USING AZURE + AI TO DETECT EMOTION

BERND VERST (@BERNDVERST)

DEVELOPER ADVOCATE, MICROSOFT

In this workshop we will build a Python app that uses Cognitive Services APIs to detect emotions in faces and then store the emotions detected.

RESOURCES

Everything covered here is available at

AKA.MS/HACKCBSWORKSHOP

WHAT ARE WE BUILDING?

- Client
  - Takes a photo from our webcam
  - Sends the photo to our Backend API
- Server
  - Receives the photo, detects faces and emotions
  - Stores the emotions in a database

AZURE TECHNOLOGIES

- Azure App Service
- Cognitive Services
- CosmosDB

OPEN SOURCE TECHNOLOGIES

- Python 3.7
- Flask
- OpenCV

PREREQUISITES

- Python 3.7
- Microsoft Azure Account
- Visual Studio Code
- Python Extension for Visual Studio Code
- Azure App Service Extension for Visual Studio Code
- Azure Cosmos DB Extension for Visual Studio Code

LET'S GET STARTED

BUILD AN APP TO TAKE A PHOTO

```
pip3 install opencv-python
```

Create `picturetaker.py` with code:

```
import cv2

cam = cv2.VideoCapture(0)
cv2.namedWindow('Press space to take a photo')

while True:
  ret, frame = cam.read()
  cv2.imshow('Press space to take a photo', frame)

  key = cv2.waitKey(1)
  if key%256 == 32:
    break

cam.release()
cv2.destroyAllWindows()
```

CREATE A FLASK WEB APP

Create file `requirements.txt` with content:

```
opencv-python
flask
```

Install requirements by running terminal command

```
pip3 install -r requirements.txt
```

CREATE A FLASK WEB APP (CONTINUED)

Create a file called `app.py` with code:

```python
from flask import Flask

app = Flask(__name__)

@app.route('/')
def home():
  return 'Hello World'
```

DEPLOYING TO AN APP SERVICE

Using Visual Code and the App Service extension we are deploying our web app to the cloud.

Detailed steps

ADD A REST (WEB) API ROUTE TO THE FLASK APP

We want to be able to receive image data

In `app.py` add imports:

```
from flask import Flask, request
import base64
```

Add code

```
@app.route('/image', methods=['POST'])
def upload_image():
  json = request.get_json()
  base64_image = base64.b64decode(json['image'])

  return 'OK'
```

ANALYSE THE PHOTO USING AI

Using the *Face API* from *Cognitive Services* we will detect a face and emotion.

- Open the Azure portal at portal.azure.com
- Create a resource of type *Face*
- Once completed, take note of the *Endpoint* and *Keys*

ANALYSE THE PHOTO USING AI (CONTINUED)

Add the Face API SDK to `requirements.txt` by adding the line

```
azure-cognitiveservices-vision-face
```

Install the requirements with

```
pip3 install -r requirements.txt
```

ANALYSE THE PHOTO USING AI (CONTINUED)

Add imports to `app.py`

```
from azure.cognitiveservices.vision.face import FaceClient
from msrest.authentication import CognitiveServicesCredentials
import io
import uuid
```

Add two variables:

```
face_api_endpoint = 'https://centralus.api.cognitive.microsoft.co
face_api_key = '<key>'
```

ANALYSE THE PHOTO USING AI (CONTINUED)

Initialize the Face API client and add a helper function

```python
credentials = CognitiveServicesCredentials(face_api_key)
face_client = FaceClient(face_api_endpoint, credentials=credentia

def best_emotion(emotion):
  emotions = {}
  emotions['anger'] = emotion.anger
  emotions['contempt'] = emotion.contempt
  emotions['disgust'] = emotion.disgust
  emotions['fear'] = emotion.fear
  emotions['happiness'] = emotion.happiness
  emotions['neutral'] = emotion.neutral
  emotions['sadness'] = emotion.sadness
  emotions['surprise'] = emotion.surprise
  return max(zip(emotions.values(), emotions.keys()))[1]
```

ANALYSE THE PHOTO USING AI (CONTINUED)

Update the `upload_image` function by adding this code snippet before the return statement.

```
image = io.BytesIO(base64_image)
faces = face_client.face.detect_with_stream(image,
    return_face_attributes=['emotion'])

for face in faces:
    doc = {
        'id'  : str(uuid.uuid4()),
        'emotion': best_emotion(face.face_attributes.emotion)
        }
```

SAVE THE FACE DETAILS TO A DATABASE

We will store data in a *Cosmos DB* database. Let's create one.

- Database `workshop`
- Collection `faces`
- Partition key blank
- Initial Throughput 400

Detailed instructions

SAVE THE FACE DETAILS TO A DATABASE (CONTINUED)

Add the Cosmos SDK to the `requirements.txt`

```
azure.cosmos
```

Install the requirements

```
pip3 install -r requirements.txt
```

SAVE THE FACE DETAILS TO A DATABASE (CONTINUED)

Import the SDKs

```
import azure.cosmos.cosmos_client as cosmos_client
```

Initialize the Cosmos DB client

```
cosmos_url = ''
cosmos_primary_key = ''
cosmos_collection_link = 'dbs/workshop/colls/faces'
client = cosmos_client.CosmosClient(url_connection=cosmos_url,
                                    auth = {'masterKey': cosmos_p
```

Using the Cosmos DB extension, get the connection string and replace the variable values.

SAVE THE FACE DETAILS TO A DATABASE (CONTINUED)

Now use the Cosmos DB SDK

In the `upload_image` function, inside the loop after the `doc` is created update the code:

```
...
for face in faces:
  ...
  client.CreateItem(cosmos_collection_link, doc)
...
```

Deploy the code using the App Service extension.

CALL THE WEB API FROM THE PHOTO TAKING APP

Add `requests` to `requirements.txt` and install via

```
pip3 install -r requirements.txt
```

Add these imports to `picturetaker.py`

```
import requests
import base64
```

CALL THE WEB API FROM THE PHOTO TAKING APP

Add the image upload code

```python
imageUrl = 'https://<Your Web App>.azurewebsites.net/image'

def upload(frame):
  data = {}
  img = cv2.imencode('.jpg', frame)[1]
  data['image'] = base64.b64encode(img).decode()
  requests.post(url=imageUrl, json=data)
```

CALL THE WEB API FROM THE PHOTO TAKING APP

Add a call to the upload function.

```
...
if k%256 == 32:
  upload(frame)
  break
...
```

Start the Debugger

CREATE A WEB PAGE TO VIEW THE RESULTS

Create a template to display the emotions stored in the DB

Create a folder `templates` and file `home.html` within that folder.

```
<!doctype html>
<html>
  <body>
    <table border = 1>
      <tr>
        <td>Emotion</td>
      </tr>
      {% for row in result %}
        <tr>
          <td> {{ row.emotion }} </td>
        </tr>
      {% endfor %}
    </table>
  </body>
</html>
```

CREATE A WEB PAGE TO VIEW THE RESULTS

Now update `app.py` and add this import

```
from flask import Flask, request, render_template
```

Replace the `home` function with this code

```
@app.route('/')
def home():
  docs = list(client.ReadItems(cosmos_collection_link))
  return render_template('home.html', result = docs)
```

CLEANING UP

Delete the resource group we created by visiting the Azure Portal.

THANK YOU!

Everything covered here is available at

AKA.MS/HACKCBSWORKSHOP

You can follow me on social media at `@berndverst`