Name: Samuel Alejandro Cueva Lozano
Batch code: LISUM01
Submission date: 07/07/2021
Submitted to: Data Glacier

1. Model Creation

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 Building

Preprocess function

```
import tensorflow as tf

path_weights = 'weights_covidResnet/cp-0030.ckpt'

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’ 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

Hyper parameters and model loading, Keras function ‘load_model’ is used to deserialize the model that was saved in the file ‘covid_resnet_model’

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

IMAGE_SIZE = (256, 256)
PATHOLOGIES = ['COVID-19', 'NORMAL', 'PNEUMONIA']
path_img = 'images/image.jpg'
# To load the serialized 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 render_template('index.html',
                           prediction='The x-ray result is: {} with a {:.2f} % probability'.format(
                               prediction, p))
```

HTML Template (index.html) to send the image

```
<body>
<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="th
     <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>{{prediction}}</p></div>
</div>
</body>
```

Add CSS and Javascript files to load and preview the image

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

 $(function() {
  $('#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);
     }
```

```
.container{
  background-color: #fafafa;
  margin: 1rem;
  padding: 1rem;
  border: 2px solid #ccc;
  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;
```

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
```

4. Predicting results using Enpoints

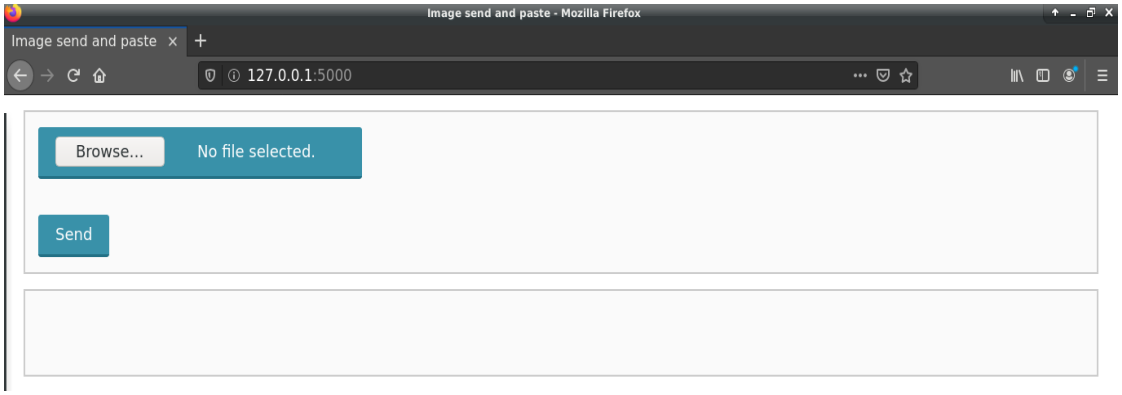
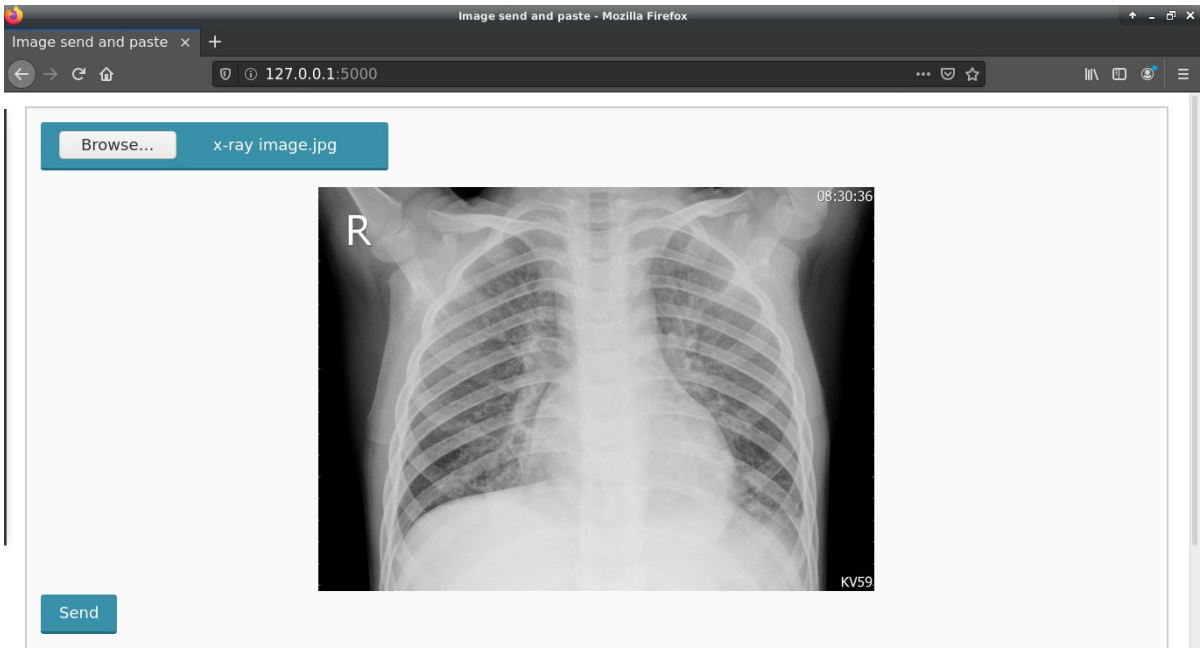


Image Loading



Response