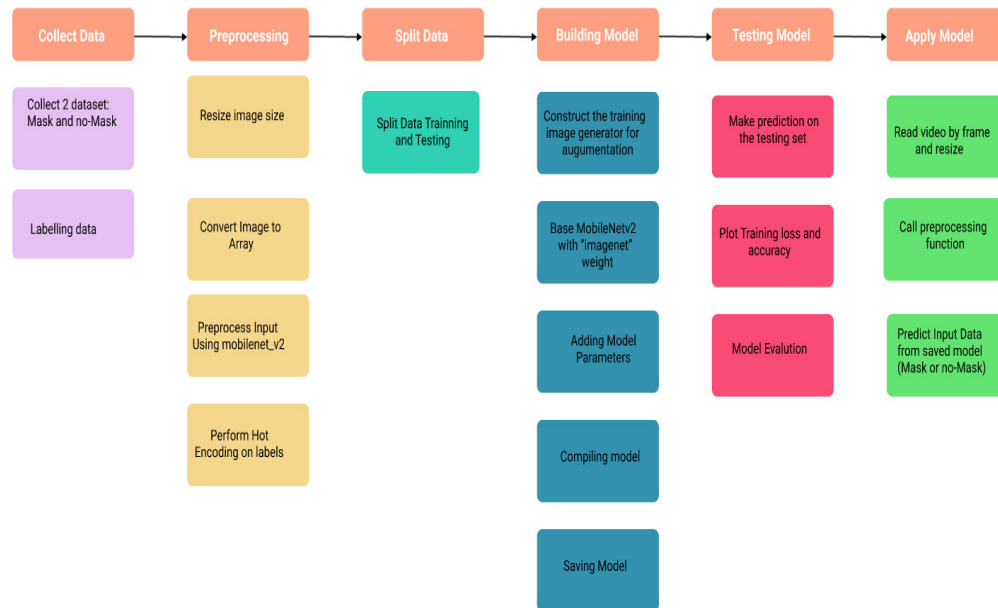


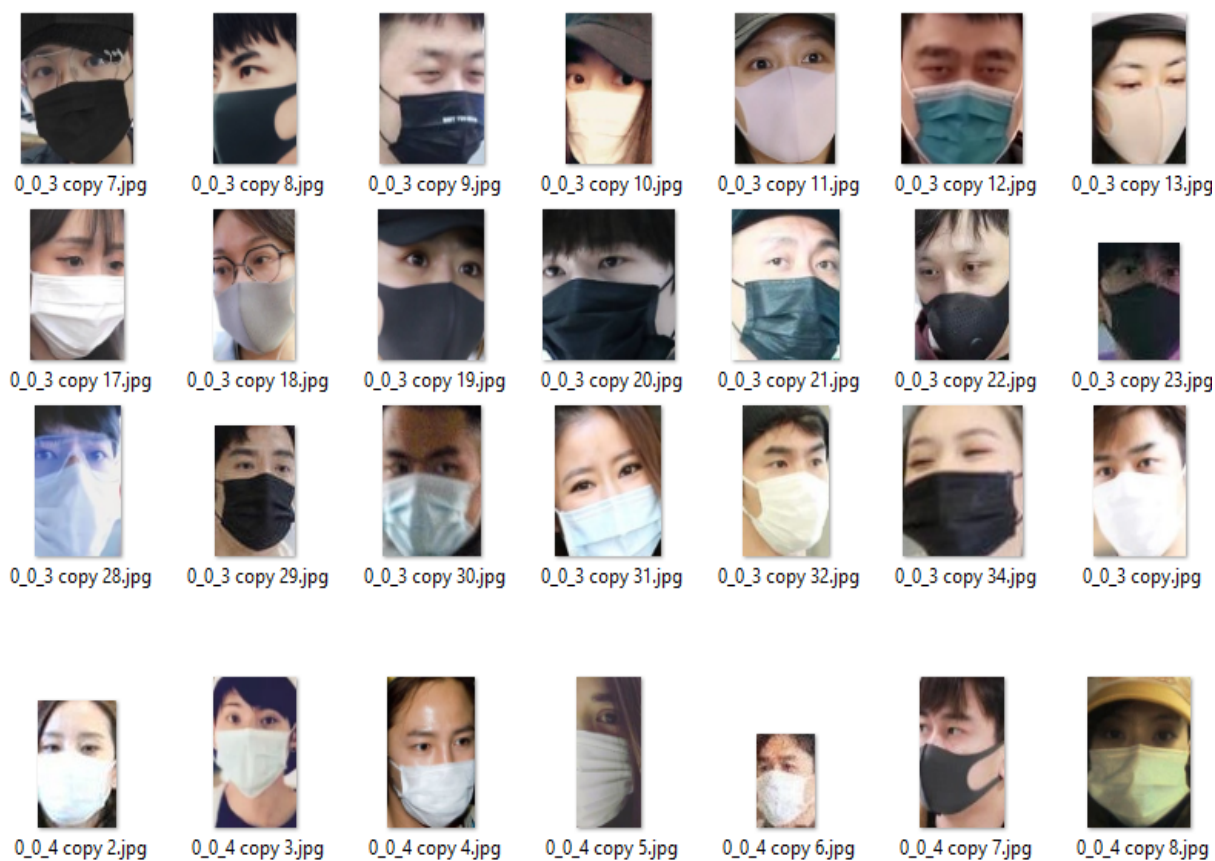
## Classification with Resnet50 vs MobileNetv2 : Steps in Building Model



