Northumbria University NEWCASTLE

Department of Computer and Information Sciences

KV4004 AI Fundamentals

Workshop 5

October 2024

1

Explore Computer Vision with Azure AI Services

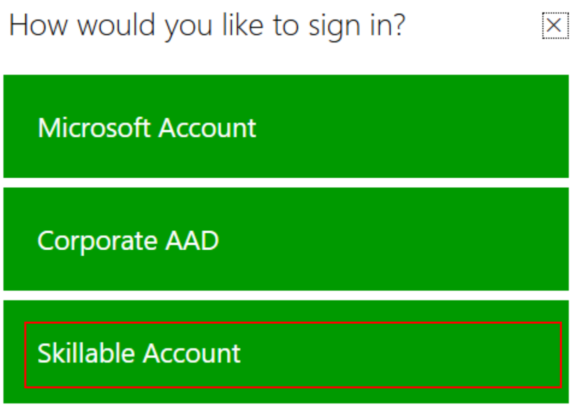
In the second part of this workshop, you'll learn how to train and test an object detection model on Microsoft Azure. Object detection is a form of computer vision in which a machine learning model is trained to classify individual instances of objects in an image and indicate a bounding box that marks its location. You can think of this as a progression from image classification to building solutions where we can ask the model what objects are in this image and where are they. For example, a road safety initiative might identify pedestrians and cyclists as being the most vulnerable road users at traffic intersections. By using cameras to monitor intersections, images of road users could be analyzed to detect pedestrians and cyclists to monitor their numbers or even change the behaviour of traffic signals.

> Before proceeding, make sure you have created an Azure Machine Learning workspace and a computing instance.

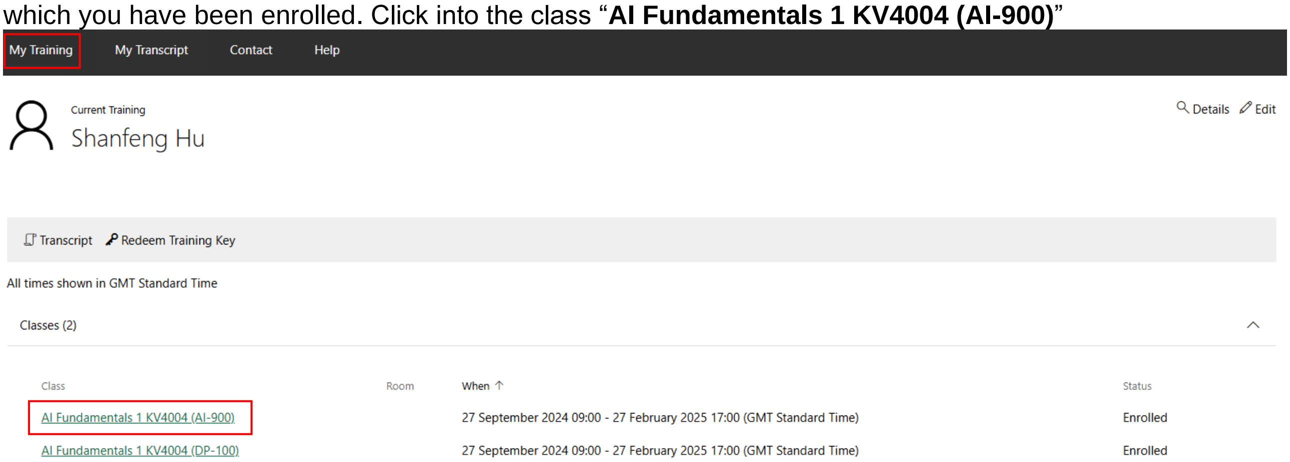
Exercise 1: Create an Azure AI services resource

1. Go to link https://msle.learnondemand.net/ and click on "**Sign In**"

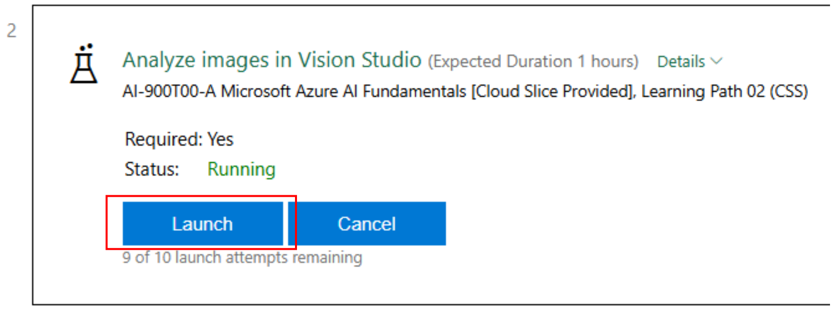
2. Click on "**Skillable Account**" and then provide your username and password on the login page



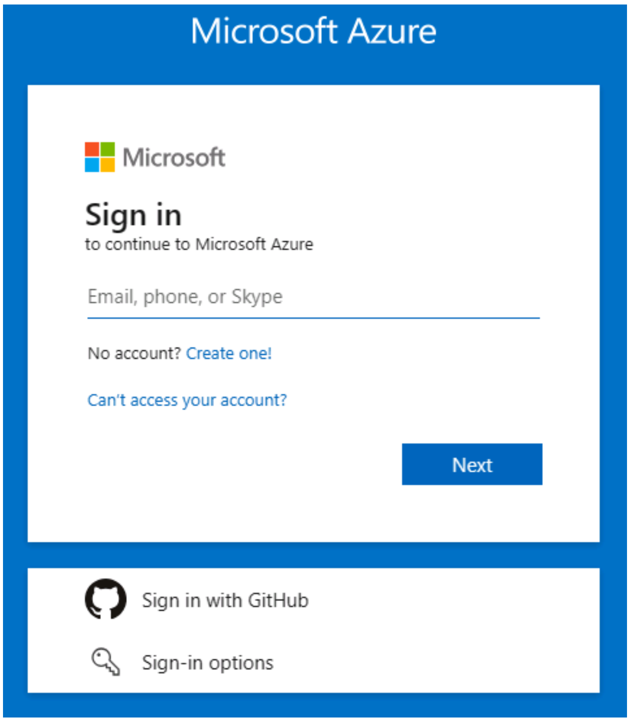
3. Once you are logged in, go to "**My Training**" and click on it, you will then be shown a list of classes on which you have been enrolled. Click into the class "**AI Fundamentals 1 KV4004 (AI-900)**"



