# Deep Learning

## End-Course Test

edureka!

## Scenario 1

### Problem Statement

> India has been fighting the COVID-19 pandemic since 30 January 2020 when the first case of COVID-19 was reported. With the Unlock 4.0 phase set to begin in September, the need to be proactive is now more than ever. The objective is to create a Real-Time Face Mask Detector which can solve monitoring issues in crowded areas such as Airports, Metros, etc. using CNN and OpenCV.

### Dataset Description

The dataset is an artificial set of face mask images

- Total Images: 1376
- with_mask images:
- without_mask images:

The goal is to create a Deep Learning model to detect in real-time whether a person is wearing a face mask or not

## Tasks to be performed

**.ipynb file:** Create a Notebook File and perform the following tasks

As a part of this test, you will be performing the following tasks:

- Prepare a detailed python notebook using CNN for detecting Face Masks in Real-time
- Import Required Libraries
  - Hint: TensorFlow, NumPy, etc.
- Load and Pre-process the dataset
  - Pre-process the images present in the dataset using the TensorFlow preprocessing module
  - Encode the categorical data using an encoder of your choice because Machine Learning algorithms can only understand numerical data
  - Split the data into training and testing set using sklearn's **train_test_split** function

Hint: Use ImageDataGenerator to perform data augmentation

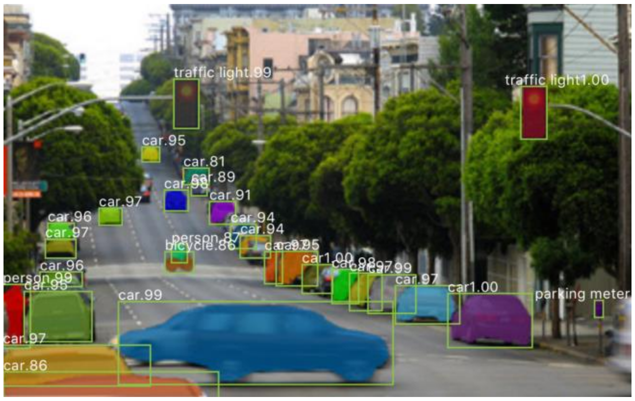
- Visualize the images present in the dataset
- Design a Convolutional Neural Network (CNN) Model using AveragePooling2D, Flatten, Dense, and Dropout layers
  - **Hint:** Use MobileNetv2 as the base model
- Compile the Model using Adam optimizer, Binary Crossentropy loss, and accuracy metric functions
- Train the Model for 30 epochs
  - Hint: Use EarlyStopping Callback to terminate the training if there is no improvement in the monitor performance measure of your choice for certain epochs in a row

  - Note: If you do not define the number of epochs, the model will only train for 1 epoch
- Plot the training history using the Tensorflow History object returned by **model.fit**
  - Make a plot for the loss function to visualize the change in the loss at every epoch
  - Make a plot for the accuracy metric to visualize the accuracy at every epoch
- Evaluate the Model using **model.evaluate** method
- Save the Entire Model using **model.save**
  - **Note:** To save the entire model that includes the model's architecture, weights, and training configuration, you must use the **model.save** method. If you only want to save the weights of a model, you can use the **model.save_weights** method
- Now that you have trained the model, test it using a webcam using OpenCV, and detect the Face Masks in real-time

Required Files

# Scenario 2

## Problem Statement



Around 1.5 million people are killed in road accidents on roadways each year. Barry Research Lab is developing a smart surveillance system that would detect cars on the road using using Mask RCNN. The objective is to use the Mask RCNN technique, for instance, segmentation on the COCO dataset

## Dataset Description

### COCO Dataset

COCO is large-scale object detection, segmentation, and captioning dataset. COCO has several features:

- Object segmentation
- Recognition in context
- Superpixel stuff segmentation
- 330K images (>200K labeled)
- 1.5 million object instances
- 80 object categories
- 91 stuff categories
- Five captions per image
- 250,000 people with keypoints

## Tasks to be performed

.ipynb file

As a part of this test, you will be performing the following tasks:

- Clone the Facebook Detectron Model using the following link
  - 'https://github.com/facebookresearch/Detectron.git'
- Import Required Libraries
- Load the pre-trained weights using the following link
  - https://dl.fbaipublicfiles.com/detectron/36494496/12_2017_baselines/e2e_mask_rcnn_X-101-64x4d-FPN_1x.yaml.07_50_11.fkwVtEvg/output/train/coco_2014_train%3Acoco_2014_valminusminival/generalized_rcnn/model_final.pkl

- Use the Mask RCNN Architecture and the pre-trained weights to generate predictions for your own images or images from the COCO dataset
- Visualize the Results with Matplotlib's **image.imread** method