Intermediate progress report

Title: Image Forgery Detection using Deep Learning

Objective:Using a convolutional neural network, we can distinguish between the authentic image and the fake image.

Team members:

1. Sreelekha Indarapu (Student ID: 16322036) Section-1 (10 Am Class)
2. Deepak Kumar Chada (Student ID: 16338496) Section-1 (10 Am Class)
3. Sasi Kiran Gandepalli(Student ID: 16336380) Section-2 (1 pm Class)

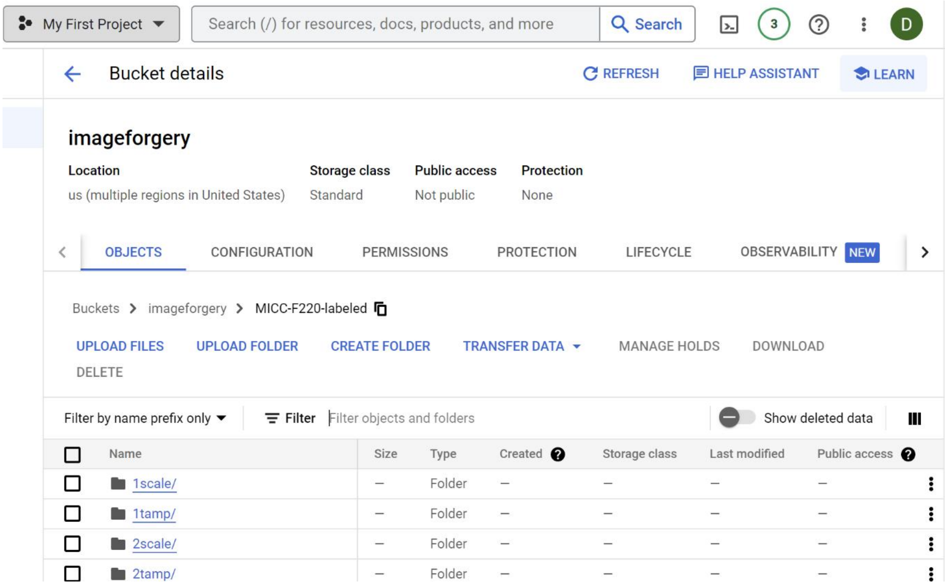
Roles and Responsibilities:

- Sreelekha and Deepak are responsible for designing and implementing the user desgined code on the google colab. They will use Python to upload images and detect the duplicate image which is stored in the GCP Bucket.

- Sasi Kiran and Sreelekha are responsible for storing the image datasets. They will use Virtual Machine and Google Collab to train and test the dataset. They will also set up the GCP infrastructure using GCP Bucket and GCP Server to deploy and manage the platform.

Bucket Created and Dataset File : -

- **Created image forgery bucket and uploaded the dataset folder inside this bucket:**



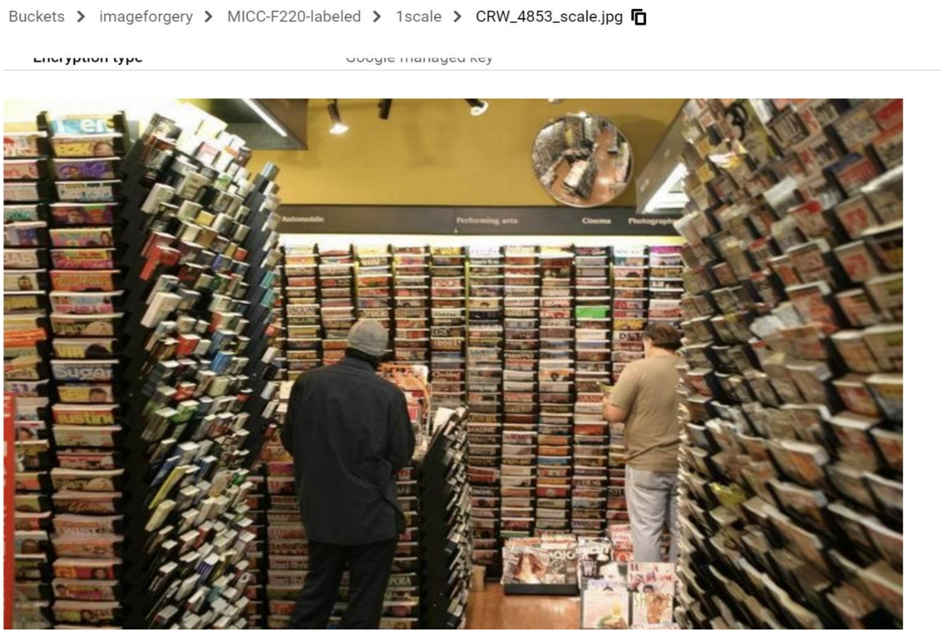
imageforgery

Location	Storage class	Public access	Protection
us (multiple regions in United States)	Standard	Not public	None