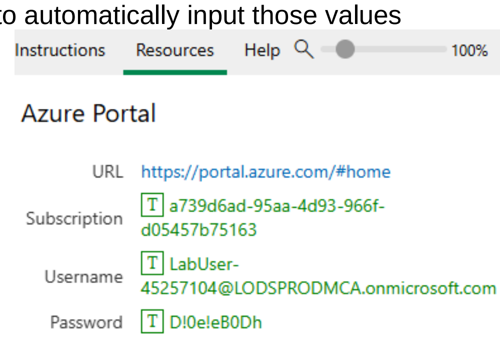
4. In the new page, there are multiple virtual machine options which come with different services provided. Launch the second one "**Analyze images in Vision Studio**"



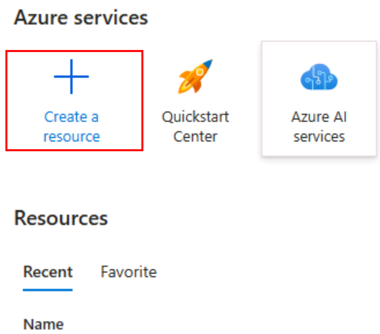
5. Once the virtual machine has been launched, open the Edge browser and navigate to the Azure login in page



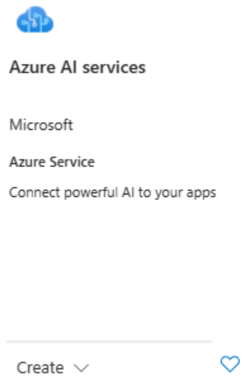
6. Use the username and password provided under the "**Resources**" tab to login. You can simply click on the Username and Password to automatically input those values



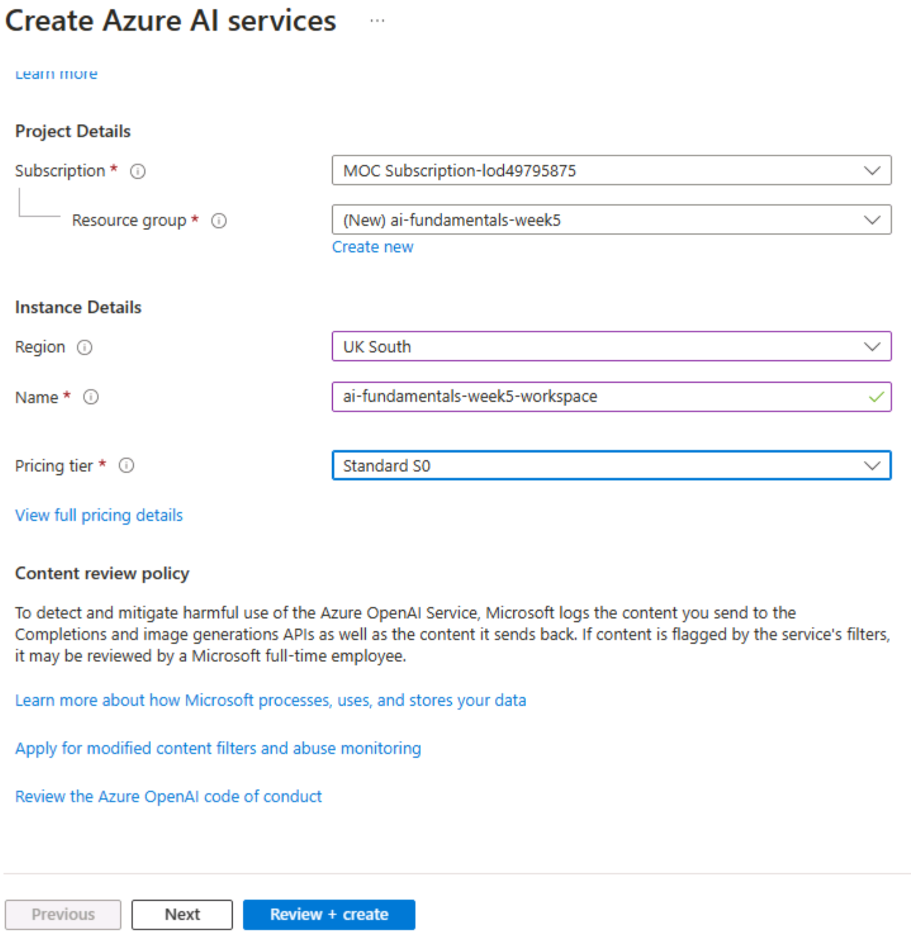
7. After you have logged into Microsoft Azure, go to the top left corner of the landing page and click on "**Create a resource**"



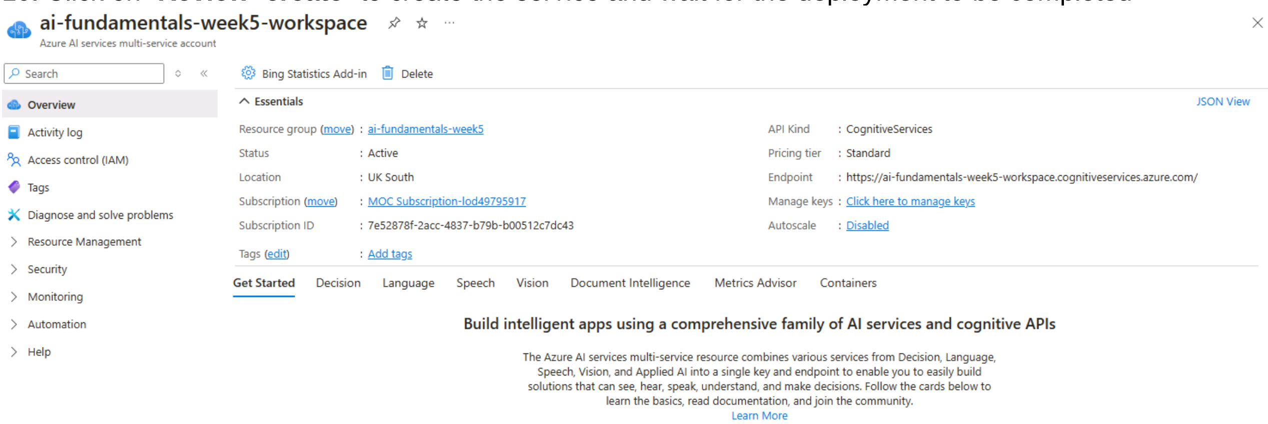
8. In the new page, user the keyword "Azure AI services" to search for this service and then click on it



