CUSTOM PREDICTION ROUTINE WITH GOOGLE AI PLATFORM

→ It is a framework set in step 5 of model deployment.

TASK → We will log in google cloud platform and deploy a model on Google's AI Platform.

Any model created with Tensorflow is easy to deploy on AI platform

But we will look at Custom Prediction Routine.

Be familiar with → Keras & Python.

BASICALLY THIS TUTORIAL WAS ON HOW TO ~~PRODUCTION~~ TAKE A TRAINED MODEL TO PRODUCTION STATE. BUT WE TRY TO REPLICATE MOST OF IT ON GOOGLE COLAB BECAUSE OF LACK OF ACCESS TO GOOGLE CLOUD.

TASK-1 IMPORT MODEL

```
import tensorflow as tf.
Model = tf.keras.models.load_model('name.h5')
```

→ Loading the ~~model~~ pre-trained model with

__/__/____

-h5 extension

model.summary() → Summary of model

TASK-2 TESTING THE MODEL

%matplotlib inline → With this the output of plotting command is displayed.
import matplotlib.pyplot as plt. inline within frontend. i.e in jupyter notebook directly below the code cell that produced it.
import numpy as np.

Preprocessing the images

Classes = ['Cat', 'Dog'] (0-0.5 → Cat, 0.5-1 → Dog)

def display_pred (image_path): (pred ← prediction)

plt.imshow (plt.imread(image_path))

x = tf.keras.preprocessing.image.load_img(
image_path,
target_size = (128,128))

Rendering the image as our model was trained for image of this size.

Converting to numpy → x = tf.keras.preprocessing.image.img_to_array(x)

because the model was trained on images preprocessed with mobilenet_v2 preprocessing function → x = tf.keras.applications.mobilenet_v2.preprocess_input(x)

x = np.expand_dims (x, axis = 0)

__/__/____

Making prediction pred = model.predict(x)[0] ← Because only one image so first in array

pet.title('Pred: {}'.format(classes[int(pred > 0.5)]))

pet.show()

Raw value → print(pred)

If returns true or false

If converts it 0 or 1

It will give the name of class

images = ['1.jpg', '2.jpg', '3.jpg']

display_pred(images[0]).

TASK-3 Custom Prediction Class → Make a python file to derive our predict.

%%writefile prediction.py → When we execute this python file will be created.

import tensorflow as tf
import numpy as np
import os
import keras

→ imported.
→ Again for python file

MODEL_NAME = 'cats_vs_dogs.hs'
CLASS_NAMES = ['CAT', 'DOG']

class CatsVsDogsPrediction:
    def __init__(self, model):
        self.model = model

- @classmethod.

def ~~from_path~~ frompath (cls, model_dir):

model = tf.keras.models.load_model(os.path.join(model_dir, MODEL_NAME))

return cls(model)

↳ For Loading Model from dir. i.e google cloud but in our case colab.

TASK-4 - Preprocess.

kwargs → key word arguments
instances → images to predict.

def predict(self, instances, **kwargs):

~~#preprocess~~ Even though we can send tensors directly when we deploy in AI platform but that's not good practice. There are limits like how much data can we ~~transfer~~ transmit from AI. And if image is big we may exceed limit and hence cost more. Hence encoding when sending over internet. So the instances should be decoded when received via internet from client.

if 'size' in kwargs:

size = int(kwargs.get('size'))

else:

size = 128

#preprocess

x_batch = self._preprocess(instances, size)

#predict

preds = self._model.predict(x_batch)

Continued on next page.

(default if not passed.)

def _preprocess(~~self~~ self, instances, size=128):

num_examples = len(instances) // total images sent by user

x_batch = np.zeros((~~num~~num_examples, ~~128~~size, ~~128~~size, 3))

for i in range(num_examples):

x = np.array(bytearray(base64.b64decode(instances[i])))

↳ from bytes to numpy array. ↳ decode from string to bytes

x = np.reshape(x, (size, size, 3)) Reshape to desired size.

Preprocessing: x = tf.keras.applications.mobilenet_v2.preprocess_input(x)

x_batch[i] = x

return x_batch.

Reshape needs (224,224,3) else it will fail as per our model. Hence Reshape depends on model.

TASK-4 Post process

```
def _postprocess(self, preds):
    results = []
    for i, pred in enumerate(preds):
        p = np.squeeze(pred)
        results.append({
            'index': i,
            'class_name': CLASS_NAMES[int(p > 0.5)],
            'raw_value': '{:.4f}'.format(p)
        })
    return results.
```

enumerate over predictions.

// Removing extra dimension if any for each pred in preds.

3 Keys to be sent to user for each prediction.

int(p > 0.5) gives 0 or 1 (false or true) and hence cats or dog in class_names

Raw value upto 4 decimal places

Continued function

```
def predict(self, instances, **kwargs):
    # post process
    results = self._postprocess(preds)
    return results.
```

On Execution the python file is created

__/__/____

TASK-4. SETUP SCRIPT.

Create a setup script, which we ~~EMM~~ will run to create a distribution or a package that we will upload along with model artifact. Dr. ~~Jasbir~~ Kaur.

```
%%writefile setup.py

from setuptools import setup

setup(
    name = 'cats_vs_dogs',
    version = '0.0.1',
    include_package_data = False,
    scripts = ['prediction.py']
)
```

Creating a package on commandline.

```
! python3 setup.py sdist --formats=gztar
```

we will get a .tar.gz file to be uploaded to Google cloud. We can upload it in bucket where we upload model.

```
! gsutil cp dist/cats_vs_dogs-0.0.1.tar.gz gs://thyme-bucket/dist/
```

(cp = copy; source; destination)

When coding properly with cloud

TASK-5 DEPLOY CODE :→ On Google Cloud Platform. → Models → Etc Etc ← Create Model ←

TASK-6 GET Predictions (Use as client) → In jupyter lab only.

in the set $[p_1, p_2, \ldots, p_k]$. # But we assumed that this set is finite and there exists no other prime. Hence our assumption is wrong.

So the prime are not finite but infinite.

TASK-6 GET Predictions.

```
from googleapiclient import discovery
from PIL import Image
import os
import base64 → to encode images before sending because we decode them.

Creating a service. service = discovery.build('ml', 'v1', cache_discovery=False)

def get_pred_from_model(body, project_name, model_name):
return service.projects().predict(
        name='projects/{}/models/{}'.format(project_name,
                                            model_name)
        body=body
    ).execute() → so we can get decode the req we receive back

project_name = 'rhyme-264117' → we set it when we begin google cloud. So it is diff for all.

model_name='cats_vs_dogs' → we set it when deploying model

instances = []
size = 128
images = ['1.jpg', '2.jpg', ...]
for image in images:
    img = Image.open(image)
    img = img.resize((size, size), Image.ANTIALIAS)

    instances.append(base64.b64encode(img.tobytes()).decode('utf-8'))
img.close()
```

Send payload to model receive and response.

body = {'instances': instances, 'size': size}
response = get_pred_from_model (body, project_name, model_name
print (response) → Get Results.

PREDICTING HOUSE PRICES with REGRESSION USING TENSORFLOW

Features → Year of sale, age at time of sale, distance from city center, no. of stores in locality, lat, long.

TASK 1- Import Library

TASK 2.1 → Import Data.