Buckets > imageforgery > MICC-F220-labeled

□	Name	Size	Type	Created	Storage class	Last modified	Public access
□	1scale/	–	Folder	–	–	–	–
□	1tamp/	–	Folder	–	–	–	–
□	2scale/	–	Folder	–	–	–	–
□	2tamp/	–	Folder	–	–	–	–

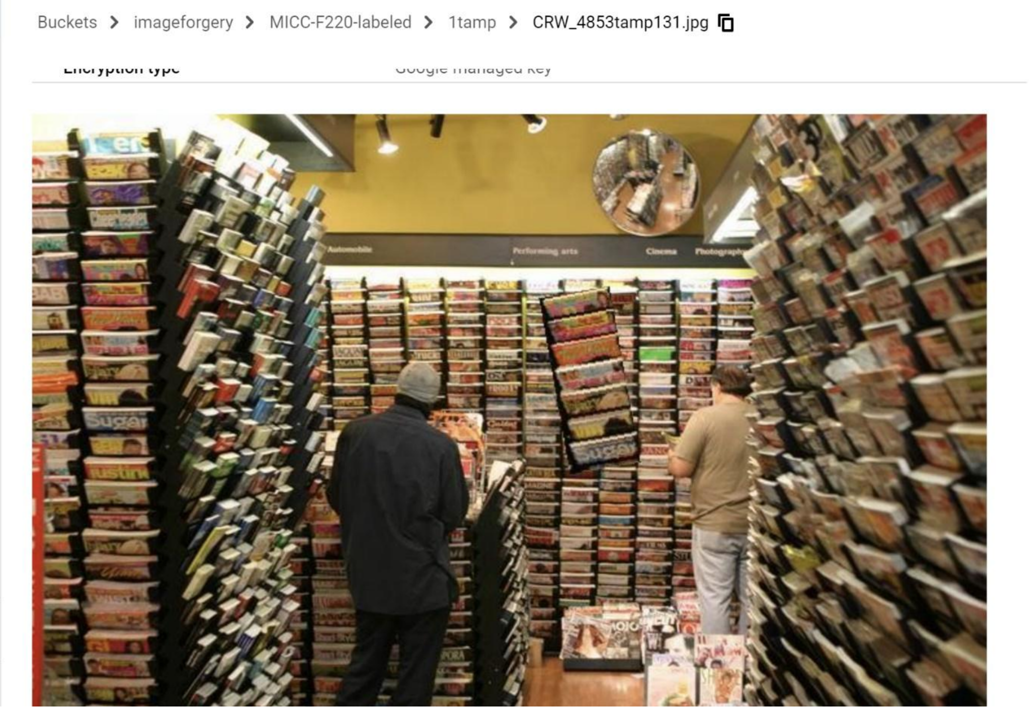
➢ Scale image Display inside bucket: -



Buckets > imageforgery > MICC-F220-labeled > 1scale > CRW_4853_scale.jpg

Encryption type Google managed key

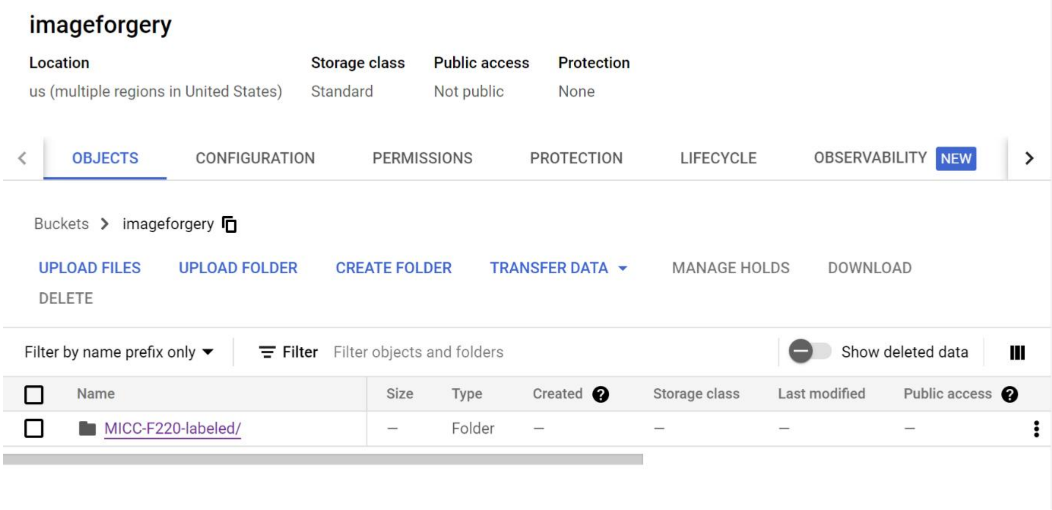
➢ Tamp image display inside bucket: -



Buckets > imageforgery > MICC-F220-labeled > 1tamp > CRW_4853tamp131.jpg

Encryption type Google managed key

- **Upload files, upload Folders, create Folder, Transfer data**



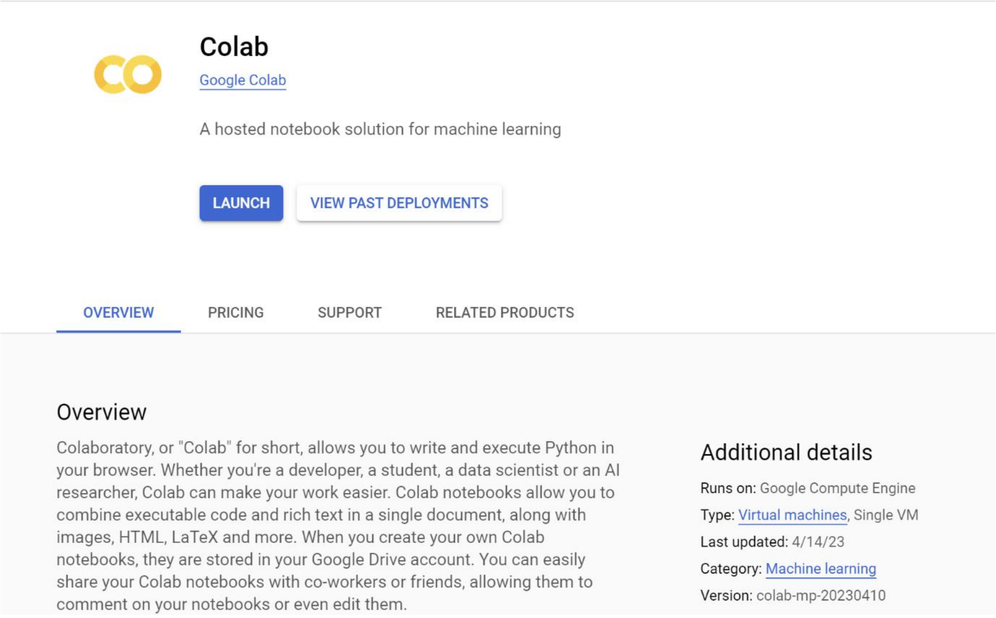
imageforgery

Location	Storage class	Public access	Protection
us (multiple regions in United States)	Standard	Not public	None

Buckets > imageforgery

□	Name	Size	Type	Created	Storage class	Last modified	Public access
□	MICC-F220-labeled/	—	Folder	—	—	—	—

- **Working with Google Cloud and Executing code Inside Collab: -**



Colab

Google Colab

A hosted notebook solution for machine learning

Overview

Colaboratory, or "Colab" for short, allows you to write and execute Python in your browser. Whether you're a developer, a student, a data scientist or an AI researcher, Colab can make your work easier. Colab notebooks allow you to combine executable code and rich text in a single document, along with images, HTML, LaTeX and more. When you create your own Colab notebooks, they are stored in your Google Drive account. You can easily share your Colab notebooks with co-workers or friends, allowing them to comment on your notebooks or even edit them.