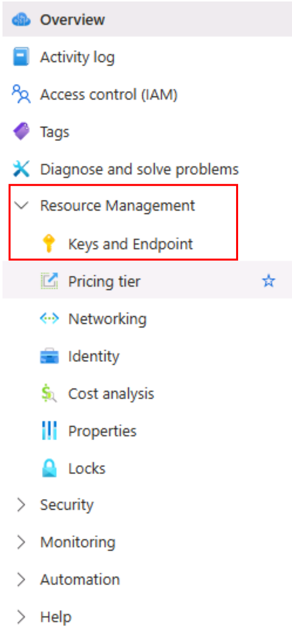
9. When you create the service, use the following details (they are for reference and can be changed as long as the names are valid)



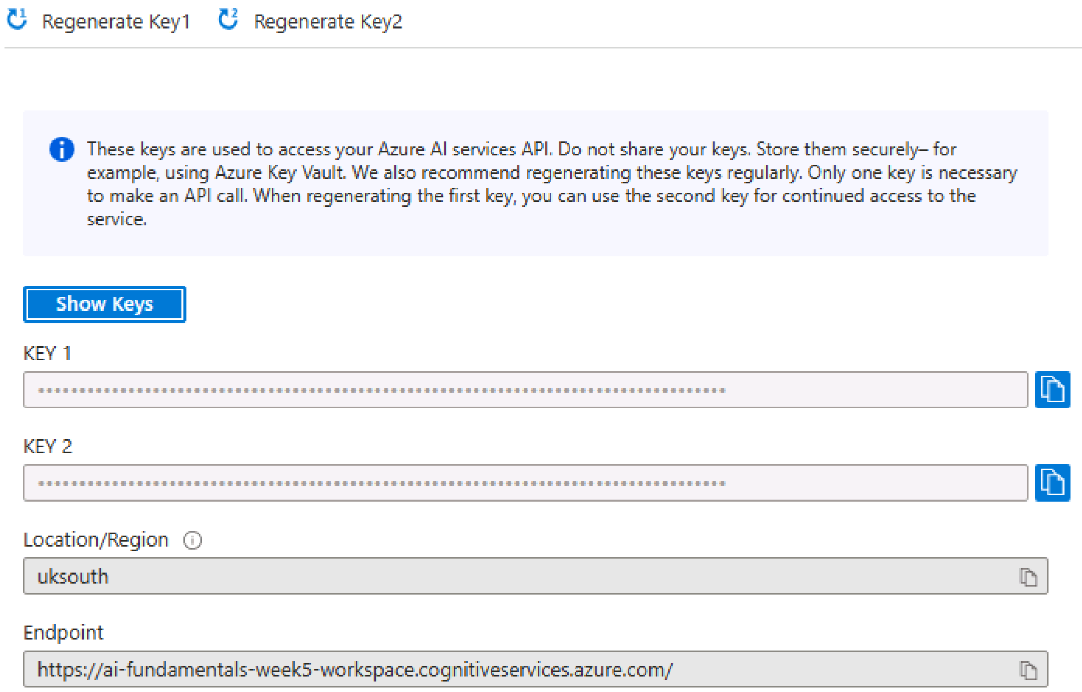
10. Click on “**Review+create**” to create the service and wait for the deployment to be completed



11. On the left panel of the page, click on “**Resource Management**” to expand it and then click on “Keys and Endpoint” the sub menu

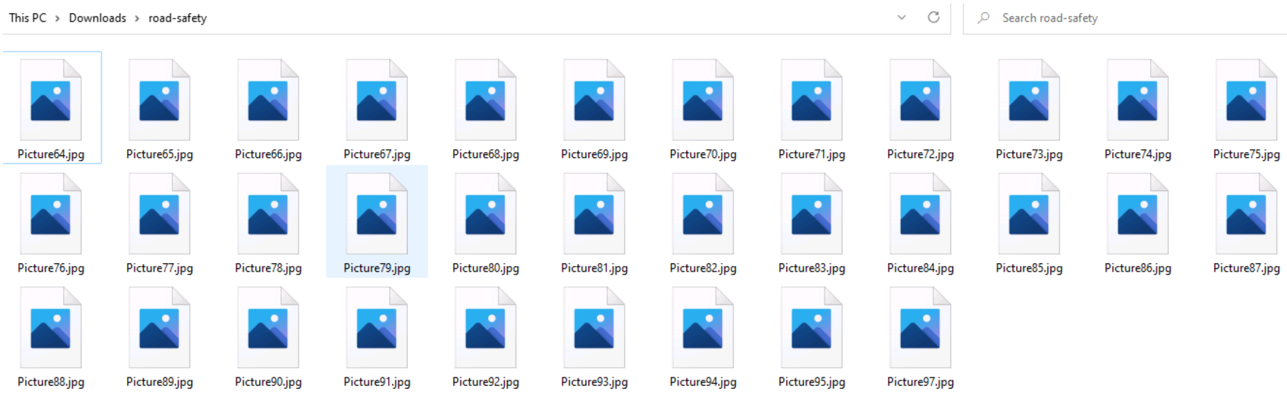


12. On the new page, you will be able to view the keys and endpoint string, which will be needed to run the following task

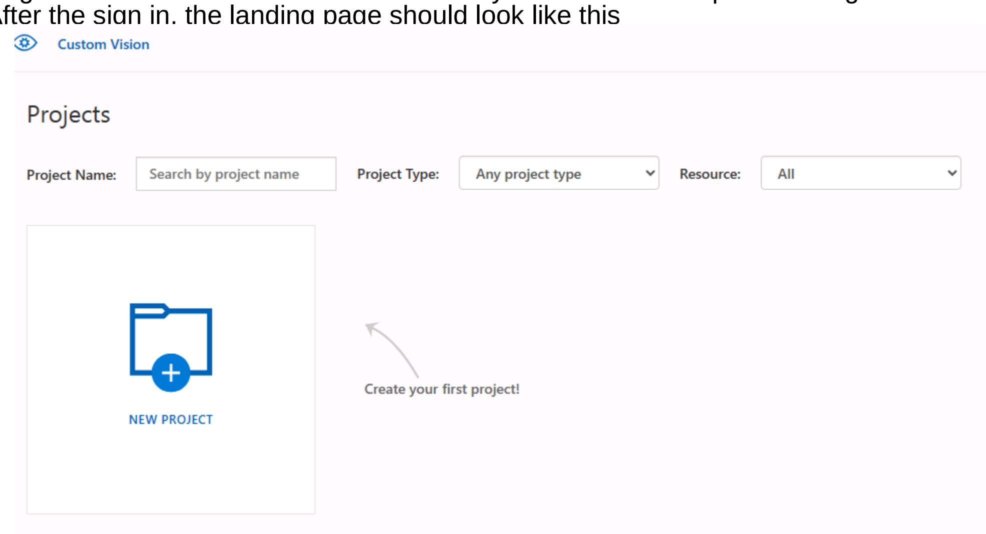


Exercise 2: Train Object Detection

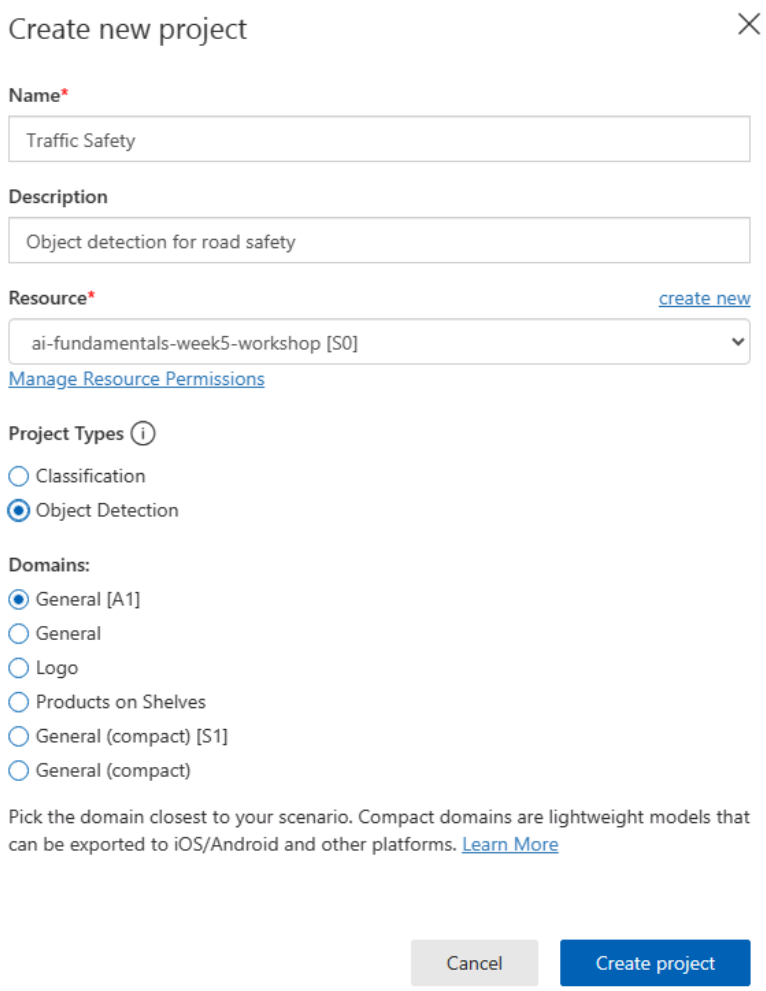
1. To train an object detection model, you need to upload images that contain the classes you want the model to identify and tag them to indicate bounding boxes for each object instance. Download and extract the training images from https://aka.ms/traffic-images. The extracted folder contains a collection of images of cyclists and pedestrians.



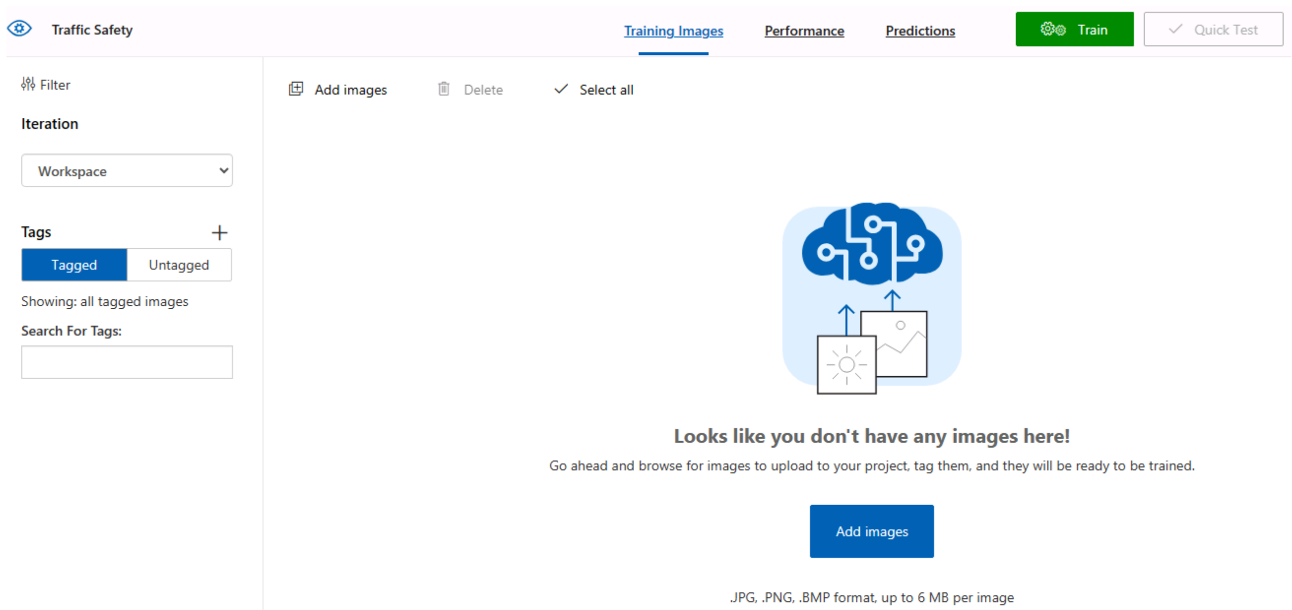
2. Open a new browser tab and browse to the Custom Vision portal at https://customvision.ai. If prompted, sign in using the Microsoft account associated with your Azure subscription and agree to the terms of service. After the sign in, the landing page should look like this



