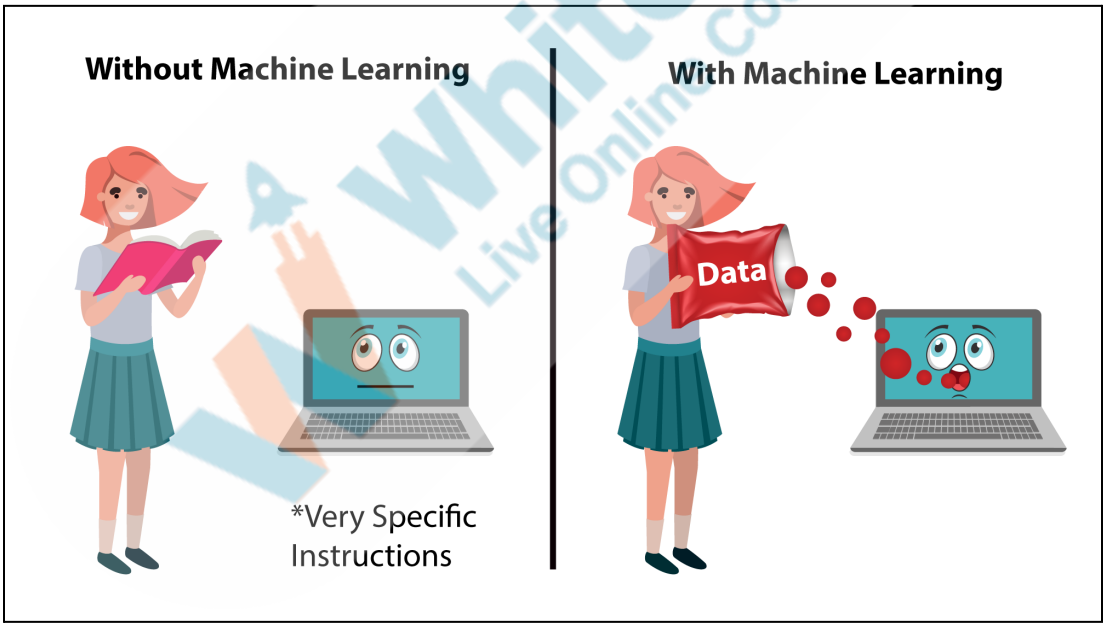
Topic	INTRODUCTION TO MACHINE LEARNING	
Class Description	**The student will learn about the machine learning process by classifying images.**	
Class	**PRO C110**	
Class time	**50 mins**	
Goal	• Learn about the basics of the machine learning process. • Learn about the image classification process.	

Resources Required	• Teacher Resources:
	  ○ Laptop with internet connectivity
	  ○ Webcam
	  ○ Earphones with mic
	  ○ Notebook and pen
	  ○ Smartphone
	• Student Resources:
	  ○ Laptop with internet connectivity
	  ○ Webcam
	  ○ Earphones with mic
	  ○ Notebook and pen	

Class structure	**Warm-Up** **Teacher-led Activity 1** **Student-led Activity 1** **Wrap-Up**	**10 mins** **10 mins** **20 mins** **05 mins**
Credit	Teachable Machines by Google. Tensorflow by Google Brain team. Keras by Google Engineer Francois Chollet.	
WARM-UP SESSION - 10 mins		

Teacher Starts Slideshow  **Slide 1 to 4** Refer to speaker notes and follow the instructions on each slide.	
Hey <student's name>. How are you? It's great to see you! Are you excited to learn something new today? **Following are the WARM-UP session deliverables:** • Greet the student. • Revision of previous class activities. • Quizzes.	**ESR**: Hi, thanks! Yes I am excited about it! Click on the slide show tab and present the slides
Teacher Action	**Student Action**
Hey <student's name>! It's great to see you! How are you?	**ESR**: Hi, I am good!
Can you tell me what we learned in the previous class? ***Note**: Encourage the student to give answers and be more involved in the discussion.* Great!	**ESR**: We learned about object detection.
Specific Instructions Vs Machine Learning: *Open Teacher Activity 1 to show the difference between giving specific instruction and machine learning while explaining.* Till now, we have been telling computers to do certain tasks using instructions in different programming, and it does the same, right?	**ESR**: Yes.

Now it's time that we should learn to: • **Teach the computer to do a task**, and then the • **The computer should be able to do the given task on its own once they learn**. That would be amazing, right? When computers/systems/machines can **learn from data**, **predict**(that can do the task on its own), and also **improve** the results if possible, we call it **Machine Learning**. **Machine Learning teaches computers to think as humans do**.	**ESR**: Yes.
	
Before we can teach a machine to think like humans, let's understand how humans think.	

Do you know how humans think?

We, humans, learn by:

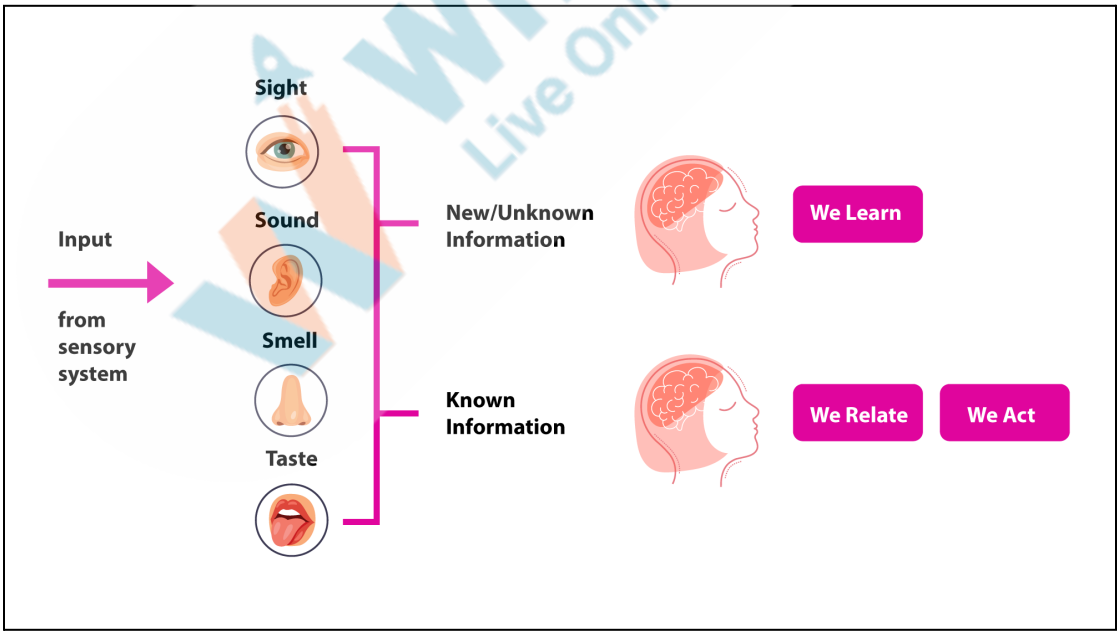
- Taking input data/information from our sensory system into our brain.
- The brain then retains the ideas and thoughts related to the input into the memory to learn based on the input information.
- The brain pulls the learned information out to figure out what and how to do certain tasks when needed.

To summarize, the human brain:

- **Takes raw data/information.**
- **Trains itself on the data.**
- **Takes action(making decisions) based on the learned data**.

Open Teacher Activity 2 to show the Human Learning Process.

ESR: Varied.

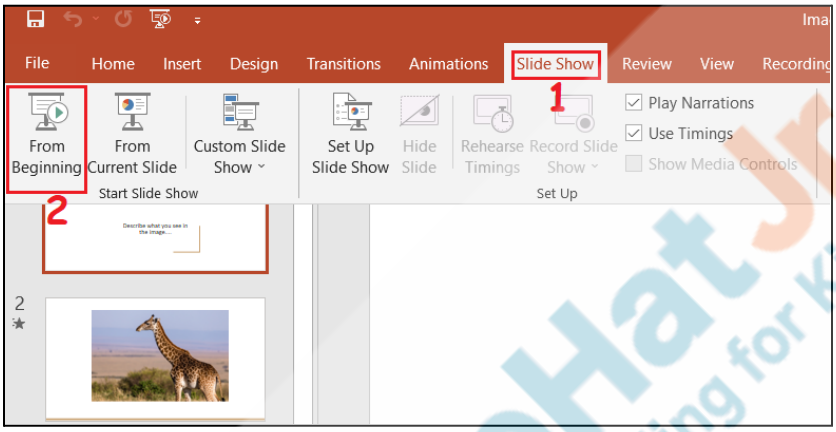
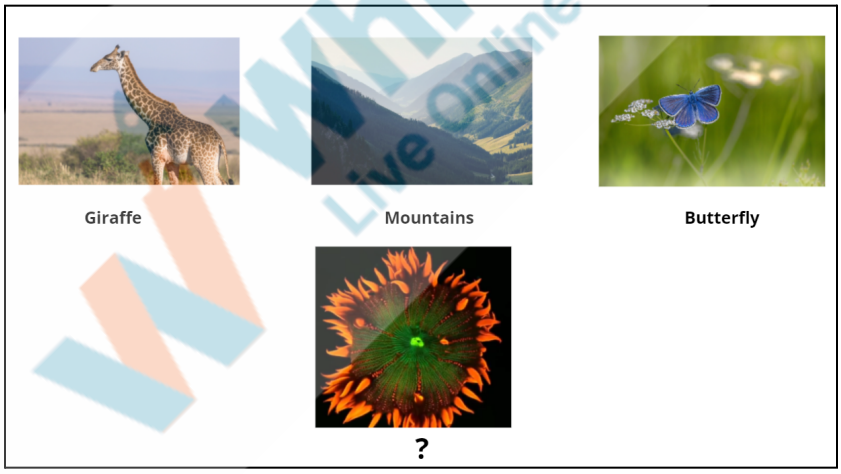


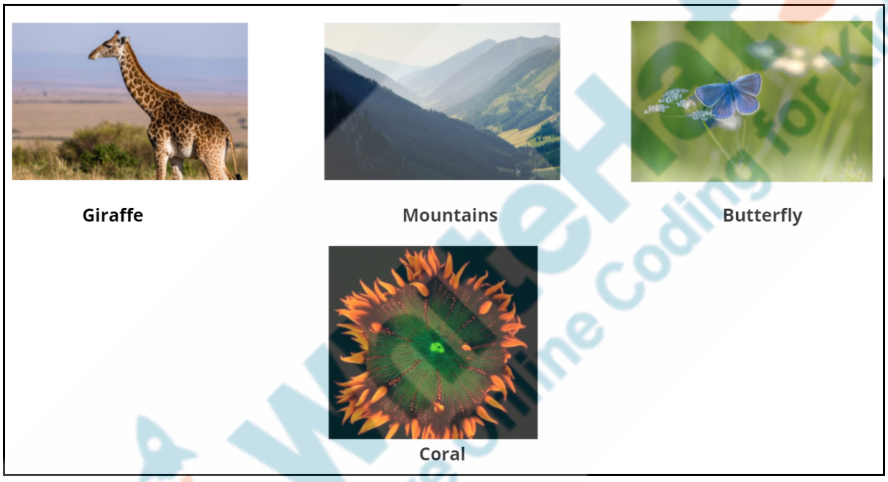
The human brain gets a lot of input information daily, that is why it does not retain everything that comes to it, that's why we forget, and we need to re-learn somethings again. We can follow a similar process to teach machines too. Although it would take years to teach machines to match human thinking levels, we can certainly begin to teach the machines to think! In today's class, we are going to learn about how we can teach the image classification process to computers/systems/machines. Are you excited? Let's get started.	**ESR:** Yes.
WARM-UP QUIZ Click on In-Class Quiz	
Continue WARM-UP Session **Slide 5 to 14** 	
Following are the session deliverables: ● Appreciate the student. ● Narrate the story by using hand gestures and voice modulation methods to bring in more interest in students.	
Teacher Ends Slideshow 	
TEACHER-LED ACTIVITY - 10 mins	
Teacher Initiates Screen Share	

