VIRTUAL INTERNSHIP

DATA SCIENCE
LISUM01

DEPLOYMENT ON FLASK

Samuel Alejandro Cueva Lozano

07/07/2021

Sumario

1. Introduction

I'm going to use a Tensorflow model to detect COVID-19 in chest x-rays, this model has already been trained by me and its trained weights will be loaded into a restored model.

- The code to train the model is on my Github
- The trained weights is on my Google Drive

2. Model Creation

Preprocess function

```
import tensorflow as tf

path_weights = 'weights_covidResnet/cp-0030.ckpt'
path_images = 'resnet_images'


def preprocess_image(path, image_size):
    raw_img = tf.keras.preprocessing.image.load_img(path)
    img_array = tf.keras.preprocessing.image.img_to_array(raw_img)
    img = tf.keras.preprocessing.image.smart_resize(img_array, image_size)
    img = tf.expand_dims(img, 0)
    return img
```

Function to create COVID-RESNET model.

```
def create_covidResnet(base_model):
    """create top layers to customize ResNet50
       to a network for Covid-19 detection"""

    input = tf.keras.Input(shape=(256, 256, 3))
    preprocess_input = tf.keras.applications.resnet.preprocess_input(input)
    model_base = base_model(preprocess_input)
    global_average_layer = tf.keras.layers.GlobalAveragePooling2D()(model_base)
    drop_out_1 = tf.keras.layers.Dropout(0.4)(global_average_layer)
    dense_layer = tf.keras.layers.Dense(4096, activation='relu')(drop_out_1)
    drop_out_2 = tf.keras.layers.Dropout(0.4)(dense_layer)
    output_model = tf.keras.layers.Dense(3, activation='softmax')(drop_out_2)

    # create the model
    model = tf.keras.models.Model(inputs=input, outputs=output_model)

    return model
```

Create and serialize the model by using a Resnet-50 as base model, the method ‘save’ from Keras is used to serialize the model.

```
# create the base model(Resnet 50)
resnet50 = tf.keras.applications.ResNet50(include_top=False, input_shape=(256, 256, 3),
                                          weights=None)
# create COVID-RESNET
covid_resnet = create_covidResnet(resnet50)

# Load the trained weights into the model
covid_resnet.load_weights(path_weights)

# Save the trained model
covid_resnet.save('covid_resnet_model')
```

3. Define an endpoint for API

app.py

Import dependencies

```
from flask import Flask, render_template, request,jsonify
import tensorflow as tf
import numpy as np
from model import preprocess_image
```

Define some variables

```
IMAGE_SIZE = (256, 256)
PATHOLOGIES = ['COVID-19', 'NORMAL', 'PNEUMONIA']
path_img = 'images/image'
```

Load the serialized model

Keras function ‘load_model’ is used to deserialize the model that was saved in the file ‘covid_resnet_model’ .

```
model = tf.keras.models.load_model('covid_resnet_model')
```

Create the endpoint to submit a chest x-ray image and return the pathology (COVID19, pneumonia, normal)

```
app = Flask(__name__)


@app.route('/')
def home():
    return render_template('index.html')


@app.route('/predict', methods=['POST'])
def predict():
    # print(type(request.files['image']))
    request.files['image'].save(
        path_img)  # this save the image in a path but in production maybe it's not the better
    output = model.predict(preprocess_image(path_img, IMAGE_SIZE))[0]

    prediction = PATHOLOGIES[np.argmax(output)]
    p = output[np.argmax(output)] * 100

    return jsonify({'prediction':prediction,'percentage':p})
    # return


if __name__ == '__main__':
    app.run(port=5000, debug=True)
```