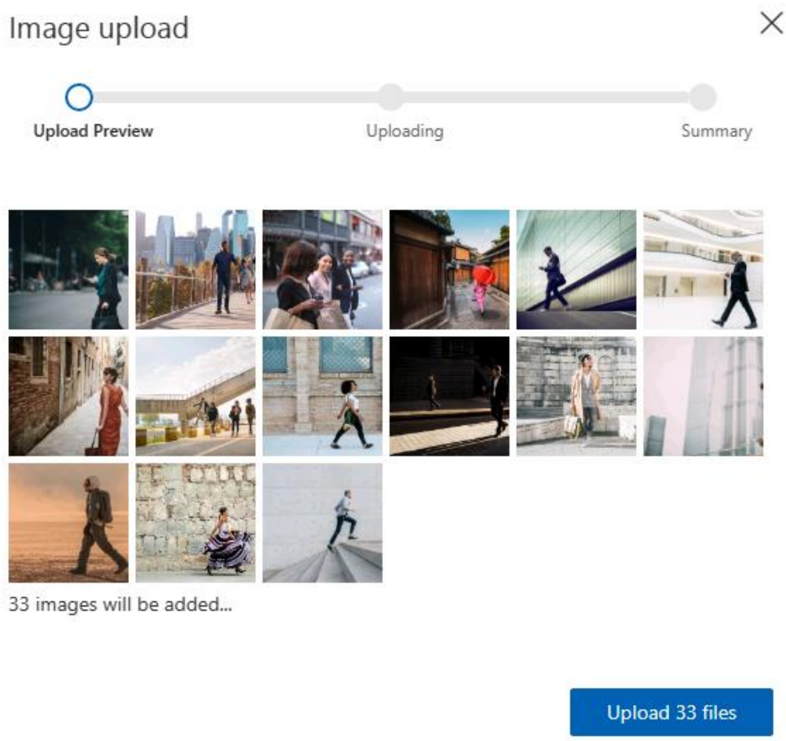
3. In the Custom Vision portal, create a new project with the following settings. Note that the “Resource” setting is automatically specified for you



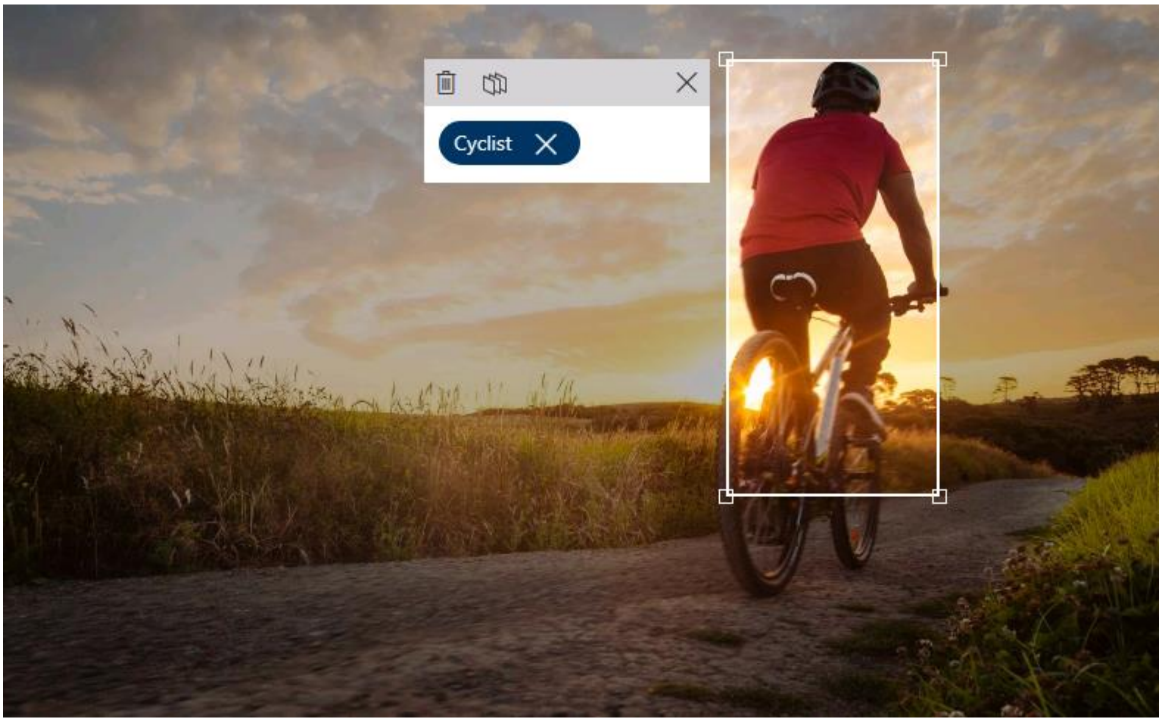
4. Once the new project has been created, you will be redirected the new project portal, which is empty now and looks likes the following



5. Now, click on “**Add images**” and in the opened window, select all the images that you have previously downloaded and unzipped



