DOCUMENTATION OF AI USING AZURE CUSTOM VISION

To use the azure custom vision service, we will need to create a custom vision training and prediction resources in Azure. To do so in Azure portal , fill out the dialog window on the Create Custom Vision page to create both a Training and Prediction resource.

CREATE A NEW PROJECT

In the web browser , navigate to the https://www.customvision.ai/ and select Sign in. Sign in with the same account you used to sign in to the Azure portal.

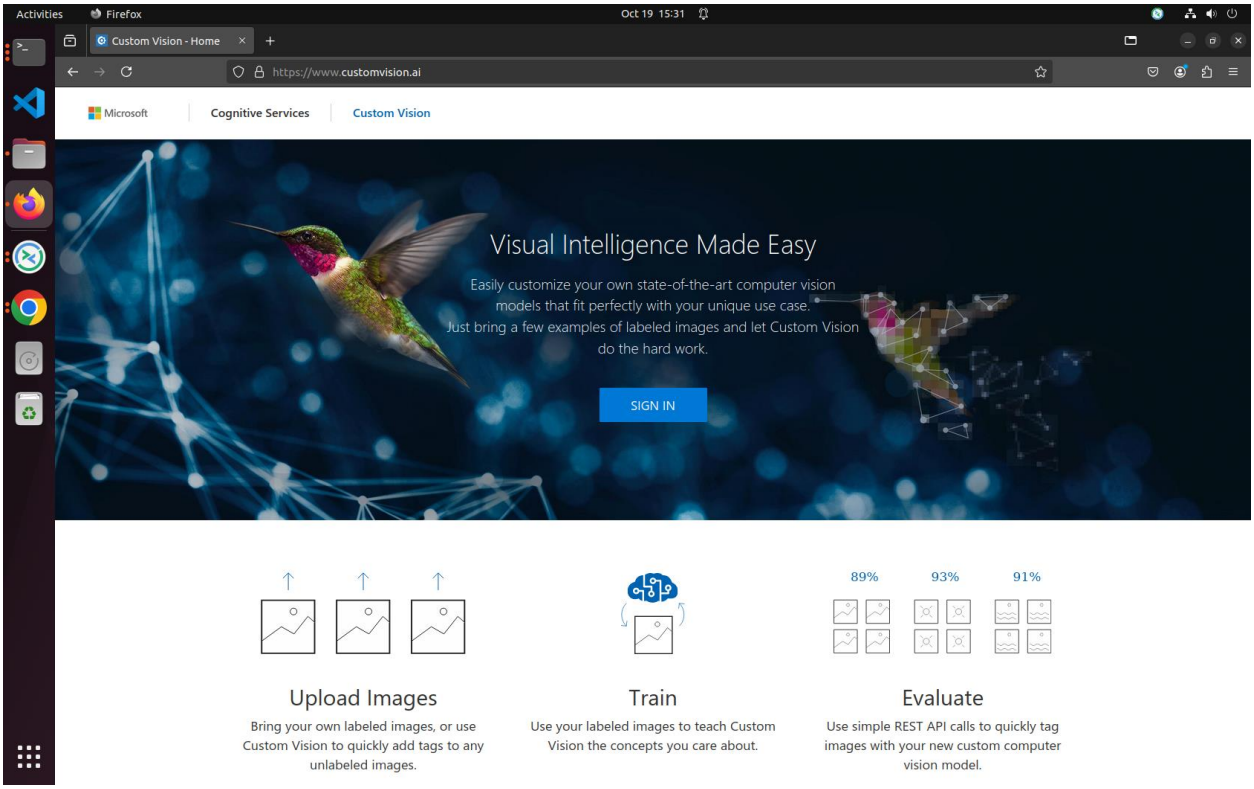

Fig 1: Sign in page of the custom vision portal

1. To create your first project, select **New Project**. The **Create new project** dialog box appears.

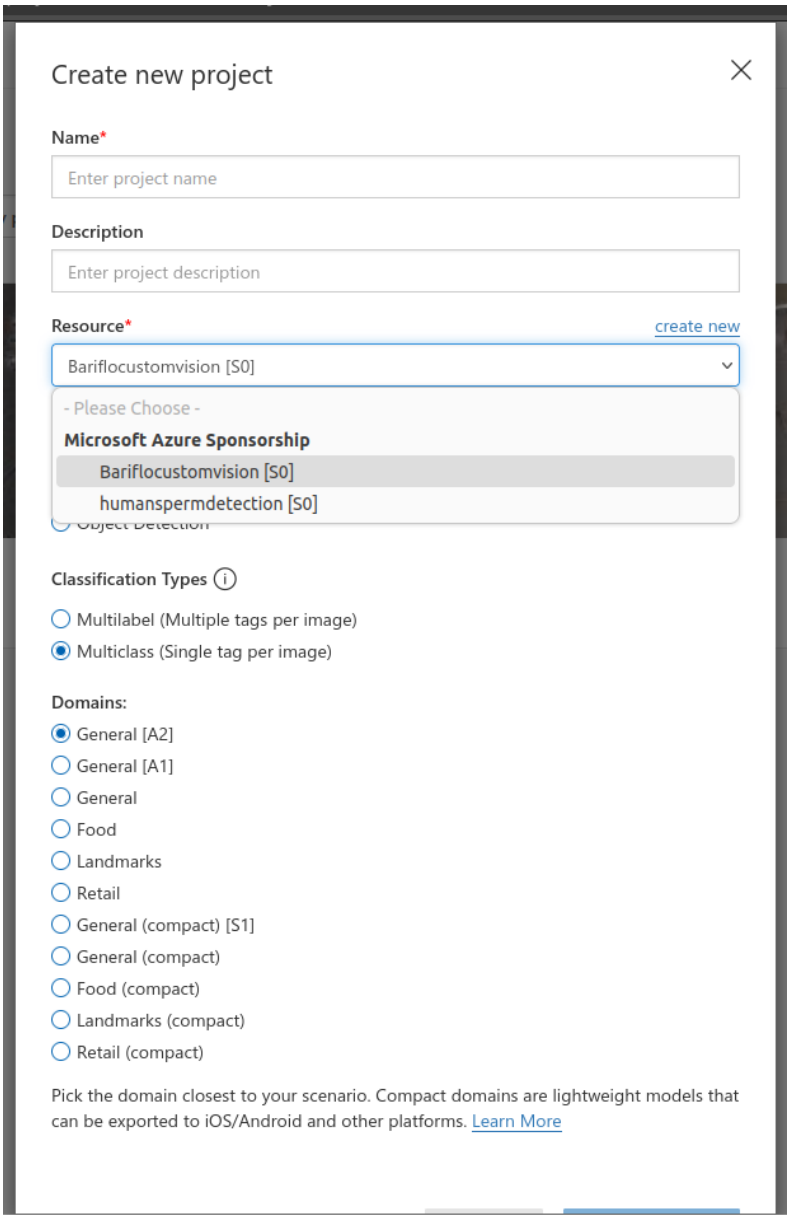