*Figure Steps in building model using Resnet50 and MobileNetv2*

### a) Data Collecting

This Face Mask Detector model is based on data from Kaggle. People with masks and those without mask are both included in the dataset. The model will distinguish between persons wearing masks and those who aren't. 2.165 data with-mask and 1.930 data without-mask are used to build the model. The image is cropped at this point until the only viewable object is the object's face. With and without a mask image, the collected data are classified.. The following is an example of data:



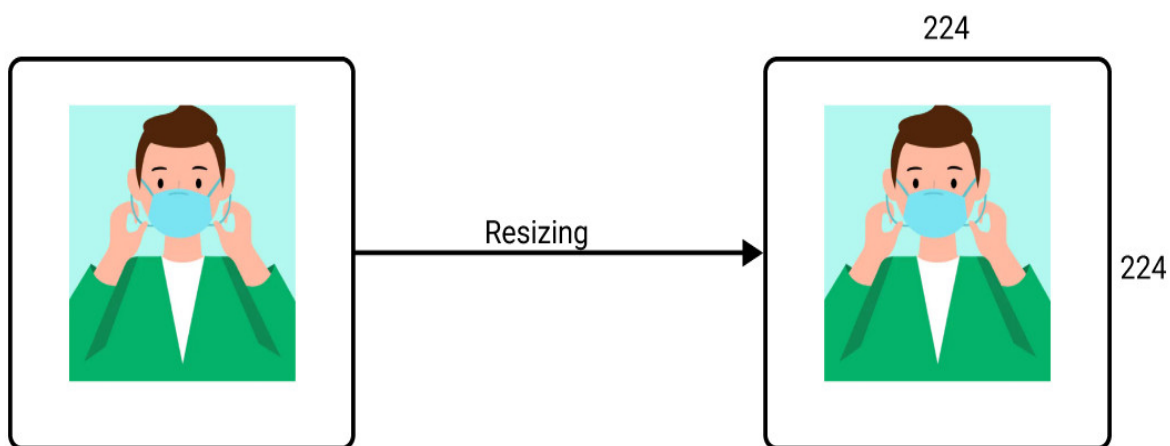
*Figure 3.4 Data with-mask (Data using for Resnet50 and MobileNetv2)*