ACTIVITY

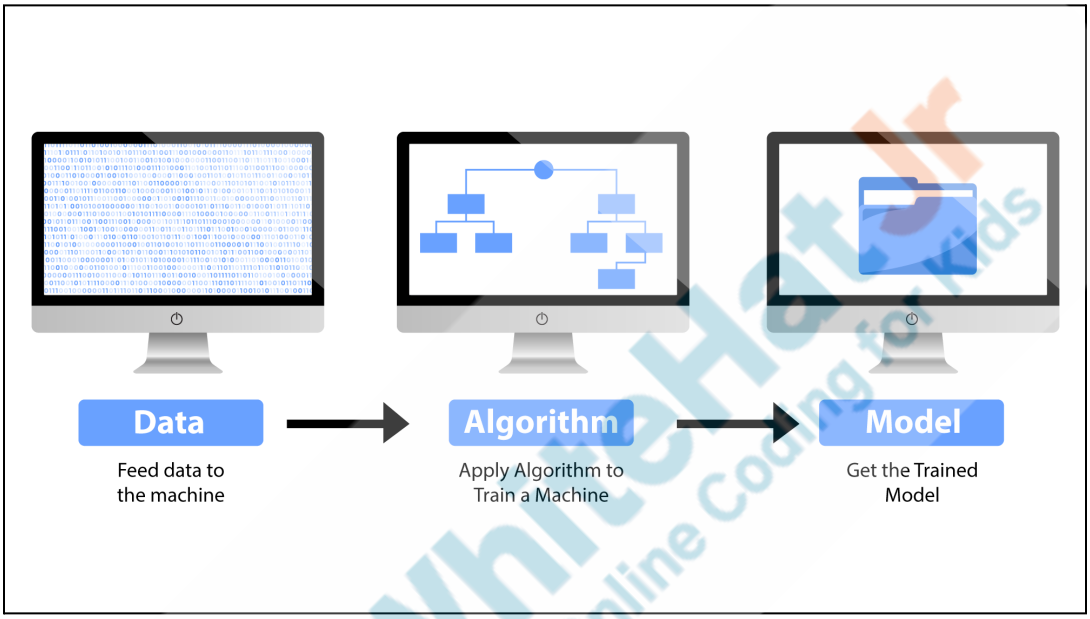
- **Understanding the image classification process.**
- **Train the model to classify images using Teachable Machines.**
- **Write a program to train the model to classify between faces with masks and without masks.**

Teacher Action	Student Action
Image Classification: Before we can teach image classification to a machine, let's start understanding how we identify images. Whenever we see an image, we can almost always tell what that image is. Can you tell me why? Well, whenever we see an image, we can search through our brain memory which was earlier trained on seeing the same picture or things in real life to identify the image. But you know, if we haven't seen something before then our brain would not have any memory of that, and we won't be able to identify what the image is. The brain will try to relate it to what it has already seen. **The brain can only tell what it has already seen!** Let's understand that a little more. Let's begin with looking at a few of the images first: **Image Identification by Human Activity**: *Open the link to the Teacher Activity 3. This will download the .ppt (or .pptx) file.*	**ESR**: Varied.

Open the downloaded file & start the Slideshow from the beginning: 1. *Click on "**Slide Show**".* 2. *Click on "**From Beginning**".* 	
Ask the student to guess what they see in the image and wait for the response before revealing the answer slide. 	
We could easily guess the first 3 images! But the 4 the image indeed looks like a flower as our brain already knows what a flower looks like.	

But this is an image of a CORAL. Had our brain seen this formation before, we would have known that it could be a coral, right? ***Note**: CORALS are marine animals generally found in the depth of an ocean. The amazing stone-like structures that are formed from the skeleton of the corals are called coral reefs.*	**ESR**: Yes.
	
We could see that we only identify based on what we have learned before. Now we have to teach the computer to learn to understand the different images. Do you remember how the computer sees an image?	**ESR**: Yes. Computers see every image in the form of pixels which are a range of numbers representing Red, Blue, and Green color bands.

Perfect! All we have to teach the computer to learn is a set of pixel values. The next time the computer sees that set of values, it would be able to identify the image. Can you try to think about what image classification could be? Great! And for machines**, image classification is the process of categorizing a set of pixels into different groups(or classes).** We will understand what image classification process we can follow while teaching machines. **Machine Learning Process**: Before we can understand the process of image classification, we need to understand a **few terminologies used in the Machine Learning field**: **Data**: The data is the information that is used as input for teaching(or training) the machines. **Algorithm**: The algorithm is a set of rules or procedures that are applied to the data to create a machine learning model. The algorithms are written to identify the patterns in the input data. The data can be in any form, text, images, or sounds. **Model**: The output that we get after applying algorithms to the data. The model can be saved as a file and used later	**ESR**: Learning how to classify images into different categories.

to recognize/predict the new pattern in new data and see how well the model identifies the new pattern. So, we feed the data to the machines and train them using some algorithms.	
	
We will understand more deeply about **data**, **machine learning algorithms**, and **model creation** in the upcoming classes. Now, can you tell me what input data we should use in teaching/training the image classification process to machines? Yes. Great! Anything else we should be telling the machine before teaching/training it?	**ESR**: The images. **ESR**: Varied.

Well, we should also tell different categories of the images beforehand too, to make the machine know what it looks like before it can predict, right?

To summarize, similar to the humans learning process, we need:

- Image input data with different categories.
- Train the machine using the input data.
- Use the trained information(i.e. model) to identify if the new image belongs to any trained category.

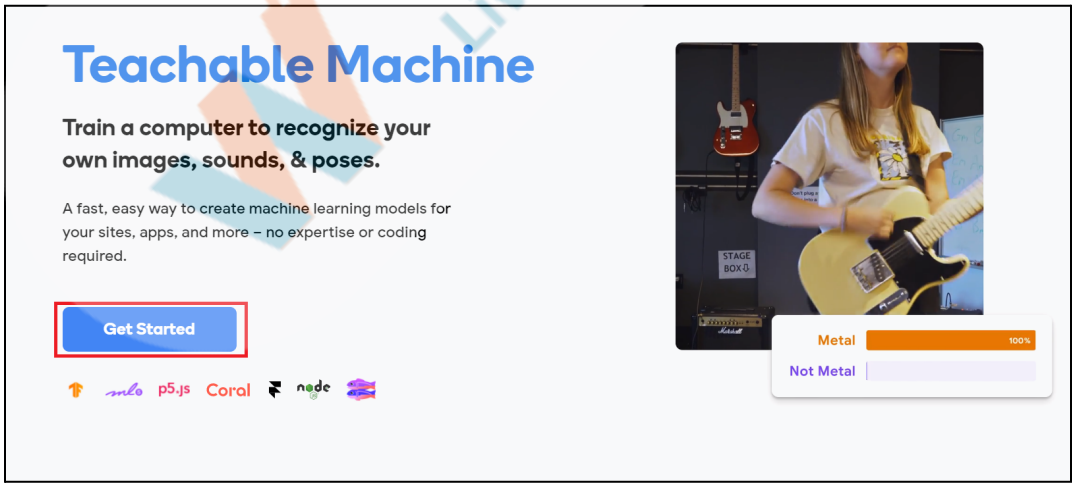
Image Classification Process in Machine Learning:

Let's try to teach the machine on detecting a face with a mask and a face without a mask.

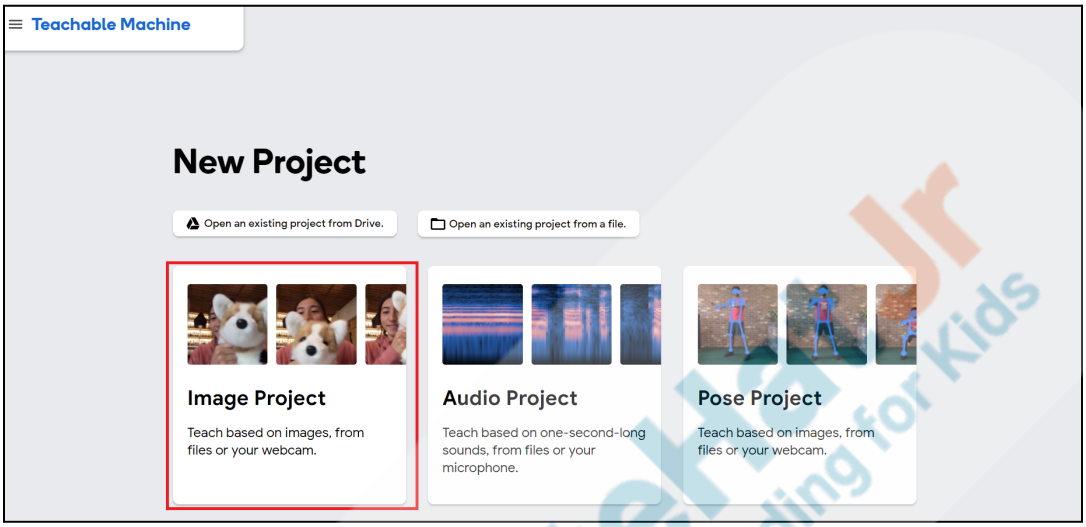
For this, we will use the Teachable Machine platform by Google(*Steps 1 to 5, Note that these steps can be completed in 2-5 mins maximum using Teachable machine*):

1. Open Teacher Activity 4 and click on "**Get Started**".

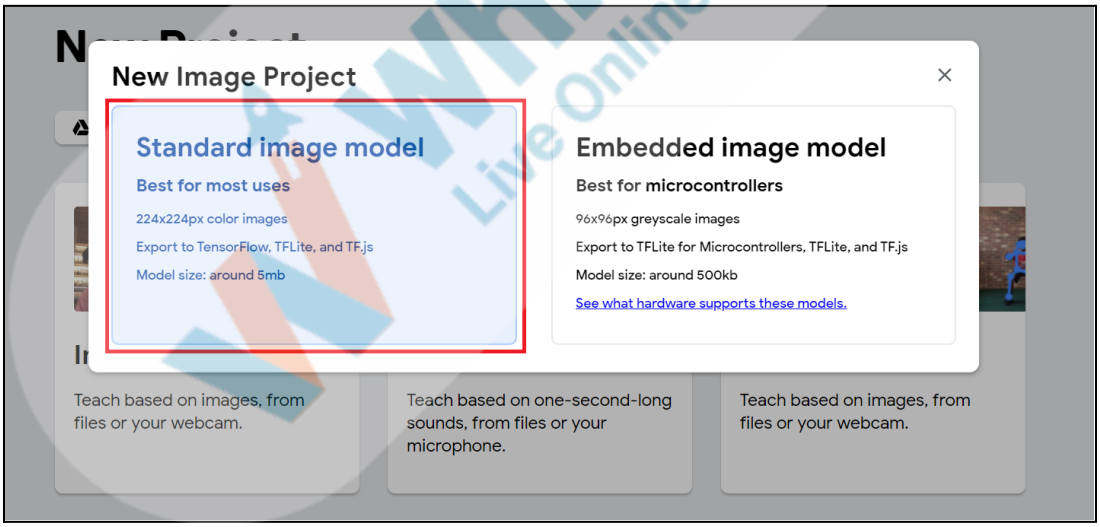
ESR: Yes.



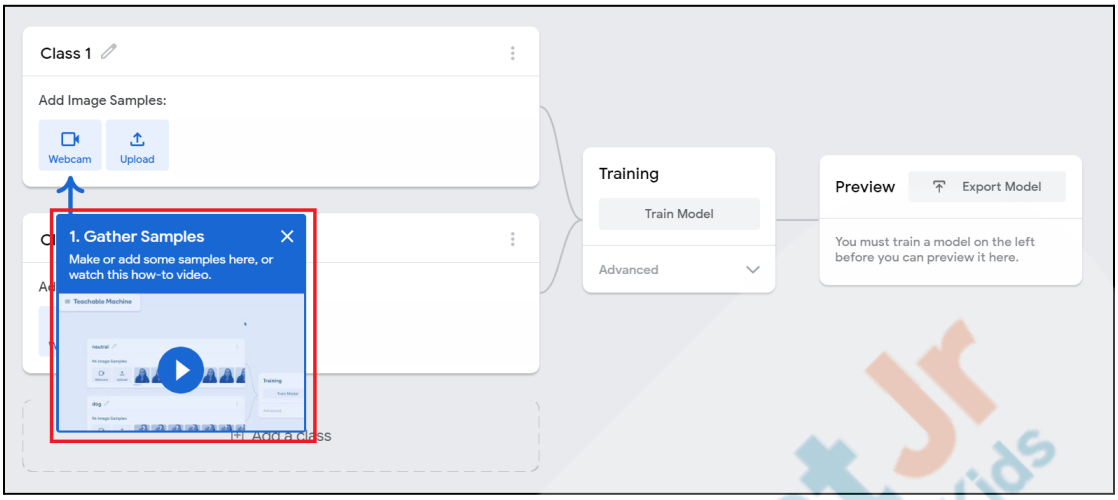
2. Click on the “**Image Project**”.
   Since we are working on the image classification project.



3. Click on “**Standard image model**”.



4. Close this “**1. Gather Samples**” popup. This is a short tutorial video that can be closed.



Train the Model:

Now let's understand how to use the Teachable Machine platform to teach/train a model.