Additional details

Runs on: Google Compute Engine

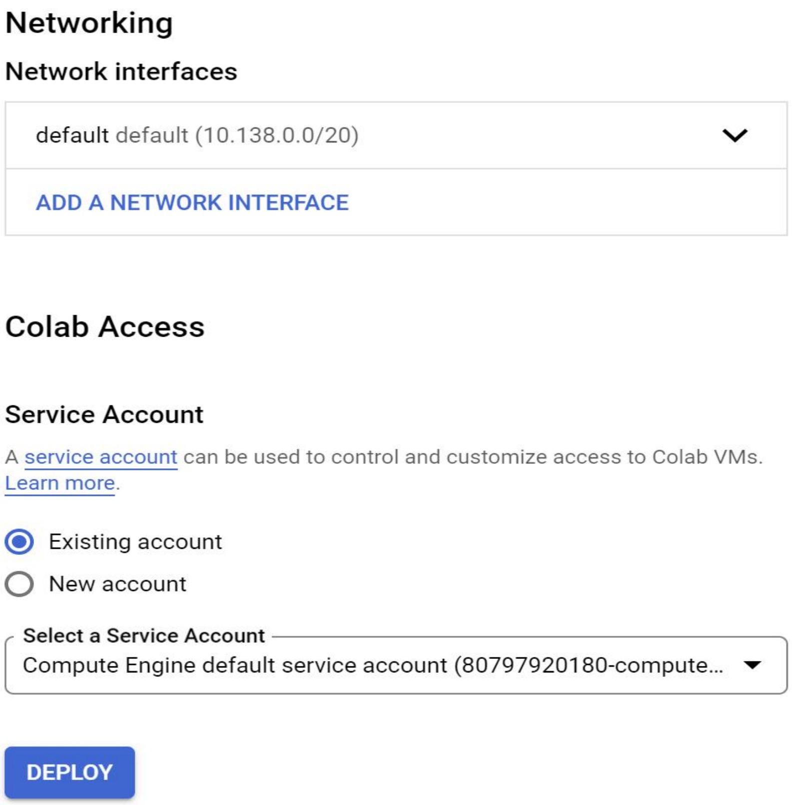
Type: Virtual machines, Single VM

Last updated: 4/14/23

Category: Machine learning

Version: colab-mp-20230410

➢ Deploying Code inside Google Cloud Server: -



Networking

Network interfaces

default default (10.138.0.0/20)

ADD A NETWORK INTERFACE

Colab Access

Service Account

A service account can be used to control and customize access to Colab VMs. Learn more.

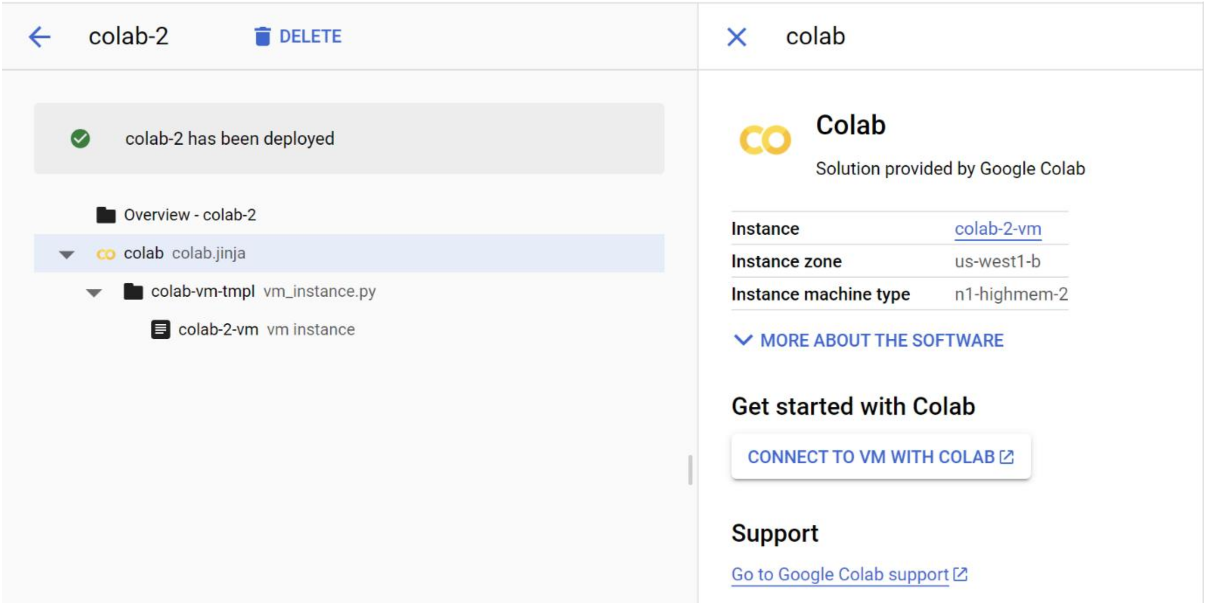
● Existing account

○ New account

Select a Service Account: Compute Engine default service account (80797920180-compute...

DEPLOY

➢ Deployment Success Page: -



colab-2 DELETE

colab-2 has been deployed

Overview - colab-2
colab colab.jinja
colab-vm-tmpl vm_instance.py
colab-2-vm vm instance

