Custom object detection(GUI)

In [1]:
```
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://001customvision.cognitiveservices.azure.com/"
training_key = "f57767e29090480fb01108a36f77a878"
prediction_key = "0f35eed01c30477d9aaad60d6e38ee50"
prediction_resource_id = "/subscriptions/54c4256e-bb50-4fbd-895d-da32982a5dad/resourceGroups/ashish/providers/Mi
```

In [3]:
```
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 [4]:
```
publish_iteration_name = "classifyModel"

credentials = ApiKeyCredentials(in_headers={"Training-key": training_key})
trainer = CustomVisionTrainingClient(ENDPOINT, credentials)

# Create a new project
print ("Creating project...")
project = trainer.create_project("My New Project")
```

```
Creating project...
```

In [5]:
```
# Make two tags in the new project
cat_tag = trainer.create_tag(project.id, "cat")
dog_tag = trainer.create_tag(project.id, "dog")
```

In [7]:
```
# Update this with the path to where you downloaded the images.
import glob
import cv2
base_image_location = "C:/Users/Jaswanth Reddy/Desktop/Image dataset/api_cat_dog/"

print("Adding images...")
image_list = []


for image_num in range(1, 26):
    file_name = "cat_{}.jpg".format(image_num)
    with open(base_image_location + "cat/" + file_name, "rb") as image_contents:
        image_list.append(ImageFileCreateEntry(name=file_name, contents=image_contents.read(), tag_ids=[cat_tag.


for image_num in range(1, 26):
    file_name = "dog_{}.jpg".format(image_num)
    with open(base_image_location + "dog/" + file_name, "rb") as image_contents:
        image_list.append(ImageFileCreateEntry(name=file_name, contents=image_contents.read(), tag_ids=[dog_tag.


upload_result = trainer.create_images_from_files(project.id, ImageFileCreateBatch(images=image_list))
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 [8]:
```
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)

# 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!")
```

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

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

In [9]:
```
# Now there is a trained endpoint that can be used to make a prediction
prediction_credentials = ApiKeyCredentials(in_headers={"Prediction-key": prediction_key})
predictor = CustomVisionPredictionClient(ENDPOINT, prediction_credentials)
with open(r"C:\Users\Jaswanth Reddy\Desktop\Image dataset\cat_dog\train\cat\cat.2981.jpg", "rb") as image_conter
    results = predictor.classify_image(project.id, publish_iteration_name, image_contents.read())
    # Display the results.
    for prediction in results.predictions:
        print("\t" + prediction.tag_name +": {0:.2f}%".format(prediction.probability * 100))
```

```
        cat: 61.15%
        dog: 40.10%
```

GUI Program

In [10]:
```
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 [25]:
```
def showImage():
    file= filedialog.askopenfilename(initialdir= os.getcwd(), title= "Select an image", filetypes= (('JPG File',
    text=detect_object1(file)
    img=cv2.imread(file)
    img=cv2.resize(img,(100,100))
    img=Image.fromarray(img)
    img= ImageTk.PhotoImage(img)
    label= Label(root,image=img)
    label.image=img
    label.pack()
    mylabel=Label(root,text=text)
    mylabel.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()
```

In [20]:
```
def detect_object1(img_path):
    test_img= cv2.imread(img_path)
    width= test_img.shape[1]
    height= test_img.shape[0]
    with open(img_path,mode= "rb") as image_contents:
        results = predictor.classify_image(project.id, publish_iteration_name, image_contents.read())

    for prediction in results.predictions:
        return ("\t" + prediction.tag_name +": {0:.2f}%".format(prediction.probability * 100))
```

In [ ]:
```
```