Do you remember the process we just discussed to teach/train machines to identify different images?

Superb!

ESR: Yes.
We need:

- Image input data with different categories.
- Train the machine using the input data.
- Use the trained information(i.e. model) to identify if the new image belongs to any one of the trained categories.

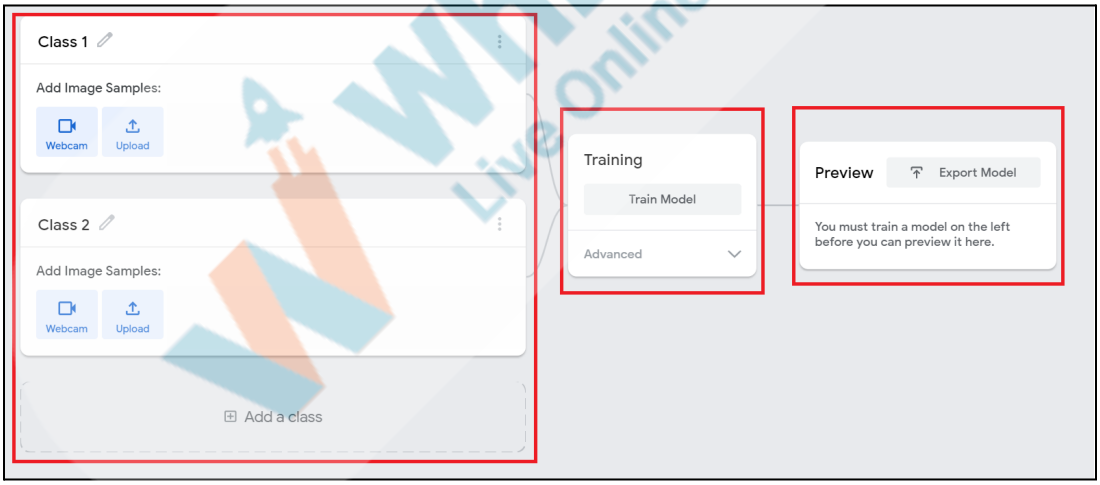
On the Teachable Machine platform, we have 3 sections for this process:

- **Add a class**: The class represents the different categories of the image to be used for teaching/training the machine.

  We can train the model with as many classes as we want, but training a model can sometimes be time-consuming and costly. We should only train the machines based on our requirements.

- **Training**: The model is trained using some machine algorithm.

- **Preview**: To see the trained model and export to save it in a file. This can be used later to identify the class of the new image(s).



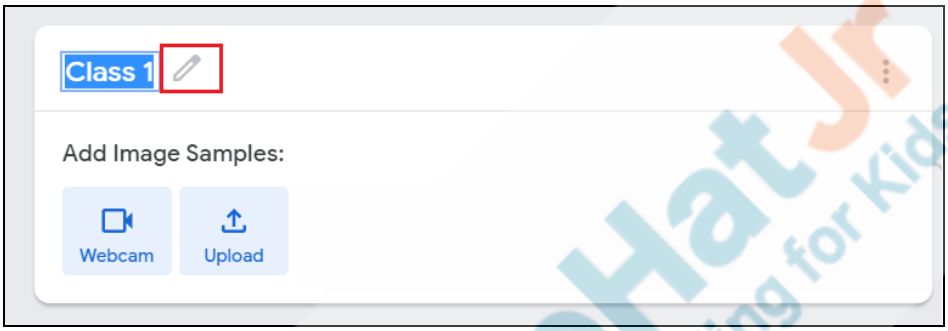
Since we want to teach/train the machine on detecting a face with a mask and a face without a mask, can you tell me how many classes we will need to train this model?

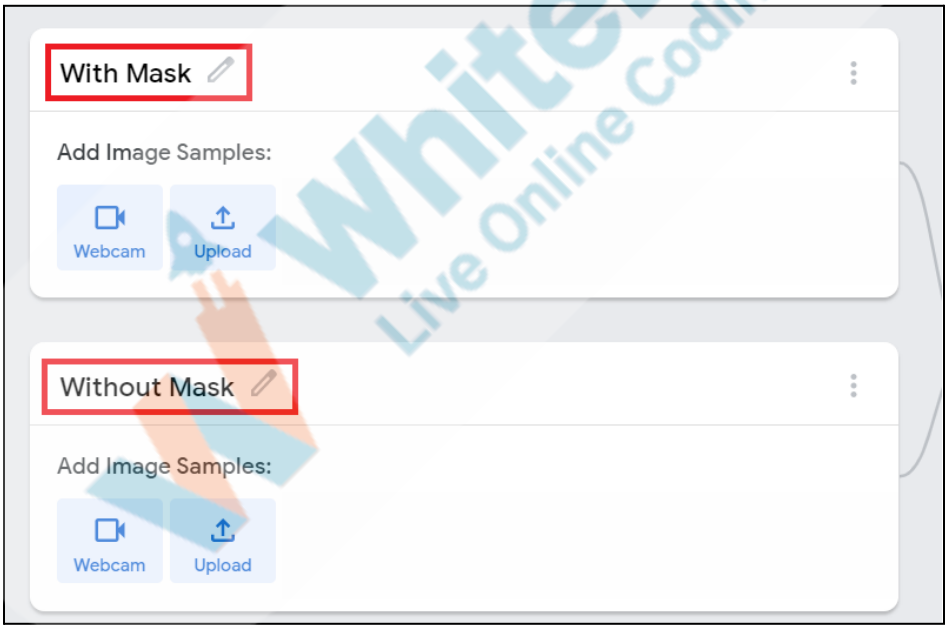
ESR: Two classes.

Superb!

Whenever we want to train a machine, we should keep the names of the classes relevant to what our machine model is going to do.

Click on the pencil icon to edit the class names.





Now we should create input data for the training of the model.

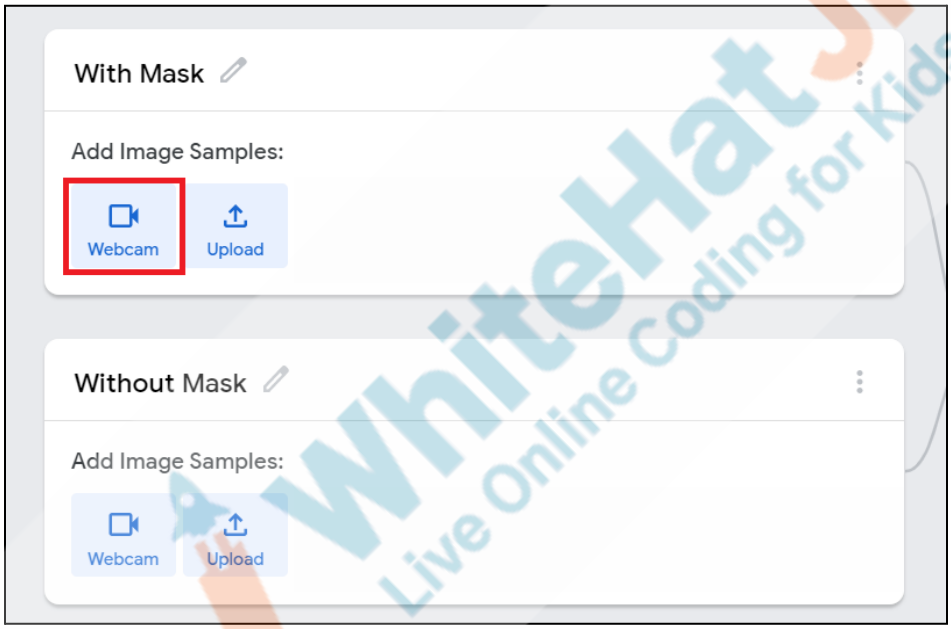
The input data that we use to teach/train a machine is

called “**Training Dataset**”.

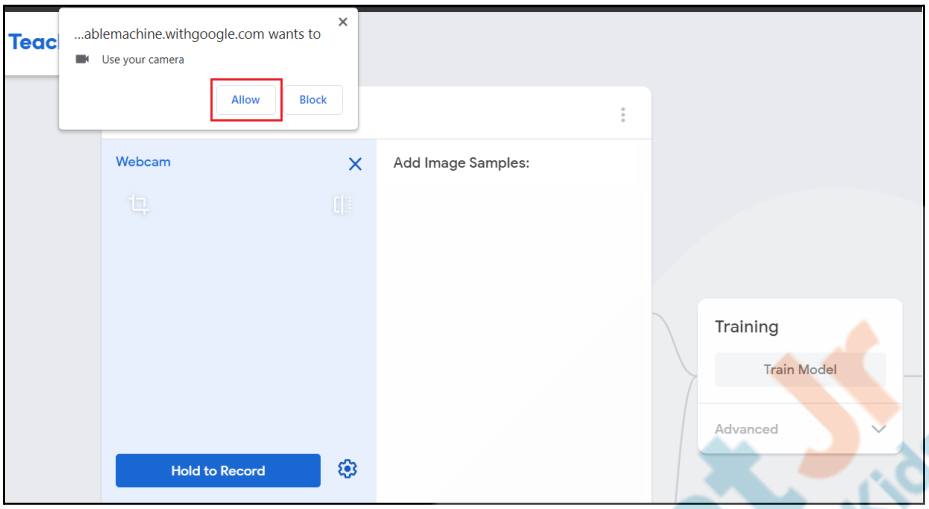
We can use the “**Webcam**” to capture images or “**Upload**” if we already have images, to create a training dataset.

Let’s capture images of the face covered with the mask and without the mask:

1. Click on “**Webcam**” to capture images. This will ask for permission to use your camera.

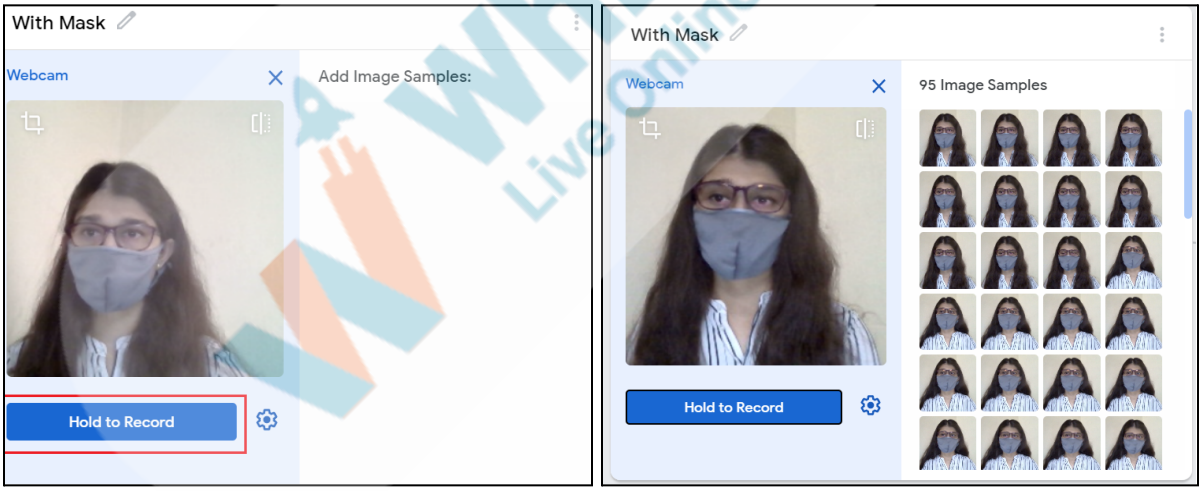


2. Click on “**Allow**” to give the camera’s permission



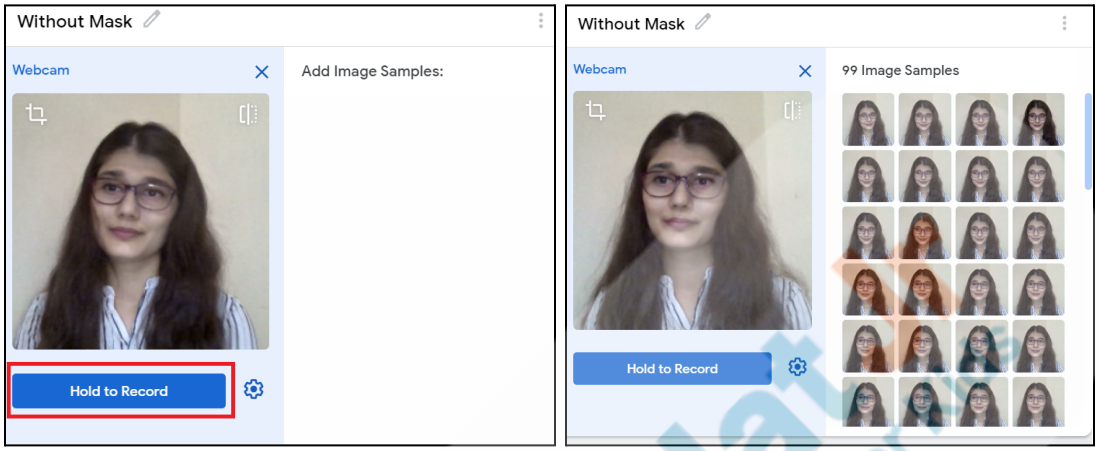
3. Click on "**Hold to Record**" and keep to the button in the clicked state to capture around 50-100 images **with a mask**.

