**Real-Time Face Mask Detection:**

**Objective:** The focus is on implementing a system that can detect whether a person is wearing a face mask or not, this is to prevent the spread of coronavirus. The system is built using the python language, this process will consist of various steps. The first step is to create a dataset that is further used for training and testing the model. The second step is the labeling of the dataset using labelIMG API. The third step is using transfer learning using a pre-trained model, the model used here is TensorFlow Object Detection API. Lastly, the model can make real-time predictions.

**Tech Stack:**

1. Python: Is an interpreted, high-level, and general-purpose programming language. It is used to develop the whole system.





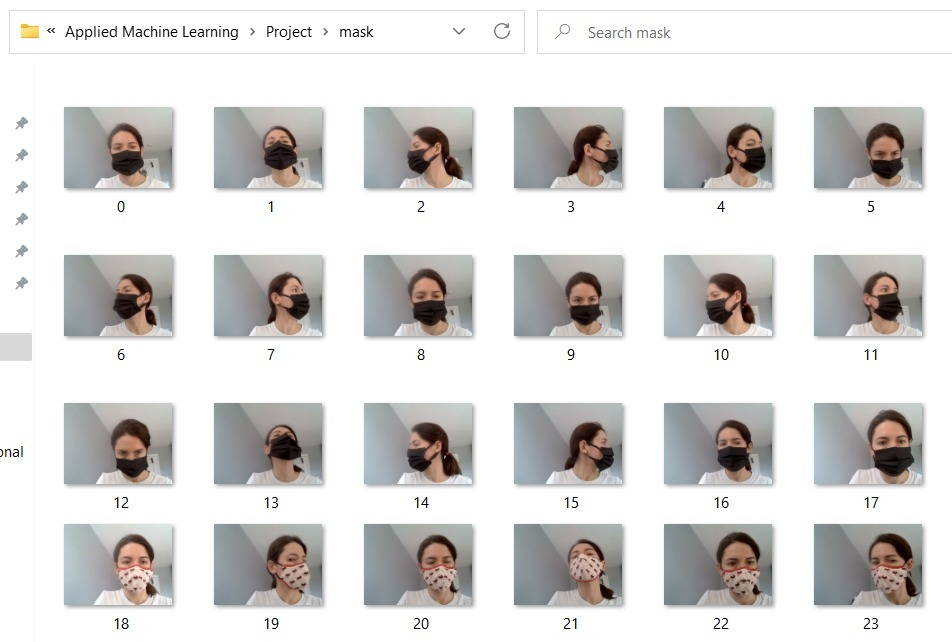


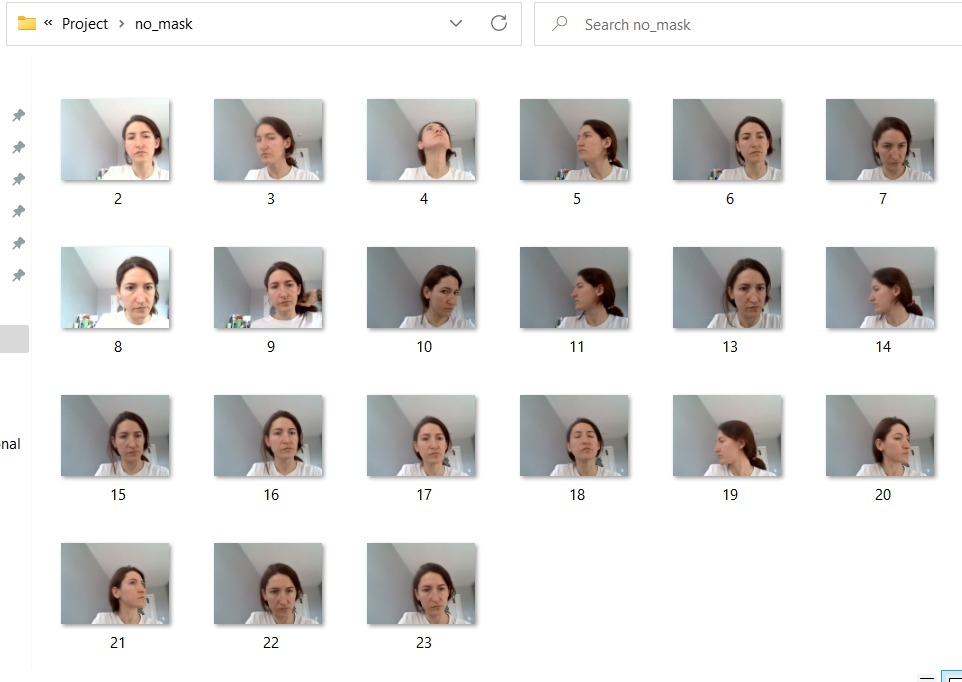


1. GitHub: It’s a software development platform to store, track, and collaborate on projects. It is used to manage the version control, share, and check the progress of the project.
2. TensorFlow Object Detection API: It’s an open-source framework built on top of TensorFlow. It has pre-trained models in the framework that makes object detection easy.
3. LabelIMG API: It’s an open-source tool for graphically labeling images. It is used in object detection.