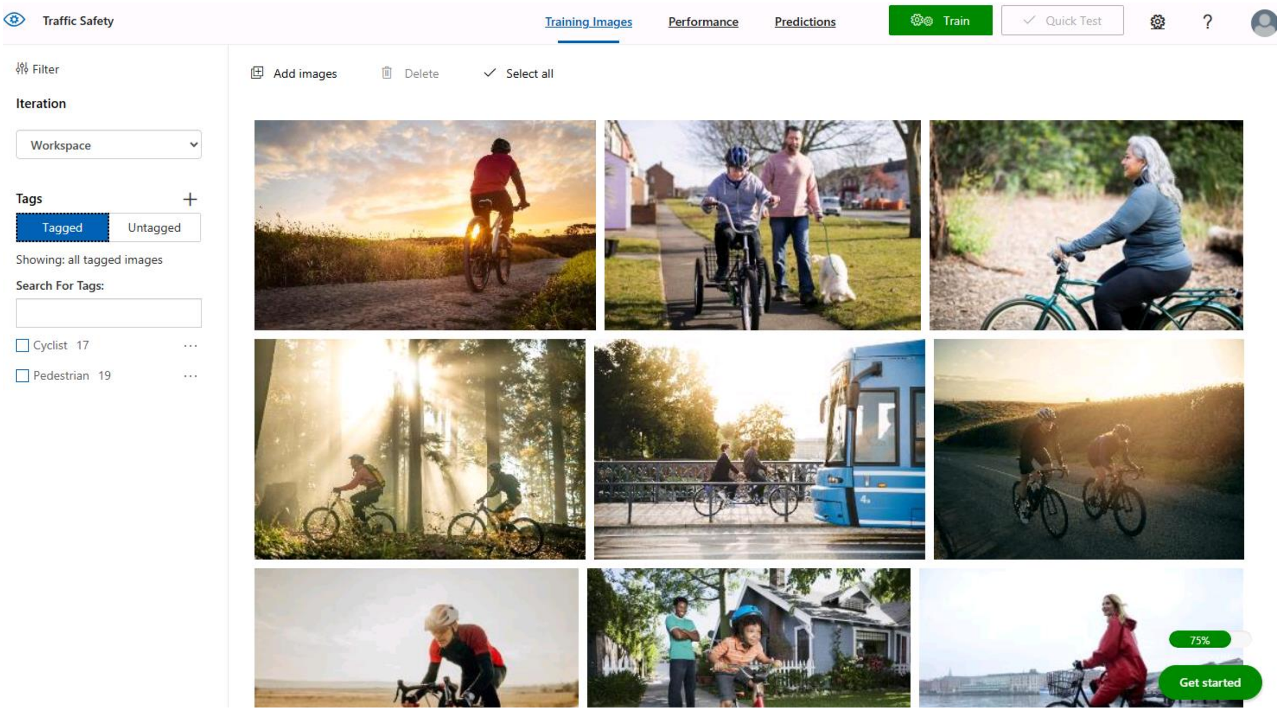
6. After all the images have been uploaded, select the first one and double click to open it. Hold the mouse over any object (cyclist or pedestrian) in the image until an automatically detected region is displayed. Then select the object, and if necessary, resize the region to tightly surround it. Alternatively, you can simply drag around the object to create a region.



7. When the object is tightly selected within the rectangular region, enter the appropriate tag for the object (**Cyclist** or **Pedestrian**) and use the Tag region (+) button to add the tag to the project.

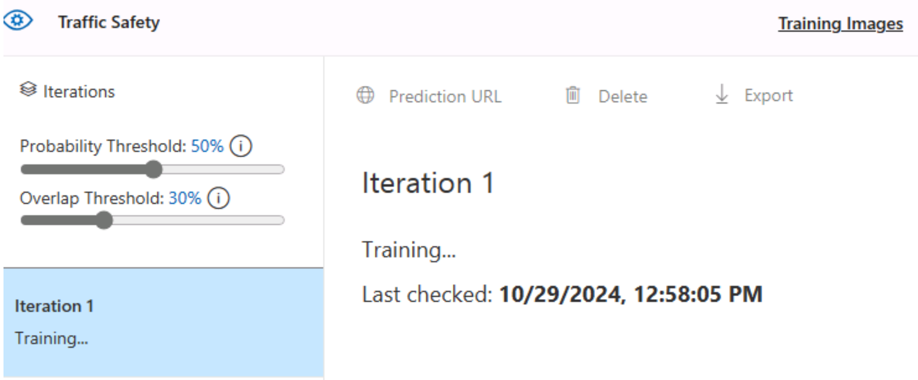
8. Use the Next (>) link on the right to go to the next image and tag its objects. Then just keep working through the entire image collection, tagging each cyclist and pedestrian. ***As you tag the images, note that some images contain multiple objects, potentially of different types. Tag each one, even if they overlap.*** After a tag has been entered once, you can select it from the list when tagging new objects. You can go back and forward through the images to adjust tags.

9. When you have finished tagging the last image, close the Image Detail editor and on the Training Images page, under Tags, select **Tagged** to see all of your tagged images.



10. Now that you’ve tagged the images in your project, you’re ready to train a model. Click on “**Train”** to train an object detection model using the tagged images. Select the “**Quick Training**” option.

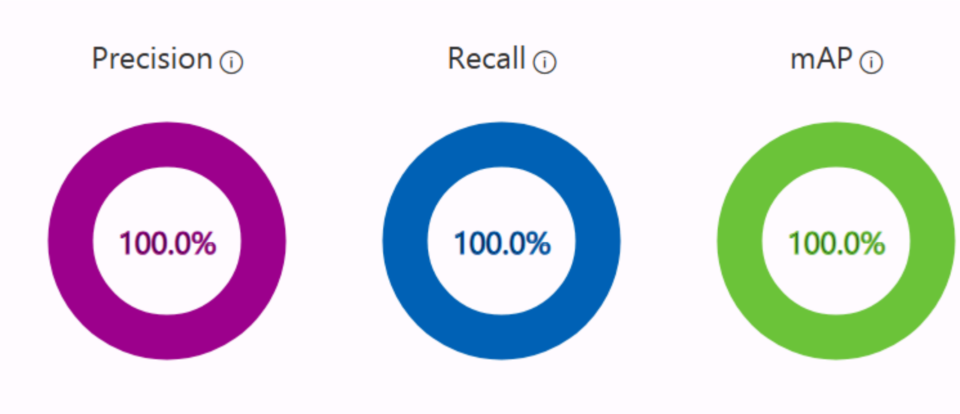
11. When the training is in progress, you should be able to those iterations like the following



12. When training is complete, review the Precision, Recall, and mAP performance metrics - these measure the prediction goodness of the object detection model, and should all be reasonably high.

Iteration 1

Finished training on **10/29/2024, 1:05:27 PM** using **General [A1]** domain
Iteration id: **dfb5d285-50b4-4502-af30-2552a5791eac**



Performance Per Tag

Tag	Precision	Recall	A.P.	Image count
Cyclist	100.0%	100.0%	100.0%	17
Pedestrian	100.0%	100.0%	100.0%	19

13. Adjust the Probability Threshold on the left, increasing it from 50% to 90% and observe the impact on the performance metrics. This setting determines the probability value that each tag evaluation must meet or exceed to be counted as a prediction.

Exercise 3: Test the Trained Model

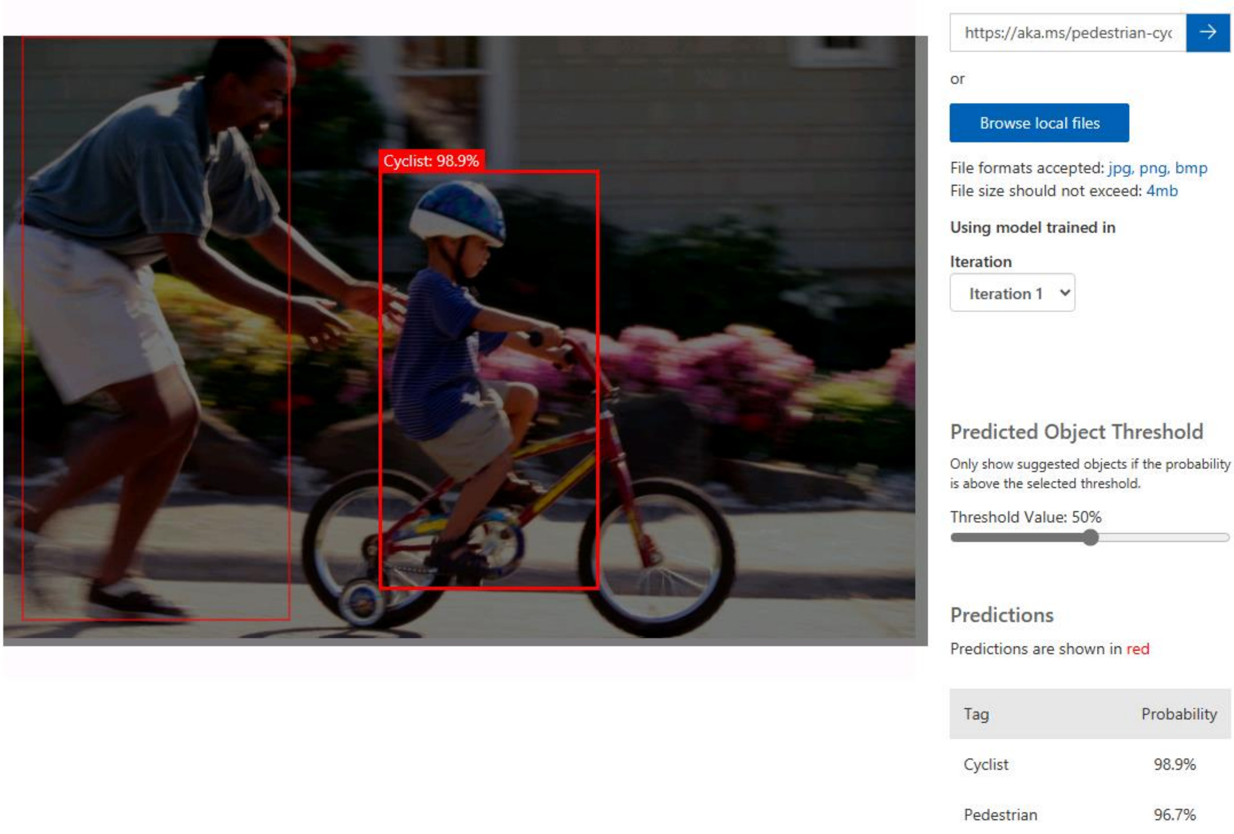
1. Before publishing this iteration of the model for applications to use, you should test it. Above the performance metrics, click on “**Quick Test**”.

2. In the Image URL box, type **https://aka.ms/pedestrian-cyclist** and click the quick test image button.

3. In the panel on the right, under Predictions, each detected object is listed with its tag and probability. Select each object to see it highlighted in the image.

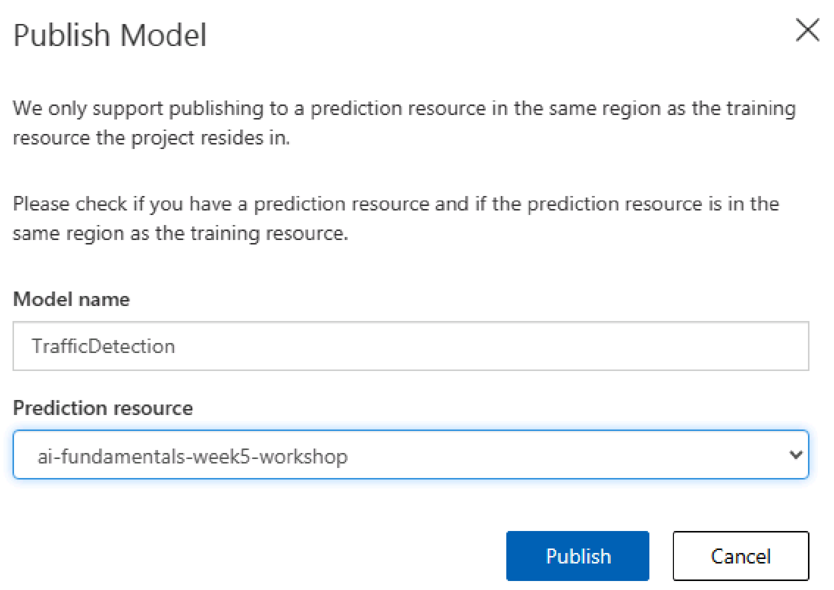
4. The predicted objects may not all be correct - after all, cyclists and pedestrians share many common features. The predictions that the model is most con dent about have the highest probability values. Use the Threshold Value slider to eliminate objects with a low probability. You should be able to select a point at which only correct predictions are included (probably at around 50%).

5. Close the Quick Test window.

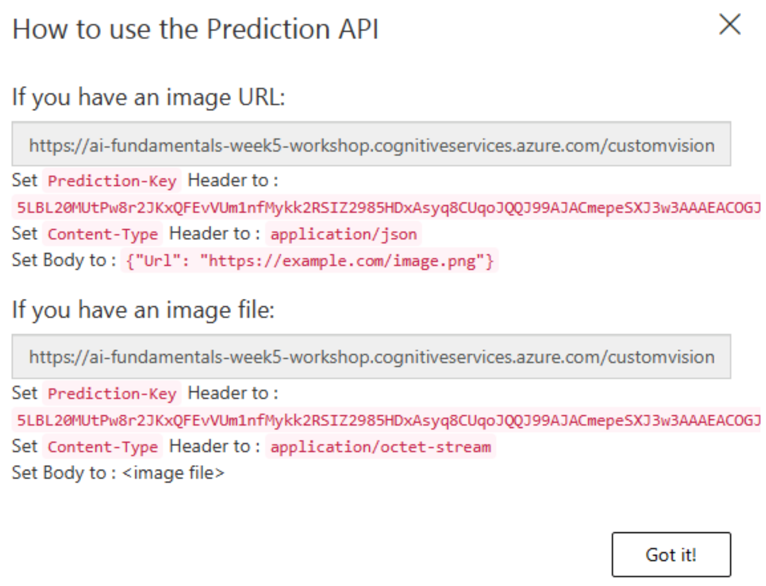


Exercise 4: Publish the Object Detection Model

1. Now you’re ready to publish your trained model and use it from a client application. Click Publish to publish the trained model with the following settings.