*Figure 3.5 Data without-mask (Data using for Resnet50 and MobileNetv2)*

## b) Preprocessing

Preprocessing data is required before it can be used for training and testing.. Scaling the image size, converting the image to an array, preprocessing input using MobileNetV2 (or Resnet50), and using hot encoding on labels are the four processes in the preprocessing. Picture scaling is an important preprocessing step in CV because of the efficacy of training models. We employed Convolutional Neural Networks, hence the dataset pictures were shrunk to 224x224 pixels. The next step is to generate an array from the whole dataset's photos. A loop function is used to turn the picture into an array. The picture will then be used to preprocess input using MobileNetV2 ( or Resnet50). Because many ML algorithms can't operate directly on data labeling, this phase's last step is to employ hot encoding on labels. All of the input variables and output variables, including this algorithm, must be numeric. So that the algorithm can interpret and analyse the data, the labeled data will be transformed to a numerical label.



*Figure : A diagram to show the resizing of an image*

## c) Split the Data

Following the preprocessing step, the data is divided into two files: training (80%), and testing data (the remaining 20%). Images with and without masks are included in each batch

## d) Building model

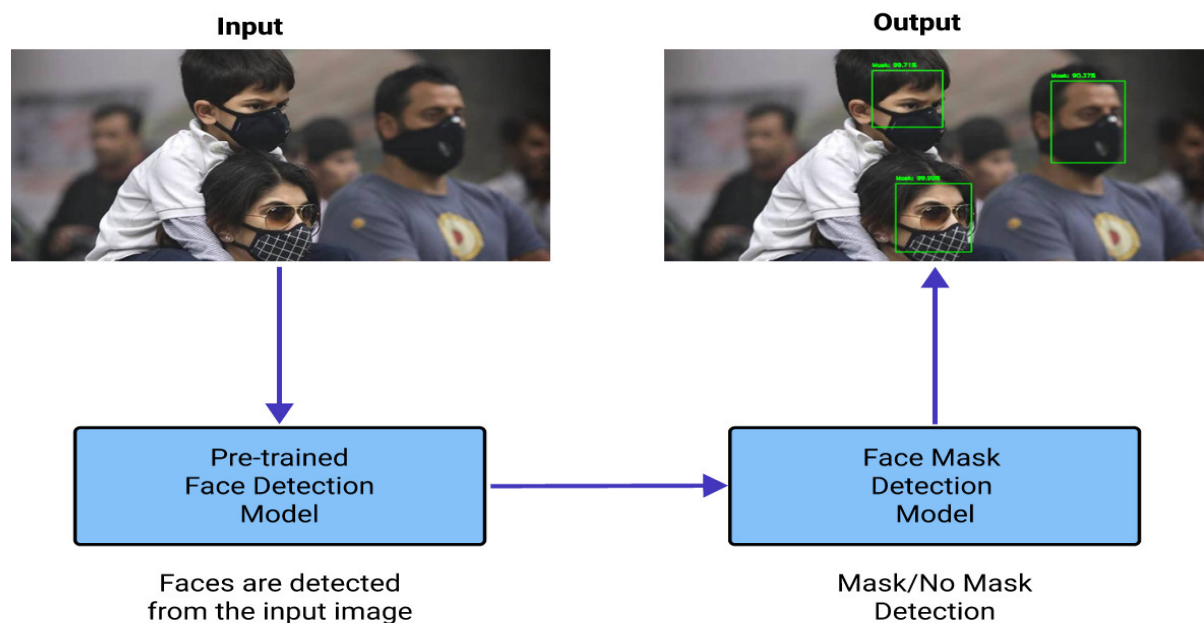
The model is created in six steps: building the augmented training image generator, constructing the base model using MobileNetV2 (or Resnet50), adding model parameters, compiling, training the model, and lastly storing the model for future processes.

### e) Testing model

There are procedures in testing the model to ensure that it can predict effectively. Making predictions about the testing set is the first stage.