Fig 2: Create new project dialog box

2. Enter a **Name** and a description for the project. Then select your Custom Vision Training **Resource**. If your signed-in account is associated with an Azure account, the Resource dropdown displays all of your compatible Azure resources.

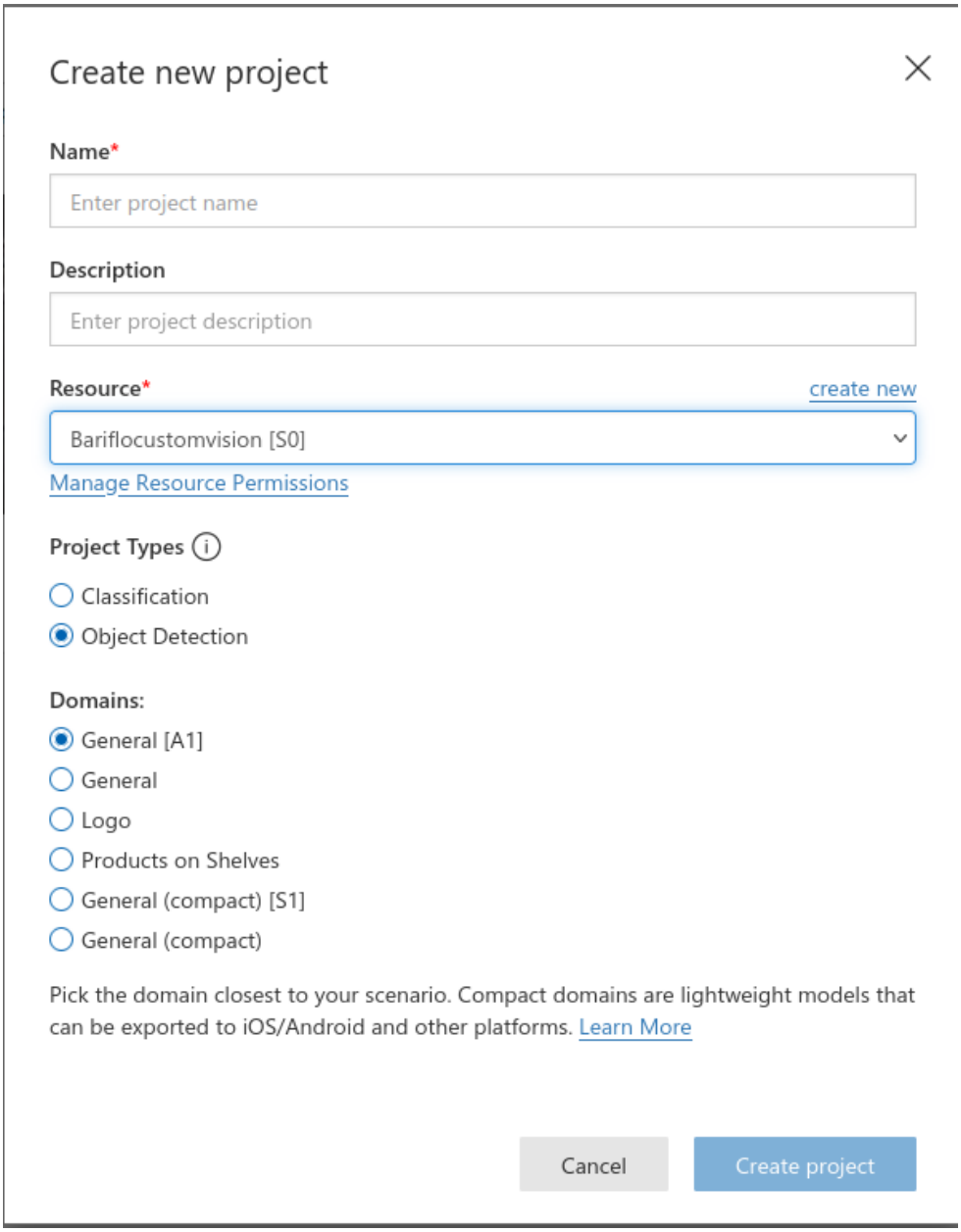

Fig 3: Selection of object detection

3. Select the **Object Detection** under **Project Types**.

4. In case of **Domains** select **General[A1]**.

5. Finally, select on **Create Project**.

UPLOAD THE DATASET IMAGES AND TAGGING

1. Click on the project , and then click on **Add images** and then select **Browse local files**. Select **Open** to upload the images.

