Custom Vision on Fork scissors dataset using GUI

In [27]:
```
from azure.cognitiveservices.vision.customvision.training import CustomVisionTrainingClient
from azure.cognitiveservices.vision.customvision.prediction import CustomVisionPredictionClient
from azure.cognitiveservices.vision.customvision.training.models import ImageFileCreateBatch, ImageFileCreateEnt
from msrest.authentication import ApiKeyCredentials
import time
```

In [2]:
```
ENDPOINT = "https://fvdzfczs.cognitiveservices.azure.com/"
training_key = "de004d779e2842408c896a6688efbe36"
prediction_key = "2d5c54b285274300bc0924a6512d9d5b"
prediction_resource_id ="/subscriptions/f468ceaa-a610-4b88-9742-2b3e8f4ef76c/resourceGroups/Day2/providers/Micro
```

In [28]:
```
credentials = ApiKeyCredentials(in_headers={"Training-key": training_key})
trainer = CustomVisionTrainingClient(ENDPOINT, credentials)
prediction_credentials = ApiKeyCredentials(in_headers={"Prediction-key": prediction_key})
predictor = CustomVisionPredictionClient(ENDPOINT, prediction_credentials)
```

In [29]:
```
# Detection model
publish_iteration_name = "detectModel"

# Find the object detection domain
obj_detection_domain = next(domain for domain in trainer.get_domains() if domain.type == "ObjectDetection" and d

# Create a new project
print ("Creating project...")
project = trainer.create_project("My Detection Project", domain_id=obj_detection_domain.id)
```

```
Creating project...
```

In [30]:
```
# Make two tags for the project
fork_tag = trainer.create_tag(project.id, "fork")
scissors_tag = trainer.create_tag(project.id, "scissors")
```

In [31]:

```
fork_image_regions = {
    "fork_1": [ 0.145833328, 0.3509314, 0.5894608, 0.238562092 ],
    "fork_2": [ 0.294117659, 0.216944471, 0.534313738, 0.5980392 ],
    "fork_3": [ 0.09191177, 0.0682516545, 0.757352948, 0.6143791 ],
    "fork_4": [ 0.254901975, 0.185898721, 0.5232843, 0.594771266 ],
    "fork_5": [ 0.2365196, 0.128709182, 0.5845588, 0.71405226 ],
    "fork_6": [ 0.115196079, 0.133611143, 0.676470637, 0.6993464 ],
    "fork_7": [ 0.164215669, 0.31008172, 0.767156839, 0.410130739 ],
    "fork_8": [ 0.118872553, 0.318251669, 0.817401946, 0.225490168 ],
    "fork_9": [ 0.18259804, 0.2136765, 0.6335784, 0.643790841 ],
    "fork_10": [ 0.05269608, 0.282303959, 0.8088235, 0.452614367 ],
    "fork_11": [ 0.05759804, 0.0894935, 0.9007353, 0.3251634 ],
    "fork_12": [ 0.3345588, 0.07315363, 0.375, 0.9150327 ],
    "fork_13": [ 0.269607842, 0.194068655, 0.4093137, 0.6732026 ],
    "fork_14": [ 0.143382356, 0.218578458, 0.7977941, 0.295751631 ],
    "fork_15": [ 0.19240196, 0.0633497, 0.5710784, 0.8398692 ],
    "fork_16": [ 0.140931368, 0.480016381, 0.6838235, 0.240196079 ],
    "fork_17": [ 0.305147052, 0.2512582, 0.4791667, 0.5408496 ],
    "fork_18": [ 0.234068632, 0.445702642, 0.6127451, 0.344771236 ],
    "fork_19": [ 0.219362751, 0.141781077, 0.5919118, 0.6683006 ],
    "fork_20": [ 0.180147052, 0.239820287, 0.6887255, 0.235294119 ]
}

scissors_image_regions = {
    "scissors_1": [ 0.4007353, 0.194068655, 0.259803921, 0.6617647 ],
    "scissors_2": [ 0.426470578, 0.185898721, 0.172794119, 0.5539216 ],
    "scissors_3": [ 0.289215684, 0.259428144, 0.403186262, 0.421568632 ],
    "scissors_4": [ 0.343137264, 0.105833367, 0.332107842, 0.8055556 ],
    "scissors_5": [ 0.3125, 0.09766343, 0.435049027, 0.71405226 ],
    "scissors_6": [ 0.379901975, 0.24308826, 0.32107842, 0.5718954 ],
    "scissors_7": [ 0.341911763, 0.20714055, 0.3137255, 0.6356209 ],
    "scissors_8": [ 0.231617644, 0.08459154, 0.504901946, 0.8480392 ],
    "scissors_9": [ 0.170343131, 0.332957536, 0.767156839, 0.403594762 ],
    "scissors_10": [ 0.204656869, 0.120539248, 0.5245098, 0.743464053 ],
    "scissors_11": [ 0.05514706, 0.159754932, 0.799019635, 0.730392158 ],
    "scissors_12": [ 0.265931368, 0.169558853, 0.5061275, 0.606209159 ],
    "scissors_13": [ 0.241421565, 0.184264734, 0.448529422, 0.6830065 ],
    "scissors_14": [ 0.05759804, 0.05027781, 0.75, 0.882352948 ],
    "scissors_15": [ 0.191176474, 0.169558853, 0.6936275, 0.6748366 ],
    "scissors_16": [ 0.1004902, 0.279036, 0.6911765, 0.477124184 ],
    "scissors_17": [ 0.2720588, 0.131977156, 0.4987745, 0.6911765 ],
    "scissors_18": [ 0.180147052, 0.112369314, 0.6262255, 0.6666667 ],
```

```
    "scissors_19": [ 0.333333343, 0.0274019931, 0.443627447, 0.852941155 ],
    "scissors_20": [ 0.158088237, 0.04047389, 0.6691176, 0.843137264 ]
}
```

