

Project : IA on the Cloud

Detect skin lesion through Android Application

Team



Olivier Randavel

Data Scientist

@ olivier.randavel@gmail.com



Hippolyte Mayard

Data Scientist

@ hippolyte.mayard@dauphine.eu

Skanner App



Skanner App



Take a picture of your skin and get an advice
regarding your lesion

AWS Rekognition

AWS Rekognition

AWS Services Groupes de ressources ★

vocstartsoft/user717349=hip... Virginie du Nord Support

Amazon Rekognition

Custom Labels New

Use Custom Labels

Démos

Détection d'objets et de scènes

Modération des images

Analyse faciale

Reconnaissance de célébrité

Comparaison faciale

Texte dans l'image

Démonstrations de vidéos

Analyse vidéo

Métriques

Ressources supplémentaires

Manuel de mise en route

Télécharger les kits SDK

Détection d'objets et de scènes

Rekognition étiquette automatiquement les objets, concepts et scènes de vos images, et fournit un score de fiabilité.

Need to detect specific objects and scenes unique to your business?
Use Rekognition Custom Labels to quickly build a custom model, no machine learning experience required.



Choisir un exemple d'image Utiliser votre propre image
L'image doit être sous format .jpeg ou .png

Read feature documentation to learn more
Issues or questions? Use feedback button on bottom-left.

▼ Résultats

Skin	99,9 %
Face	96,4 %

► Demande

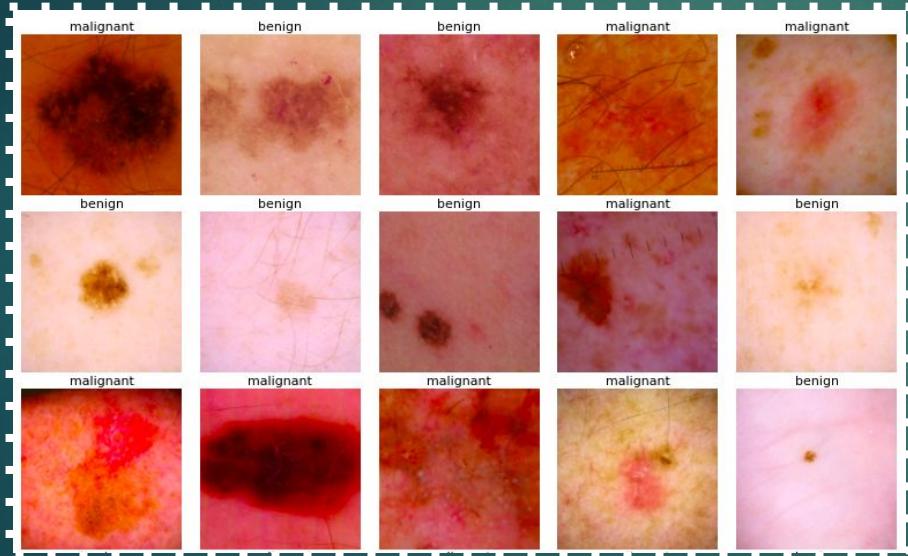
► Réponse



Model & Architecture

The data

ArchitecturSIIM-ISIC Melanoma Classification Kaggle competition



- Over 34 000 skin lesion images
- Highly Imbalanced dataset:
 - 32610 benign lesions
 - 585 malignant lesions

Data pre-processing

TensorFlow input pipeline

- **Oversampling**
- **Data augmentation**
- **Solution for the imbalanced dataset**



Image rotation



Image Zooming



Horizontal flip

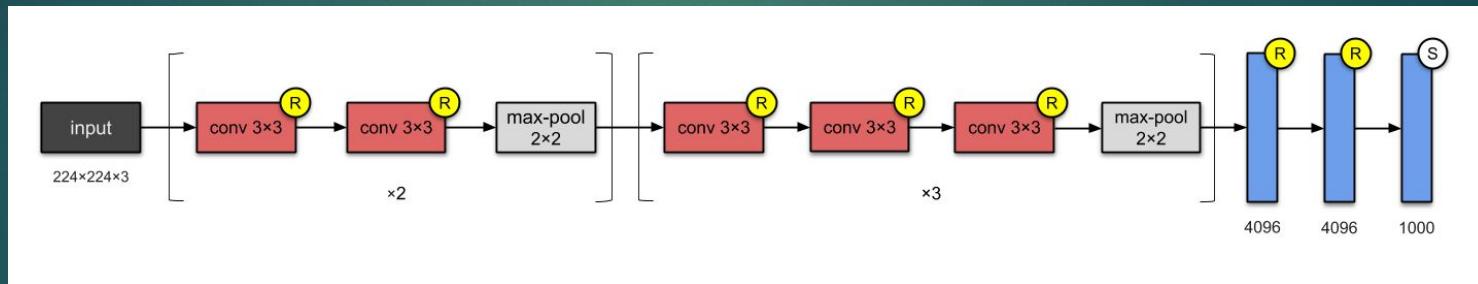


Image Brightness



Model Architecture

Transfer Learning



VGG16 architecture

Model Evaluation

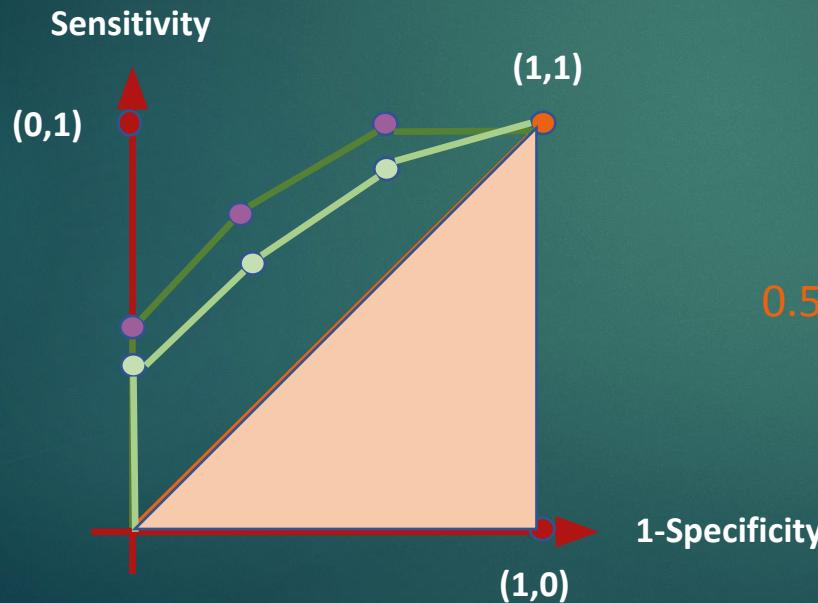
Confusion Matrix

		Predicted Class		
		Positive	Negative	
Actual Class	Positive	True Positive (TP)	False Negative (FN) Type II Error	Sensitivity $\frac{TP}{(TP + FN)}$
	Negative	False Positive (FP) Type I Error	True Negative (TN)	Specificity $\frac{TN}{(TN + FP)}$
		Precision $\frac{TP}{(TP + FP)}$	Negative Predictive Value $\frac{TN}{(TN + FN)}$	Accuracy $\frac{TP + TN}{(TP + TN + FP + FN)}$

Model Evaluation

ROC & AUC metrics

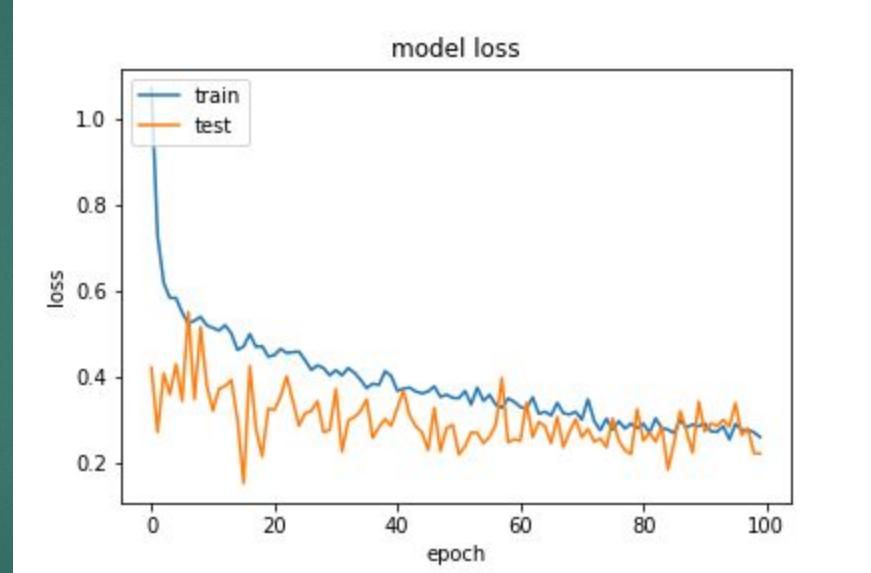
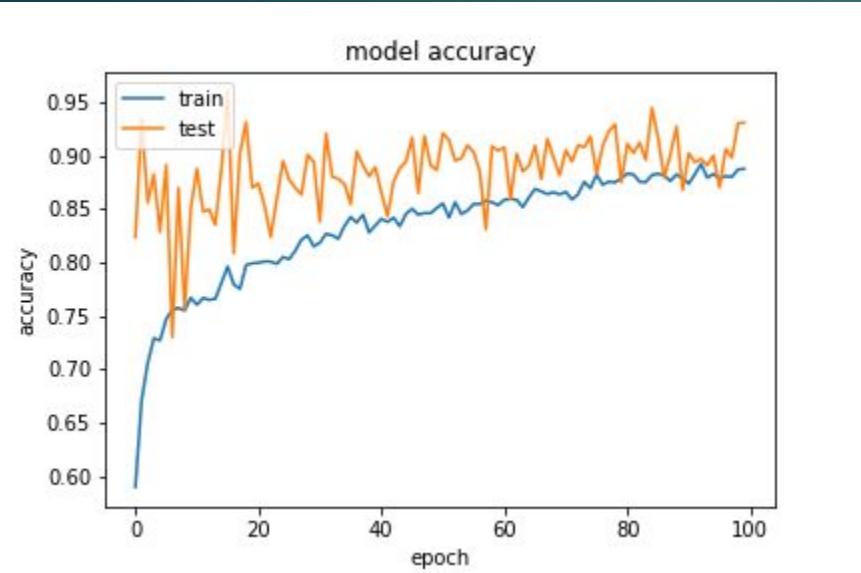
AUC or Area Under the Curve, is a value between 0 and 1 that computes the area under the ROC curve. It is usually higher than 0.5 for non-random models. The higher the value the better is the model overall.



0.5 represents the displayed area.

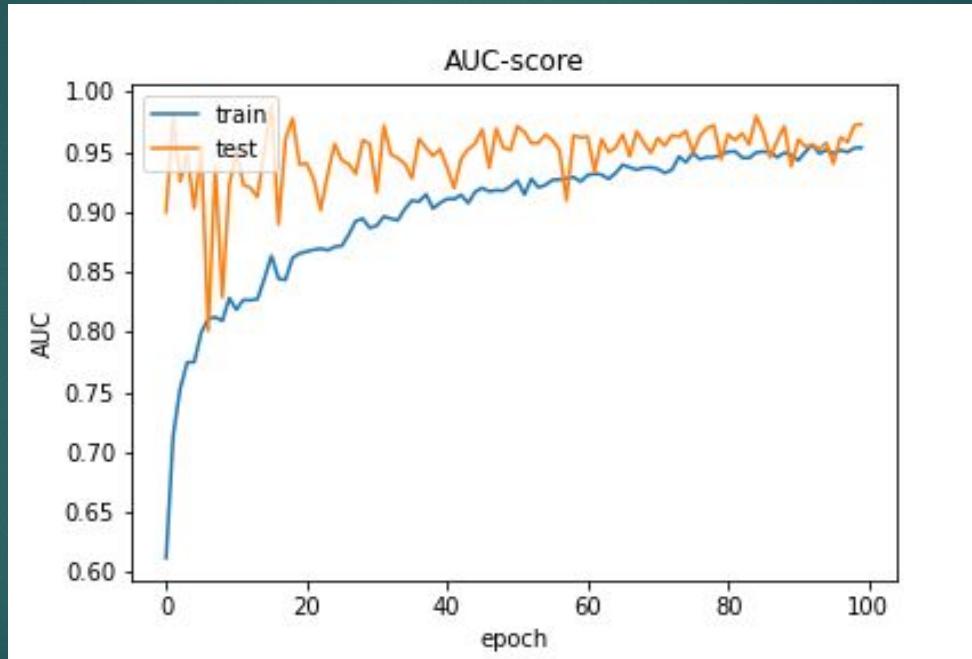
Model Evaluation

Accuracy & Loss



Model Evaluation

AUC metric



Summary

Our Solution :

- ❖ Technology used
- ❖ Test the code locally
- ❖ EC2
- ❖ FileZilla
- ❖ Android Studio
- ❖ Demonstration
- ❖ Limits of the solution

Possible Improvements (1) :

- ❖ Technology used
- ❖ Lambda
- ❖ Lambda Settings
- ❖ Lambda demo
- ❖ Demonstration

Possible Improvements (2) :

- ❖ Technology used
- ❖ Cloud Formation
- ❖ Lambda
- ❖ Demonstration





Our Solution

Technology used

Architecture



Python



Colab



Amazon
EC2



FileZilla



Android
Studio

Libraries



TensorFlow



Flask



Scikit-image

Test the code locally



Python

```
def get_prediction(image):
    start = time.time()

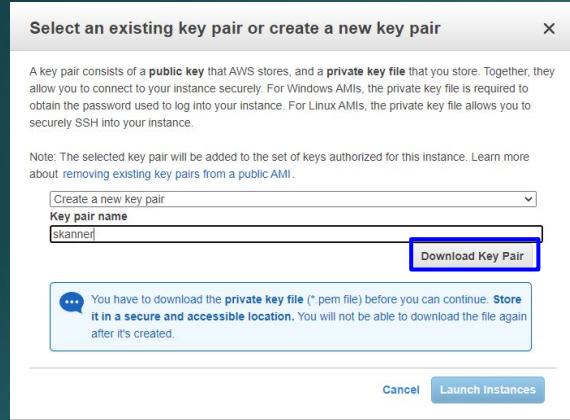
    # We use the keras module here instead of tensorflow because it's
    # easier to work with.
    model = tf.keras.models.load_model('model.h5')
    model.load_weights('weights_only.h5')
    # New model config
    json_config = model.to_json()
    with open(model_config_file, 'w') as json_file:
        json_file.write(json_config)

    # with open(model_config_file) as json_file:
    #     json_config = json.load(json_file)
    # model = tf.keras.models.model_from_json(json_config)

    # Load weights
    # model.load_weights('weights_only.h5')
    arr = tf.keras.preprocessing.image.load_img(image)
    arr = tf.keras.preprocessing.image.img_to_array(arr) / 255
    image = np.array([arr])
    t = model.predict(image, verbose=0)[0]
    logger.info('The prediction is %s with a precision of %s' % (t.argmax(), str(round(t[t.argmax()]*100, 2)) + '%'))
    print('The prediction is %s with a precision of %s' % (t.argmax(), str(round(t[t.argmax()]*100, 2)) + '%'))
    print('The job ended after %s seconds' % (time.time() - start))
    return t.argmax()
```

PyCharm

EC2



```
C:\Users\OlivierRANDEV\Downloads>ssh -i "skanner2.pem" ubuntu@ec2-100-25-33-156.compute-1.amazonaws.com
Welcome to Ubuntu 18.04.4 LTS (GNU/Linux 5.3.0-1023-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

System information as of Sun Jul  5 11:06:36 UTC 2020

System load:  0.0          Processes:      91
Usage of /:   48.7% of 7.69GB  Users logged in:   0
Memory usage: 18%
Swap usage:   0%

20 packages can be updated.
13 updates are security updates.

Last login: Sun Jul  5 07:12:59 2020 from 185.228.230.179
ubuntu@ip-172-31-59-186:~$
```



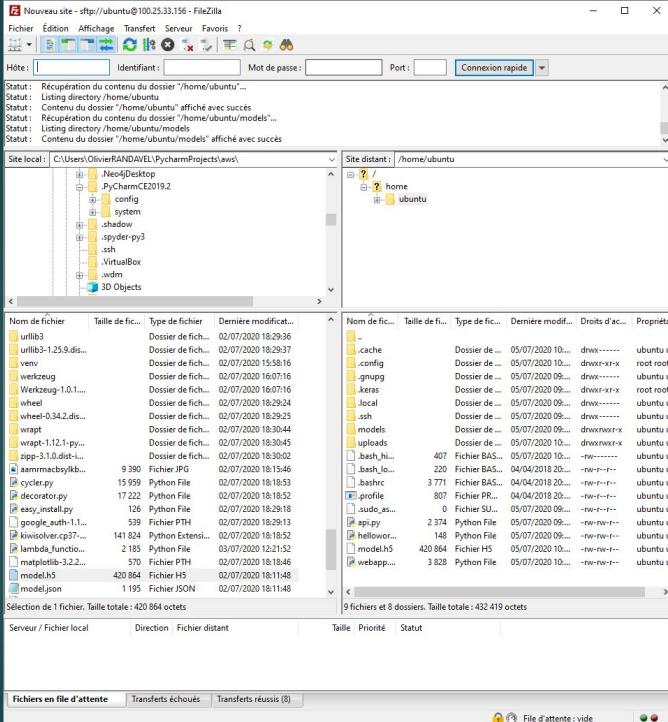
Create a VM :

- Ubuntu Server 18.04 LTS (HVM), SSD Volume Type
- Python3, Pip3

- Download the key pair model to access to the EC2

- Install all libraries

FileZilla

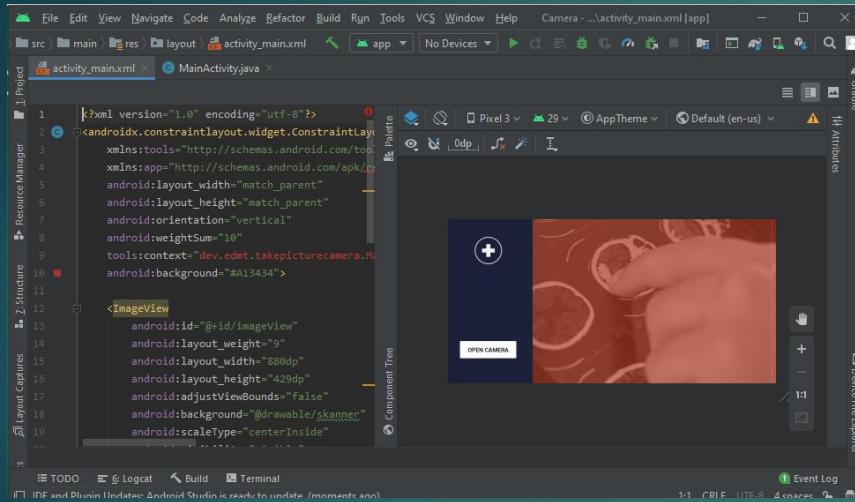


FileZilla

Send file from Windows to EC2 (VM)

Use ssh key with the pem file downloaded previously

Android Studio



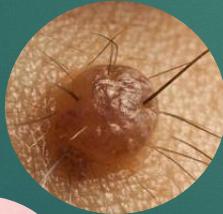
Android
Studio

Implement the app using javascript

Adding a button to take picture

Make a call API to the EC2

Demonstration



Visit the website and follow the instruction

You will upload a picture and get the prediction
regarding the lesion

Limits of the solution



- The VM need to be on in order to use the app, this imply huge cost.
- Besides the EC2 is not scalable in fact multiple calls at the same time will consume more RAM. The VM could run out of memory





Possible Improvements (1)

Technology used

Architecture



AWS Lambda



Python



API Gateway



Cloud Watch



Colab



Android
Studio

Libraries

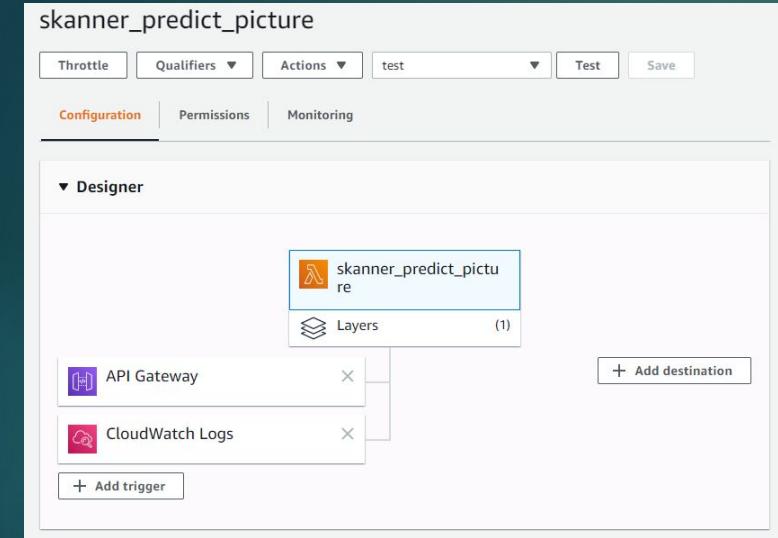


TensorFlow Lite

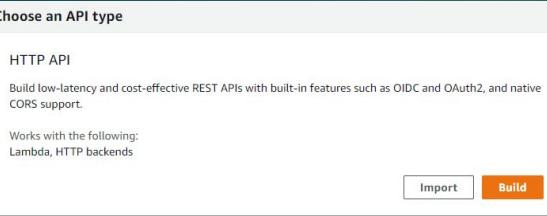


OpenCV

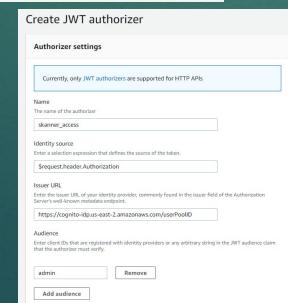
Lambda



The screenshot shows the AWS Lambda function configuration interface for a function named "skanner_predict_picture". The "Designer" tab is selected, displaying a graph where the function is triggered by an "API Gateway" and sends logs to "CloudWatch Logs". There are also "Throttle", "Qualifiers", "Actions", "Test", and "Save" buttons at the top.



The screenshot shows the AWS API Gateway configuration interface, specifically the "Choose an API type" section. It highlights the "HTTP API" option, which is described as providing low-latency and cost-effective REST APIs with built-in features like OAuth2 and CORS support. It also mentions that it works with Lambda and HTTP backends. There are "Import" and "Build" buttons at the bottom.



The screenshot shows the "Create JWT authorizer" interface. It includes sections for "Authorizer settings" (with a note that only JWT authorizers are supported for HTTP APIs), "Identity source" (specifying "skanner_access"), "Issuer URL" (set to "https://cognito-idp.us-east-2.amazonaws.com/userPoolID"), and "Audience" (specifying "admin").



AWS Lambda

Use a lambda function and launch it using an API

Use logs to get information regarding execution

Lambda : settings



Amazon
EC2



TensorFlowLite



- The code needs to be compatible with Amazon-Ubuntu
- The code including all packages need to be lower than 250 Mo



AWS Lambda

Lambda : Demo

skanner_predict_picture

Throttle Qualifiers Actions test Test Save

Execution result: succeeded (logs)

▼ Details

The area below shows the result returned by your function execution. [Learn more about returning results from your function.](#)

```
"rc"
```



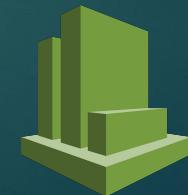
CloudWatch > 2020/07/05/[\$.LATEST]18201302bbc54769aebcb880e7ece8d6 Switch to the original view

Log events

Actions Create Metric Filter

Filter events Clear 1m 30m 1h 12h custom

Timestamp	Message
There are older events to load. Load more .	
▶ 2020-07-05T13:45:32.199+02:00	START RequestId: b74b9881-d5f6-4a5b-a324-1922c32d11cb
▶ 2020-07-05T13:45:32.199+02:00	rc
▶ 2020-07-05T13:45:32.200+02:00	END RequestId: b74b9881-d5f6-4a5b-a324-1922c32d11cb
▶ 2020-07-05T13:45:32.200+02:00	REPORT RequestId: b74b9881-d5f6-4a5b-a324-1922c32d11cb
No newer events at this moment. Auto retry paused. 5	



Cloud Watch

Demonstration



The function receives the call and return “rc”



Possible Improvements (2)

Technology used

Architecture



AWS Lambda



Python



API Gateway



S3



CloudFormation



Android
Studio

Libraries



Cloud Formation

Events (81)		
Timestamp	Logical ID	Status
2020-07-04 20:42:05 UTC+0200	deploy-ml-api	CREATE_COMPLETE
2020-07-04 20:42:02 UTC+0200	s3Operations	CREATE_COMPLETE
2020-07-04 20:42:02 UTC+0200	IntegrationAPI	CREATE_COMPLETE
2020-07-04 20:42:02 UTC+0200	s3Operations	CREATE_IN_PROGRESS
2020-07-04 20:41:58 UTC+0200	s3Operations	CREATE_IN_PROGRESS
2020-07-04 20:41:57 UTC+0200	InferenceAPIProductionStage	CREATE_COMPLETE
2020-07-04 20:41:56 UTC+0200	InferenceAPIProductionStage	CREATE_IN_PROGRESS
2020-07-04 20:41:55 UTC+0200	s3Lambda	CREATE_COMPLETE
2020-07-04 20:41:55 UTC+0200	s3Lambda	CREATE_IN_PROGRESS
2020-07-04 20:41:55 UTC+0200	InferenceAPIProductionStage	CREATE_IN_PROGRESS
2020-07-04 20:41:53 UTC+0200	AppDistribution	CREATE_COMPLETE
2020-07-04 20:41:53 UTC+0200	InferenceAPIDeploymenta799996c84	CREATE_COMPLETE



CloudFormation

We specify the template of the html page, the code of the lambda function, and all libraries in the form of layers stored in the S3

Lambda

deploy-ml-api-Inference-4T28DRD20U57

Throttle Qualifiers Actions Test Save Related functions: Select a function

API Gateway + Add destination + Add trigger

Layers (4)

Function code Info Actions

File Edit Find View Go Tools Window Save Test

Environment deploy-ml-api-Inference-4T28DRD20U57 inference.py

```
1 from mxnet.image import imdecode
2 from gluoncv import model_zoo, data, utils
3 import requests
4 import io
5 import json
6 import base64
7
8 net = model_zoo.get_model('yolo3_mobilenetv1_0_coco', pretrained=True, root='~/mxnet/pretrained')
9
10 def lambda_handler(event, context):
11     try:
12         url = event['img_url']
13         response = requests.get(url)
14         img = imdecode(response.content)
15         x, img = data.transforms.presets.yolo.transform_test((img), short=320)
16         net_IDs, net_outs, net_outs_logit = net(x)
17         output = utils.viz.plot_box(img, bounding_boxes[0], scores[0],
18                                     class_IDs[0], class_names=net.classes)
19         output = BytesIO()
20         output.figure.savefig(f, format='jpeg', bbox_inches='tight')
21         f = BytesIO()
22         output.figure.savefig(f, format='jpeg', bbox_inches='tight')
23         return base64.b64encode(f.getvalue())
24     except Exception as e:
25         print(e)
```

Layers Info Add a layer Edit

Merge order	Name	Layer version	Version ARN
1	MXNet	1	arn:aws:lambda:us-east-1:222789047898:layer:MXNet:1
2	SciPy	1	arn:aws:lambda:us-east-1:222789047898:layer:SciPy:1
3	Pillow	1	arn:aws:lambda:us-east-1:222789047898:layer:Pillow:1
4	GluonCV	1	arn:aws:lambda:us-east-1:222789047898:layer:GluonCV:1



AWS Lambda

GLUON

mxnet

SciPy

pillow

Use a lambda function and launch it using an API

Use logs to get information regarding execution

Demonstration



The model recognize objects among 100 items
(person, plane, bag..)

It helps to segment item and give the prediction accuracy

You can use picture example to test the model

Conclusion



Merci de votre attention