**GitHub Link:** <https://github.com/esraersoy/Object-Detection-System---detecting-mask->

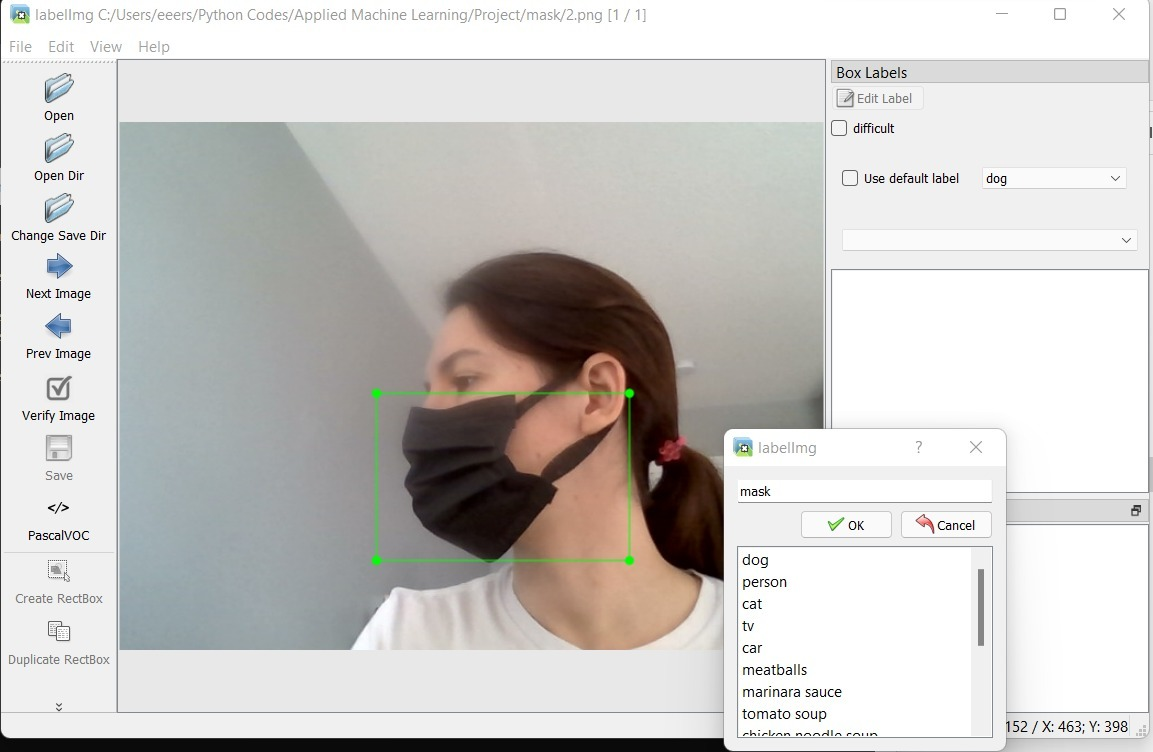
**Step 1: Dataset**





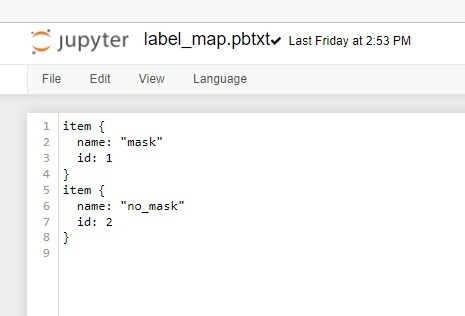
The process of collecting the dataset was completed through the python script. We used OpenCV to capture the video through a webcam. Later on, the image was captured each second from the video and saved as an image file using a scaling factor of 0.4. Finally, all the images were resized as 640 x 480 size. In total the dataset contained 46 images, 23 images masked images, and 23 unmasked images. The screenshots of the dataset are shown above.

**Step 2: Labelling of Data**



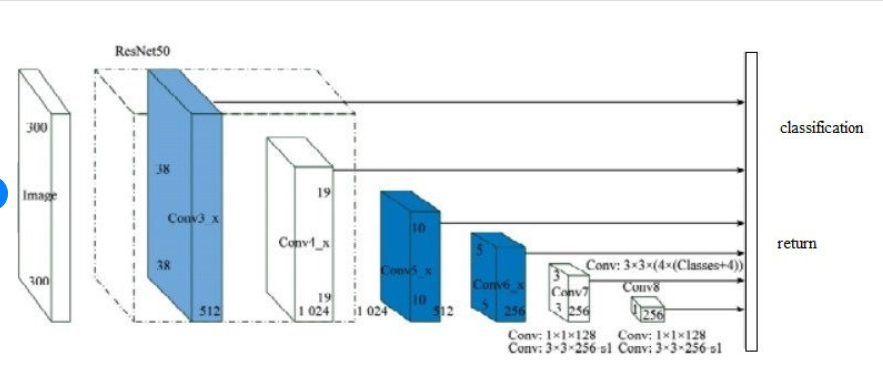
The labeling process was implemented through a free, open-source tool called LabelIMG. The dataset was organized in the folder and we begin the process of annotations where each image was annotated with a mask or no mask depending upon the image and then annotations XML files are generated. These XML files will be further used by our model.

A sample image of annotating a mask is shown above and a screenshot of a label-map is shown below of annotations mask and no mask.