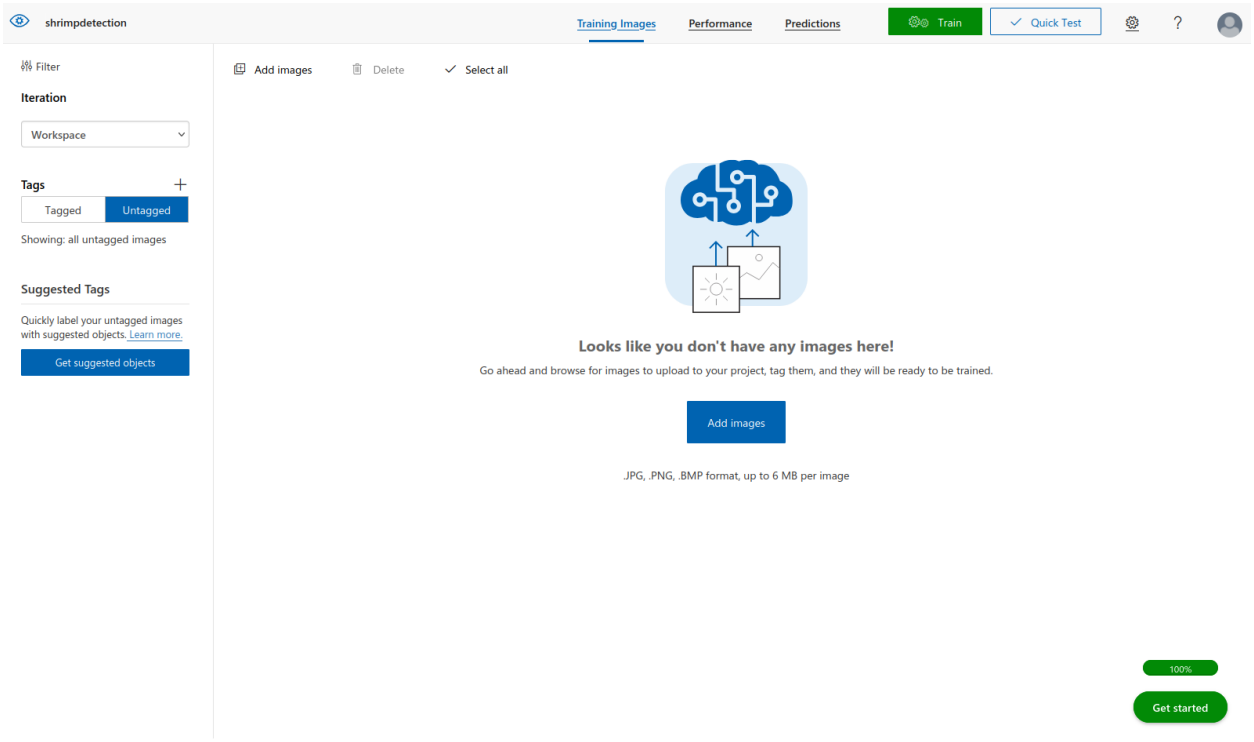

Fig 4: Interface to Add the image in the dataset

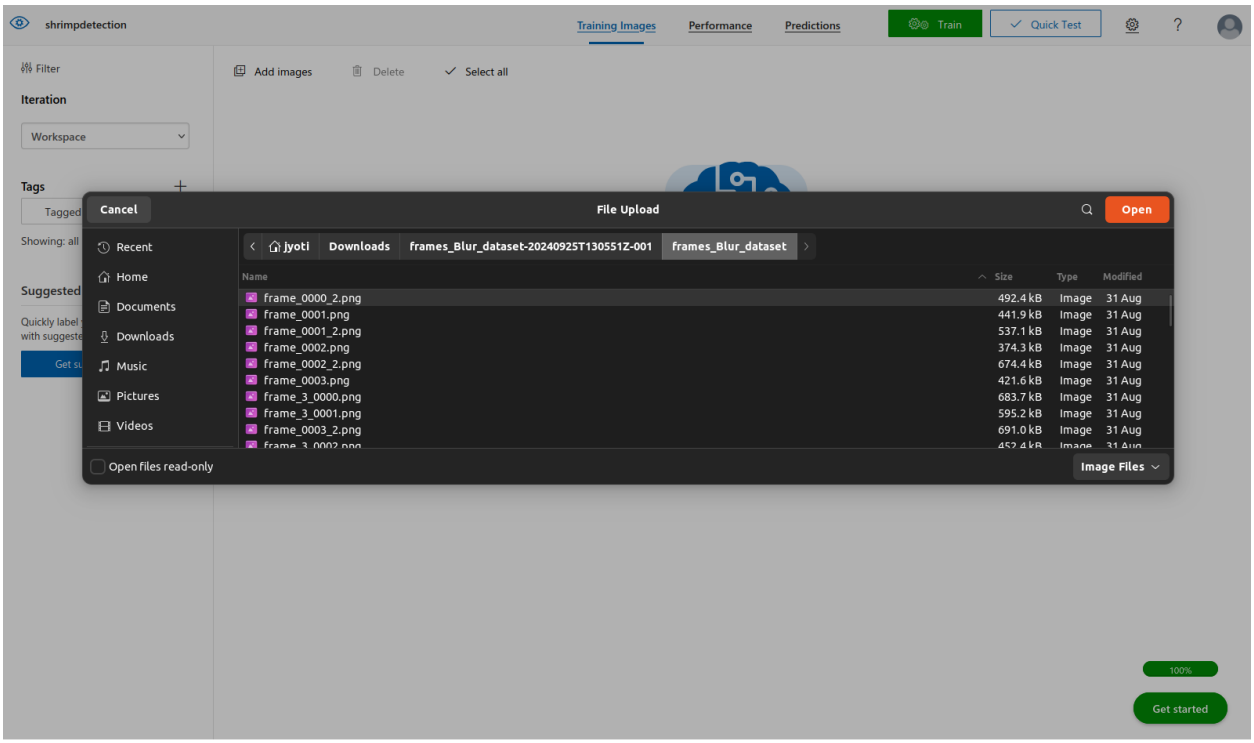

Fig 5: On clicking on Add images

2. After clicking on **Open** then a interface will be opened , as shown in below Fig6.

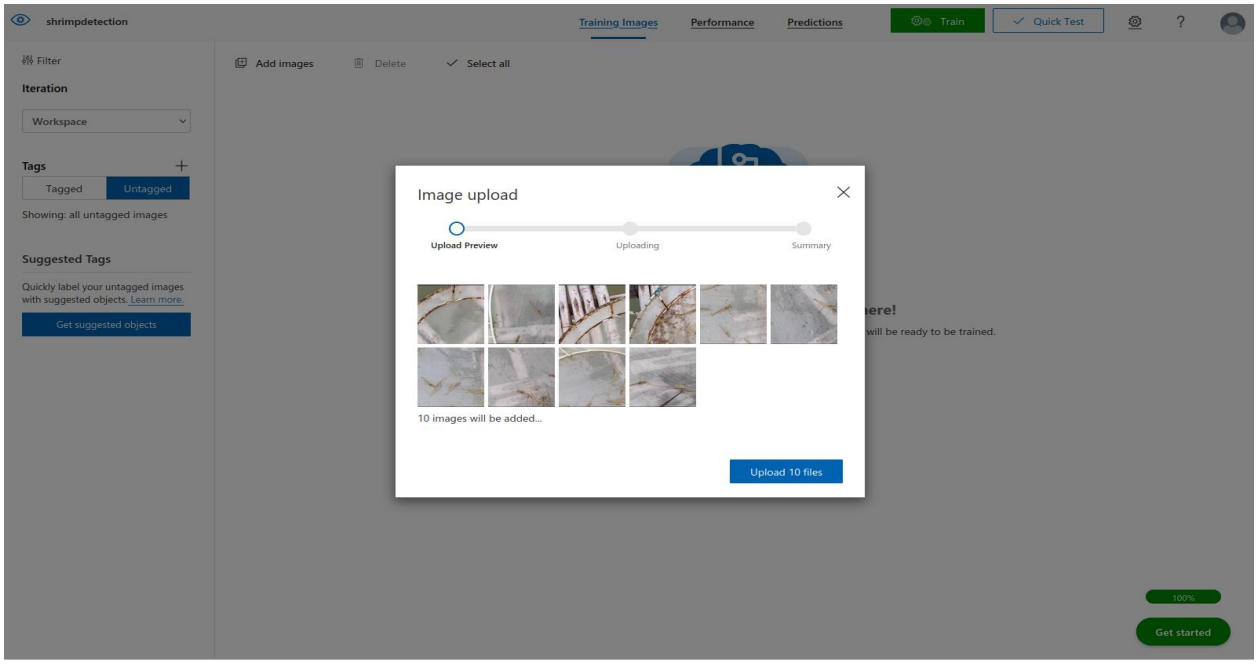

Fig 6: Interface on clicking on Open

3. Click on **Upload** button.