***Note**: Turn your face around to capture images from different face angles.*

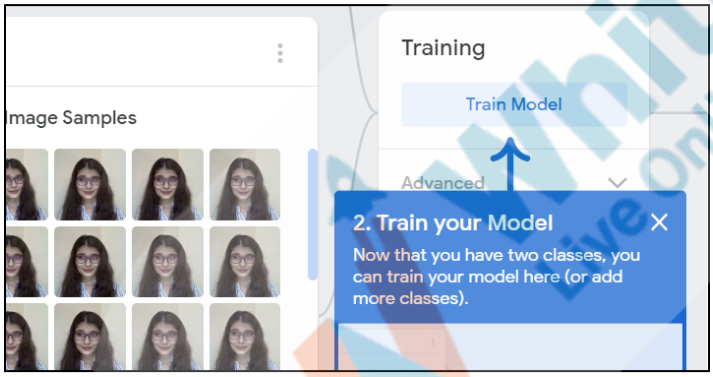


4. Click on "**Hold to Record**" and keep to the button in the clicked state to capture around 50-100 images **without the mask**.

***Note**: Turn your face around to capture images from different face angles.*

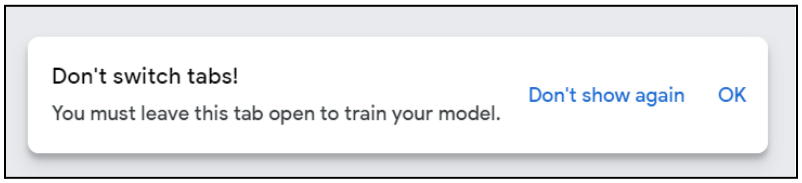


Close this "**2. Train your Model**" popup. This is a short tutorial video that can be closed.

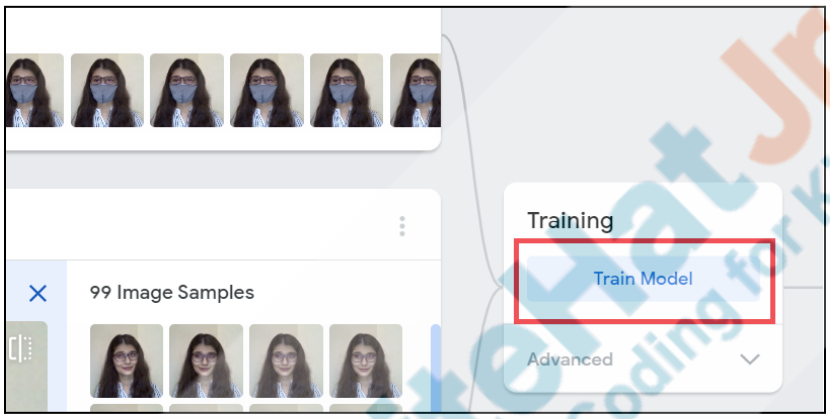


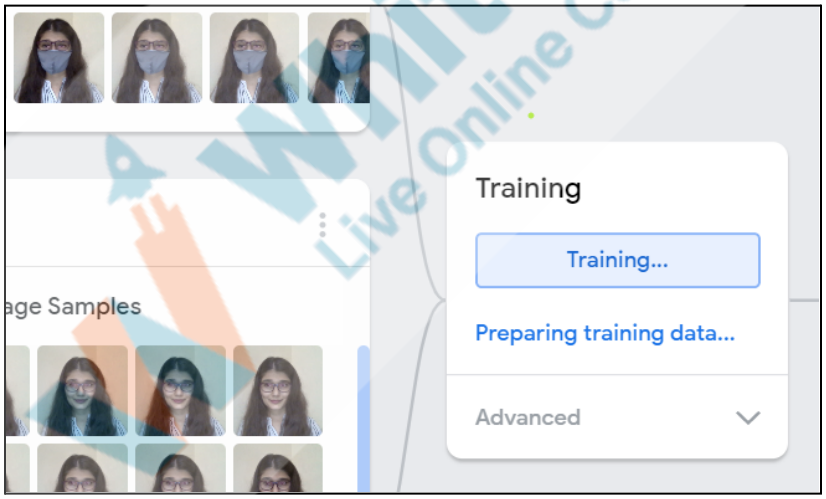
5. Click on "**Train Model**"

***Note 1**: Stay on the browser tab while the model is being trained.*



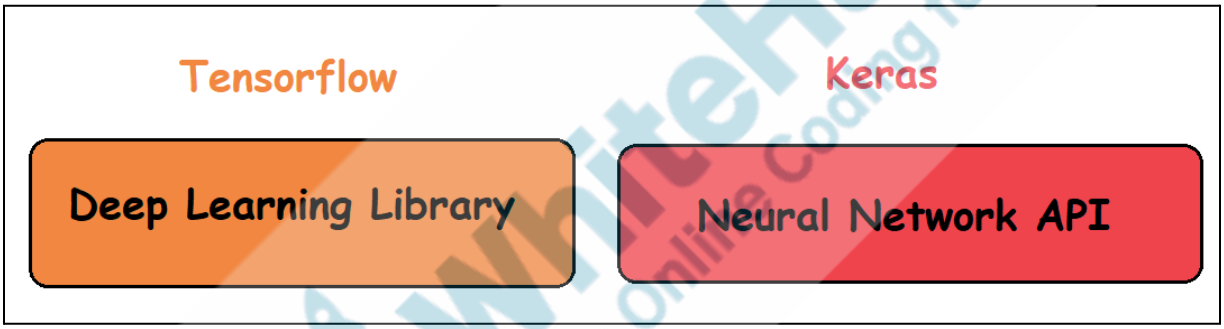
***Note 2**: The model "**Preview**" mode opens automatically once the model is trained.*

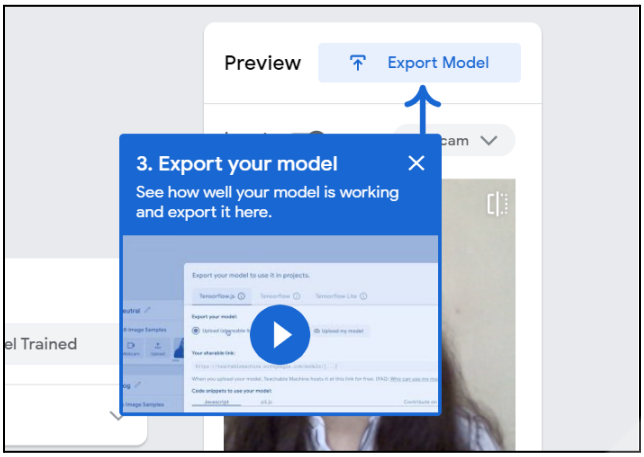




The models are trained using certain machine learning libraries, which contain machine learning algorithms to train the machine.

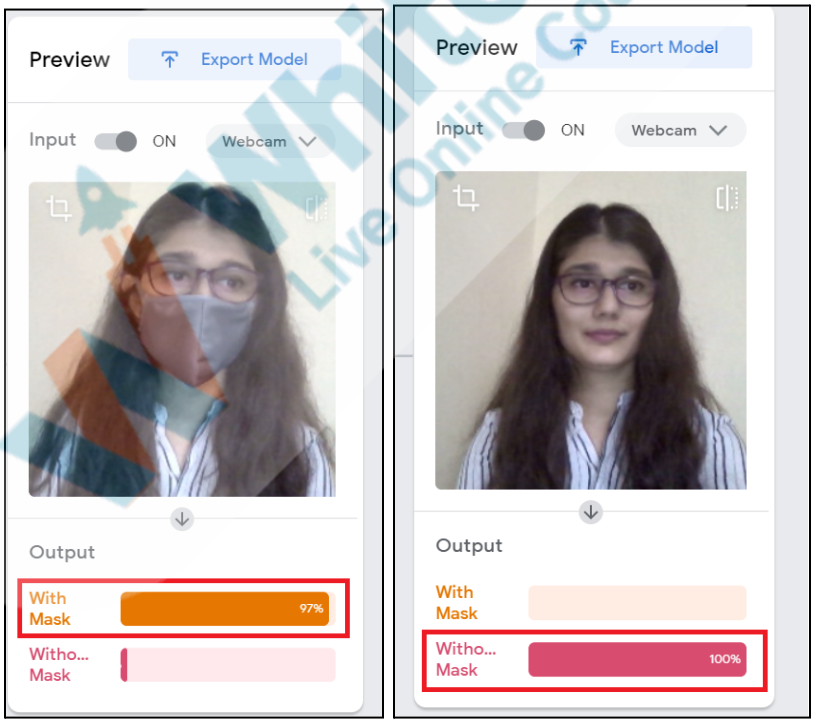
There are many machine learning libraries that can be used to train machines.

Teachable Machine is using **Tensorflow** and **Keras** to apply machine learning algorithms to the input image dataset. We will study a few machine learning libraries in detail in the upcoming classes. For now, we can understand that Tensorflow and Keras are the machine learning libraries that can be used to teach machines how to identify images. Once you click on the "**Train Model**" the machine algorithm present in the Tensorflow(and Keras) library is applied to teach the machine to identify the face with a mask and a face without a mask.	
	
Close this "**3. Export your Model**" popup. This is a short tutorial video that can be closed.	

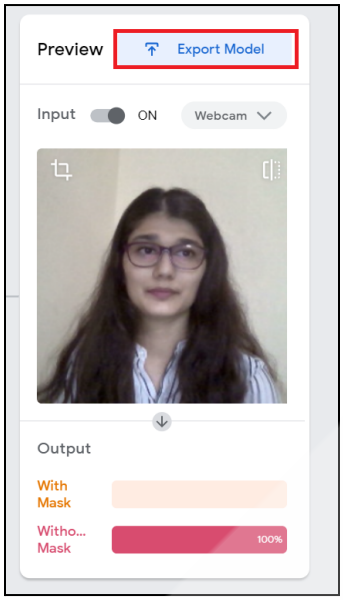


The model "**Preview**" mode opens automatically once the model is trained.

Check the results in the Preview for different classes before exporting the model.

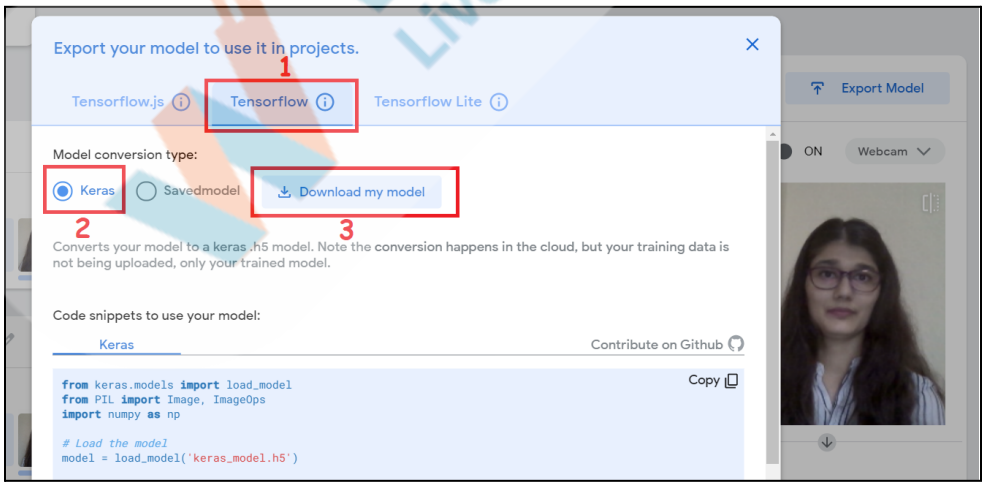


6. Click on “**Export Model**” to download the trained model.

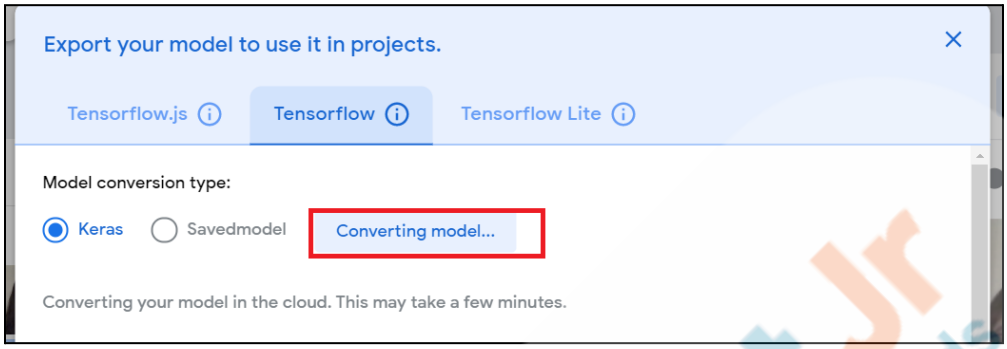


7. Click on “**Tensorflow**”.
8. Click on “**Keras**”.
9. Click on “**Download my model**”.

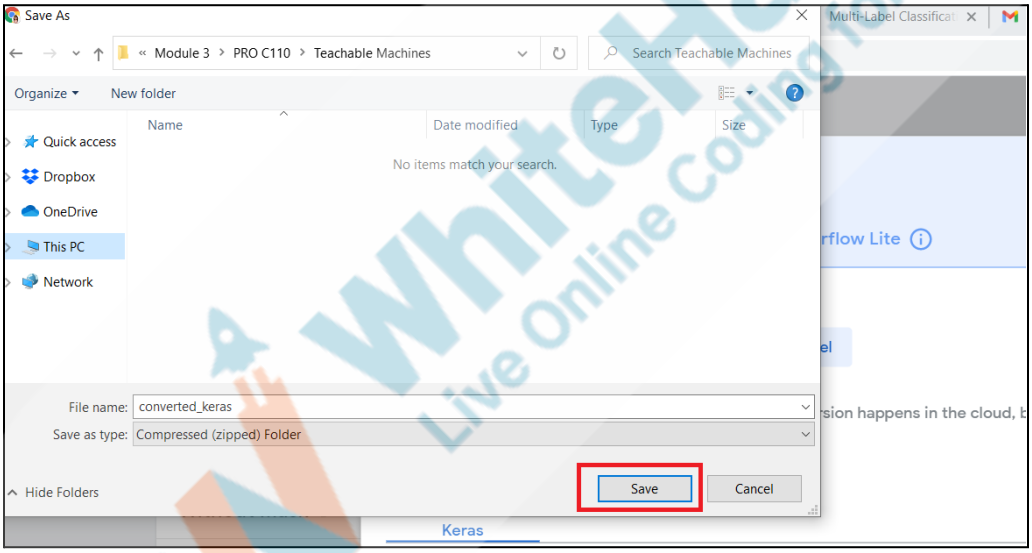
A **zipped file** will be downloaded.



Converting models may take some time. If in case, the model fails to convert. Repeat the steps to capture the images and train the model again, and then export.



*Choose the folder to save the model files and click on "**Save**".*

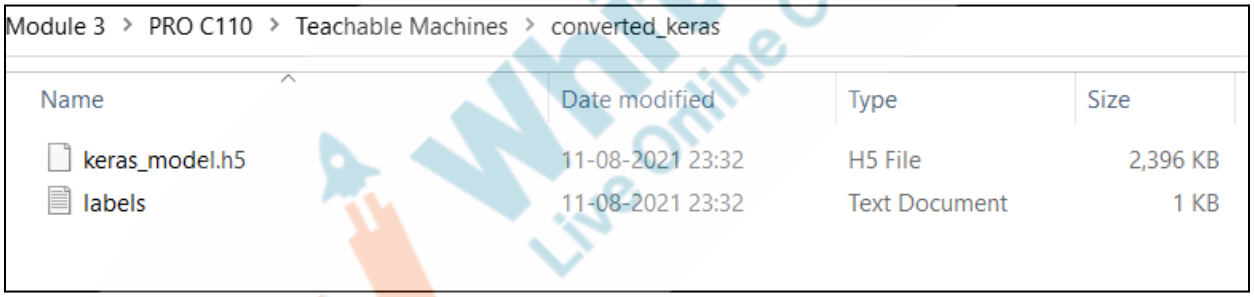


*The model files will be saved as a zipped file named "**converted_keras.zip**".*

We have successfully downloaded the trained model. Now let's find out if the model is working. For this:

10. Go to the folder where the model was downloaded(saved) in your computer and **Extract** all the files.

The files extracted from download a **zipped file** we will be:
keras_model.h5
labels.txt

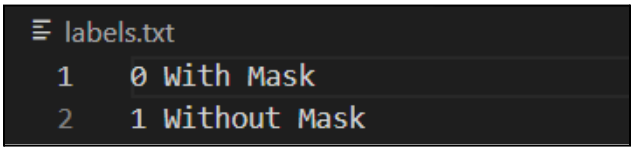


The **trained model file** is in **.h5 file format** which stands for Hierarchical Data Format version 5. This format stores data in a tree-like structure. This format is designed to store a large amount of data. For now, we do not have to understand the depth of how data is stored in a hierarchical format.

In this model file, we have all information stored which can help the computer identify images with people wearing masks and people not wearing masks.

The **labels file** is in **.txt format**. This keeps the information of the classes with numbers assigned as the label.

*Open the **labels.txt** to see these labels.*



In our model:

Class 1 is "**With Mask**" and **0** has been assigned as the label to this class during the model training by Teachable Machine.

Class 2 is "**Without Mask**" and **1** has been assigned as the label to this class during the model training by Teachable Machine.

Once the model is trained, what do you think we should do as a next step?

Remember what happens once you learn something at school, your school takes a test to see what you learned. Similarly, as we have already taught the machine to identify the face with a mask or without a mask, and now it is time to take its test!

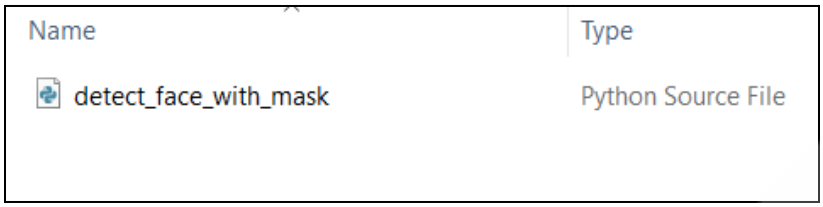
Test the working of the model:

To test the working of the model:

- We can write a program to capture video frames using OpenCV.
- And then integrate the model to check if it is able to identify the faces correctly.

ESR: Test if the model is able to identify the face with a mask or without a mask correctly!

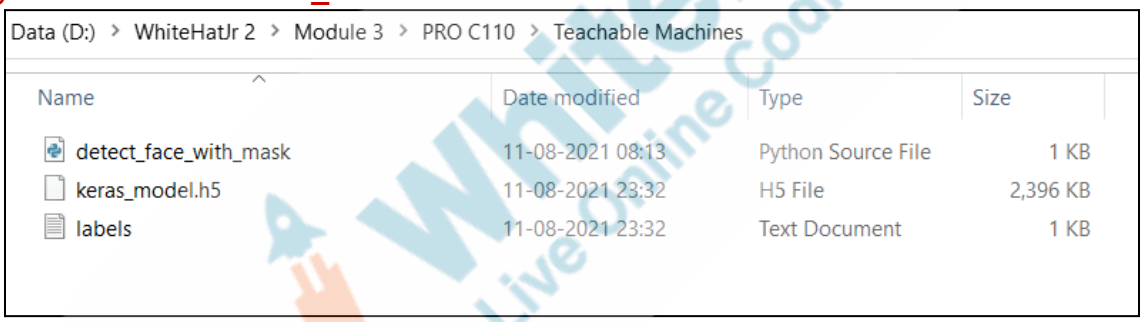
Open Teacher Activity 5 to download the Boilerplate Code and extract files. This will contain a Python file ***detect_face_with_mask.py***



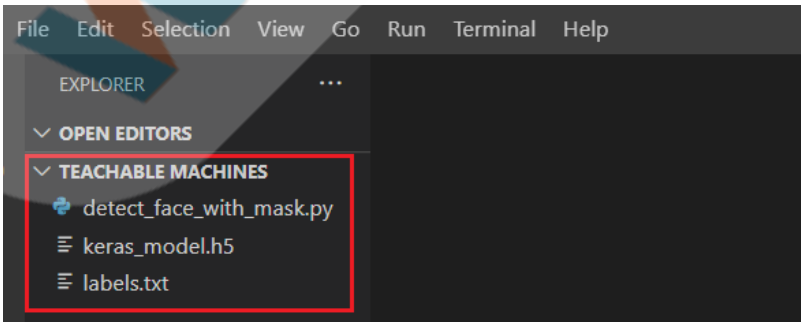
1. Copy and Paste the **keras_model.h5** & **labels.txt** into **the** same folder.

2. Open the folder with **detect_face_with_mask.py**, **keras_model.h5** & **labels.txt** in Visual Studio Code Editor.

Copy and Paste the ***keras_model.h5*** *&* ***labels.txt*** *into* ***the*** *same folder.*



Open the folder(with ***detect_face_with_mask.py*** *,* ***keras_model.h5*** *&* ***labels.txt****) in Visual Studio Code Editor.*



The boilerplate code has the program file (**detect_face_with_mask.py**) which contains program:

- To open the webcam
- Capture the video frames using OpenCV.
- Show the video frames using OpenCV in a new pop window.
- Close the pop window by pressing spacebar(key value 32).

```
import cv2
import numpy as np

video = cv2.VideoCapture(0)

while True:

    check,frame = video.read()

    cv2.imshow("Result",frame)

    key = cv2.waitKey(1)

    if key == 32:
        print("Closing")
        break

video.release()
```

Now we will be learning to use the model file to detect the face with and without a mask when the video will be captured through a webcam!

For this, we need to install the **tensorflow** module for our project.

Before we do that, we are going to learn about creating virtual environments in Python.

Virtual Environments in Python is used to manage packages/modules specific to only one project.

Can you try to think why you would need virtual environments?

If we keep all the libraries that we need in different projects stored at one place in our computer(which is the default environment in the system) this can break the system tools and might affect the working of other projects.

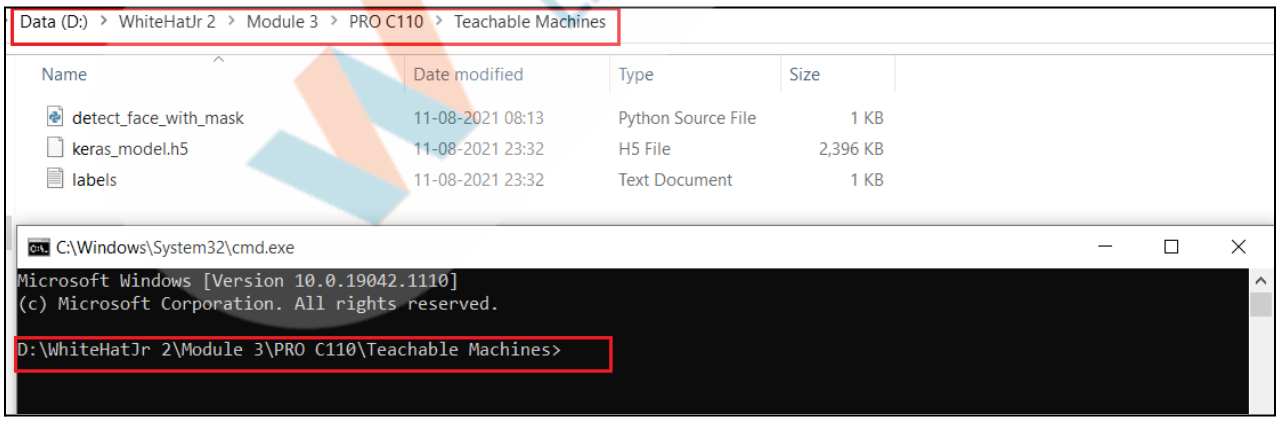
Every project might need a different set of libraries. That's why we should create virtual environments with a specific version of Python and all the dependencies that we need for that project to avoid problems in the future while working on different projects.

In Python, we can use the built-in **venv** module to create virtual environments.

Let's first quickly create a virtual environment in the working folder to install the Python modules that are needed for this project:

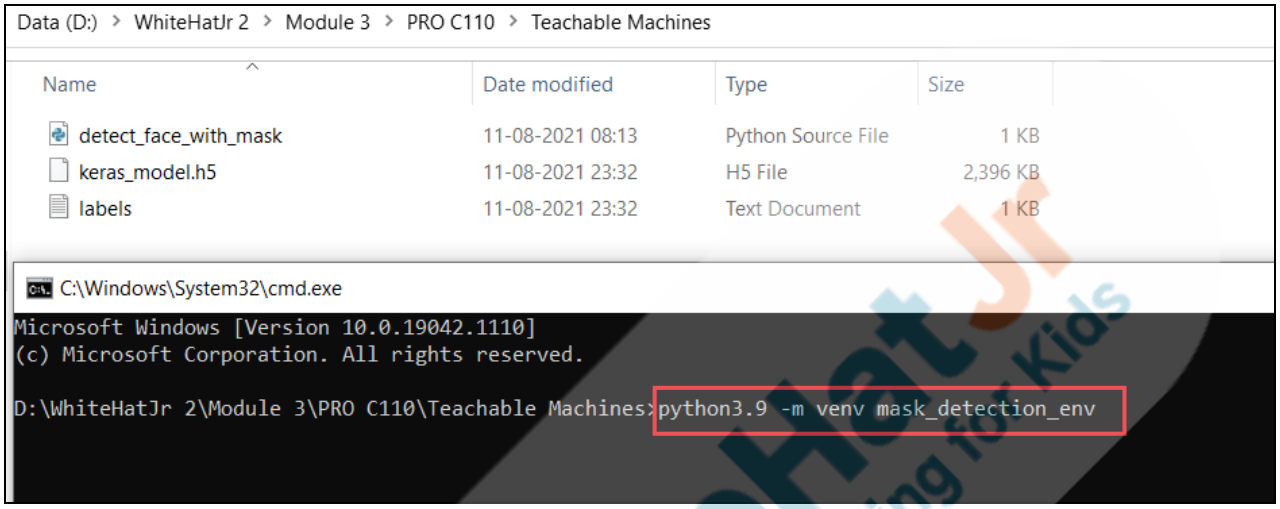
Open Command Prompt(cmd) in Windows/Terminal on Mac and traverse to the working folder.

ESR: Varied.

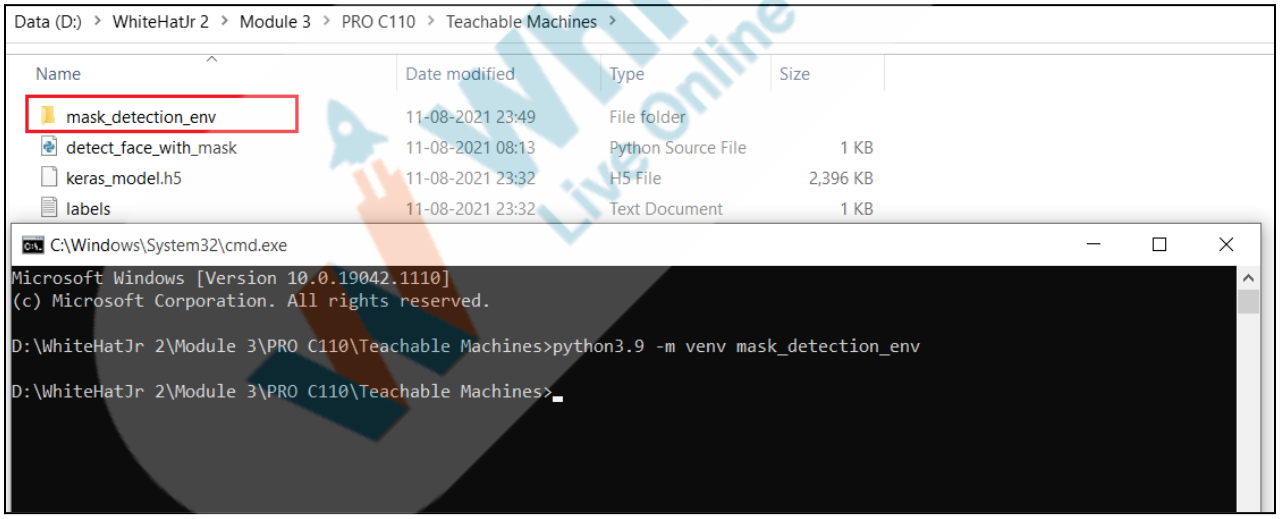


To create virtual environment run command(Windows/Mac):
 	
`python3.9 -m venv <name_of_the_environment>`
 	
Note that in the above we are specifying the version of the python we are using, this is an application for a system having multiple versions of the Python installed.
 	
If your system has only one Python version installed use the below command(without the version) to create virtual environment(Windows/Mac):
 	
`python -m venv <name_of_the_environment>`
 	
We are using the **venv** built-in **module**. The "**-m**" in the command indicates the module.
 	
When the "**-m**" is used with a command on the command-line interface, followed by a **<module_name>**, it allows the module to be executed as an executable file.
 	
Once the command is executed, a folder is created with **<name_of_the_environment>** as folder name.	

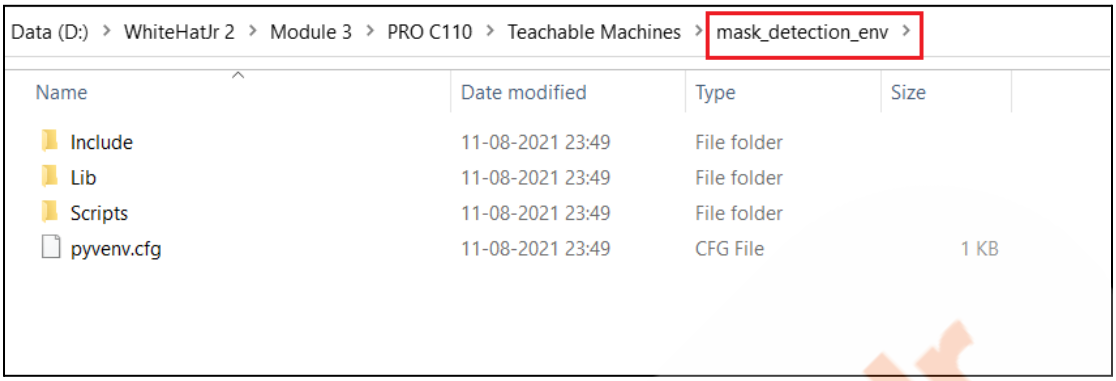
*Run the command to create a virtual environment with the name as "**mask_detection_env**". The environment is user-defined. We can keep the name as we want relevant to our project.*



After the command is executed, a folder virtual environment folder is created.

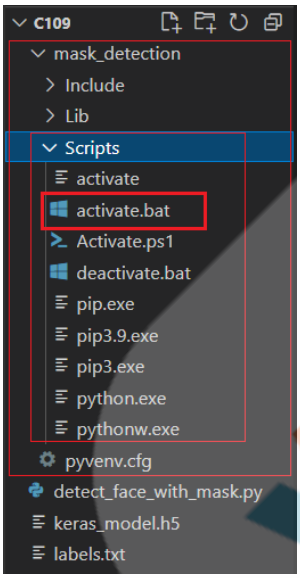


Virtual environment folder has some files to manage the packages/modules specific to this project.



Once the environment is created, we need to **activate** it:

Inside the virtual environment folder we have **Scripts** folder which contains the **activate.bat** file to activate the virtual environment.



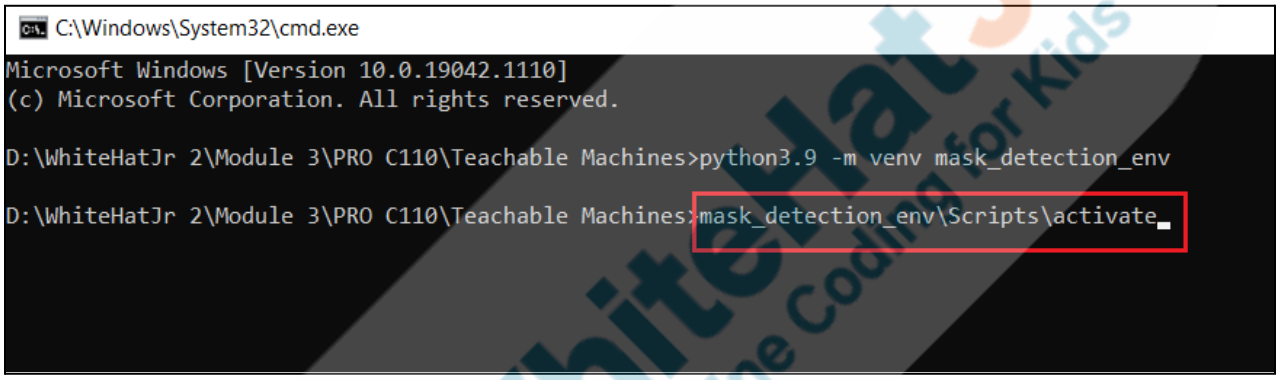
To **activate** the virtual environment command to be used on **Windows**:

```
<name_of_the_environment>\Scripts\activate
```

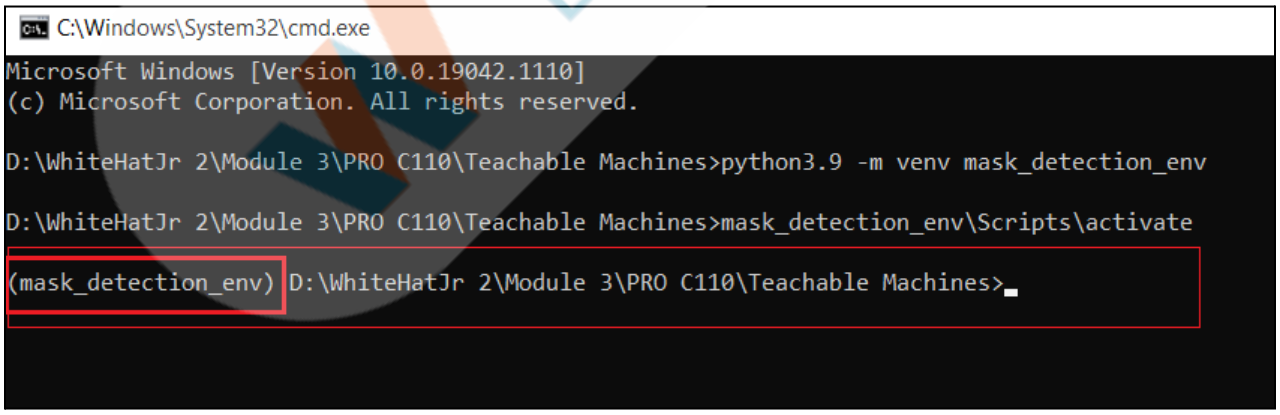
Once the command is executed, the virtual environment is activated. We can confirm this by checking the name of the virtual environment at the beginning of the working folder path.

To **activate** the virtual environment command to be used on **Mac**:

```
source <name_of_the_environment>/bin/activate
```



Once the command is executed, the virtual environment is activated. We can confirm this by checking the name of the virtual environment at the beginning of the working folder path.



Once that environment is activated, everything is set to install specific libraries for our project.

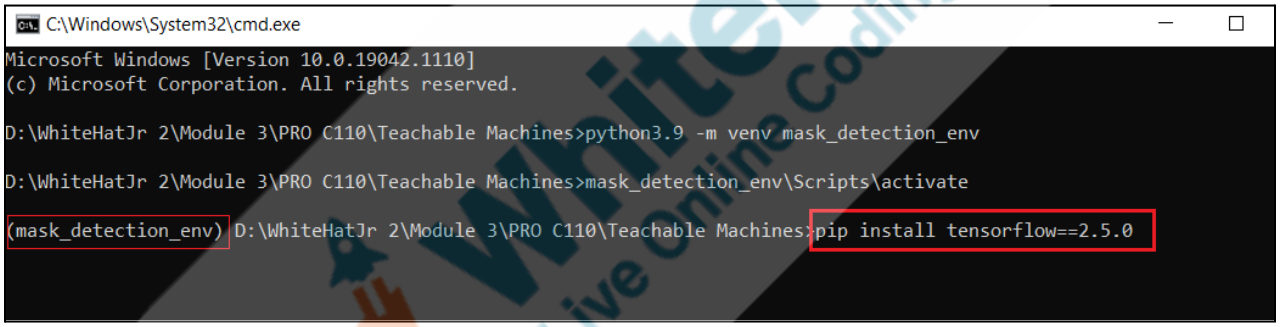
Now we can install the **tensorflow** module using **pip** for our project and import it.

Run **pip** command to install **tensorflow**:

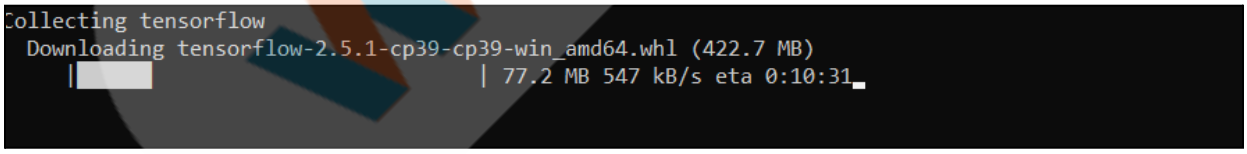
```
pip install tensorflow==2.5.0
```

***Note 1**: The tensorflow version 2.5.0 works with python version 3.9. Make sure Python version 3.9 or higher is installed.*

***Note 2**: Use **pip install tensorflow_cpu=2.5.0** if the above is not working.*



Downloading in progress…



All the libraries required with tensorflow will be installed.

```
     |                                | 59 kB 1.7 MB/s
Collecting oauthlib>=3.0.0
  Downloading oauthlib-3.1.1-py2.py3-none-any.whl (146 kB)
     |                                | 146 kB 6.8 MB/s
Using legacy 'setup.py install' for termcolor, since package 'wheel' is not installed.
Using legacy 'setup.py install' for wrapt, since package 'wheel' is not installed.
Installing collected packages: urllib3, pyasn1, idna, charset-normalizer, certifi, six, rsa, requests, pyasn1-
t, tensorboard-data-server, protobuf, numpy, markdown, grpcio, google-auth-oauthlib, absl-py, wrapt, typing-ex
y, h5py, google-pasta, gast, flatbuffers, astunparse, tensorflow
    Running setup.py install for wrapt ... done
    Running setup.py install for termcolor ... done
Successfully installed absl-py-0.13.0 astunparse-1.6.3 cachetools-4.2.2 certifi-2021.5.30 charset-normalizer-2
cio-1.34.1 h5py-3.1.0 idna-3.2 keras-nightly-2.5.0.dev2021032900 keras-preprocessing-1.1.2 markdown-3.3.4 nump
.26.0 requests-oauthlib-1.3.0 rsa-4.7.2 six-1.15.0 tensorboard-2.6.0 tensorboard-data-server-0.6.1 tensorboard
.3 urllib3-1.26.6 werkzeug-2.0.1 wheel-0.37.0 wrapt-1.12.1
```

***Note**: Since we have created a separate virtual environment for this project, make sure to install the **opencv-python** & **numpy** module also in the virtual environment using:*

*eg: **pip install opencv-python.***

```
(mask_detection_env) D:\WhiteHatJr 2\Module 3\PRO C110\Teachable Machines>pip install opencv-python
Collecting opencv-python
  Using cached opencv_python-4.5.3.56-cp39-cp39-win_amd64.whl (34.9 MB)
Requirement already satisfied: numpy>=1.19.3 in d:\whitehatjr 2\module 3\pro c110\teachable machines\mask_detection_env\
lib\site-packages (from opencv-python) (1.19.5)
Installing collected packages: opencv-python
Successfully installed opencv-python-4.5.3.56
```

*After successful installation, import the **tensorflow** library in the code file. (The cv2 & numpy are part of the boilerplate code).*

```
import cv2
import numpy as np

import tensorflow as tf
```

Now we **load_model()** to load the trained model file. This model is part of the **keras** library which is a part of tensorflow.

```
import tensorflow as tf

model = tf.keras.models.load_model('keras_model.h5')
```

Once the model is loaded, what should we do next? Yes will be testing the model by using the image data capture through the webcam. The boilerplate code has the program file (**detect_face_with_mask.py**) to open the webcam and capture the video frames using OpenCV. Remember how we capture frames from the webcam using OpenCV? Superb! After reading the frames from the webcam, we will use the **predict()** method to test if the model is able to predict the correct result of the frame captured: 1. Take a variable **prediction**. 2. Assign the value given by **predict()** method to the **prediction** variable. 3. Use the **print()** method to print the **prediction** variable value.	**ESR:** We should run the model to test it. **ESR**: We used: **VideoCapture()** method to get the video. **read()** method to read the video frames. **imshow()** method to show the frames.

```
while True:

    check,frame = video.read()

    #Prediction Result
    prediction = model.predict(frame)
    print("Prediction : ", prediction)

    cv2.imshow("Result",frame)

    key = cv2.waitKey(1)

    if key == 32:
        print("Closing")
        break

video.release()
```

Now let's test the output to see if the model is able to detect the face with a mask and face without a mask.

***Note**: Run the **detect_face_with_mask.py** file only after activating the virtual environment.*

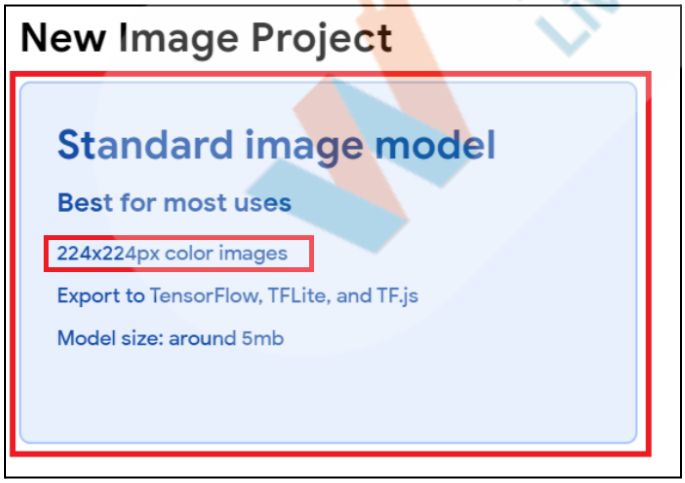
```
(mask_detection_env) D:\WhiteHatJr 2\Module 3\PRO C110\Teachable Machines>python detect_face_with_mask.py
```

```
ValueError: Input 0 is incompatible with layer sequential_4: expected shape=(None, 224, 224, 3), found shape=(32, 640, 3)
```

What happened when we ran the program?

Once we run the program, we get an error!

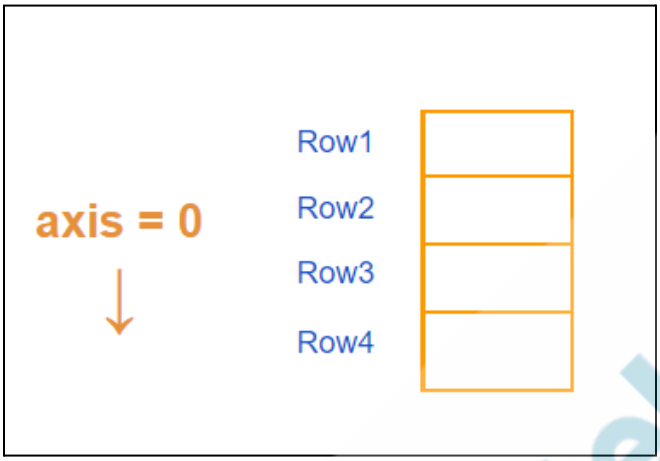
ESR: We get an error.

Can you try to describe what the error is about? The **error is related to the shape of the input data** which the image frame captured through the webcam. The **webcam captured the image in 32x640x3** shape as a **3 dimensional array**. Can you tell what these values stand for? Amazing! The expected shape of the input image frame to be used in the **predict()** method is **224x224x3** as a **4 dimensional array**. This means we need to send the image frames in the 224x224x3 shape to the model for prediction. Can you try to think we should send the image frame for testing in this particular shape? Remember, while choosing the image project, we chose the standard model with **224x224 image size.** 	**ESR:** Varied. **ESR**: The values are width, height and BRG color range. **ESR**: Varied.

Since we want to test the trained model file to detect the face with and without a mask when the video will be captured through a webcam, we need to send the video frame captured into the model for testing. But we cannot directly send the frame captured through the webcam. We need to send the image with the size in which model was trained. Can you tell me what should we do for this? Great! We also need to convert the 3 dimensional array to 4 dimensional array. To change the dimensions of the image, we need first convert the image into an array. Remember which module did we use to handle image arrays? Yes. Correct! We know images can be converted to the **Numpy arrays**, now let's understand what the NumPy axis is which will be used to expand the dimension of the array! ● The **Numpy axis represents the dimension of an array**. ● The Numpy array axes are numbered starting with 0. For example axis in:	**ESR:** We should use the **resize()** method from the OpenCV module**.** **ESR:** We use Numpy module

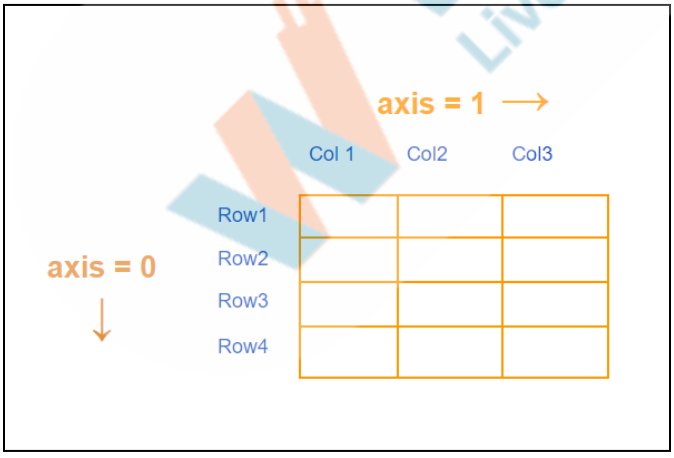
1D array:

- The 1D array will have only one axis.
- Axis 0 is along the rows.



2D array:

- The 2D array will have two axes.
- Axis 0 is along the rows of the 2D arrays
- Axis 1 is along the columns of the 2D array.



The axis value starts from 0, and it keeps on increasing as

the dimensions of the array increases.

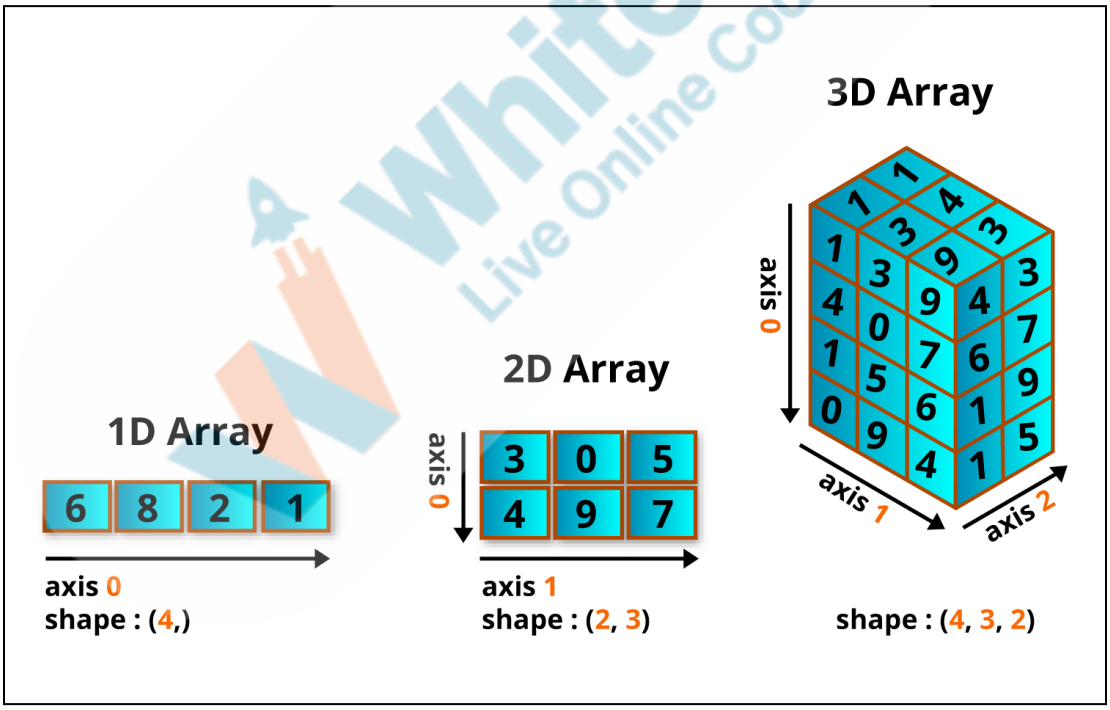
The **shape** property of a **numpy array** gives the number of elements along each axis.

For example the below image represents:

1D array shape which has:
4 elements along axis 0

2D array shape which has:
2 elements along axis 0,
3 elements along axis 1.

3D array shape which has:
4 elements along axis 1,
3 elements along axis 2,
2 elements along axis 3.



