Creación de una aplicación web que use un modelo entrenado con Deep Learning a partir del curso de Fastai

[Practical Deep Learning for Coders](https://course.fast.ai/)

# Motivación

En la lección 2 se ofrecen diferentes opciones para poder usar el modelo entrenando en el curso a través de la web. Recomiendan hacerlo a través de [Voila](https://pypi.org/project/voila/) y [Binder](https://mybinder.org/) por ser una forma sencilla de publicar Jupyter notebooks.

Yo, en cambio, he querido ir más allá y crear una aplicación web en la cual se realice la predicción del modelo entrenado en el lado del servidor. Esta aproximación tiene la ventaja de poder implementarse de forma fácil también en apps de dispositivos móviles.

De esta forma, se podrá subir la imagen desde la web al servidor, devolviendo el resultado obtenido.

# Preparación del entorno

En el terminal:

* Crea un entorno virtual donde trabajar:

|  |
| --- |
| python3 -m venv env1 source env1/bin/activate |

*Si se quiere salir del entorno:*

|  |
| --- |
| deactivate |

* Instala [Flask](https://flask.palletsprojects.com/): framework que permite crear aplicaciones web rápidamente

|  |
| --- |
| pip3 install flask python -m flask --version |

* Crea directorio donde alojar la web y dos subcarpetas que usará la app

|  |
| --- |
| mkdir web cd web  mkdir templates  mkdir static |

*Nota: no cambiar el nombre de templates y static ya que los usará Flask*

# Archivos de la aplicación

* Crea un archivo de python con tu editor de textos (yo uso [VS Code](https://code.visualstudio.com/)) en la carpeta “web”. Combinaremos el código de nuestro modelo de Deep Learning usado para el curso y escrito en un [Jupyter Notebook](https://jupyter.org/) y el código específico de Flask que generará la web.

**import** fastbook  
**from** fastbook **import** \*  
**from** fastai.vision **import** \*  
**from** flask **import** Flask, render\_template, request, redirect, url\_for, abort, send\_from\_directory  
**import** os  
**from** werkzeug.utils **import** secure\_filename  
**import** imghdr  
  
fastbook.setup\_book()  
  
  
*# Se importa nuestro modelo preentrenado “export.pkl”*  
  
path=Path()  
path.ls(file\_exts='.pkl')  
learn\_inf=load\_learner(path/'export.pkl')  
  
  
*# valores para uso de Flask*  
app = Flask(\_\_name\_\_)  
app.config['MAX\_CONTENT\_LENGTH'] = 2 \* 1024 \* 1024  
app.config['UPLOAD\_EXTENSIONS'] = ['.jpg', '.png', '.gif']  
app.config['UPLOAD\_PATH'] = 'static'  
  
  
*# función que valida la extensión del archivo*  
**def** **validate\_image**(stream):  
 header = stream.read(512)  
 stream.seek(0)  
 format = imghdr.what(**None**, header)  
 **if** **not** format:  
 **return** **None**  
 **return** '.' + (format **if** format != 'jpeg' **else** 'jpg')  
  
*# función que valida el tamaño del archivo*  
**@app.errorhandler(413)**  
**def** **too\_large**(e):  
 **return** "File is too large", 413  
  
  
*# función de genera la página pricipal*  
**@app.route('/')**  
**def** **index**():  
 **for** f **in** os.listdir(app.config['UPLOAD\_PATH']):  
 os.remove(os.path.join(app.config['UPLOAD\_PATH'], f))  
 **return** render\_template('inicio.html')  
  
*# función generada tras pulsar boton de Clasificación*  
**@app.route('/', methods=['POST'])**  
**def** **upload\_files**():  
 files = os.listdir(app.config['UPLOAD\_PATH'])  
 uploaded\_file = request.files['file']  
 filename = secure\_filename(uploaded\_file.filename)  
 **if** filename != '':  
 file\_ext = os.path.splitext(filename)[1]  
 **if** file\_ext **not** **in** app.config['UPLOAD\_EXTENSIONS'] **or** \  
 file\_ext != validate\_image(uploaded\_file.stream):  
 **return** "Invalid image", 400  
   
 filenamefull =os.path.join(app.config['UPLOAD\_PATH'], filename)   
 uploaded\_file.save(filenamefull) *#guardar la imagen, se mostrará en el resultado*  
   
 img = PILImage.create(os.path.join(filenamefull))  
 pred,pred\_inx,prob=learn\_inf.predict(img) *# uso del modelo para hacer la predicción*  
 prediccion=f'Prediccion: {pred}; probabilidad: {prob[pred\_inx]:.04f}'  
   
 **return** render\_template('resultado.html',prediccion=prediccion, files=files,imagen=filenamefull)

* Dentro de “templates” creamos dos archivo html “inicio.html” y “resultado.html”