4. After clicking on **Upload** button then click on **Done.** Then the interface is appeared.

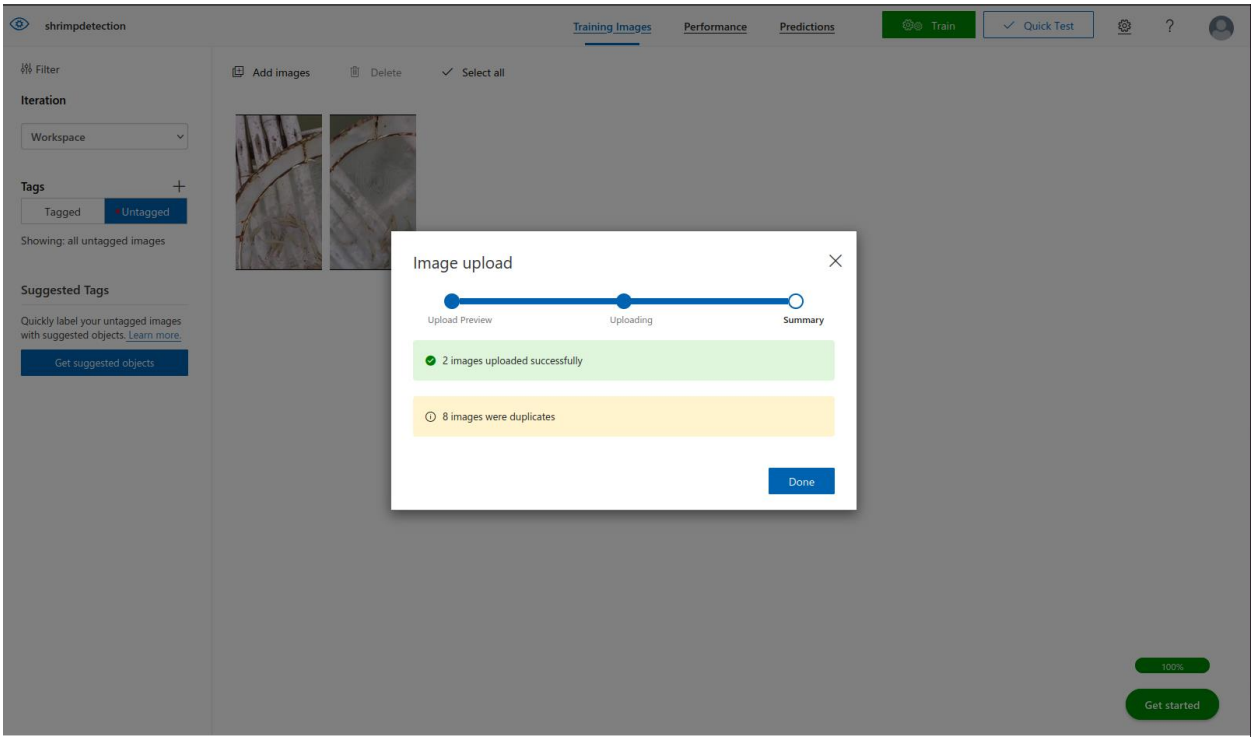

Fig 7: Interface on clicking on upload button

5. After clicking on **Done** then it is in the section of **untagged** image.

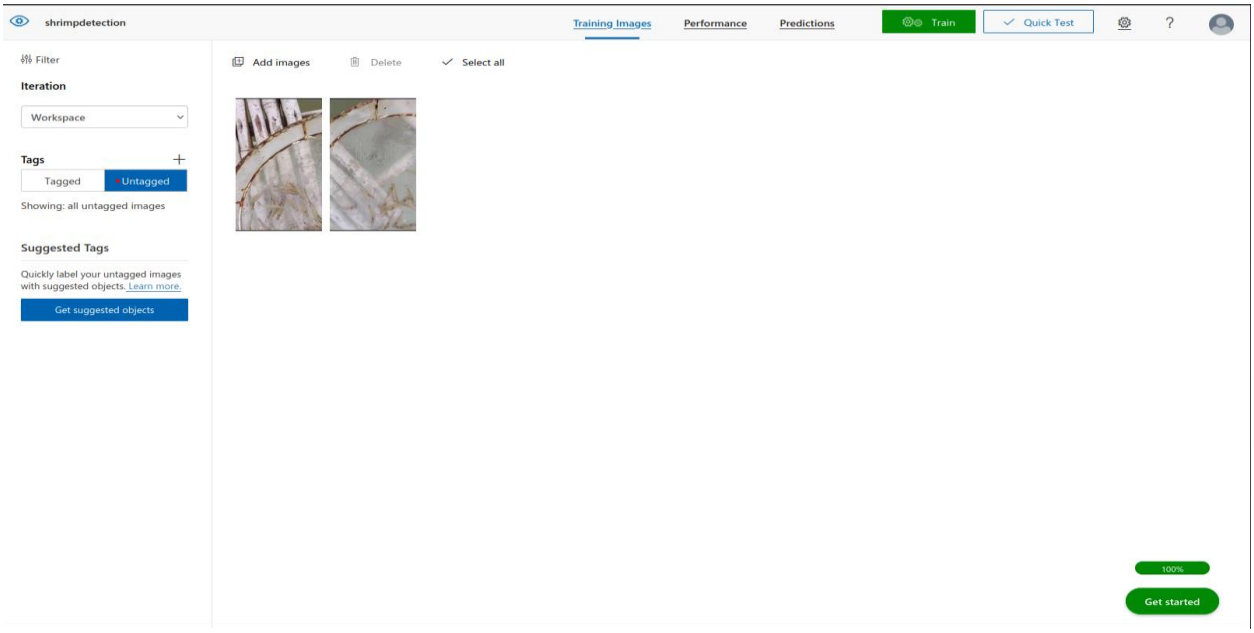

Fig 8: Interface of the untagged image