colab

Colab

Solution provided by Google Colab

Instance	colab-2-vm
Instance zone	us-west1-b
Instance machine type	n1-highmem-2

MORE ABOUT THE SOFTWARE

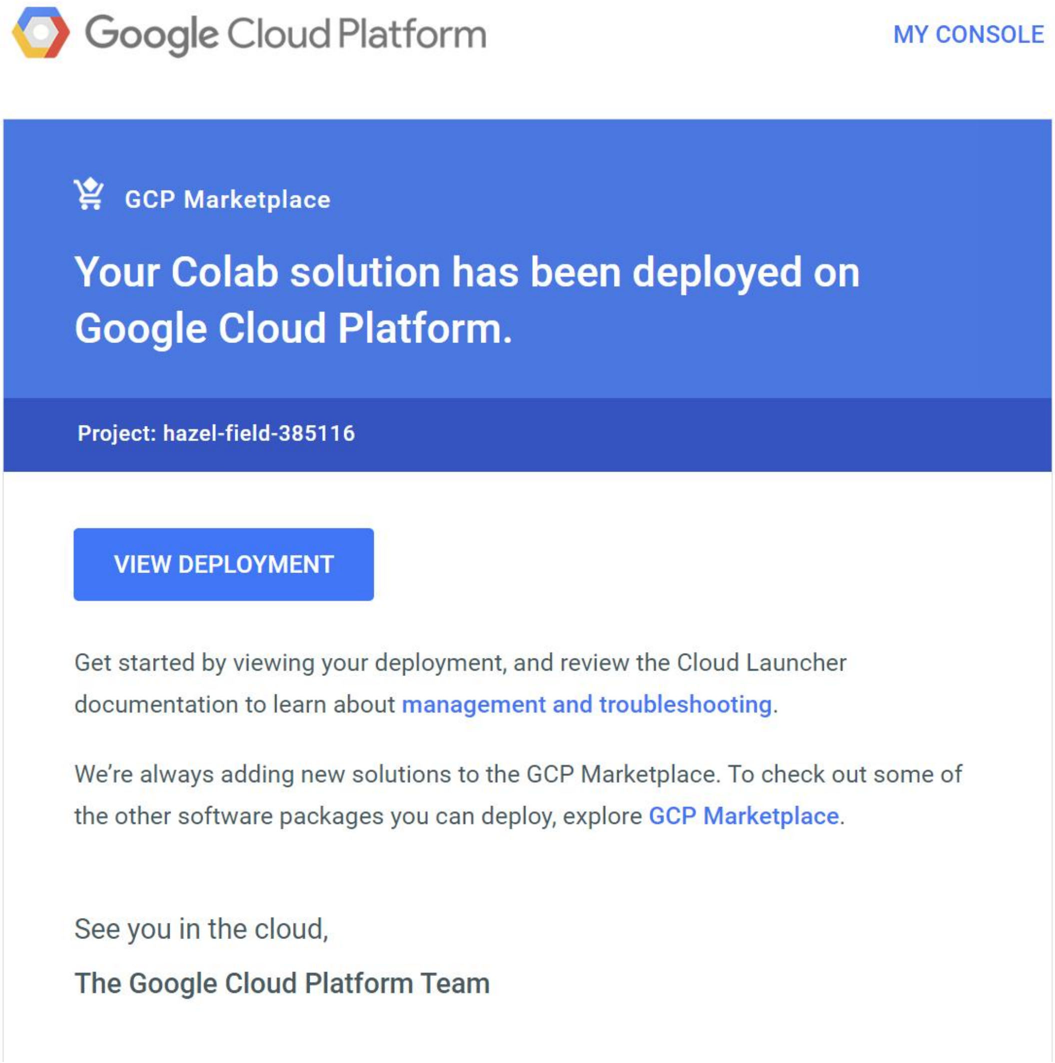
Get started with Colab

CONNECT TO VM WITH COLAB

Support

Go to Google Colab support

- **Solution Deployment confirmation mail: -**



Google Cloud Platform

MY CONSOLE

GCP Marketplace

Your Colab solution has been deployed on Google Cloud Platform.

Project: hazel-field-385116

VIEW DEPLOYMENT

Get started by viewing your deployment, and review the Cloud Launcher documentation to learn about management and troubleshooting.

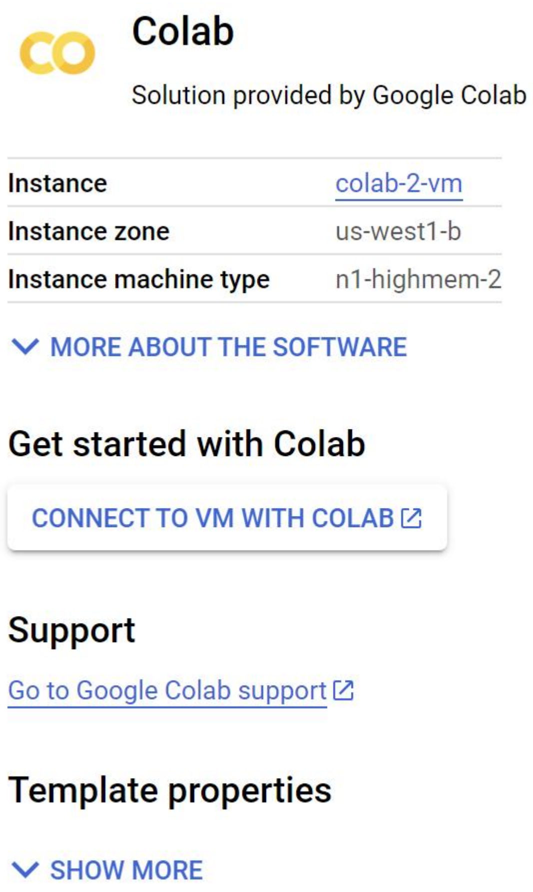
We're always adding new solutions to the GCP Marketplace. To check out some of the other software packages you can deploy, explore GCP Marketplace.

