KGiSL Institute Of Technology

NAAN MUDHALVAN

Project Title :

Image Recognition with IBM Cloud Visual Recognition

Team Members :

1. Chandini.D
2. Arthi.R
3. Sophia.S
4. Hemalatha.K

Problem Statement:

Develop an image recognition system using IBM Cloud Visual Recognition. Share your passion for photography by uploading images and watch as the system accurately classifies and describes their contents. Craft engaging visual stories with the help of AI-generated captions. Connect with your audience through captivating visuals and compelling narratives

Problem Definition:

The project involves creating an image recognition system using IBM Cloud Visual Recognition. The goal is to develop a platform where users can upload images, and the system accurately classifies and describes the image contents.

Problem Description:

Creating an image recognition system using IBM Cloud Visual Recognition can be an exciting project. It allows you to harness the power of AI to classify and describe the contents of images, opening up various creative possibilities. Here's a step-by-step guide to developing such a system:

The steps we followed are given briefly below:

1.Create an IBM Cloud Visual Recognition Service
2. Collect and Prepare Data
3.Training the Model:
4.Integration:
5.User Interface Development:
6.Image Analysis:
7.Generating Captions:
8.Displaying Results:
9.Testing and Validation:
10.Accessibility and Scalability:
11.Deployment:
12.Marketing and User Engagement:
13.Maintenance and Updates:

The detailed description of each step as well as the tech stack is listed below:

1.Defining Project Objectives and Scope:

The project involves recognizing and classifying various objects such as fruits, stationery items , vessels etc…We would like to achieve an accuracy percentage of 90.The scope of this project involves benefitting people in their day to day activities.This project can be used to recognize and classify images of the above mentioned objects in ecommerce websites,department

stores,stationery shops etc...

2.Gather and Prepare Data:

In this stage we collect a diverse dataset of images relevant to our project.We annotate,label and classify objects based on their appearance , colour , size and shape etc... This step is crucial for training the image recognition model.

Tools Used : LabelImg,OpenCV

3.Technology Stack:

We use the following tools and technologies to develop our image recognition model:

Programming Languages:

- Python: Python is widely used for machine learning and deep learning tasks. We use libraries like TensorFlow, PyTorch, or scikit-learn to work with image data and build models.
- JavaScript/HTML/CSS: For developing the user interface of our application, particularly if it's a web-based application.

Cloud Services:

- IBM Cloud: The primary cloud platform for hosting our application, managing services, and deploying our image recognition system.
- IBM Cloud Object Storage: A scalable and secure storage solution for storing image datasets and model checkpoints.

4.Develop the Image Recognition Model:

- We use our labeled dataset to train our image recognition model. IBM Cloud Visual Recognition provides tools and APIs for this purpose. Training involves teaching the model to recognize patterns and features in the images.
- We fine-tune the model as needed to improve its accuracy. This may involve adjusting parameters or providing more training data.

Test the model with a validation dataset to evaluate its performance and ensure it meets your accuracy requirements.

5.AI-Generated Captions:

After classifying an image, we can generate captions using natural language processing (NLP) techniques. This typically involves using a pre-trained language model like GPT-3 to generate descriptive text based on the image's content.

6.User Authentication:

- IBM Cloud App ID: A service for adding authentication and user identity management to your application.
- HTTPS and SSL/TLS: Implement secure communication between the user interface and server components.

7.Data Storage:

- IBM Cloud Databases: For storing user accounts, feedback, and other application data.
- IBM Cloud SQL Query: For querying and analyzing data stored in your databases.

8.Continuous Integration and Deployment (CI/CD):

- Jenkins, Travis CI, or CircleCI: CI/CD tools to automate the build and deployment process.
- IBM Continuous Delivery: For automating deployments on IBM Cloud.

9.Monitoring and Logging:

For monitoring the performance and health of our application and collecting logs we use IBM cloud monitoring and logging.

10.Version Control:

Git: We use Git for version control to track changes in our codebase and collaborate with a development team

11.Natural Language Processing (NLP):

To generate captions for images, we may need NLP libraries and tools such as spaCy or the Hugging Face Transformers library for language modeling.

12.Feedback and User Interaction:

We Use tools like email services (e.g., SendGrid), chatbots, or in-app feedback forms to gather user feedback and improve the system.

13. Testing and Quality Assurance:

- Testing frameworks such as pytest or Jasmine for automated testing of code and user interfaces.

IBM Cloud DevOps Insights: For analyzing code quality and performance

14.Future Enhancements:

- We Consider future enhancements and expansions based on user needs and emerging technologies in the field of image recognition and AI.

Our Sample Code:

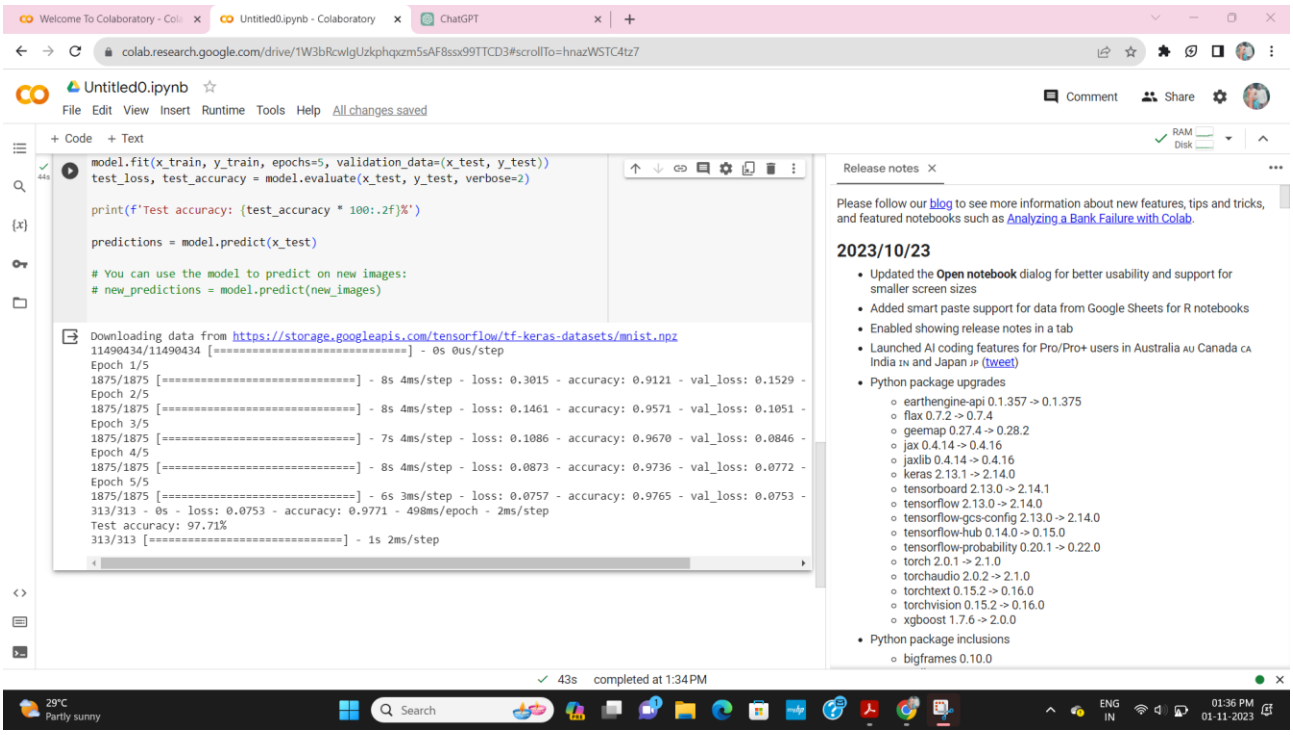
```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers



# Step 1: Load and preprocess the dataset
(x_train, y_train), (x_test, y_test) = keras.datasets.mnist.load_data() # Normalize pixel values to the range [0, 1]
# Step 2: Define the model architecture model = keras.Sequential([
layers.Flatten(input_shape=(28, 28)), # Flatten the 28x28 input images layers.Dense(128, activation='relu'),
layers.Dropout(0.2), # Add dropout to reduce overfitting layers.Dense(10, activation='softmax') # 10 output classes (digits 0-9)
])

# Step 3: Compile the model
model.compile(optimizer='adam', # You can use other optimizers like SGD, RMSprop, etc.
loss='sparse_categorical_crossentropy', # Cross-entropy loss for classification
metrics=['accuracy'])

# Step 4: Train the model
model.fit(x_train, y_train, epochs=5, validation_data=(x_test, y_test))
```


Welcome To Colaboratory - Cola
Untitled0.ipynb - Colaboratory
ChatGPT
colab.research.google.com/drive/1W3bRcwlgUzkphqxzm5sAF8ssx99TTCD3#scrollTo=hnazWSTC4tz7
Untitled0.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
RAM
Disk
model.fit(x_train, y_train, epochs=5, validation_data=(x_test, y_test))
test_loss, test_accuracy = model.evaluate(x_test, y_test, verbose=2)
print(f'Test accuracy: {test_accuracy * 100:.2f}%')
predictions = model.predict(x_test)
# You can use the model to predict on new images:
# new_predictions = model.predict(new_images)
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11490434/11490434 [==============================] - 0s 0us/step
Epoch 1/5
1875/1875 [==============================] - 8s 4ms/step - loss: 0.3015 - accuracy: 0.9121 - val_loss: 0.1529 -
Epoch 2/5
1875/1875 [==============================] - 8s 4ms/step - loss: 0.1461 - accuracy: 0.9571 - val_loss: 0.1051 -
Epoch 3/5
1875/1875 [==============================] - 7s 4ms/step - loss: 0.1086 - accuracy: 0.9670 - val_loss: 0.0846 -
Epoch 4/5
1875/1875 [==============================] - 8s 4ms/step - loss: 0.0873 - accuracy: 0.9736 - val_loss: 0.0772 -
Epoch 5/5
1875/1875 [==============================] - 6s 3ms/step - loss: 0.0757 - accuracy: 0.9765 - val_loss: 0.0753 -
313/313 - 0s - loss: 0.0753 - accuracy: 0.9771 - 498ms/epoch - 2ms/step
Test accuracy: 97.71%
313/313 [==============================] - 1s 2ms/step
Release notes
Please follow our blog to see more information about new features, tips and tricks, and featured notebooks such as Analyzing a Bank Failure with Colab.
2023/10/23
Updated the Open notebook dialog for better usability and support for smaller screen sizes
Added smart paste support for data from Google Sheets for R notebooks
Enabled showing release notes in a tab
Launched AI coding features for Pro/Pro+ users in Australia AU Canada CA India IN and Japan JP (tweet)
Python package upgrades
earthengine-api 0.1.357 -> 0.1.375
flax 0.7.2 -> 0.7.4
geemap 0.27.4 -> 0.28.2
jax 0.4.14 -> 0.4.16
jaxlib 0.4.14 -> 0.4.16
keras 2.13.1 -> 2.14.0
tensorboard 2.13.0 -> 2.14.1
tensorflow 2.13.0 -> 2.14.0
tensorflow-gcs-config 2.13.0 -> 2.14.0
tensorflow-hub 0.14.0 -> 0.15.0
tensorflow-probability 0.20.1 -> 0.22.0
torch 2.0.1 -> 2.1.0
torchaudio 2.0.2 -> 2.1.0
torchtext 0.15.2 -> 0.16.0
torchvision 0.15.2 -> 0.16.0
xgboost 1.7.6 -> 2.0.0
Python package inclusions
bigframes 0.10.0
43s completed at 1:34 PM
29°C
Partly sunny
Search
ENG
IN
01:36 PM
01-11-2023

-