In [32]:

```
base_image_location = "C:/Users/Jaswanth Reddy/Desktop/Image dataset/"
print ("Adding images...")
tagged_images_with_regions = []

for file_name in fork_image_regions.keys():
    x,y,w,h = fork_image_regions[file_name]
    regions = [ Region(tag_id=fork_tag.id, left=x,top=y,width=w,height=h) ]

    with open(base_image_location + "fork/" + file_name + ".jpg", mode="rb") as image_contents:
        tagged_images_with_regions.append(ImageFileCreateEntry(name=file_name, contents=image_contents.read(), r

for file_name in scissors_image_regions.keys():
    x,y,w,h = scissors_image_regions[file_name]
    regions = [ Region(tag_id=scissors_tag.id, left=x,top=y,width=w,height=h) ]

    with open(base_image_location + "scissors/" + file_name + ".jpg", mode="rb") as image_contents:
        tagged_images_with_regions.append(ImageFileCreateEntry(name=file_name, contents=image_contents.read(), r

upload_result = trainer.create_images_from_files(project.id, ImageFileCreateBatch(images=tagged_images_with_regi
if not upload_result.is_batch_successful:
    print("Image batch upload failed.")
    for image in upload_result.images:
        print("Image status: ", image.status)
    exit(-1)
```

```
Adding images...
```

In [33]:
```
# Training
print ("Training...")
iteration = trainer.train_project(project.id)
while (iteration.status != "Completed"):
    iteration = trainer.get_iteration(project.id, iteration.id)
    print ("Training status: " + iteration.status)
    time.sleep(1)
```

```
Training status: Training
Training status: Training
Training status: Training
Training status: Training
Training status: Training
Training status: Training
```

In [34]:
```
# The iteration is now trained. Publish it to the project endpoint
trainer.publish_iteration(project.id, iteration.id, publish_iteration_name, prediction_resource_id)
print ("Done!")
```

```
Done!
```

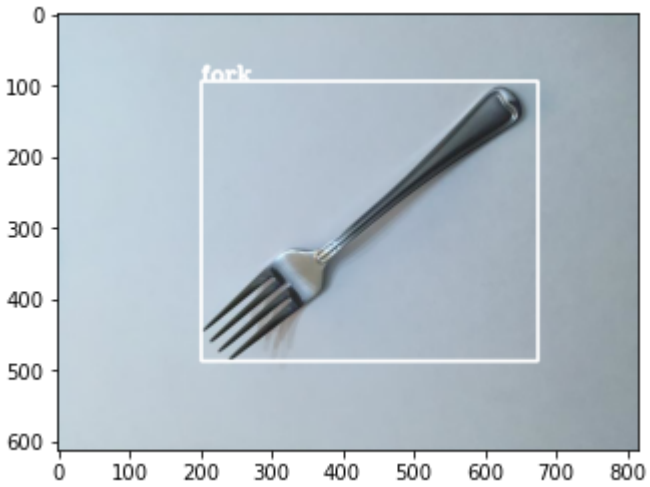
In [35]:
```
# Predicting an image

with open(base_image_location + "/fork/fork_5.jpg", mode="rb") as test_data:
    results = predictor.detect_image(project.id, publish_iteration_name, test_data)

# Display the results.
for prediction in results.predictions:
    print("\t" + prediction.tag_name + ": {0:.2f}% bbox.left = {1:.2f}, bbox.top = {2:.2f}, bbox.width = {3:.2f}
```

```
        fork: 91.81% bbox.left = 0.25, bbox.top = 0.16, bbox.width = 0.58, bbox.height = 0.64
        fork: 0.99% bbox.left = 0.25, bbox.top = 0.51, bbox.width = 0.18, bbox.height = 0.32
        fork: 0.65% bbox.left = 0.30, bbox.top = 0.16, bbox.width = 0.19, bbox.height = 0.33
        fork: 0.53% bbox.left = 0.20, bbox.top = 0.22, bbox.width = 0.17, bbox.height = 0.34
```