6. Then click on each image and tag the object with the name. As shown in Fig 9.

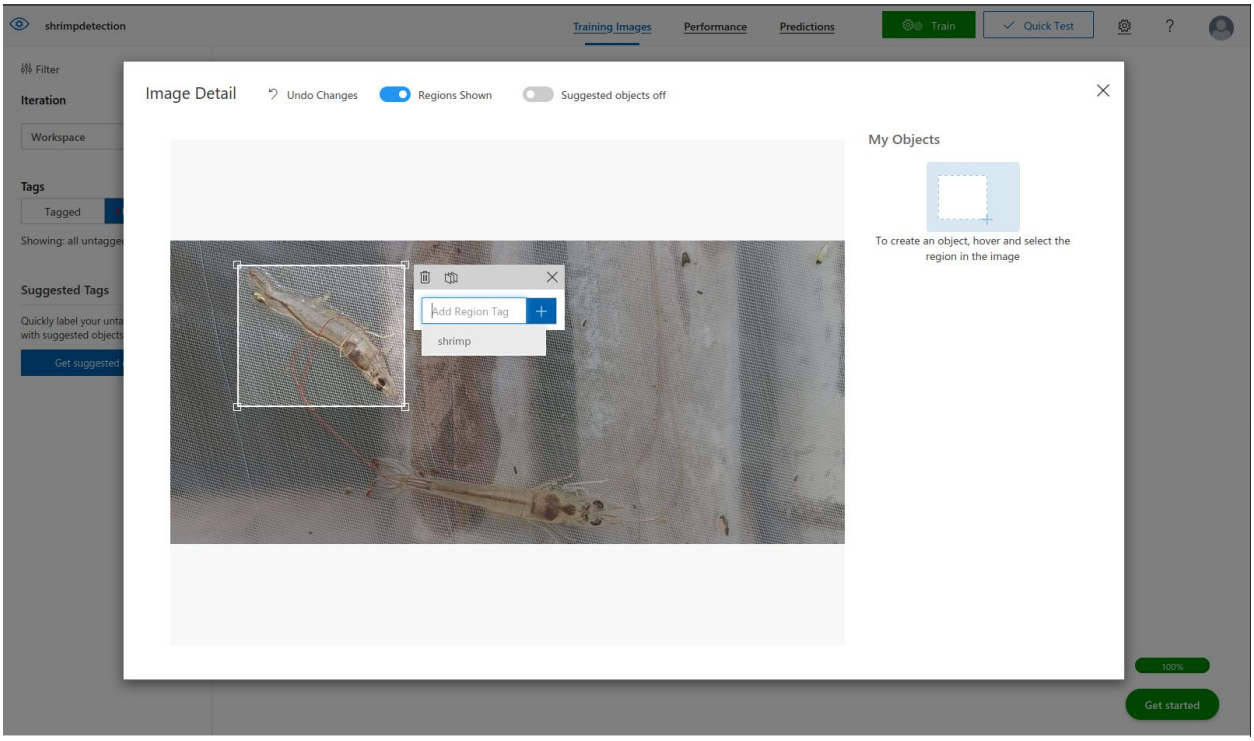

Fig 9: Images tagged with the suitable name

7. After tagging all the images then all the images will be shift to **Tagged** and make sure that there is no images at **Untagged**. Images are shift to tagged as shown in figure Fig 10.

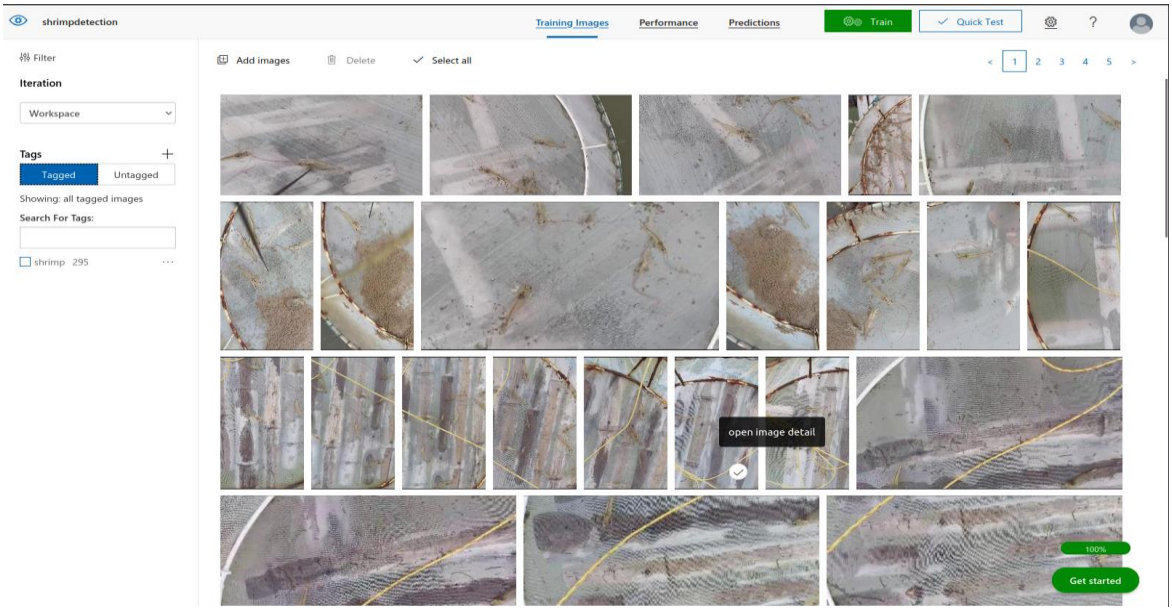

Fig 10: Images stored in tagged

TRAINING OF THE MODEL

After ensuring that all the images are tagged then click on **Train** button. Then select the training types i.e. **Quick Training** and **Advanced Training .** As shown in figure Fig 11:

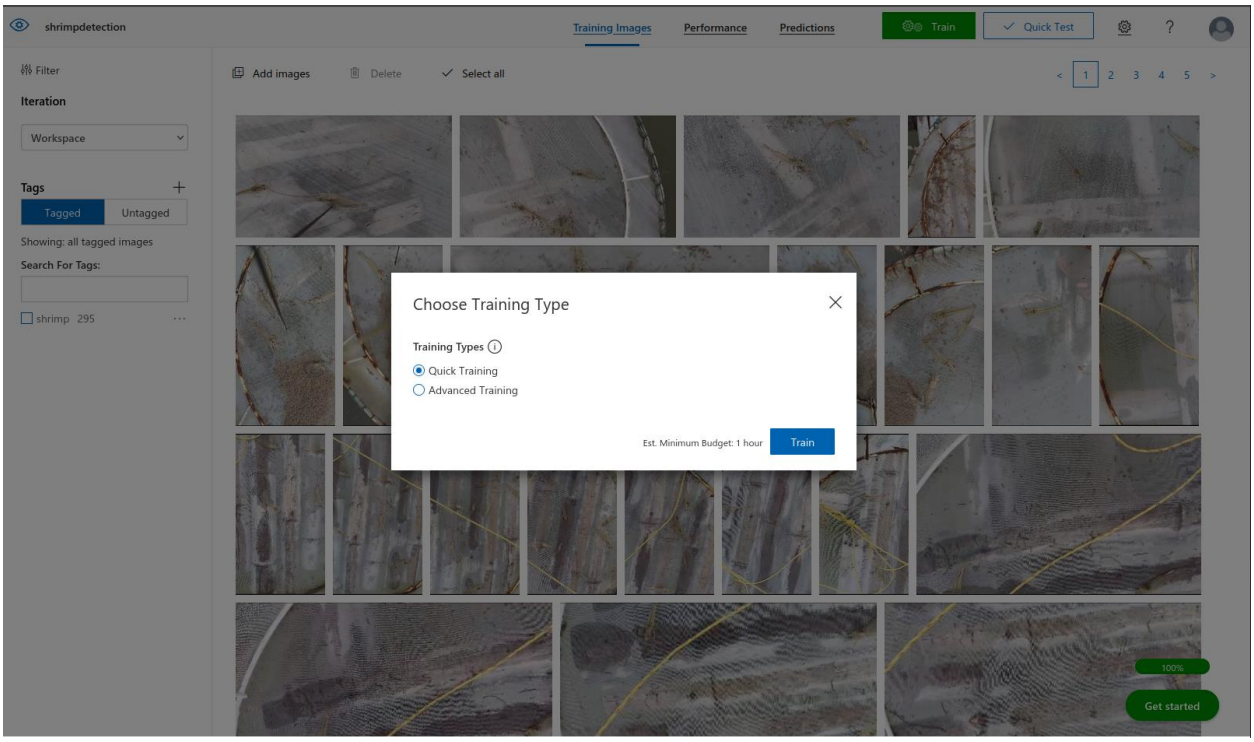

Fig 11: Interface on clicking on Train button

On clicking on **Quick Training ,** then click on **Train** button **,** It will start training the model. But In case of **Advanced Training ,** we have to set the training hour . And then click on **Train.** As shown in figure Fig 12.

