CODEBOXX

AIC-111 Video Reconstruction

Summary

This document outlines the requirements for Module 11 of the CodeBoxx "Smart Journey." In this module, candidates will focus on video reconstruction tasks, including using Azure OCR to extract data from images and utilizing OpenCV to reassemble shuffled video frames. The module also covers best practices for deploying OCR models on Azure, handling image data, and reconstructing videos from disorganized frames.

Objectives

- Extract data from images using Azure OCR.
- Organize and classify frames.
- Reconstruct video footage using OpenCV.
- Practice deploying OCR models in Azure

Requirements

Oh No! We lost the elevator footage!

The Rocket Elevators Company recently experienced a hard drive issue, resulting in the loss of their surveillance footage. Apparently, some unusual activities were taking place in the elevator. Fortunately, they were able to recover frames from the damaged hard drive, but the files are a complete mess—shuffled and assigned random names. However, when you examined them, you had a brilliant idea. Each frame contains a date and frame number. By using OCR, you can extract this information and reassemble the footage. Your boss thinks you're a genius. Now it's time to shine!

- Use Azure OCR to extract the date and frame number from each image using a cognitive service resource: Azure Cognitive Services
- Classify each frame by its date.

- Reconstruct the videos from the frames using OpenCV.

Hints

- The color of the text printed on each image is always the same.
- The text is always located in the same place on the image.
- Don't hesitate to use intensive processing techniques to handle the images and reconstruct the video.

Resources

- Create an Azure account
- Create a cognitive services resource on the Azure web interface.
- Follow this Quickstart guide for Optical Character Recognition (OCR) in Azure Cognitive Services.

Important: To avoid using up all your credits, be sure to delete resources after you're done. As a best practice, use resource groups to manage resources easily.

Putting a model into production

Over the past few weeks, we've developed several models. Now, how do we deploy them and make them accessible for others to use for predictions? Let's walk through the process of deploying a simple model on AWS.

API and Notification Service

Create an API (What is a RESTful API?) that takes a number as input and returns the predicted value of the approximated function. You will deploy the PyTorch model using SAM, an open-source tool provided by AWS to help create all necessary resources. Follow the steps below:

1. Install Docker, AWS CLI, and SAM.
2. Log in to AWS using the CLI with the `aws configure` command.
3. Create the project template using the SAM command: `sam init`. Options and default values may change, so pay close attention when executing `sam init`. See the configurations below for reference.
4. The result will be files and subfolders in the `./test-inference` folder.
5. Modify the `app.py` and `Dockerfile` to return a prediction for a number provided as input.
6. Run the `sam build` command.

7. Execute `sam local start-api` to start the API locally and test it.
8. Once it works on your machine, deploy the API to AWS using the `sam deploy -g` command. Be sure to carefully review the questions and answers, as the order and default settings may vary (see Sam deploy configuration below).
9. Connect to the AWS web console and verify the resources that were just created, including S3 buckets, ECR, Lambda functions, and API Gateway.
10. Test the deployed API to ensure it returns the correct prediction.
    a. Note: The API call has a 30-second timeout. This should be sufficient, but sometimes starting the Docker container may take longer. If your API call times out, simply retry it, and it should work.
11. Share the URL with us so we can test the API.
12. **To avoid unnecessary charges, once we've confirmed it works, run sam delete to remove all AWS resources.**

- Use at least 90% of the frames to reconstruct the video.
- The video quality should be clear enough to observe and understand the actions inside the elevator.

Resources

- Create a Microsoft Azure free account with free credits: Azure Free Account.
- To avoid using all your credits, make sure to delete resources after you are done using them. A best practice is to use resource groups for easier management: Manage Resource Groups.

Sam init configuration

```
Unset
>> sam init
You can preselect a particular runtime or package type when
using the `sam init` experience.
Call `sam init --help` to learn more.

Which template source would you like to use?
     1 - AWS Quick Start Templates
     2 - Custom Template Location
```

```
Choice: 1

Choose an AWS Quick Start application template
     1 - Hello World Example
     2 - Multi-step workflow
     3 - Serverless API
     4 - Scheduled task
     5 - Standalone function
     6 - Data processing
     7 - Infrastructure event management
     8 - Machine Learning
Template: 8

Which runtime would you like to use?
     1 - python3.9
     2 - python3.8
Runtime: 1

Based on your selections, the only Package type available
is Image.
We will proceed to selecting the Package type as Image.

Based on your selections, the only dependency manager
available is pip.
We will proceed by copying the template using pip.

Select your starter template
     1 - PyTorch Machine Learning Inference API
     2 - Scikit-learn Machine Learning Inference API
     3 - Tensorflow Machine Learning Inference API
     4 - XGBoost Machine Learning Inference API
Template: 1

Would you like to enable X-Ray tracing on the function(s)
in your application?  [y/N]: N

Project name [sam-app]: test-inference
```

```
Cloning                                                    from
https://github.com/aws/aws-sam-cli-app-templates    (process
may take a moment)

    -----------------------
    Generating application:
    -----------------------
    Name: test-inference
    Base Image: amazon/python3.8-base
    Architectures: x86_64
    Dependency Manager: pip
    Output Directory: .

       Next  steps  can  be  found  in  the  README  file  at
./test-inference/README.md


    Commands you can use next
    =========================
     [*] Create pipeline: cd test-inference && sam pipeline
init --bootstrap
    [*] Validate SAM template: sam validate
     [*] Test Function in the Cloud: sam sync --stack-name
{stack-name} --watch
```

Sam deploy configuration

Unset

```
>> sam deploy -g

Configuring SAM deploy
======================

     Looking for config file [samconfig.toml] :  Not Found
     Reading default arguments  :  Success
```

```
        Setting default arguments for 'sam deploy'
        =========================================
        Stack Name [test-inference]:
        AWS Region [us-east-1]:
        #Shows you resources changes to be deployed and
require a 'Y' to initiate deploy
        Confirm changes before deploy [Y/n]:
        #SAM needs permission to be able to create roles to
connect to the resources in your template
        Allow SAM CLI IAM role creation [Y/n]:
        #Preserves the state of previously provisioned
resources when an operation fails
        Disable rollback [y/N]:
        InferenceFunction may not have authorization defined,
Is this okay? [y/N]: Y
        Save arguments to configuration file [Y/n]:
        SAM configuration file [samconfig.toml]:
        SAM configuration environment [default]:
```

Deliverables

A text file containing

- Your full name
- A link to your five videos, following one of these formats: **mod11_video<number>.avi** or **mod11_video<number>.mp4**.
- Everything we need to test your deployment according to the previous tutorial. Don't forget to include the URL to access your API after deployment.