We will use the **expand_dims(image_array, axis)** method to convert the 3D array to 4D array.

We need to pass an image array and axis value along which we want to increase the dimension.

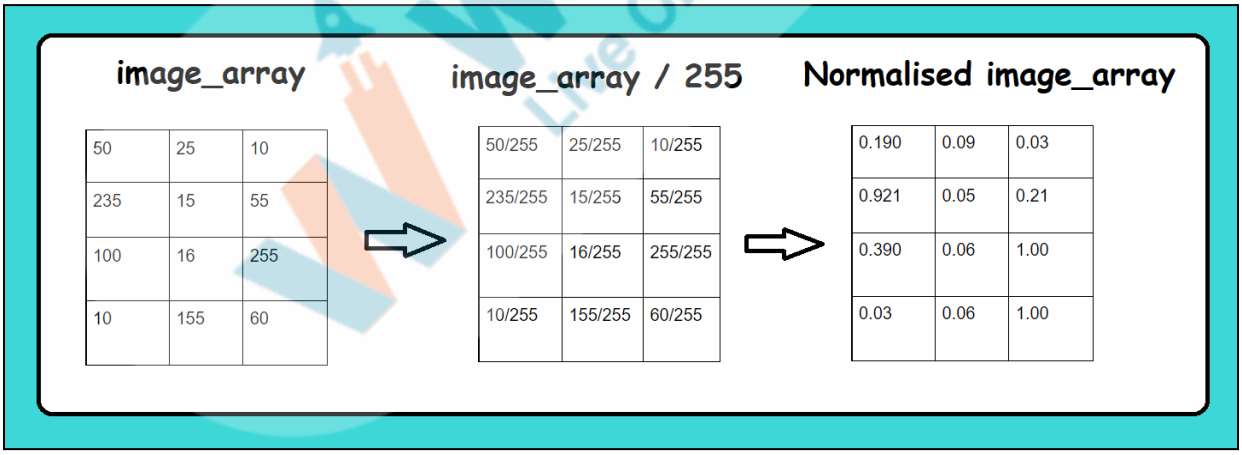
Before we can test the image after resizing and increasing the dimension, we also need to normalise the images array too.

Normalisation is used to convert all the image pixel values within 0 and 1 only. This helps the model to predict results with more accuracy.

Currently, when we convert an image to an array, the values of pixels vary from 0 to 255.

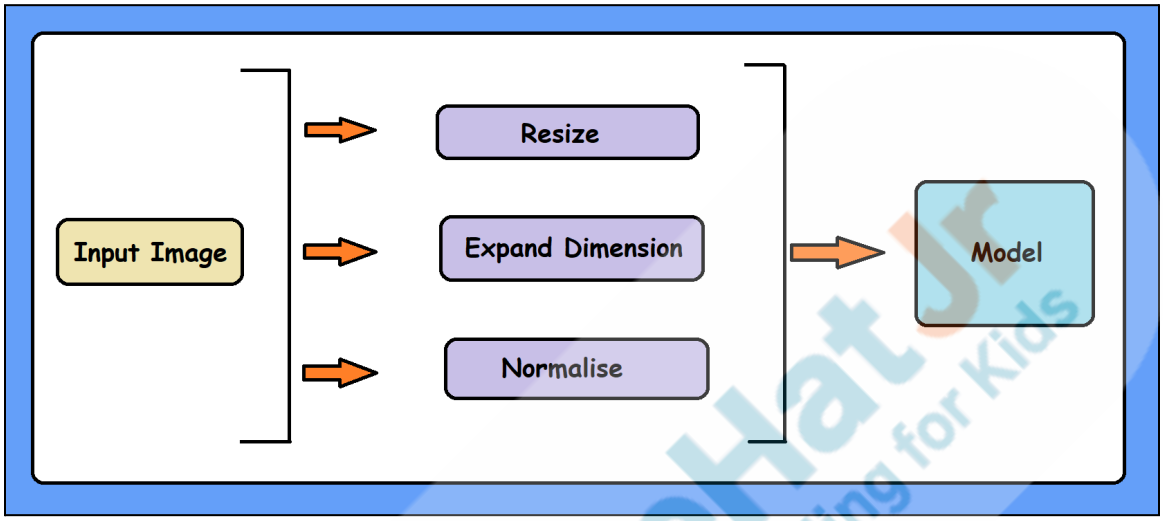
To convert all pixel values within 0 and 1, we need to divide all the pixel values in an image by the highest value of the pixel, which is 255.

The final image array will have pixels only in the range of 0 and 1.



To summarize, before we send an image to the model for testing, we need to:

1. Resize the image input.
2. Increase the dimension of the image array.
3. Normalise the image array.



Let's write the code to modify the image and pass it to the model for prediction:

1. Take an **img** variable and assign the result after **resize().**
2. Take the **test_image** variable and convert it into a numpy array using the **array()** method.
3. Convert the **test_image** from 3D array to 4D array using **expand_dims()** method.
4. Take a variable **nomalised_image** and assign the result after dividing the **test_image** by **255**.
5. Pass the **nomalise_image** variable in the model **predict()** method and assign the result to the **prediction** variable.
6. Use the **print()** method to print the prediction variable.

```
# Modify the input data by:

# 1. Resizing the image

img = cv2.resize(frame,(224,224))

# 2. Converting the image into Numpy array and increase dimension

test_image = np.array(img, dtype=np.float32)
test_image = np.expand_dims(test_image, axis=0)

# 3. Normalizing the image
normalised_image = test_image/255.0

# Predict Result
prediction = model.predict(normalised_image)

print("Prediction : ", prediction)

cv2.imshow("Result",frame)
```

Now let's test the output:

Run **detect_face_with_mask.py.**

Note: Make sure to activate the virtual environment before running the .py file.

```
(mask_detection_env) D:\WhiteHatJr 2\Module 3\PRO C110\Teachable Machines>python detect_face_with_mask.py
```

Output:

```
Prediction :  [[0.7263644  0.27363566]]
```

Analysing Output:

The output consists of an 2D array with 2 values.

The number of values in the prediction array depends on the number of the classes taken while training the model. Since we trained the model using 2 classes(0 With Mask and 1 Without mask), the prediction array has two values.

The first values is for first class(0 With Mask)
The second value is for the second class(1 Without Mask)

The values in the prediction array show the percentage of the belongingness of the input test images in each class.

The first value shows the probability(chance) of belongingness to 1st class and the second value shows the probability of belongingness to 2nd class.

That means, in the current output the test image frame has 72% (72/100=0.72) chance that it is "With Mask"(class 0) and 27%(27/100=0.27) chance that it is "Without Mask"(class 0).

Class	Label	Probability
With Mask	0	0.72
Without Mask	1	0.27

That was interesting!

We learned to teach machines how to classify images. The model could classify between the face with the mask and the face without the mask.

Are you excited?

ESR: Yes.

Teacher Stops Screen Share	
Teacher Starts Slideshow  **Slide 15 to 21** Refer to speaker notes and follow the instructions on each slide.	
We have one more class challenge for you. Can you solve it? Let's try. I will guide you through it.	
Teacher Ends Slideshow 	
STUDENT-LED ACTIVITY - 20 mins	
• **Ask the student to press the ESC key to come back to the panel.** • **Guide the student to start Screen Share.** • **The teacher gets into Fullscreen.**	
ACTIVITY • **Train the model to classify images using Teachable Machines.** • **Write a program to train the model to classify between faces with masks and without masks.**	
Teacher Action	**Student Action**
Guide the student to download the boilerplate code Student Activity 1. ***Note 1**: The student will perform the same activities as done by the teacher.* ***Note 2**:Please refer to Teacher Activities to follow through and guide the student to do the activities.*	
Guide the student to train a model using Teachable	

Machine.

Click on "**Webcam**" to capture images. This will ask for permission to use your camera.

1. Click on "**Allow**" to give the camera's permission
2. Click on "**Hold to Record**" and keep to the button in the clicked state to capture around 30-50 images with a mask.
3. Click on "**Hold to Record**" and keep to the button in the clicked state to capture around 30-50 images without the mask.
4. Click on "**Train Model**"
5. Click "**Export Model**" to down the trained model.
    i. Click on "**Tensorflow**".
    ii. Click on "**Keras**" .
    iii. Click on "**Download my model**".
6. A **zipped file** will be downloaded.
7. **Extract** all the files.

Guide the student to create & activate a virtual environment in the working directory.

Create Virtual Environment(**Windows/Mac**):

```
python3.9 -m venv <name_of_the_environment>
```

Activate Virtual Environment:

Windows:

```
<name_of_the_environment>\Scripts\activate
```

Mac:

```
source <name_of_the_environment>/bin/activate
```

Guide the student to write a program to test the trained model: ● Install tensorflow and opencv-python ● Load the Model ● Modify the input data ● Predict the model results. *Encourage the student to test this with his/her family/friends faces too and check if the model can tell whether it belongs to a class with a mask or a class without mask!*	
We could teach machines to identify between two classes of images dataset. You did amazing work today!	
Teacher Guides Student to Stop Screen Share	
WRAP-UP SESSION - 05 mins	
Teacher Starts Slideshow **Slide 22 to 27** 	
Activity Details: **Following are the WRAP-UP session deliverables:** ● Appreciate the student. ● Revise the current class activities. ● Discuss the quizzes.	
WRAP-UP QUIZ Click on In-Class Quiz	

Continue WRAP-UP Session
Slide 28 to 33

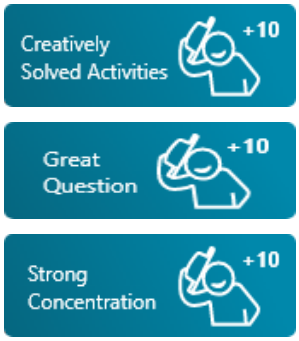
Activity Details:

Following are the session deliverables:

- Explain the facts and trivia
- Next class challenge
- Project for the day
- Additional Activity (Optional)

FEEDBACK

- **Appreciate the student for his/her efforts in the class.**
- **Ask the student to make notes for the reflection journal along with the code they wrote in today's class.**

Teacher Action	Student Action
You get Hats off for your excellent work!	*Make sure you have given at least 2 Hats Off during the class for:* 

PROJECT OVERVIEW DISCUSSION
Refer the document below in Activity Links Sections

Teacher Clicks	✖ End Class

ADDITIONAL ACTIVITIES	
Additional Activities *Encourage the student to write reflection notes in their reflection journal using Markdown.* Use these as guiding questions: ● What happened today? ○ Describe what happened. ○ The code I wrote. ● How did I feel after the class? ● What have I learned about programming and developing games? ● What aspects of the class helped me? What did I find difficult?	*The student uses the markdown editor to write her/his reflections in the reflective journal.*

ACTIVITY LINKS		
Activity Name	**Description**	**Link**
Teacher Activity 1	Specific Instruction Vs ML	https://s3-whjr-curriculum-uploads.whjr.online/ed194ebb-d0cb-47f5-821f-cf8c49874bf5.png
Teacher Activity 2	Humans Learning Process	https://s3-whjr-curriculum-uploads.whjr.online/3b880e2b-d752-49f4-981d-641da21e5931.png
Teacher Activity 3	Image Identification Activity	https://s3-whjr-curriculum-uploads.whjr.online/750c7c3c-a59d-4ade-9b5f-b80ad5dd5f53.pptx
Teacher Activity 4	Google Teachable	Teachable Machine

	Machine	
Teacher Activity 5	Teacher Boilerplate Code	https://github.com/procodingclass/PRO-110-Teacher-Boilerplate
Teacher Activity 6	Reference Code	https://github.com/procodingclass/PRO-C110-Reference-Code
Student Activity 1	Student Boilerplate Code	https://github.com/procodingclass/PRO-C110-Student-Boilerplate
Teacher Reference 1	Tensorflow Guidebook	https://docs.google.com/document/d/1zuuyL6dK6HapCIMWX1uEAjm4FTDuQO1I0xQ7JDqfxy8/edit?usp=sharing
Teacher Reference 2	Project Document	https://s3-whjr-curriculum-uploads.whjr.online/03585471-d7de-4f23-8d51-b8d0a7f63f00.pdf
Teacher Reference 3	Project Solution	https://github.com/procodingclass/PRO-C110-Project-Solution.git
Teacher Reference 4	Visual Aid Link	https://s3-whjr-curriculum-uploads.whjr.online/de2489e8-1560-4d08-aa1e-ae2af5cc0032.html
Teacher Reference 5	In Class Quiz	https://s3-whjr-curriculum-uploads.whjr.online/71dd2053-a73a-4692-aef6-949e09a136c0.pdf