See you in the cloud,

The Google Cloud Platform Team

Connecting to Virtual Machine with Collab: -

- **Instance and Zone is created**



Colab

Solution provided by Google Colab

Instance	colab-2-vm
Instance zone	us-west1-b
Instance machine type	n1-highmem-2

Get started with Colab

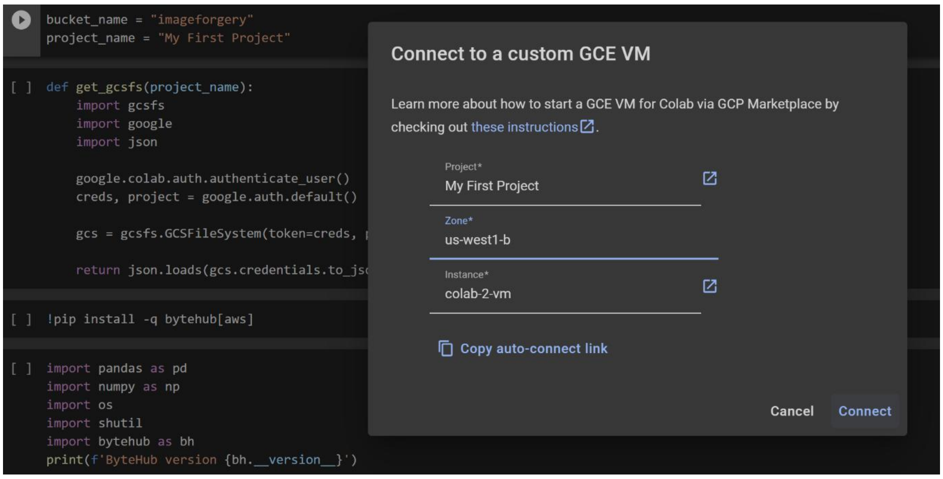
Support

Go to Google Colab support

Template properties

Connect to Custom GCE VM: -

- **While Connecting GCE VM we have to configure instance, zone, Project Name.**



```
bucket_name = "imageforgery"
project_name = "My First Project"

def get_gcsfs(project_name):
    import gcsfs
    import google
    import json

    google.colab.auth.authenticate_user()
    creds, project = google.auth.default()

    gcs = gcsfs.GCSFileSystem(token=creds, p

    return json.loads(gcs.credentials.to_js

!pip install -q bytehub[aws]

import pandas as pd
import numpy as np
import os
import shutil
import bytehub as bh
print(f'ByteHub version {bh.__version__}')
```

Connect to a custom GCE VM

Learn more about how to start a GCE VM for Colab via GCP Marketplace by checking out these instructions.

Project*: My First Project

Zone*: us-west1-b

Instance*: colab-2-vm

After connecting to VM GCE we need to run the code

```
# -*- coding: utf-8 -*-
"""imageTampering.ipynb

Automatically generated by Colaboratory.

Original file is located at

https://colab.research.google.com/drive/1ouMwyetqxovNQk6izOUvENa6qhTBONkb
"""

bucket_name = "imageforgery"
project_name = "My First Project"

def get_gcsfs(project_name):
    import gcsfs
    import google
    import json
    google.colab.auth.authenticate_user()
    creds, project = google.auth.default()

    gcs = gcsfs.GCSFileSystem(token=creds, project=project_name)

    return json.loads(gcs.credentials.to_json())

!pip install -q bytehub[aws]

import pandas as pd
import numpy as np
import os
import shutil
import bytehub as bh
print(f'ByteHub version {bh.__version__}')

fs = bh.FeatureStore()
fs

import torch
import torch.nn as nn
import torch.optim as optim
from torch.optim import lr_scheduler
import numpy as np
import torchvision
from torchvision import datasets, models, transforms
import matplotlib.pyplot as plt
import time
import os
import copy
from PIL import Image
from joblib import dump, load

plt.ion()   # interactive mode

"""# Load Dataset

x = dataitems

y = 1 | Tampered

y = 0 | Not Tampered
"""

transform = transforms.Compose([
        transforms.Resize(224),
        transforms.CenterCrop(224),
        transforms.ToTensor(),
        transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])
    ])

model_conv = torchvision.models.alexnet(pretrained=True)
classifier = list(model_conv.classifier.children())
model_conv.classifier = nn.Sequential(*classifier[:-1])
```