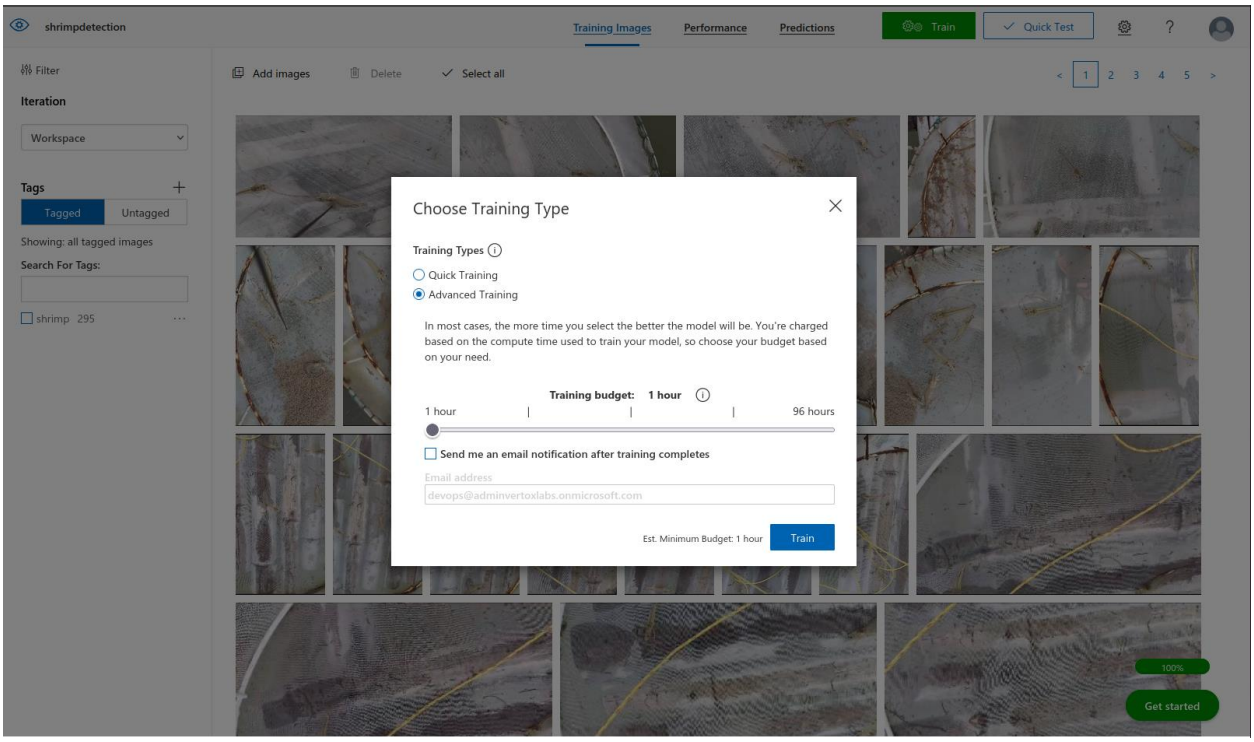

Fig 12: Interface of Advance Training

COMPLETION OF TRAINING

After the completion of training then click on **Performance** to have the report how the model is trained. As shown in Fig 13.

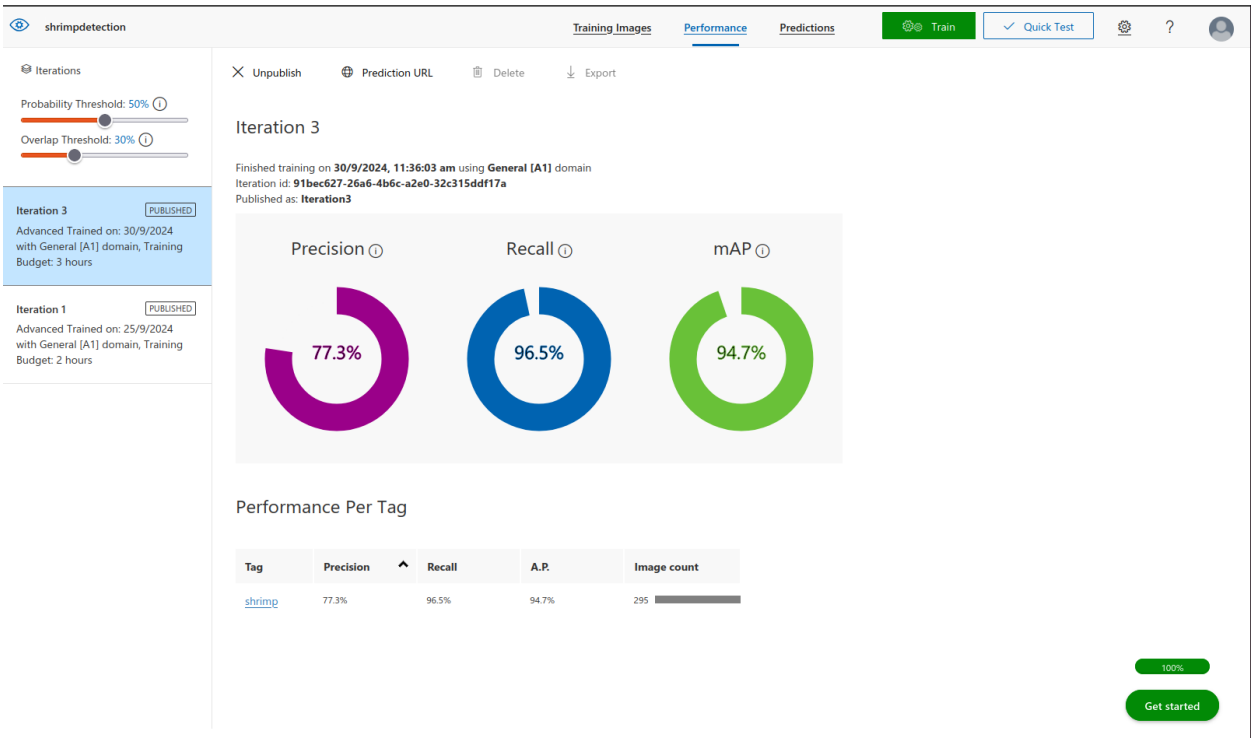

Fig 13: Report of the model trained

PREDICTION

For the prediction we have to required the prediction URL and the Key. For the Prediction URL , click on the **Publish** button at the top right corner of the Report , after that click on the **Prediction URL ,** we will be getting the **Prediction URL** and the **Key.** As shown in Fig 14.

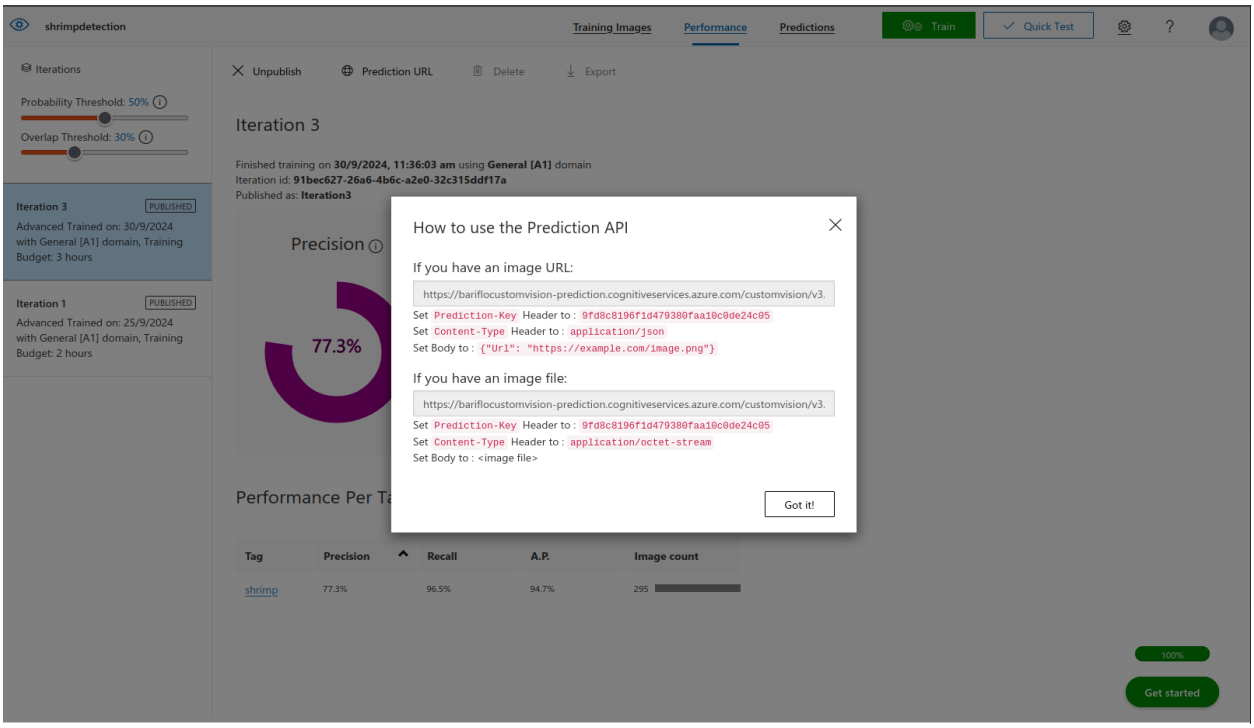

Fig 14: Prediction URL and Key

Take the Prediction URL and Key and use it in the python code for the prediction . If we have the image file then choose the API for **image file** , If we are having the **image url** then using the API for image URL.

PYTHON CODE ALGORITHM FOR PREDICTION

Use the Prediction URL and the key in the python code. This script processes a video to detect shrimp using Azure Custom Vision. It extracts frames, enhances them, detects shrimp, and calculates their sizes. The results, including detected shrimp and their sizes, are saved as images and summarized in tables based on their size relative to the average.And the Algorithm of the python code is written as below:

Input Video:

The script starts by specifying the path to a video file that contains shrimp to be detected.

Output Directory:

An output folder is defined to store the processed frames and annotated images.

Video Processing:

The script opens the video file and retrieves its frame rate to determine how often to extract frames.

It sets an interval (e.g., every 2 seconds) to extract frames from the video.

Frame Extraction:

A loop iterates through the video frames based on the defined frame interval.

For each frame:

The script reads the frame from the video.

Converts the frame from a BGR format (OpenCV) to RGB format (PIL) for image processing.

Image Enhancement:

The extracted frame is enhanced to improve visibility for shrimp detection:

A sharpen filter is applied to reduce blur, or other enhancements like brightness and contrast adjustments can be applied if needed.

The enhanced frame is saved as an image in the output folder.

Shrimp Detection:

The saved enhanced image is sent to the Azure Custom Vision API for shrimp detection:

The image is read as binary data and sent in an HTTP POST request.

The response contains predictions of detected objects (shrimp) in the image.

Processing Predictions:

If the response is successful:

The script iterates through the predictions, filtering for those with a high probability (e.g., greater than 0.96).

For each detected shrimp:

The bounding box dimensions are calculated.

The diagonal size in pixels is converted to centimeters.

An annotation is drawn on the image with the bounding box, unique ID, probability, and size information.

Output Image Saving:

The annotated image, showing detected shrimp with bounding boxes and labels, is saved to the specified output path.

Data Collection:

The unique IDs and sizes of the detected shrimp are collected and stored in a list.

Completion:

The video capture object is released, and the script concludes its execution, summarizing the results of shrimp detection.