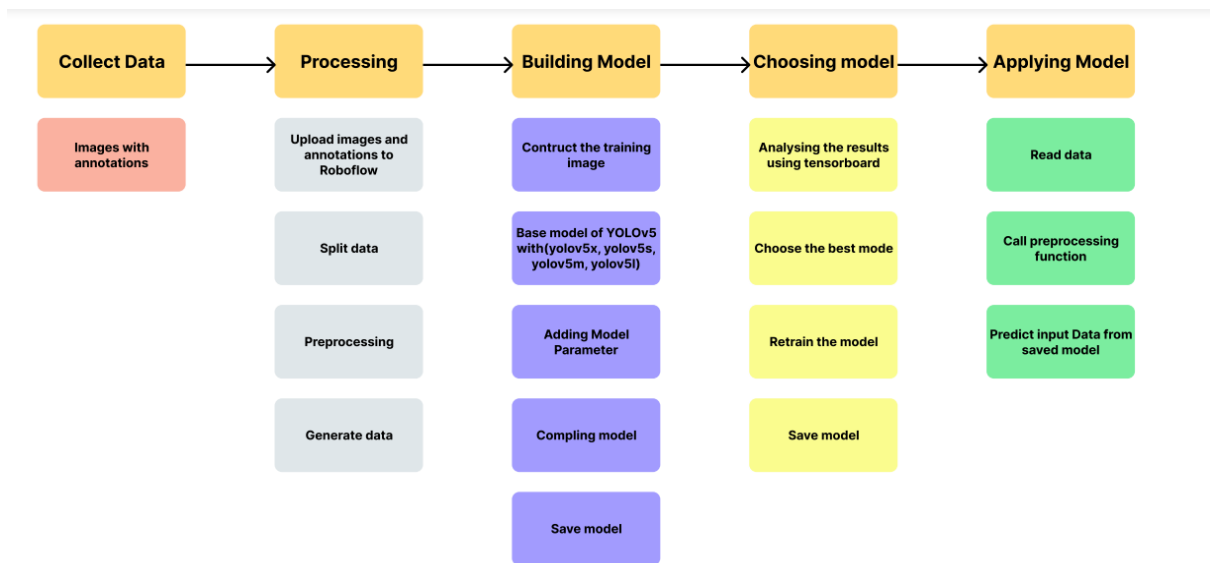
### f) Applying face mask detection model

A deep learning framework - Caffe that Berkeley AI Research (BAIR) and other collaborators built and maintain for the community as a quicker, more powerful, and more effective alternative to current object identification algorithms. The network architecture and the res10 300x300 SSD iter 140000 are described in the deploy.prototxt file. Caffemodel is the model that contains the layer weights. Two parameters are accepted by the cv2.dnn.readNet method ("path/to/prototxtfile" and "path/to/caffemodelweights"). After the model has been trained, it may be used to recognize face masks in both static photos and real-time video streams. After reading the movie frame by frame, the face detection algorithm is applied. The procedure continues if a face is discovered. On detected frames with faces, reprocessing will be conducted, including shrinking the picture size, converting to an array, and preprocessing the input using MobileNetV2 (or Resnet50). The next step is to use the stored model to predict input data. With the aid of a previously established model, predict the input picture that has been processed. In addition, whether or not the individual is wearing a mask, as well as the prediction %, will be indicated on the video frame.



*Figure :Diagram showing the implemented face mask detector model*

## II) Steps in building model using YOLOv5



*Figure 3.9 Steps in building model using YOLOv5*

### a) Collecting data: dataset (with 2 folder images, annotations)

In the absence of immunization, masks are one of the only feasible defenses against COVID-19, and they play a critical role in protecting human health from respiratory illnesses. It is possible to create a model that can detect people who are wearing masks, not wearing masks, or wearing masks improperly using this dataset. This collection contains 853 pictures from the three courses, together with their bounding-box in PASCAL VOC format. There are three types of masks: with -mask, without-mask, and incorrectly- mask. However, we only utilize two of the three classes in this thesis: with mask and without mask.



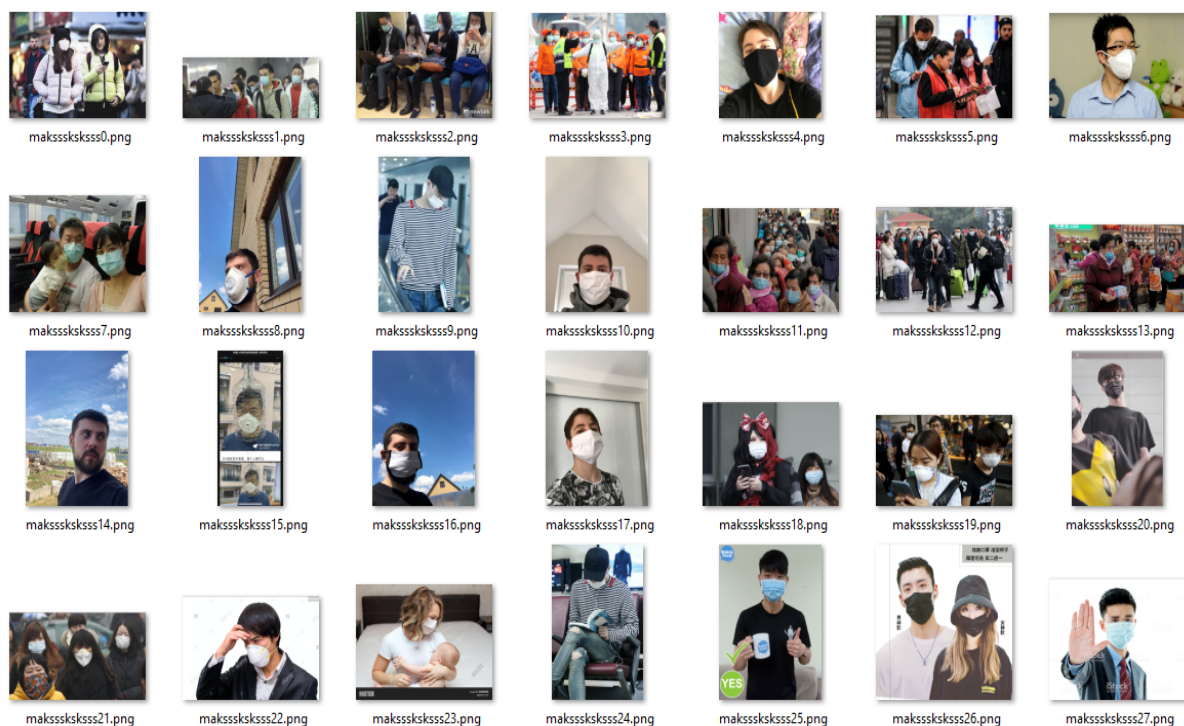


Figure 3.10 Data of YOLOv5 (Images)

	makssskskss0.xml	5/22/2020 7:18 AM	XML Document	2 KB
	makssskskss1.xml	5/22/2020 7:18 AM	XML Document	4 KB
	makssskskss2.xml	5/22/2020 7:18 AM	XML Document	2 KB
	makssskskss3.xml	5/22/2020 7:18 AM	XML Document	4 KB
	makssskskss4.xml	5/22/2020 7:18 AM	XML Document	1 KB
	makssskskss5.xml	5/22/2020 7:18 AM	XML Document	2 KB
	makssskskss6.xml	5/22/2020 7:18 AM	XML Document	1 KB
	makssskskss7.xml	5/22/2020 7:18 AM	XML Document	2 KB
	makssskskss8.xml	5/22/2020 7:18 AM	XML Document	1 KB
	makssskskss9.xml	5/22/2020 7:18 AM	XML Document	1 KB
	makssskskss10.xml	5/22/2020 7:18 AM	XML Document	1 KB
	makssskskss11.xml	5/22/2020 7:18 AM	XML Document	6 KB
	makssskskss12.xml	5/22/2020 7:18 AM	XML Document	5 KB
	makssskskss13.xml	5/22/2020 7:18 AM	XML Document	3 KB
	makssskskss14.xml	5/22/2020 7:18 AM	XML Document	1 KB
	makssskskss15.xml	5/22/2020 7:18 AM	XML Document	1 KB