**Step 3: TensorFlow Object Detection API**

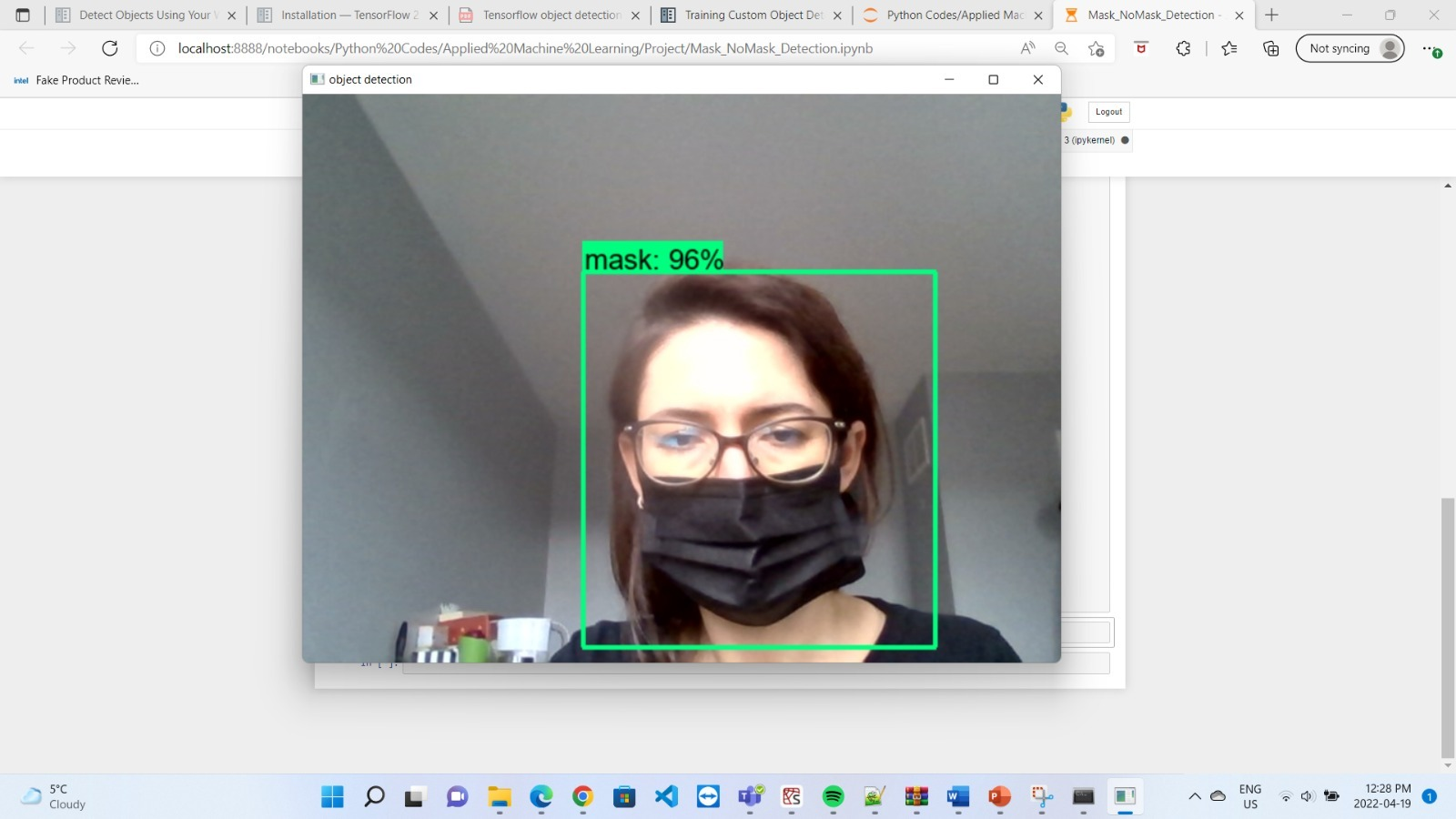
The training and method to use tensor flow object detection was the most difficult part as the training took a lot of hours. We have used a pre-trained deep learning model SSD ResNet50 V1 FPN 640x640.



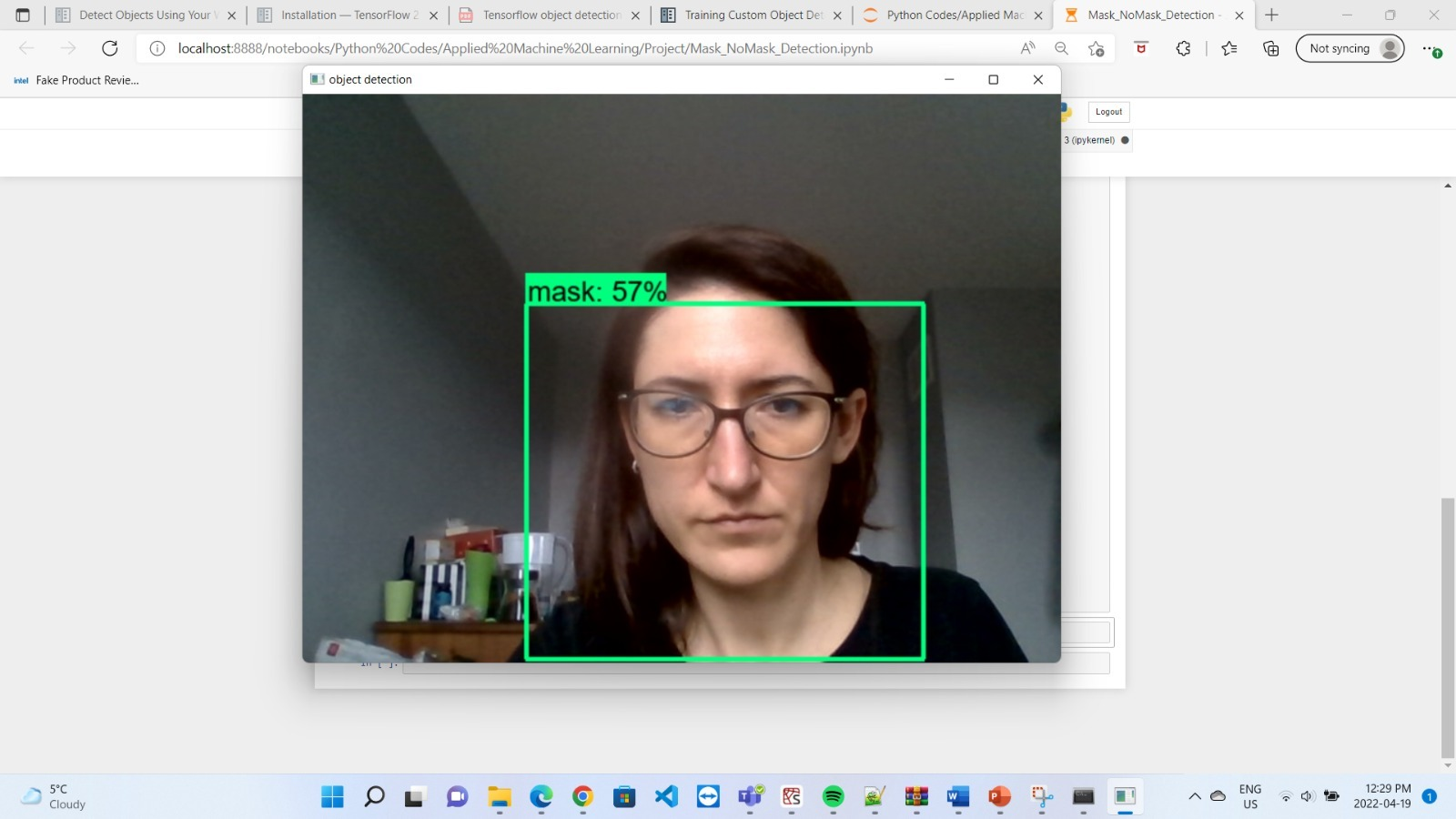
This step is all about the installation of the model and necessary libraries. This includes an object detection module available in the tensor flow because of this we are able to use a pre-trained model otherwise creating a model from scratch is a difficult process.

The next step is to load the label map file we created in step-2 which is a dataset for our model which we installed earlier. The only remaining step now is to infer results from the test images. This is done with the help of OpenCV where we try to capture the content through a webcam and our model tries to implement object detection. The final results of real-time object detection are shown in the next step.

**Step 4: Real-Time Predictions**



The result observed from our object detection model is not the best accuracy at the moment but it can be improved in a future iteration. This can be done by increasing the size of data or changing the object detection model with a higher accuracy rate as the model used here has 38% accuracy as compared to different available models at MODEL ZOO of tensor flow object detection module.



**References:**

1. Alakh Sethi, *Build you own Object Detection Model Using TensorFlow API*, Apr 2020, Analytics Vidhya <https://www.analyticsvidhya.com/blog/2020/04/build-your-own-object-detection-model-using-tensorflow-api/>
2. Gilbert Tanner, *Creating your own object detector*, Feb 2019, Towards Data Science, <https://towardsdatascience.com/creating-your-own-object-detector-ad69dda69c85>
3. *Training Custom Object Detector*, <https://tensorflow-object-detection-api-tutorial.readthedocs.io/en/latest/training.html>
4. Vighnesh Birodkar, *TensorFlow 2 Detection Model Zoo,* May 2021, GitHub, <https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/tf2_detection_zoo.md>