In [43]:
```
def detect_object(img_path):
    test_img= cv2.imread(img_path)
    width= test_img.shape[1]
    height= test_img.shape[0]
    print(test_img.shape)
    with open(img_path,mode= "rb") as test_data:
        results= predictor.detect_image(project.id, publish_iteration_name, test_data)

    for pred in results.predictions:
        if pred.probability> 0.02:
            pt1= (width*pred.bounding_box.left, height* pred.bounding_box.top)
            pt2= (width* (pred.bounding_box.left + pred.bounding_box.width),height* (pred.bounding_box.top + pre

            cv2.rectangle(test_img, (int(pt1[0]),int(pt1[1])), (int(pt2[0]),int(pt2[1])), (255,255,255), 4)
            cv2.putText(test_img,pred.tag_name, (int(pt1[0]),int(pt1[1])), cv2.FONT_HERSHEY_COMPLEX, 1,(255,255,
        resized_symm=cv2.resize(test_img,((int(test_img.shape[1]/2)),int(test_img.shape[0]/2)))
        cv2.imshow("Image",resized_symm)

        cv2.waitKey(0)
        cv2.destroyAllWindows()
        plt.imshow(test_img)
```

In [44]:
```
import cv2
import matplotlib.pyplot as plt
path=r"C:\Users\Jaswanth Reddy\Desktop\Image dataset\fork\fork_5.jpg"
detect_object(path)
```

```
(612, 816, 3)
```



GUI Program

In [51]:
```
from tkinter import *
from tkinter import filedialog
import os
import tkinter as tk
from PIL import Image, ImageTk
import cv2
import matplotlib.pyplot as plt
```

In [78]:

```
def detect_object1(img_path):
    test_img= cv2.imread(img_path)
    width= test_img.shape[1]
    height= test_img.shape[0]
    print("width",width)
    print(test_img.shape)
    with open(img_path,mode= "rb") as test_data:
        results= predictor.detect_image(project.id, publish_iteration_name, test_data)

    for pred in results.predictions:
        if pred.probability> 0.02:
            pt1= (width*pred.bounding_box.left, height* pred.bounding_box.top)
            pt2= (width* (pred.bounding_box.left + pred.bounding_box.width),height* (pred.bounding_box.top + pre

            cv2.rectangle(test_img, (int(pt1[0]),int(pt1[1])), (int(pt2[0]),int(pt2[1])), (255,255,255), 4)
            cv2.putText(test_img,pred.tag_name, (int(pt1[0]),int(pt1[1])), cv2.FONT_HERSHEY_COMPLEX, 1,(255,255,
        resized_symm=cv2.resize(test_img,((int(test_img.shape[1]/2)),int(test_img.shape[0]/2)))
        #plt.imshow(test_img)
        return test_img
```

In [*]:
```
def showImage():
    file= filedialog.askopenfilename(initialdir= os.getcwd(), title= "Select an image", filetypes= (('JPG File',
    img=detect_object1(file)
    img=Image.fromarray(img)
    img= ImageTk.PhotoImage(img)
    label= Label(root,image=img)
    label.image=img
    label.pack()

root = Tk()
frame= Frame(root)
frame.pack(side= BOTTOM, padx= 15, pady= 15)


button= Button(frame, text= "Pick an image", command= showImage)
button.pack(side= tk.LEFT)

button2= Button(frame, text= "Exit", command= root.destroy)
button2.pack(side= tk.LEFT, padx=10)

root.title("My Object Recognizer")
root.geometry("300x300")
root.mainloop()
```

```
width 816
(612, 816, 3)
```

In [ ]:

In [ ]:

In [ ]:

In [ ]: