NAAN MUDHALVAN - IBM CLOUD BASED PROJECT

PART – PHASE 5

TITLE : IMAGE RECOGNITION USING IBM CLOUD.

PRESENTED BY
K. PAVITHRA
T. ABIRAMI
K. DHARANI
S. NIVETHA

OBJECTIVE :

The primary objective of utilizing IBM Cloud for image recognition is to leverage the platform's robust and AI-driven capabilities to develop and deploy advanced image recognition solutions. This entails training machine learning models to accurately identify and classify objects, scenes, or patterns within images, which can be instrumental in various domains such as e-commerce, healthcare, and security. By harnessing IBM's cloud infrastructure and services, organizations can achieve enhanced automation, efficiency, and accuracy in tasks that require image analysis, ultimately improving decision-making processes, customer experiences, and operational efficiency in their applications and services.

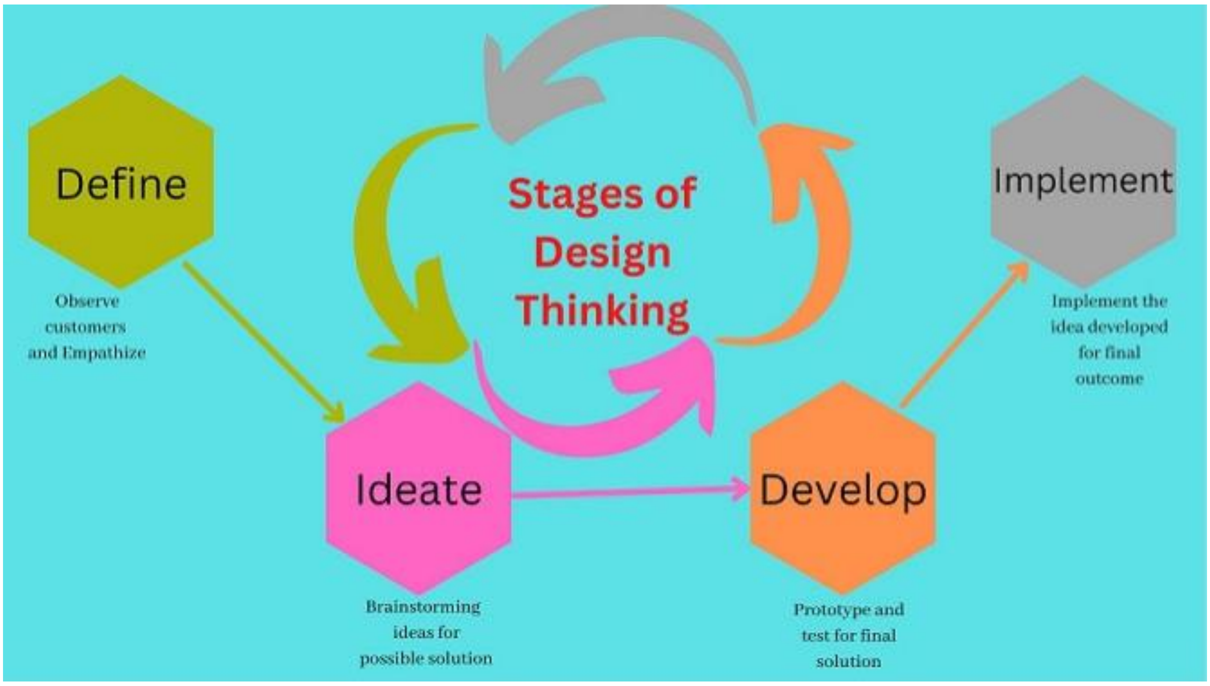
DESIGN AND THINKING PROCESS :

Design thinking is a human-centered approach to problem-solving and innovation. When applying the design thinking process to image recognition using IBM Cloud, the following stages can be considered:

1. Empathize: Begin by understanding the needs, challenges, and goals of the end-users and stakeholders who will benefit from the image recognition system. Conduct interviews, surveys, and observations to gain insights into their specific requirements. Understand the context and use cases where image recognition can be valuable.
2. Define: Clearly define the problem and goals of the image recognition project. Identify the specific tasks or challenges that image recognition can address, such as object detection, classification, or annotation. Define the success criteria and the key performance indicators (KPIs) to measure progress.

3. Ideate: Generate a wide range of ideas for how image recognition can be applied to solve the defined problem. Brainstorm potential features, functionalities, and user interactions. Consider the AI and image recognition services available in IBM Cloud, such as Watson Visual Recognition, and explore how they can be customized to meet the project's objectives.
4. Prototype: Create prototypes or proof-of-concept models using IBM Cloud resources. Experiment with different algorithms and models for image recognition. Develop a minimal viable product (MVP) to demonstrate how the system can work and gather feedback from users to refine the design.
5. Test: Gather user feedback by testing the prototype. Understand how users interact with the image recognition system, and collect data on its performance and usability. Iterate on the design and the AI model based on the feedback and testing results.
6. Implement: Once the prototype has been refined and validated, proceed to implement the full image recognition system using IBM Cloud services. Configure and train the model, integrate it with the desired application or platform, and set up the infrastructure for scalability and reliability.
7. Learn: Continuously monitor the performance of the image recognition system in a real-world environment. Gather data on its accuracy, speed, and user satisfaction. Use this data to make improvements and updates as needed, including retraining the model with new data to adapt to changing requirements.
8. Iterate: Design thinking is an iterative process. Regularly revisit the problem, collect feedback, and make refinements to the image recognition system based on evolving user needs and technology advancements. Ensure that the system remains relevant and effective over time.

Throughout the design thinking process, collaboration among cross-functional teams, including designers, developers, and domain experts, is essential. This approach emphasizes a user-centric and problem-solving mindset, ensuring that the image recognition solution built on IBM Cloud is not only technologically sound but also effectively addresses the specific needs and challenges of the target audience.


Define
Observe customers and Empathize
Stages of Design Thinking
Implement
Implement the idea developed for final outcome
Ideate
Brainstorming ideas for possible solution
Develop
Prototype and test for final solution

DEVELOPMENT PHASES :

Developing an image recognition system using IBM Cloud typically involves several phases, which can be outlined as follows:

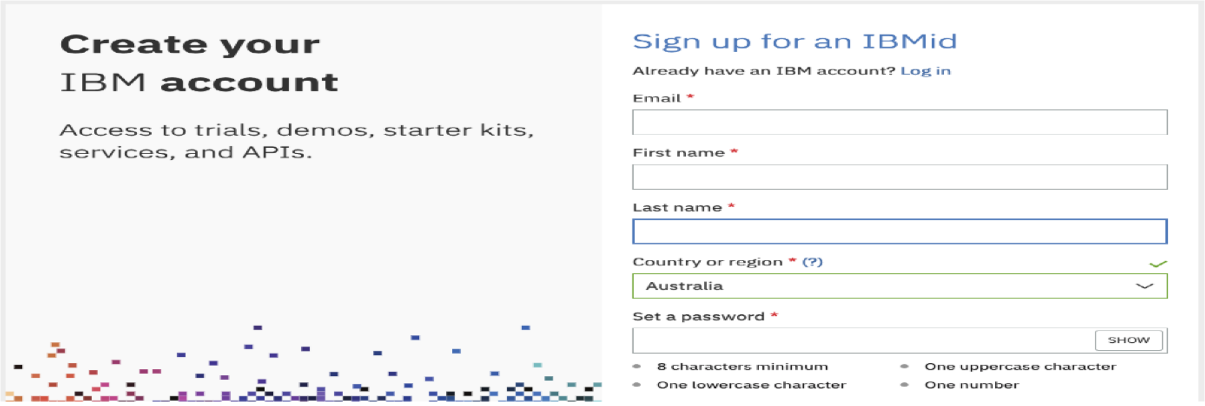
1. **Project Planning and Requirements Gathering:**
   - Define the project scope and objectives.
   - Identify specific use cases and applications for image recognition.
   - Gather requirements, including data sources, types of images, and desired outcomes.

2. Data Collection and Preparation:

- Acquire and collect a dataset of images relevant to your application.
- Clean and preprocess the data by resizing, normalizing, and augmenting the images.
- Annotate the images to label objects or features of interest.

3. IBM Cloud Setup:

- Create an IBM Cloud account if you don't have one.
- Set up the necessary cloud services, such as Watson Visual Recognition or IBM Cloud Functions for serverless execution.



Create your IBM account

Access to trials, demos, starter kits, services, and APIs.

Sign up for an IBMid

Already have an IBM account? Log in

Email *

First name *

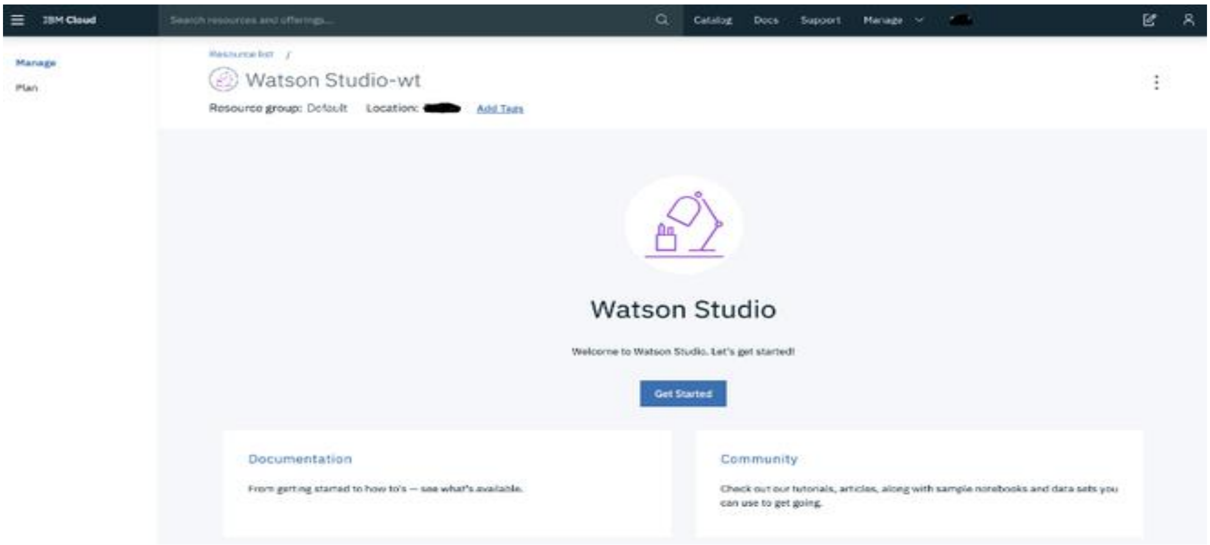
Last name *

Country or region * (?)

Australia

Set a password *

- 8 characters minimum
- One uppercase character
- One lowercase character
- One number



[redacted]

Watson Studio-wt

Resource group: Default Location: [redacted] Add Tags

Watson Studio

Welcome to Watson Studio. Let's get started!

Documentation

From getting started to how to's — see what's available.

Community

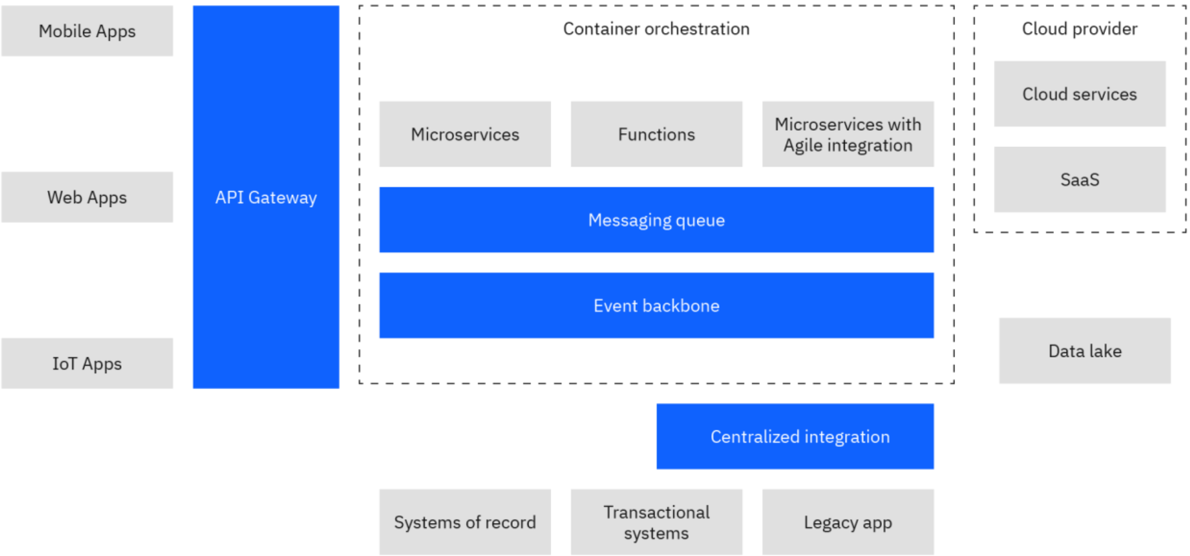
Check out our tutorials, articles, along with sample notebooks and data sets you can use to get going.

4.Model Selection and Training:

- Choose an appropriate image recognition model or framework.
- Train the model using your annotated dataset.
- Fine-tune the model to improve accuracy, if needed.
- Leverage IBM Cloud services like Watson Visual Recognition for model training and deployment.

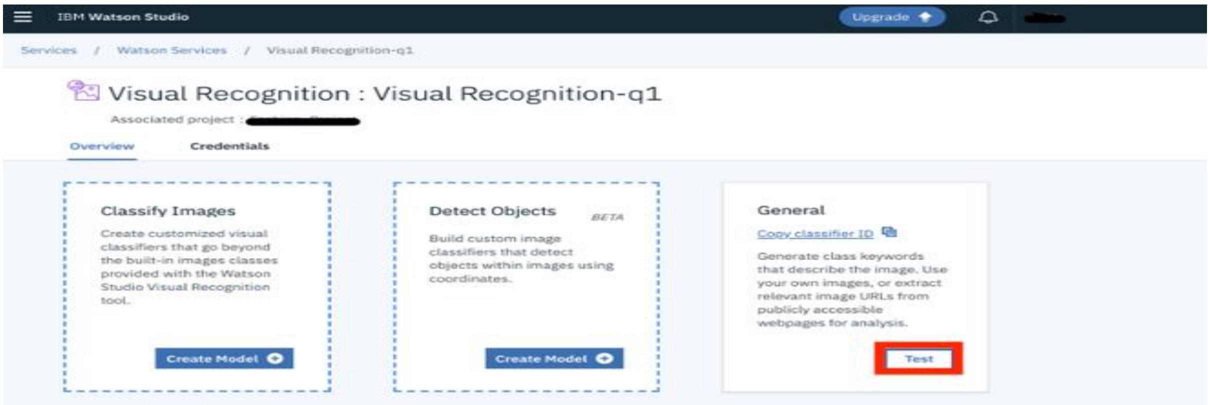
5. Integration with Applications:

- Integrate the trained model into your target application, whether it's a mobile app, website, or other software.
- Ensure the image recognition system communicates effectively with your application.



Testing and Validation:

- Test the system with a variety of real-world images to ensure accuracy and reliability.
- Collect user feedback and iteratively refine the system.



Visual Recognition : Visual Recognition-q1

Associated project : [redacted]

Classify Images

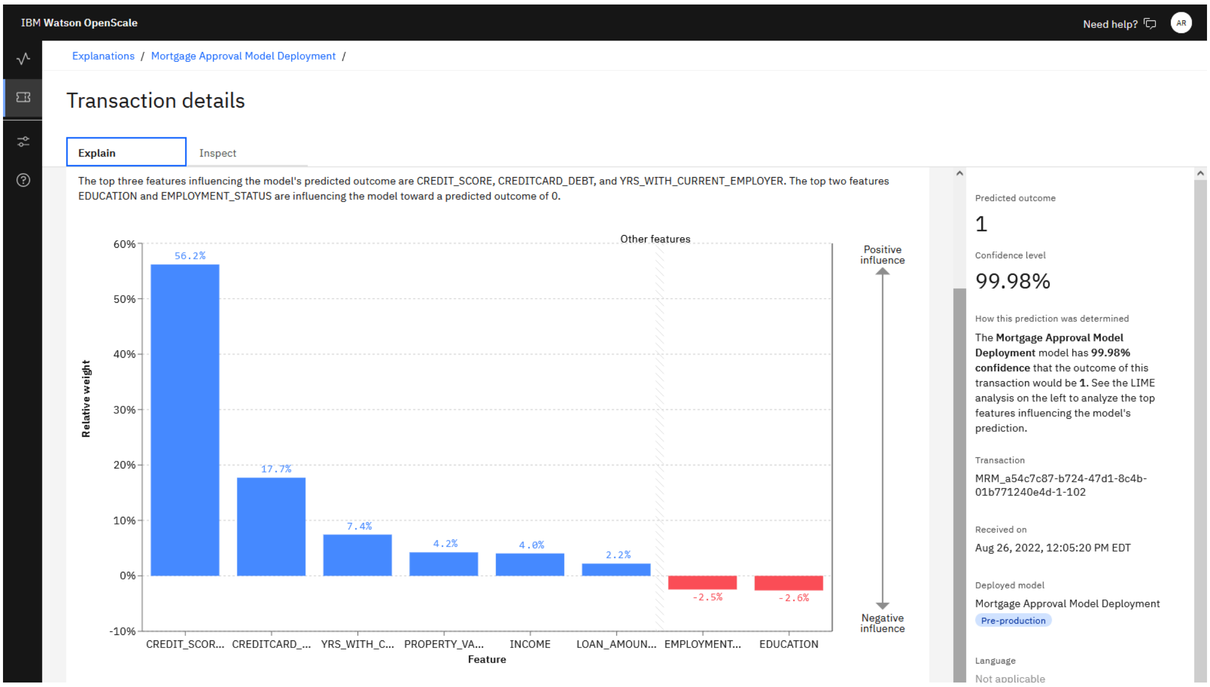
Create customized visual classifiers that go beyond the built-in images classes provided with the Watson Studio Visual Recognition tool.

Detect Objects *BETA*

Build custom image classifiers that detect objects within images using coordinates.

General

Generate class keywords that describe the image. Use your own images, or extract relevant image URLs from publicly accessible webpages for analysis.



Transaction details

The top three features influencing the model's predicted outcome are CREDIT_SCORE, CREDITCARD_DEBT, and YRS_WITH_CURRENT_EMPLOYER. The top two features EDUCATION and EMPLOYMENT_STATUS are influencing the model toward a predicted outcome of 0.

Predicted outcome
1

Confidence level
99.98%

How this prediction was determined

The **Mortgage Approval Model Deployment** model has **99.98% confidence** that the outcome of this transaction would be **1.** See the LIME analysis on the left to analyze the top features influencing the model's prediction.

Transaction
MRM_a54c7c87-b724-47d1-8c4b-01b771240e4d-1-102

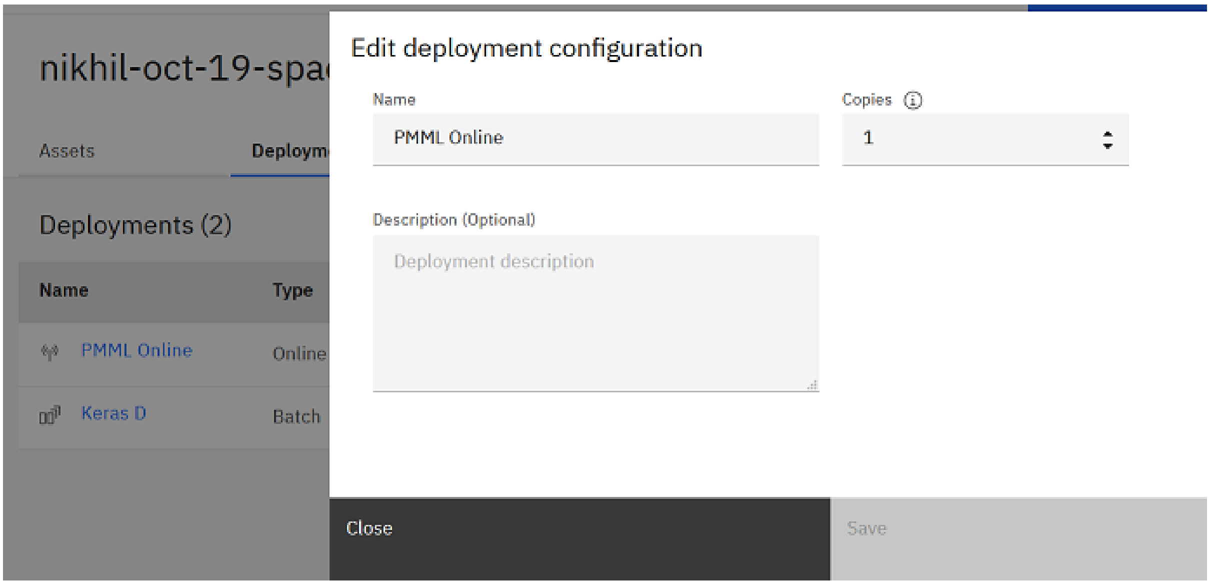
Received on
Aug 26, 2022, 12:05:20 PM EDT

Deployed model
Mortgage Approval Model Deployment

Language
Not applicable

Deployment and Scaling:

- Deploy the image recognition system on IBM Cloud infrastructure.
- Set up auto-scaling and load balancing to handle increased usage.



Deployments (2)

Name	Type
PMML Online	Online
Keras D	Batch

Edit deployment configuration

Name
PMML Online

Copies
1

Description (Optional)
Deployment description

Monitoring and Maintenance:

- Implement monitoring tools to track system performance and user interactions.
- Continuously update the model with new data to improve accuracy and adapt to changing conditions.

Security and Compliance:

- Implement security measures to protect user data and maintain compliance with relevant regulations (e.g., GDPR).
- Regularly audit and update security protocols.
- **User Training and Support:**
  - Provide user training or documentation for interacting with the image recognition system.
  - Offer customer support and troubleshoot issues as they arise.
- **Performance Optimization:**
  - Continuously optimize the system for speed and efficiency.
  - Identify and resolve bottlenecks or performance issues.
- **Feedback Loop and Iteration:**
  - Collect and analyze user feedback and usage data.
  - Use insights to make improvements and iterate on the system, including retraining the model with new data.

These development phases are typically part of the lifecycle when creating an image recognition system using IBM Cloud or any other cloud-based AI services. The specific details and requirements will vary depending on the project's complexity and objectives.

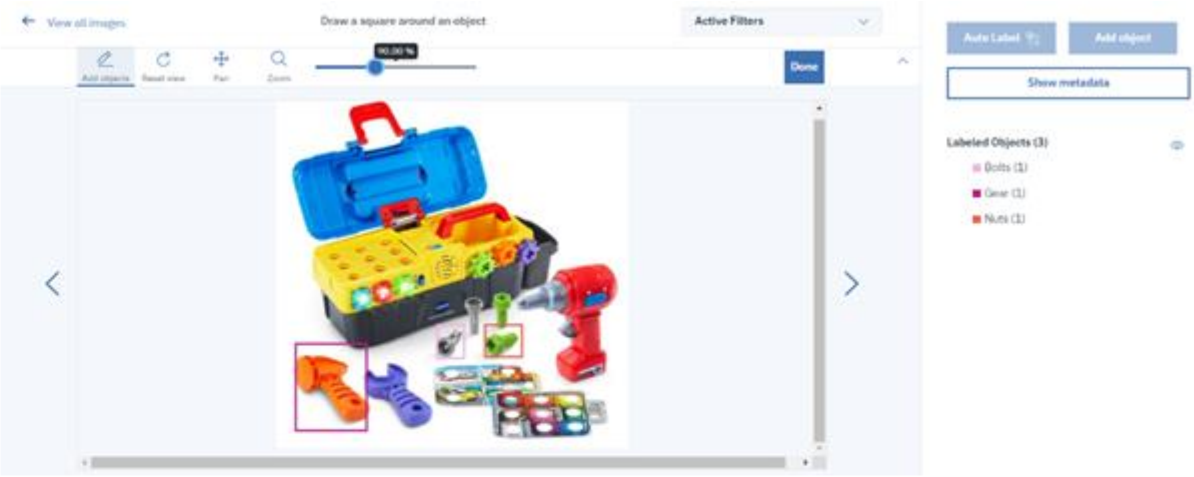
INTEGERATION OF IBM CLOUD VISUAL RECOGNITION:

IBM Watson™ Visual Recognition Custom Object Detection identifies items and their location in an image. The service detects these items based on a set of images with labelled training data that you provide.

The Custom Object Detection model is the newest feature in the Visual Recognition service, which includes classification.

Classification and object detection are similar but have different uses. In general, if you want to predict the existence of objects in an image, use classification. If you want to locate or count objects in an image, use object detection. This service identifies the location of items in the image. Like classification, the response also includes the label of each item it detected and the confidence in its identification.

Following toddler's toolbox image is a collection of different subsets of images. Now if we want to know the count of bolts or gears in this image then our conventional Image classification techniques doesn’t fit quite smart. Custom Object Detection feature allows us to train Watson by labeling *specific objects*, and training Watson to identify those objects in other images as well.



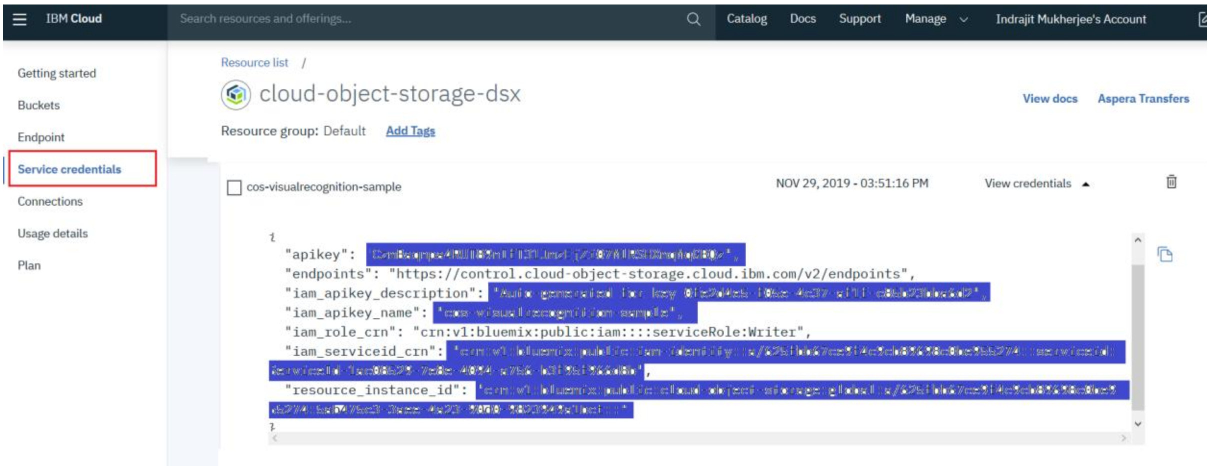
IBM Watson support V3 Visual Recognition or V4 Visual Recognition API’s to integrate Object detection model with your Client applications. This API documentation comes with ready to go code snippets of Curl, .Net, Go, Java, node.js, Python, Ruby, Swift and Unity applications.

I shall demonstrate how to call our deployed Custom object detection model using Python client using V3 Visual recognition API.

```
!pip install --upgrade "ibm-watson>=4.1.0"
!pip install ibm-cos-sdk
```

```
!pip install ibm-watson
!pip install ibm-cloud-sdk-core
```

I have store the test images in Cloud Object storage. Hence Need to authenticate my client to download the images. Following authentication details will be available in Service credential tab of your COS instance.



```
{
  "apikey": "[illegible]",
  "endpoints": "https://control.cloud-object-storage.cloud.ibm.com/v2/endpoints",
  "iam_apikey_description": "Auto generated for key [illegible]",
  "iam_apikey_name": "cos-visualrecognition-sample",
  "iam_role_crn": "crn:v1:bluemix:public:iam::::serviceRole:Writer",
  "iam_serviceid_crn": "crn:v1:bluemix:public:iam-identity::a/[illegible]::serviceid:ServiceId-[illegible]",
  "resource_instance_id": "crn:v1:bluemix:public:cloud-object-storage:global:a/[illegible]:[illegible]::"
}
```

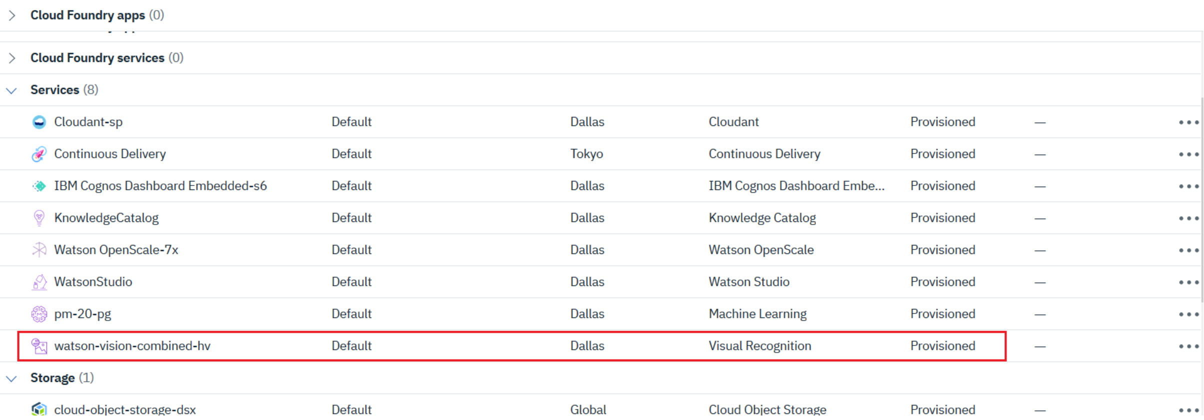
```
import ibm_boto3
from ibm_botocore.client import Config, ClientError

# Constants for IBM COS values
COS_ENDPOINT = "<Your End point>" # Current list avaiable at
https://control.cloud-object-storage.cloud.ibm.com/v2/endpoints
COS_API_KEY_ID = "<Your API Key>" # eg "W00YiRnLW4a3fTjHB-oiB-
2ySfTrFBIQQWanc--P3byk"
COS_AUTH_ENDPOINT = "https://iam.cloud.ibm.com/identity/token"
COS_RESOURCE_CRN = "<Your Resource ID>"
# eg "crn:v1:bluemix:public:cloud-object-
storage:global:a/3bf0d9003abfb5d29761c3e97696b71c:d6f04d83-6c4f-4a62-
a165-696756d63903::"

# Create resource
cos = ibm_boto3.resource("s3",
    ibm_api_key_id=COS_API_KEY_ID,
    ibm_service_instance_id=COS_RESOURCE_CRN,
    ibm_auth_endpoint=COS_AUTH_ENDPOINT,
    config=Config(signature_version="oauth"),
    endpoint_url=COS_ENDPOINT
)
```

This part is Optional, I just wanted to check how many Buckets I'm having in my COS.

```
def get_buckets():
    print("Retrieving list of buckets")
    try:
        buckets = cos.buckets.all()
        for bucket in buckets:
            print("Bucket Name: {0}".format(bucket.name))
    except ClientError as be:
        print("CLIENT ERROR: {0}\n".format(be))
    except Exception as e:
        print("Unable to retrieve list buckets: {0}".format(e))
get_buckets();
```

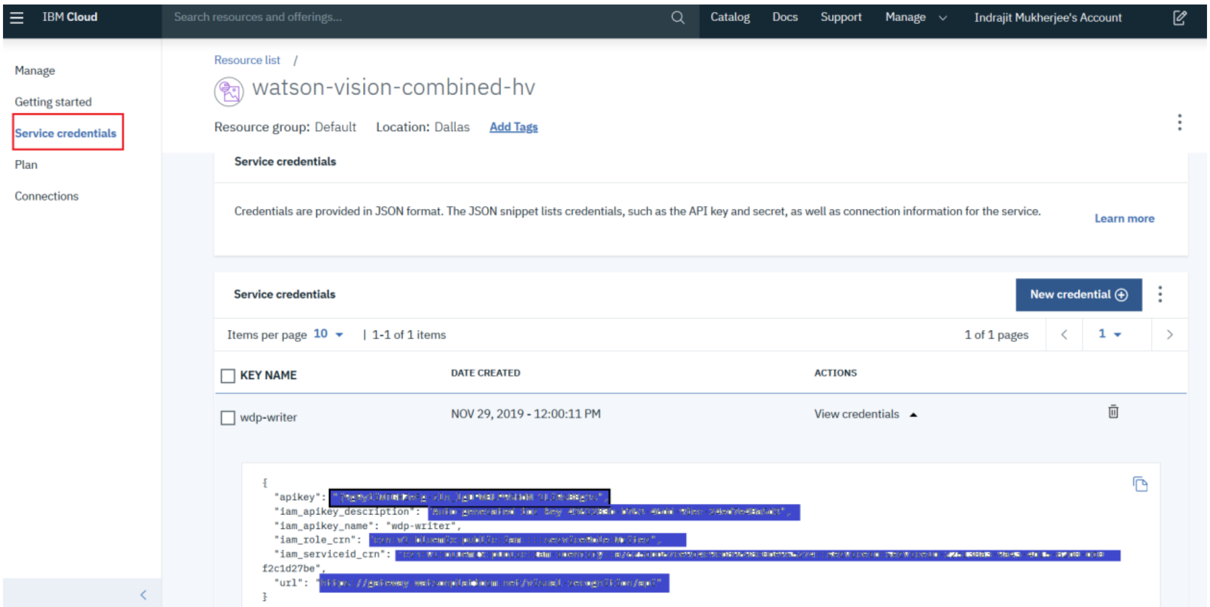
```
# downloading the sample image to pass to the Custom Object detection model
def download_file_cos(local_file_name,key,bucket_name):
    cos = ibm_boto3.client(service_name='s3',
    ibm_api_key_id=COS_API_KEY_ID,
    ibm_service_instance_id=COS_RESOURCE_CRN,
    ibm_auth_endpoint=COS_AUTH_ENDPOINT,
    config=Config(signature_version='oauth'),
    endpoint_url=COS_ENDPOINT)
    try:
        res=cos.download_file(Bucket=
bucket_name,Key=key,Filename=local_file_name)

    except Exception as e:
        print(Exception, e)
    else:
        print('File downloaded')
```

To call V3 API, first we need to authenticate our client's request by IBM's IAM authentication. Now, we need visual recognition services credentials to call our deployed model. Go, to your visual recognition service instance in cloud.



Cloud Foundry apps (0)						
Cloud Foundry services (0)						
Services (8)						
Cloudant-sp	Default	Dallas	Cloudant	Provisioned	—	•••
Continuous Delivery	Default	Tokyo	Continuous Delivery	Provisioned	—	•••
IBM Cognos Dashboard Embedded-s6	Default	Dallas	IBM Cognos Dashboard Embe...	Provisioned	—	•••
KnowledgeCatalog	Default	Dallas	Knowledge Catalog	Provisioned	—	•••
Watson OpenScale-7x	Default	Dallas	Watson OpenScale	Provisioned	—	•••
WatsonStudio	Default	Dallas	Watson Studio	Provisioned	—	•••
pm-20-pg	Default	Dallas	Machine Learning	Provisioned	—	•••
watson-vision-combined-hv	Default	Dallas	Visual Recognition	Provisioned	—	•••
Storage (1)						
cloud-object-storage-dsx	Default	Global	Cloud Object Storage	Provisioned	—	•••

On the left hand corner panel, click on the service credential. Now, if not created earlier, you need to create a new credentials, what your client application will use to connect the deployed service.



watson-vision-combined-hv

Resource group: Default Location: Dallas Add Tags

Service credentials

Credentials are provided in JSON format. The JSON snippet lists credentials, such as the API key and secret, as well as connection information for the service. Learn more

Service credentials New credential ⊕

Items per page 10 | 1-1 of 1 items 1 of 1 pages 1

☐ KEY NAME	DATE CREATED	ACTIONS
☐ wdp-writer	NOV 29, 2019 - 12:00:11 PM	View credentials ▲

```
{
  "apikey": [redacted]
  "iam_apikey_description": [redacted]
  "iam_apikey_name": "wdp-writer",
  "iam_role_crn": [redacted]
  "iam_serviceid_crn": [redacted]
f2c1d27be",
  "url": [redacted]
}
```

```
from ibm_watson import VisualRecognitionV3, ApiException
from ibm_cloud_sdk_core.authenticators import IAMAuthenticator
import json
import os

def call_Image_Recognition_API(bucket_name):
    print("Retrieving bucket contents from: {0}".format(bucket_name))
    authenticator =IAMAuthenticator('<your API Key>')
    visual_recognition = VisualRecognitionV3(
                version='2018-03-19',
                authenticator=authenticator
    )

    try:
        files = cos.Bucket(bucket_name).objects.all()
        for file in files:
            print("Item: {0} ({1} bytes).".format(file.key, file.size))
            download_file_cos(file.key,file.key,bucket_name)
            with open(file.key,'rb') as images_file:
                medical_results = visual_recognition.classify(
                    images_file=images_file,
                    threshold='0.76',
                    classifier_ids='Your Classifier ID').get_result()
            print(json.dumps(medical_results, indent=2))
    except ClientError as be:
        print("CLIENT ERROR: {0}\n".format(be))
    except ApiException as ex:
        print("Method failed with status message: {0}".format(ex.message))
    except Exception as e:

        print("Unable to retrieve bucket contents: {0}".format(e))
```

I am passing my bucket name, where I stored my image files.

```
call_Image_Recognition_API('your bucket name');
```

Sample Json output.

```
[
 {
  "classifier_id": "insurance_perm_v2_1096877345",
  "name": "insurance_perm_v2",
  "classes": [
   {
    "class": "broken_windshield",
    "score": 0
   },
   {
    "class": "flat_tire",
    "score": 0.016
   },
   {
    "class": "motorcycle_accident",
    "score": 0.902
   },
   {
    "class": "vandalism",
    "score": 0.001
   }
  ]
 }
]
```

USER INTERFACE:

Design a simple web service interface where users can upload images and view the AI generated captions.

Creating a simple web interface for users to upload images and view AI-generated captions can be achieved using HTML, CSS, and JavaScript. Below is a basic example of how you can design such an interface:

HTML (index.html):

```
<!DOCTYPE html>
<html>
<head>
    <title>Image Caption Generator</title>
    <link rel="stylesheet" type="text/css" href="styles.css">
</head>
<body>
    <h1>Image Caption Generator</h1>
    <input type="file" id="imageInput" accept="image/*" />
    <button id="generateCaption">Generate Caption</button>
    <img id="uploadedImage" src="" alt="Uploaded Image" />
    <div id="captionResult">
        <h2>Generated Caption:</h2>
        <p id="captionText"></p>
    </div>

    <script src="script.js"></script>

</body>

</html>
```

CSS (styles.css):

```
body {

    text-align: center;

    font-family: Arial, sans-serif;

}



h1 {

    margin-top: 20px;

}

input[type="file"] {

    display: none;

}

button {

    padding: 10px 20px;

    background-color: #007BFF;
    color: white;
    border: none;
    cursor: pointer;
}
img {
    max-width: 100%;
    max-height: 400px;
    display: none;
}
#captionResult {
    margin-top: 20px;
    display: none;
}
#captionText {
    font-style: italic;
}
```

JavaScript (script.js):

```
document.getElementById("generateCaption").addEventListener("click", function() {
    const imageInput = document.getElementById("imageInput");
    const uploadedImage = document.getElementById("uploadedImage");
    const captionResult = document.getElementById("captionResult");
    const captionText = document.getElementById("captionText");
  // Ensure an image is selected
    if (imageInput.files.length === 0) {
        alert("Please select an image to generate a caption.");
        return;
    }
const file = imageInput.files[0];
    const reader = new FileReader();
 reader.onload = function(event) {
        uploadedImage.src = event.target.result;
        uploadedImage.style.display = "block";

        // You can now send the uploaded image to your AI service for caption generation.
        // Replace the following line with your actual AI integration code.
        const generatedCaption = "A beautiful image.";
 captionText.textContent = generatedCaption;
        captionResult.style.display = "block";
    };
 reader.readAsDataURL(file);
```

```
});

// You can also add functionality to clear the uploaded image and caption if needed.
```

This code provides a simple web interface for users to upload an image, generate a caption (currently hardcoded), and display the result. It uses HTML for the structure, CSS for styling, and JavaScript for interaction. To make this code functional with AI-generated captions, you'll need to integrate your AI service's capabilities where indicated in the JavaScript code. This typically involves making an API request to the AI service for caption generation based on the uploaded image.

AI GENERATED CAPTIONS :

AI-generated captions can enhance user engagement and storytelling in image recognition using IBM Cloud in several ways:

1. **Contextual Understanding:** AI-generated captions provide context to the images, helping users understand what they are seeing. This contextual information enhances the storytelling by explaining the significance of the image, its contents, and any relevant details.
2. **Accessibility:** Captions make image content accessible to users with visual impairments, ensuring inclusivity. They can use screen readers to understand the content, enhancing the user engagement by extending the reach to a wider audience.
3. **Narrative Enrichment:** AI-generated captions can be used to weave a narrative around a collection of images, turning them into a cohesive story. They can provide descriptions, emotions, and thematic connections, making the storytelling more engaging and immersive.
4. **Multilingual Support:** With AI, captions can be automatically translated into different languages. This enables users from diverse linguistic backgrounds to engage with and understand the content, broadening your audience reach.
5. **Searchability:** Captioned images are more searchable, both within your application or website and on search engines. This can enhance user engagement by making it easier for users to find and discover relevant image content.

6. **User Interaction:** Captions can invite user interaction and engagement. For instance, you can encourage users to add their own captions, comments, or stories related to the images, creating a sense of community and involvement.
7. **Education and Training:** In educational contexts, AI-generated captions can provide explanations, additional information, and quiz questions related to images. This not only enhances user engagement but also improves the educational value of the content.
8. **Visual Content Marketing:** In marketing, AI-generated captions can be used to tell a brand's story through images. They can add context and emotional appeal to visual content, engaging users and reinforcing brand messaging.
9. **Content Personalization:** AI can analyze user preferences and behaviors to generate captions tailored to individual users, making the content more relevant and engaging for each user.
10. **Emotion and Sentiment Analysis:** AI-generated captions can incorporate sentiment analysis, expressing the emotions or mood conveyed by the images. This adds an emotional dimension to storytelling and can resonate more deeply with users.

By incorporating AI-generated captions into your image recognition system on IBM Cloud, you can create a more interactive, informative, and engaging user experience. Whether you're telling a story through a collection of images or providing additional context for individual images, these captions can enrich the narrative, appeal to a broader audience, and make your content more accessible and discoverable.

CONCLUSION :

In this solution, we successtuly developed an image recognition system using IBM Cloud Visual Recogniton By folowing the problem definiltian, design thiniking. development, and documentation phases, we designed and implemented a user-fnendy web interface that allows users to upload images and recelve Al-generated captions. The system accurately classifes and describes the image contents, enabling users to craft engaging visual stories and connect with their audience through captivating visuals and compeling narratives.