4. Front-end for the app

HTML Template (***index.html***) to send the image

```html
<!DOCTYPE html >
<html lang="en-US">
<head>
    <title>COVID-19 detection</title>
    <link rel="stylesheet" type="text/css" href="../static/index.css">
    <script src="//ajax.googleapis.com/ajax/libs/jquery/1.9.1/jquery.min.js"></script>
    <script>window.jQuery || document.write('<script src="{{url_for('static', filename='jquery.js')
}}">\x3C/script>')</script>
    <script type="text/javascript" src="../static/index.js"></script>

</head>
<body>
    <h1>COVID-19 DETECTION BY CHEST X-RAY</h1>
<div class="container">
<form action="/predict" method="post" enctype="multipart/form-data">
    <input class="btn" id="imageinput" type="file" name="image" onchange="readUrl(this)"
oninvalid="this.setCustomValidity('Texto personalizado')">
     <br />
    <img id="imgSalida" width="50%" height="50%" src="" />
    <br />
    <button class="btn" type="submit" name="send" id = "sendbutton">Send</button>
</form>
</div>
<div class="container" >
     <br />
    <div class="result" ><p id="div_result"></p></div>
</div>
</body>
</html>
```

static/index.js

The library jQuery is used to give functionality to the application.

```
$(window).load(function(){

 $(function() {

  $('#imgSalida').hide()
  $('#imageinput').change(function(e) {
      addImage(e);
     });

     function addImage(e){
      var file = e.target.files[0],
      imageType = /image.*/;

      if (!file.type.match(imageType))
       return;

      var reader = new FileReader();
      reader.onload = fileOnload;
      reader.readAsDataURL(file);
     }

     function fileOnload(e) {
      var result=e.target.result;
      $('#imgSalida').show();
      $('#imgSalida').attr("src",result);
     }



     $('#imageinput').click(function(){
        $('#div_result').hide()

     });


     $('form').on('submit', function(event) {
        var formData = new FormData(this);
        $.ajax({
            data : formData,
            type : 'POST',
            url : '/predict',
            contentType: false,
            processData: false
        })
        .done(function(data) {
            $('#div_result').show()
            $('#div_result').text('The x-ray result is: '+data.prediction+ ' with a
'+data.percentage+'% probability')


        });

        event.preventDefault();

    });

    });
  });
```

static/index.css

```css
container{
  background-color: #fafafa;
  margin: 1rem;
  padding: 1rem;
  border: 2px solid #ccc;
  /* IMPORTANT */
  text-align: center;
}



.btn{
 display: block;
 padding: 10px 21px;
 border: none;
 color: #fff;
 font-size: 18px;
 background-color: #3991A9;
 border-radius: 3px;
 cursor: pointer;
 border-bottom: 3px solid #237085;
 font-family: sans-serif;
 text-align: center;
}

h1{
background-color: #3991A9;
text-align: center;
color: white;
}

.result{
    font-size: 38px;
    background-color: #3991A9;
}
.btn:hover{
 transition: all .4s;
 background-color: #237085;
}
```

Work Tree

```
.
├── app.py
├── covid_resnet_model
│   ├── assets
│   ├── saved_model.pb
│   └── variables
│       ├── variables.data-00000-of-00001
│       └── variables.index
├── images
│   ├── image
│   └── image.jpg
├── model
├── model.py
├── __pycache__
│   └── model.cpython-38.pyc
├── resnet_images
│   ├── Covid_1.jpg
│   ├── covid_2.jpg
│   ├── normal_1.jpg
│   ├── normal_2.jpg
│   ├── pneumonia_1.jpg
│   └── pneumonia_2.jpg
├── static
│   ├── index.css
│   └── index.js
├── templates
│   ├── index.html
│   └── result.html
└── weights_covidResnet
    ├── checkpoint
    ├── cp-0030.ckpt.data-00000-of-00001
    └── cp-0030.ckpt.index

9 directories, 22 files
```

5. Run Flask App

Head to the project directory and activate the ***conda*** environment

```
samuelcueva@debiansc:~$ cd machine_learning/deploy_covid_resnet/
samuelcueva@debiansc:~/machine_learning/deploy_covid_resnet$ conda activate ml
(ml) samuelcueva@debiansc:~/machine_learning/deploy_covid_resnet$ ls -l
total 36
-rw-r--r-- 1 samuelcueva samuelcueva 1218 jul  8 01:47 app.py
drwxr-xr-x 4 samuelcueva samuelcueva 4096 jul  8 01:36 covid_resnet_model
drwxr-xr-x 2 samuelcueva samuelcueva 4096 jul  6 23:30 images
-rw-r--r-- 1 samuelcueva samuelcueva    0 jul  5 22:06 model
-rw-r--r-- 1 samuelcueva samuelcueva 1538 jul  7 23:59 model.py
drwxr-xr-x 2 samuelcueva samuelcueva 4096 jul  8 00:51 __pycache__
drwxr-xr-x 2 samuelcueva samuelcueva 4096 jul  8 01:07 resnet_images
drwxr-xr-x 2 samuelcueva samuelcueva 4096 jul  8 00:30 static
drwxr-xr-x 2 samuelcueva samuelcueva 4096 jul  7 00:16 templates
drwxr-xr-x 2 samuelcueva samuelcueva 4096 jul  5 18:59 weights_covidResnet
```

