-3.8.12

Configure the build process:

./configure --enable-optimizations

Build Python:

make -j4

sudo make altinstall
python3.8 --version

python3.8 -m venv myenv
source myenv/bin/activate

pip install ultralytics
pip show ultralytics
pip3 install jupyte
Jupyter notebook (optional)

Dataset for YOLOv8:

Folder Structure

```
dataset_root/
├── train/
│   ├── images/
│   │   ├── image1.jpg
│   │   ├── image2.png
│   │   └── ... (training images)
│   └── labels/  # (labels folder assumed to exist here)
│       ├── image1.txt
│       ├── image2.txt
│       └── ... (text files with bounding box annotations)
└── val/
    ├── images/
    │   ├── image3.jpg
    │   ├── image4.png
    │   └── ... (validation images)
    └── labels/  # (labels folder assumed to exist here)
        ├── image3.txt
        ├── image4.txt
        └── ... (text files with bounding box annotations)
└── dataset.yaml  # YAML configuration file
```

Yaml File Structure

```yaml
train: ../train/images
val: ../valid/images

nc: 3  #total number of classes
names: ['bus', 'car', 'truck']  #class names
```

YOLOv8 Training and Deployment:

Here, we have used the following techniques for training, validating, and testing to fine-tune the Ultralytics Yolov8 pre-trained model for vehicle detection.

```
!pip install ultralytics  #install package
from ultralytics import YOLO  #import yolo

model  # model version (yolov8s.pt, yolov8m.pt, yolov8l.pt, yolov8x.pt)
lr0  # learning rate
data = …….  #path of data.yaml
epochs  #set number of epochs
imgsz = …… # image size
```

!yolo task=detect mode=train model=model_name data=data_path epochs=epochs imgsz=image_size lr0=lr device=Cuda #training

!yolo task=detect mode=val model=model_path data=data_path device=Cuda #validation

!yolo task=detect mode=predict model=model_path conf=ct source=path_image/video/camera #testing

In order to detect vehicles in real-time, we must export the model into Jetson Nano using Pendrive or get the best.pt model from Google Drive. Then, we must run the Python script below from Jetson Nano's command prompt.

Python Script (detection.py):

```
from ultralytics import YOLO
import cv2

model = YOLO('myenv/yolov8l_0.pt')


cap=cv2.VideoCapture("/dev/video0")
cap.set(3, 840)
cap.set(4, 680)
font=cv2.FONT_HERSHEY_COMPLEX

ret = True
```

```
while ret:

    ret, frame = cap.read()

    if ret:

        results = model.track(frame, persist=True)

        frame_ = results[0].plot()

        cv2.imshow('frame', frame_)

        if cv2.waitKey(25) & 0xFF == ord('q'):

            break

cap.release()
cv2.destroyAllWindows()
```

Run Python Script:

```
python detection.py    #python name_of_your_python_file.py
```