2. After publishing, click on the “**Prediction URL**” icon (next to Unpublish) to see information required to use the published model. Later, you will need the appropriate URL and Prediction-Key values to get a prediction from an Image URL, so keep this dialog box open and carry on to the next task.



3. Now, switch back to the browser tab containing the Azure portal, and select the Cloud shell button at the top of the page to the right of the search box (we did this in the first part!). This opens a cloud shell pane at the bottom of the portal.

4. In the command shell, enter the following commands to download the les for this exercise and save them in a folder named ai-900:

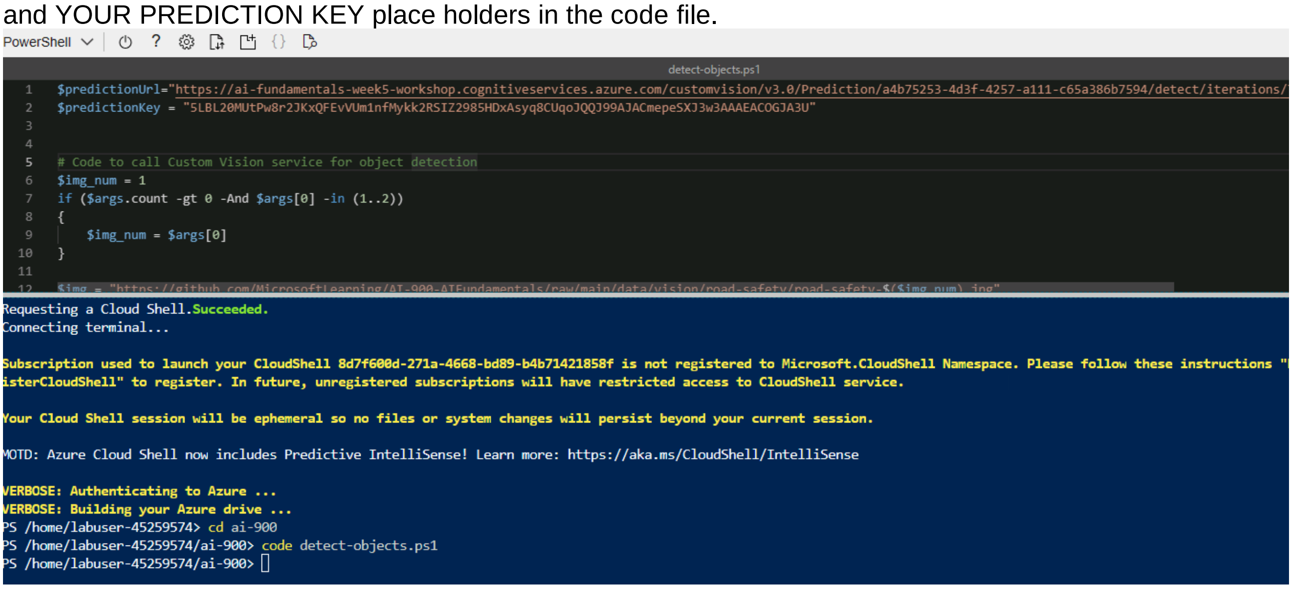
```
rm -r ai-900 -f
git clone https://github.com/MicrosoftLearning/AI-900-AIFundamentals ai-900
```

5. After the package has been downloaded, enter the following commands to change to the ai-900 directory and edit the code le for this exercise:

```
cd ai-900
code detect-objects.ps1
```

6. Notice how this opens an editor. Don't worry too much about the details of the code. The important thing is that it starts with some code to specify the prediction URL and key for your Custom Vision model.

7. Get the **prediction URL** and **prediction key** from the dialog box you left open in the browser tab for your Custom Vision project. Copy the strings and use these values to replace the YOUR PREDICTION URL and YOUR PREDICTION KEY place holders in the code file.



Exercise 5: Test the Published Model

1. Now you can use the sample client application to detect objects in an image. In the PowerShell panel, enter the following command to run the code:

```
./detect-objects.ps1 1
```

2. This code uses your model to classify the following image. Review the prediction results.

```
VERBOSE: Authenticating to Azure ...
VERBOSE: Building your Azure drive ...
PS /home/labuser-45259574> cd ai-900
PS /home/labuser-45259574/ai-900> code detect-objects.ps1
PS /home/labuser-45259574/ai-900> ./detect-objects.ps1 1
Analyzing image...

Cyclist (0.9893003%)
@{left=0.40519705; top=0.2194964; width=0.2392306; height=0.6896839}

Pedestrian (0.96727955%)
@{left=0.019545043; top=0; width=0.29049358; height=0.9594586}

PS /home/labuser-45259574/ai-900>
```

3. Now let us try another image! The command is:

```
./detect-objects.ps1 2
```

4. Hopefully, your object detection model did a good job of detecting pedestrians and cyclists in the test images.

```
VERBOSE: Authenticating to Azure ...
VERBOSE: Building your Azure drive ...
PS /home/labuser-45259574> cd ai-900
PS /home/labuser-45259574/ai-900> code detect-objects.ps1
PS /home/labuser-45259574/ai-900> ./detect-objects.ps1 1
Analyzing image...

Cyclist (0.9893003%)
@{left=0.40519705; top=0.2194964; width=0.2392306; height=0.6896839}

Pedestrian (0.96727955%)
@{left=0.019545043; top=0; width=0.29049358; height=0.9594586}

PS /home/labuser-45259574/ai-900> ./detect-objects.ps1 2
Analyzing image...

Pedestrian (0.9962042%)
@{left=0.5724398; top=0.4060511; width=0.122341275; height=0.41031224}

Pedestrian (0.99292976%)
@{left=0.022322554; top=0.387865; width=0.11137797; height=0.48916173}

Pedestrian (0.9911225%)
@{left=0.42075658; top=0.45019436; width=0.06477347; height=0.28085572}

Pedestrian (0.98948973%)
@{left=0.2044107; top=0.37521744; width=0.06941423; height=0.40398377}

Pedestrian (0.98928344%)
@{left=0.69316214; top=0.41026586; width=0.055287838; height=0.35492718}

Pedestrian (0.9743027%)
@{left=0.4948104; top=0.42852288; width=0.07228327; height=0.32701743}
```