Run the ‘app.py’ file and open the URL where the flask app is running (localhost).

```
(ml) samuelcueva@debiansc:~/machine_learning/deploy_covid_resnet$ python app.py
2021-07-08 17:40:01.985049: I tensorflow/compiler/jit/xla_cpu_device.cc:41] Not creating XLA devices, tf_xla_enable_xla_de
vices not set
2021-07-08 17:40:01.985646: I tensorflow/core/platform/cpu_feature_guard.cc:142] This TensorFlow binary is optimized with
oneAPI Deep Neural Network Library (oneDNN) to use the following CPU instructions in performance-critical operations:  SSE
4.1 SSE4.2 AVX AVX2 FMA
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
2021-07-08 17:40:02.047841: I tensorflow/core/common_runtime/process_util.cc:146] Creating new thread pool with default in
ter op setting: 2. Tune using inter_op_parallelism_threads for best performance.
2021-07-08 17:40:17.021689: W tensorflow/python/util/util.cc:348] Sets are not currently considered sequences, but this ma
y change in the future, so consider avoiding using them.
WARNING:tensorflow:No training configuration found in save file, so the model was *not* compiled. Compile it manually.
 * Serving Flask app 'app' (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

6. Predicting results using Enpoints

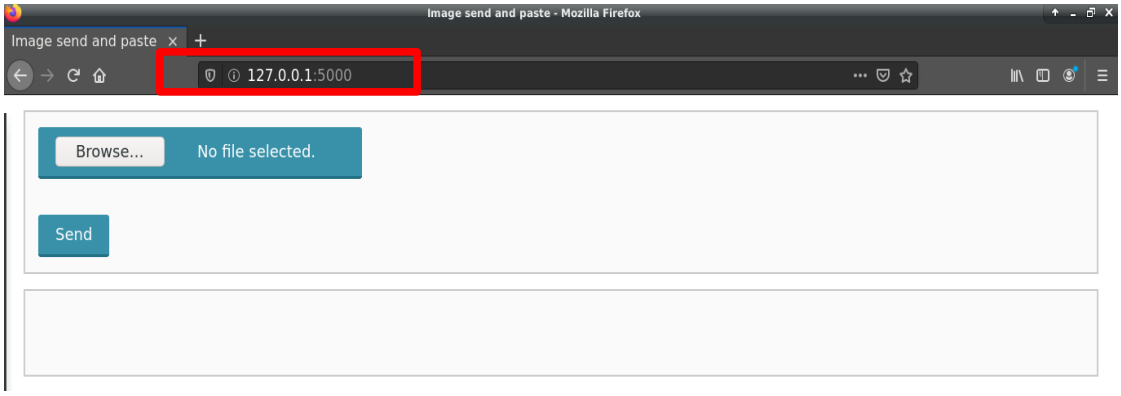
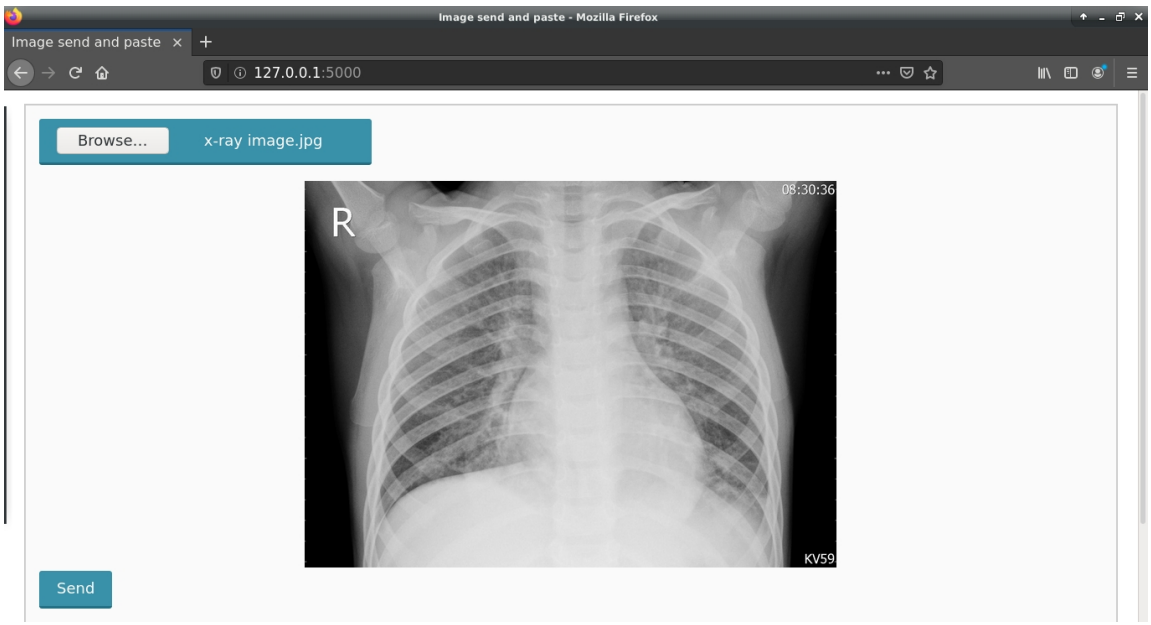


Image Loading



Response

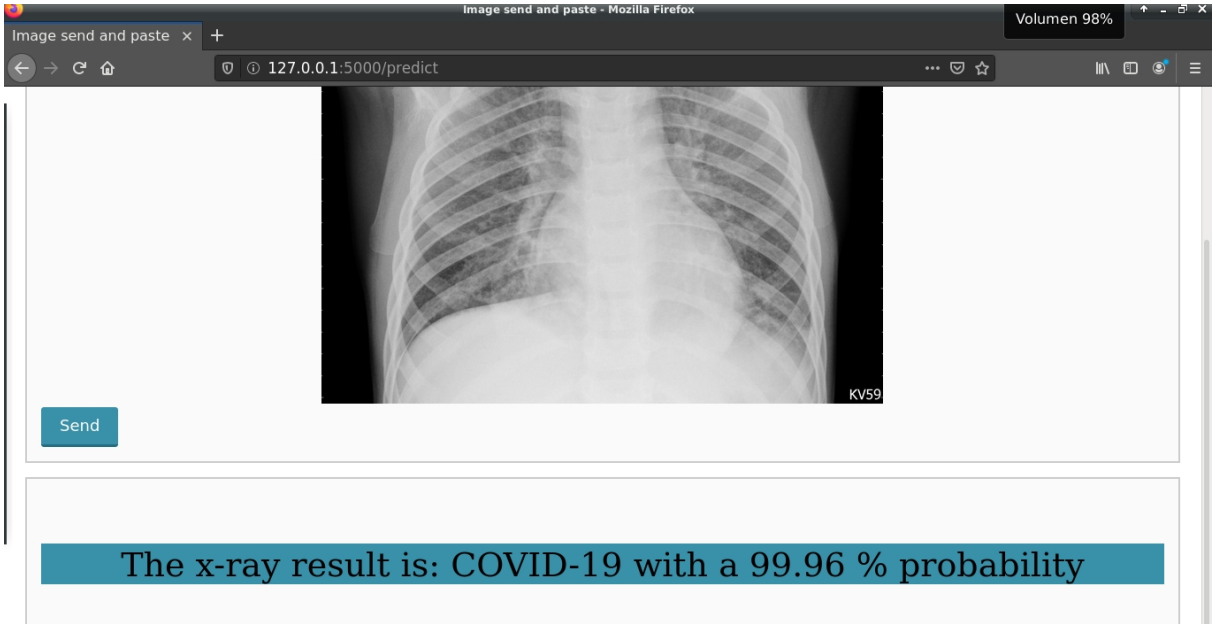


Image send and paste - Mozilla Firefox
Volumen 98%
Image send and paste
127.0.0.1:5000/predict
KV59
Send
The x-ray result is: COVID-19 with a 99.96 % probability