Figure 3.11 Data of YOLOv5 (Annotations)

The data of Resnet50 or MobileNetv2 described above is a dataset consisting of only faces divided into 2 folders (with\_mask and without\_mask). Meanwhile, YOLOv5's dataset also includes 2 folders, 1 folder for images and 1 folder for annotations corresponding to images. In YOLOv5's Image Dataset, an image may include one or more people, including both masked and non-masked people.

## b) Processing:

### Upload images and annotation to Roboflow:

We drag and drop both the images folder and annotations folder at the same time to Roboflow

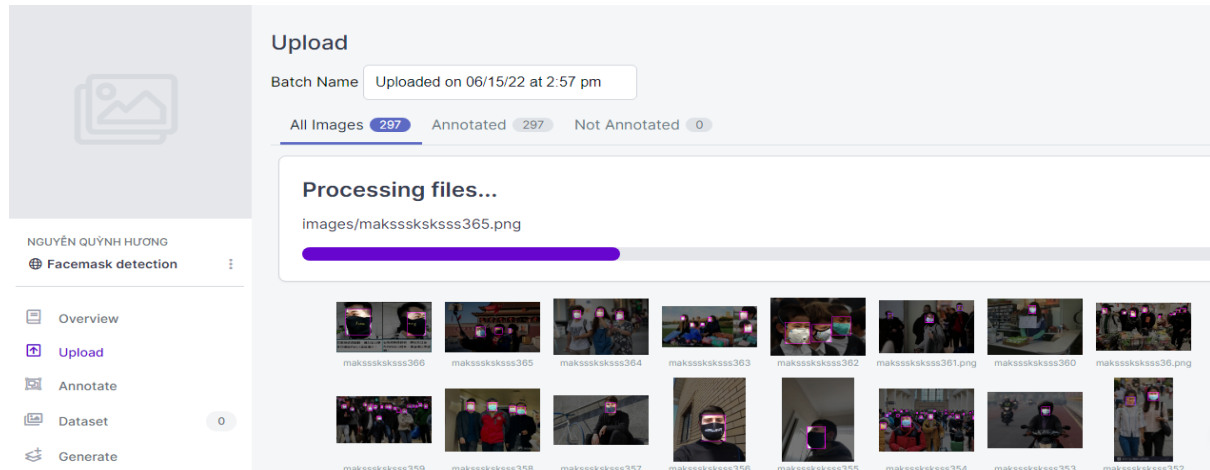


Figure 3.12 Upload images and annotation to Roboflow

**Split Data:** After we upload data, we will split dataset into Train/Valid/Test

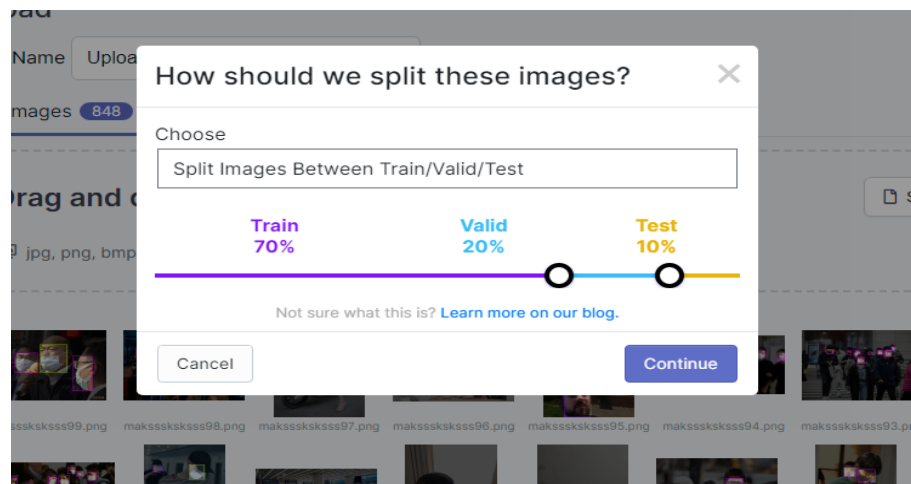


Figure 3.13 Split Data step with Roboflow

**Preprocessing:** In this step, because we just used 2 classes, with\_mask and without\_mask, I chose to modify class options to drop 1 class, mask\_worned\_incorrect. We also resized the image to 416x416 pixels.

3


### Preprocessing

Decrease training time and increase performance by applying image transformations to all images in this dataset.


Auto-Orient	Edit	×
Resize Stretch to 416×416	Edit	×
Modify Classes 0 remapped, 1 dropped	Edit	×
+ Add Preprocessing Step		

[Continue](#)

#### Resize



original



resized

#### Resize

Downsize images for smaller file sizes and faster training.

Stretch to

416 x 416

You might be resizing your images incorrectly. [↗](#)

Considerations for choosing the optimal computer vision resize settings.

[via Roboflow Blog](#)

[Cancel](#)[Apply](#)

#### Modify Classes

#### Modify Classes

Remap and exclude class labels

INCLUDE	CLASS NAME	OVERRIDE
<input type="checkbox"/>	mask_worned_incorrect	<input type="text"/>
<input checked="" type="checkbox"/>	with_mask	<input type="text"/>
<input checked="" type="checkbox"/>	without_mask	<input type="text"/>

[Cancel](#)[Apply](#)



Figure 3.14 Preprocessing step

**Generating data:** Because the initial dataset is images with PASCAL VOC-formatted bounding boxes. Therefore, for using this dataset for YOLOv5 generated new data with YOLOv5 Pytorch format as figure beblow:

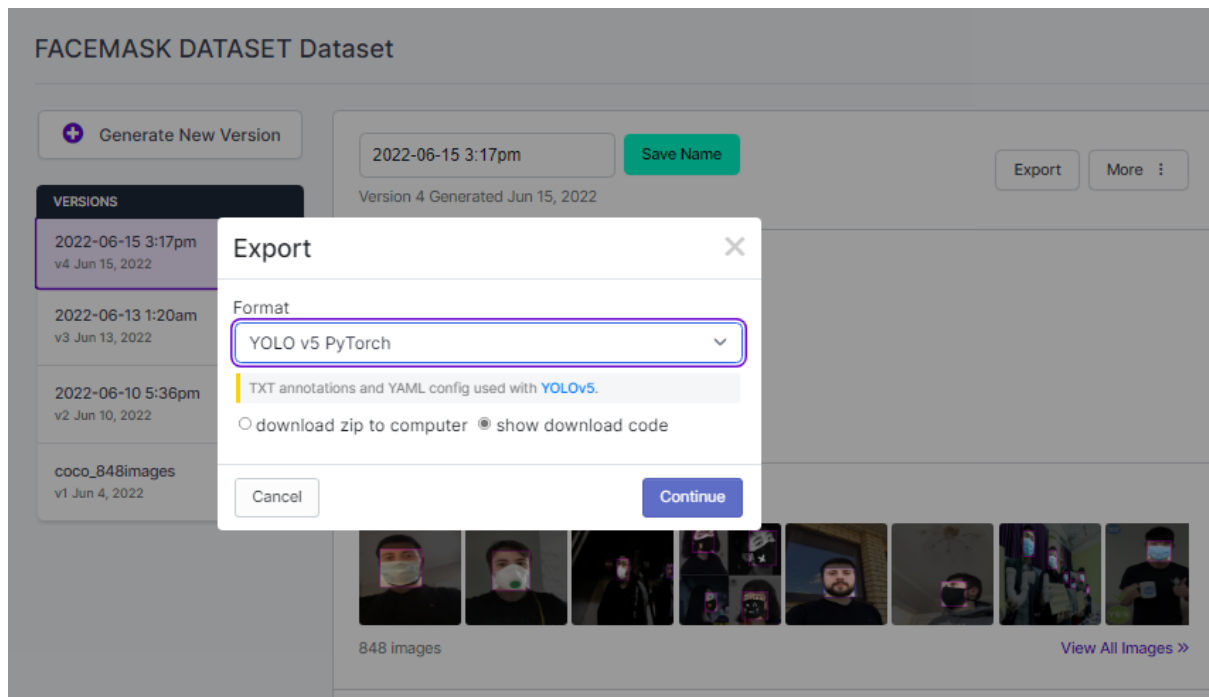


Figure 3.15 Generating data step

### c) Building Model

The model is built in six steps: producing the training image, using YOLOv5 to create the base model, adding model parameters, compiling the model, training the model, and lastly storing the model for use in the next process.. In this step, we have trained the model with four weights: YOLOv5s, YOLOv5m, YOLOv5l, YOLOv5x.

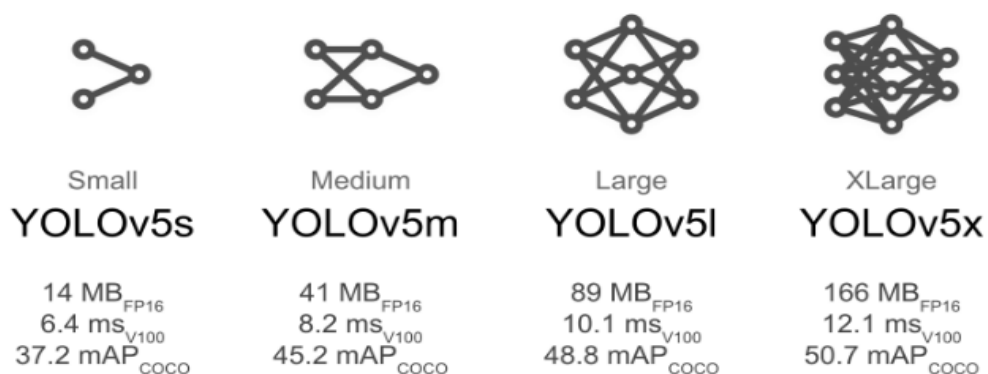


Figure 3.16 Four weights of YOLO: YOLOv5s, YOLOv5m, YOLOv5l, YOLOv5x

YOLOv5s: It is the small model in the family, with about 7.2 million parameters, and is suited for CPU-based inference.

YOLOv5m: With 21.2 million parameters, this is a medium-sized model. It's probably the best model for a wide range of datasets and training since it attempts to reach a good balance between precision and quickness. YOLOv5l: The YOLOv5 family's large model, with 46.5 million parameters. It's useful for datasets requiring the detection of tiny things.

YOLOv5x: It is the biggest of these models, as well as having the greatest mAP. Although it is slower than the others and includes 86.7 million parameters.

**d) Choosing model:**

After the previous step, we used tensorboard to analyze the results and chose the model with the best performance. Then, we trained the chosen best model with the same weight as in the previous step, but with a higher number of epochs. Lastly, saving the model after retraining it for use in future prediction processes.

**e) Applying the model:**

After training, the model may be used to detect face masks in both still images and streaming video. In order to identify faces, the video is read frame-by-frame. The process continues if a face is discovered. The stored model will be used to predict the new input data. Predict the output picture of a previously stored model that has been applied to the input image. On the video frame, the prediction % and whether or not the object is wearing mask will be labeled.