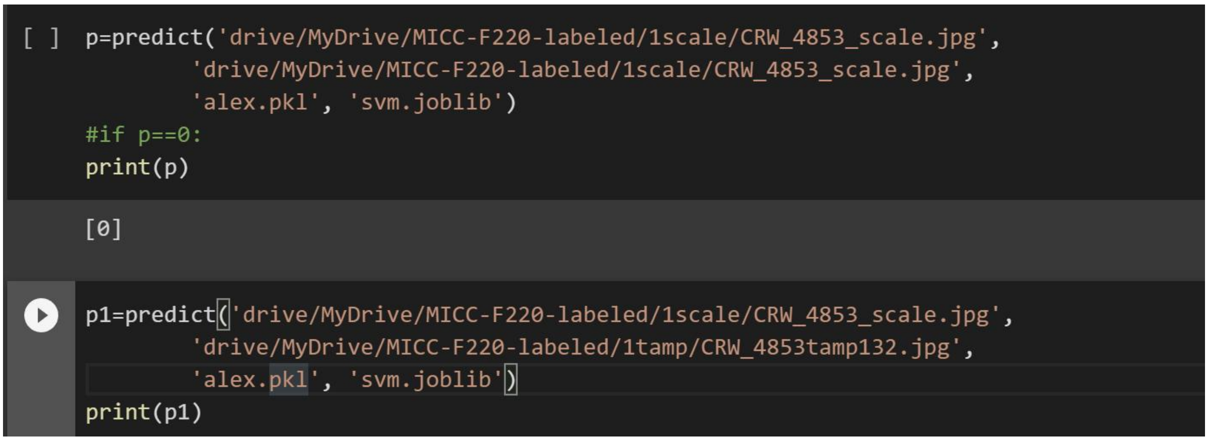
```
for param in model_conv.parameters():
    param.requires_grad = False


from google.colab import drive
drive.mount('/content/drive')


x = []
y = []
model_conv.eval()
for i in range(1,2):


    scales = None
    for scale_img in os.listdir(f'drive/MyDrive/MICC-F220-labeled/{i}scale'):
        img = Image.open(f'drive/MyDrive/MICC-F220-labeled/{i}scale/{scale_img}')
        img_tensor = transform(img)
        img_tensor.unsqueeze_(0)
        scale_ftrs = model_conv(img_tensor)
        scale_ftrs.squeeze_(0)
        scales = scale_ftrs.cpu().numpy()


        x.append(np.concatenate((scales, scales)))
        y.append(0)


    for tamp_img in os.listdir(f'drive/MyDrive/MICC-F220-labeled/{i}tamp'):
        img = Image.open(f'drive/MyDrive/MICC-F220-labeled/{i}tamp/{tamp_img}')
        img_tensor = transform(img)
    img_tensor.unsqueeze_(0)
    tamp_ftrs = model_conv(img_tensor)
    tamp_ftrs.squeeze_(0)
    tamp_ftrs = tamp_ftrs.cpu().numpy()

    x.append(np.concatenate((scales, tamp_ftrs)))
    y.append(1)

x = np.array(x)
y = np.array(y)

x.shape, y.shape

np.unique(y, return_counts=True)

from sklearn import svm

classifier = svm.SVC(kernel='linear')
classifier.fit(x,y)

import sklearn
sklearn. __version__

classifier.score(x,y)*100

classifier.predict(x)
```

```
torch.save(model_conv, 'alex.pkl')
dump(classifier, 'svm.joblib')

"""# Inference"""

def predict(img1_path, img2_path, ftr_ext_path, classifier_path):
  transform = transforms.Compose([
      transforms.Resize(224),
      transforms.CenterCrop(224),
      transforms.ToTensor(),
      transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])
    ])

  alexnet = torch.load(ftr_ext_path)
  alexnet.eval()
  classifier = load(classifier_path)

  img1 = Image.open(img1_path)
  img2 = Image.open(img2_path)

  img1_tensor = transform(img1)
  img1_tensor.unsqueeze_(0)
  img1_ftrs = alexnet(img1_tensor)
  img1_ftrs.squeeze_(0)
  img1_ftrs = img1_ftrs.cpu().numpy()

  img2_tensor = transform(img2)
```

```
    img2_tensor.unsqueeze_(0)
    img2_ftrs = alexnet(img2_tensor)
    img2_ftrs.squeeze_(0)
    img2_ftrs = img2_ftrs.cpu().numpy()

    x = np.concatenate((img1_ftrs, img2_ftrs))
    x = np.expand_dims(x, axis=0)
    return classifier.predict(x)

p=predict('drive/MyDrive/MICC-F220-labeled/1scale/CRW_4853_scale.jpg',
        'drive/MyDrive/MICC-F220-labeled/1scale/CRW_4853_scale.jpg',
        'alex.pkl', 'svm.joblib')
#if p==0:
print(p)

p1=predict('drive/MyDrive/MICC-F220-labeled/1scale/CRW_4853_scale.jpg',
        'drive/MyDrive/MICC-F220-labeled/1tamp/CRW_4853tamp132.jpg',
        'alex.pkl', 'svm.joblib')
print(p1)
```

Final Output: -

If no Forgery it will generate ‘0’ value, if forgery it will generate ‘1’ value.



```
[ ]  p=predict('drive/MyDrive/MICC-F220-labeled/1scale/CRW_4853_scale.jpg',
             'drive/MyDrive/MICC-F220-labeled/1scale/CRW_4853_scale.jpg',
             'alex.pkl', 'svm.joblib')
     #if p==0:
     print(p)

     [0]

     p1=predict('drive/MyDrive/MICC-F220-labeled/1scale/CRW_4853_scale.jpg',
             'drive/MyDrive/MICC-F220-labeled/1tamp/CRW_4853tamp132.jpg',
             'alex.pkl', 'svm.joblib')
     print(p1)
```