**inicio.html**

|  |
| --- |
| **<!doctype html> *<!--Aplicacion web basica: se elige una imagen, se envia al servidor para pasarla por el modelo entrenado-->* <html> <head>  <title>Clasificador de lunares</title> </head>  <body>  <h1>Clasificador de lunares</h1>  <form method="POST" action="" enctype="multipart/form-data">  <p><input type="file" name="file"></p>  <p><input type="submit" value="Clasificar"></p>  </form> </body> </html>** |

**resultado.html**

|  |
| --- |
| **<!doctype html> *<!--Aplicacion web basica: se muestra la imagen elegida y el resultado de la prediccion-->*  <html> <head>  <title>Clasificador de lunares</title> </head>  <body>  <h1>Clasificador de lunares</h1>  <img src="{{imagen}}" style="width: 128px">  <p>{{prediccion}}</p>  <br>  <button onclick="goBack()">Volver</button>  <script>  function goBack() {  window.history.back();  }  </script>  </body> </html>** |

# Publicación de la aplicación

* Iniciar el servidor web con Flask en modo local:

|  |
| --- |
| **export FLASK\_APP=app**  **flask run** |

* Abrir navegador y colocar en la barra de direcciones:

|  |
| --- |
| http://127.0.0.1:5000/ |

* Averiguar la dirección ip en mi red:

|  |
| --- |
| ip a |

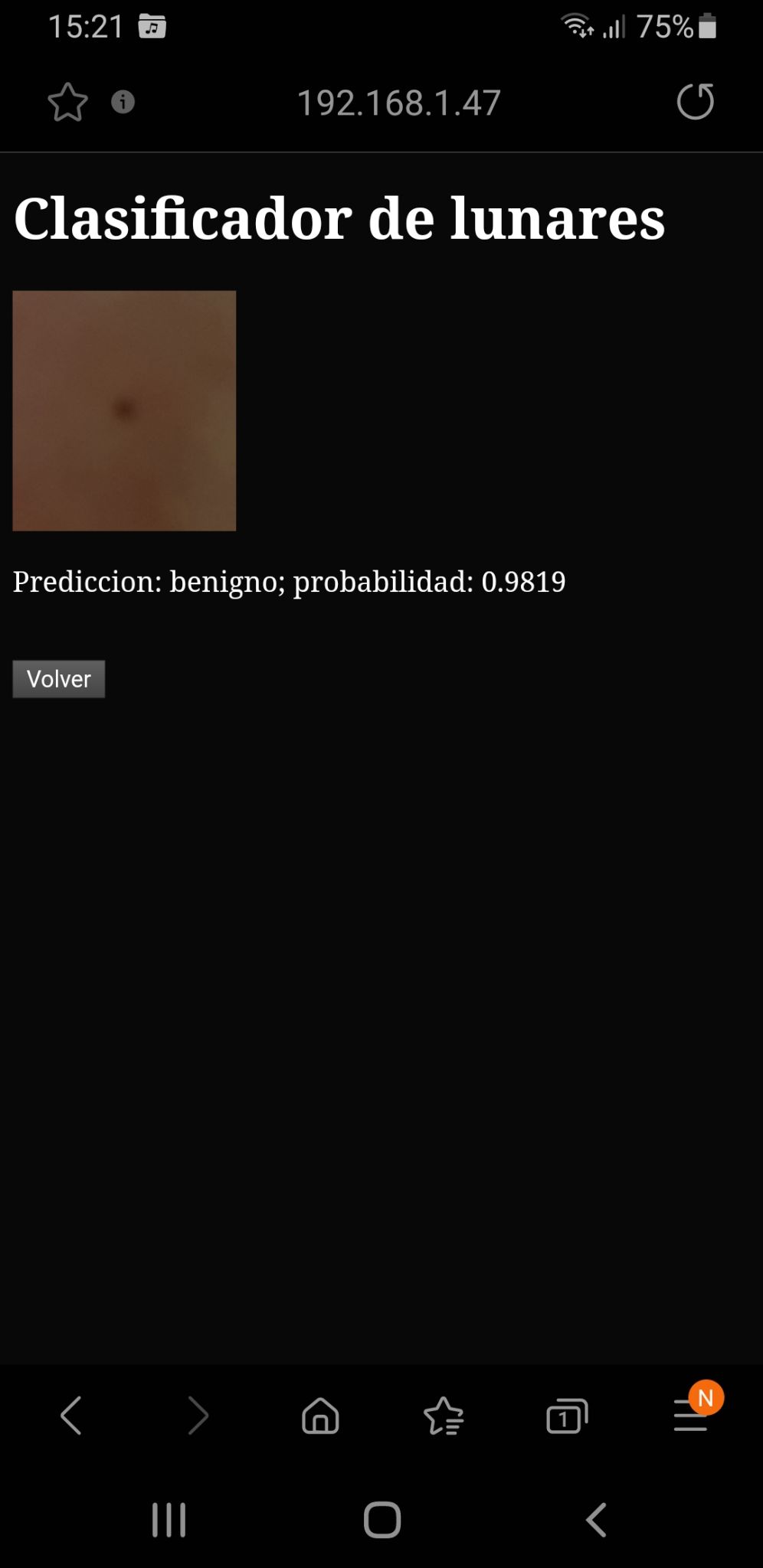
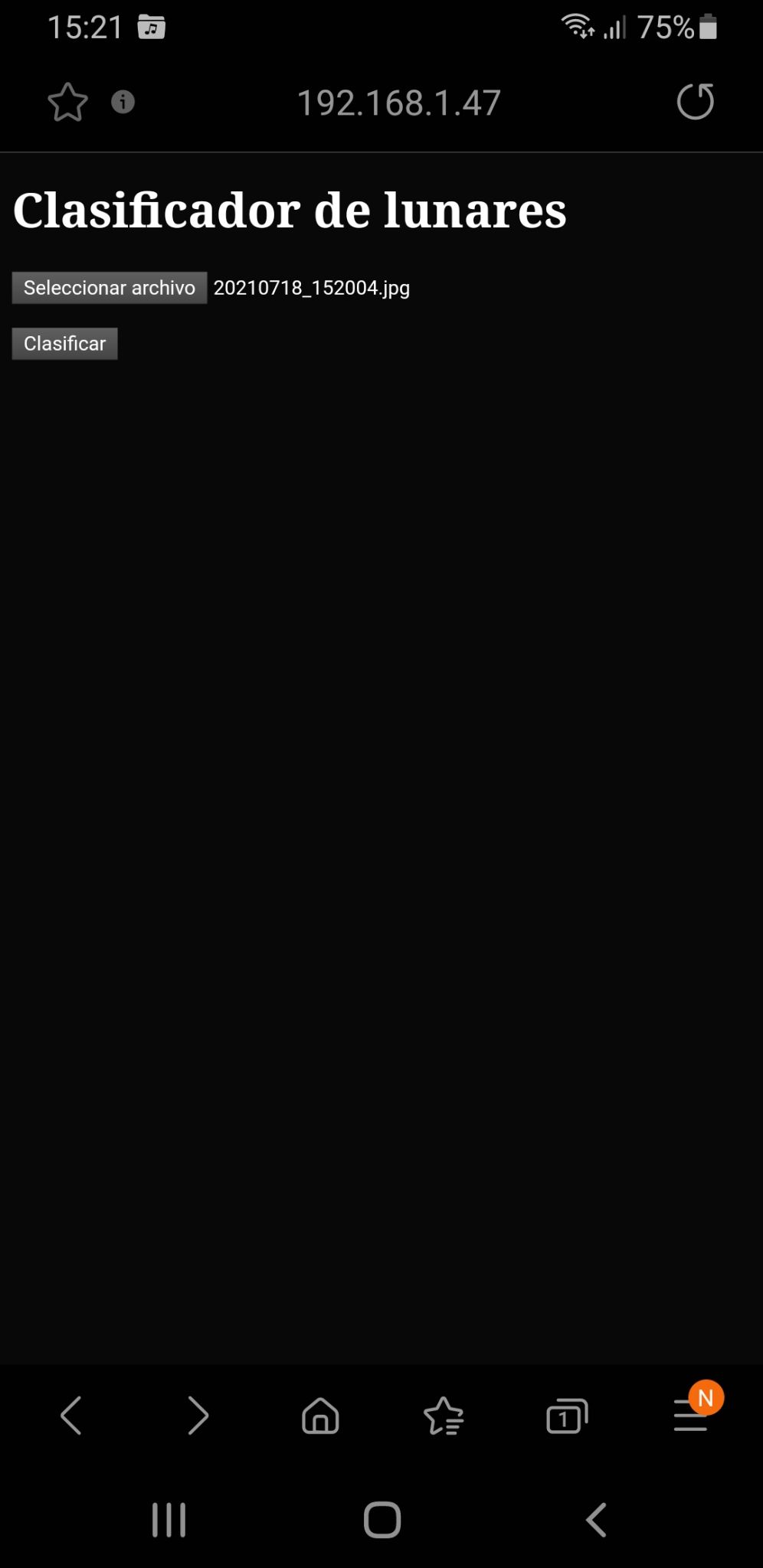


* Ejecutar la aplicación de forma visible dentro de mi red:

|  |
| --- |
| flask run --host=0.0.0.0 |

Ahora podrás acceder desde cualquier dispositivo que esté conectado a tu red (a tu WIFI, por ejemplo)





* Ejecutar flask de forma visible fuera de mi red:

Para ello necesitaremos un servidor con acceso a la red que ejecute nuestra aplicación. Podemos montarlo nosotros o usar los servicios de otras empresas como, por ejemplo:

* [Heroku](https://www.heroku.com/)
* [Microsoft Azure](https://azure.microsoft.com/es-es/)
* [Amazon AWS](https://aws.amazon.com/es/)
* [SeeMe.ai](https://course.fast.ai/deployment_seeme_ai#create-a-model-on-seemeai)