CAPÍTULO 1

APLICACIÓN WEB

Para la ejecución de la aplicación web en local se necesitará Python 3.7.6 y las librerías Tensorflow, OpenCV, ImgAug y Django. Aunque se ha desarrollado un instalador multiplataforma con los únicos requisitos de tener instalado Python 3.7.* y, en caso de querer usar aceleración por GPU, CUDA y cuDNN para NVIDIA [10].

La aplicación web almacena modelos previamente entrenados en su base de datos. Estos modelos deberán ser o de clasificación de humo binaria o de segmentación de humo. Además, podrán ser elegidos por el propio usuario del despliegue.

Además las imágenes que pasen por los modelos serán guardadas como recolecta para aumentar el *dataset*.

1.1. MANUAL DE USUARIO

Una vez se introduzca la URL de la web la página principal mostrará 4 secciones (Ver Fig. 5.21):

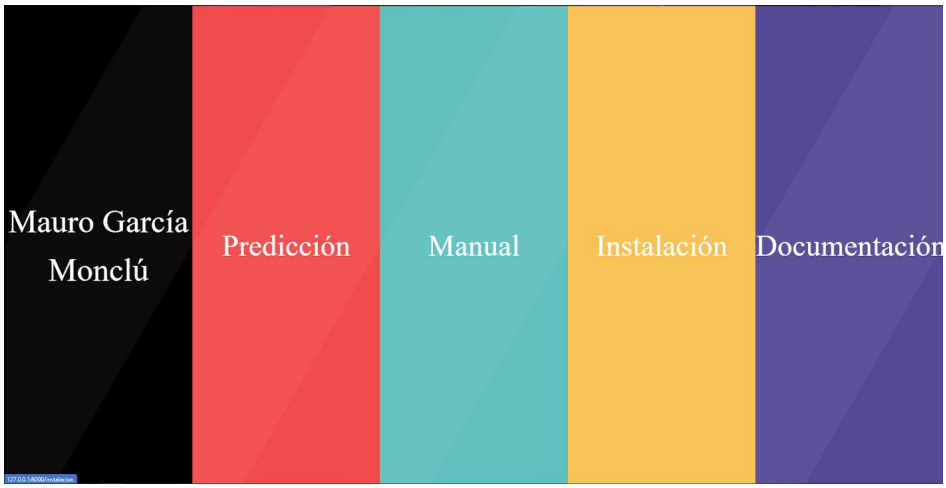
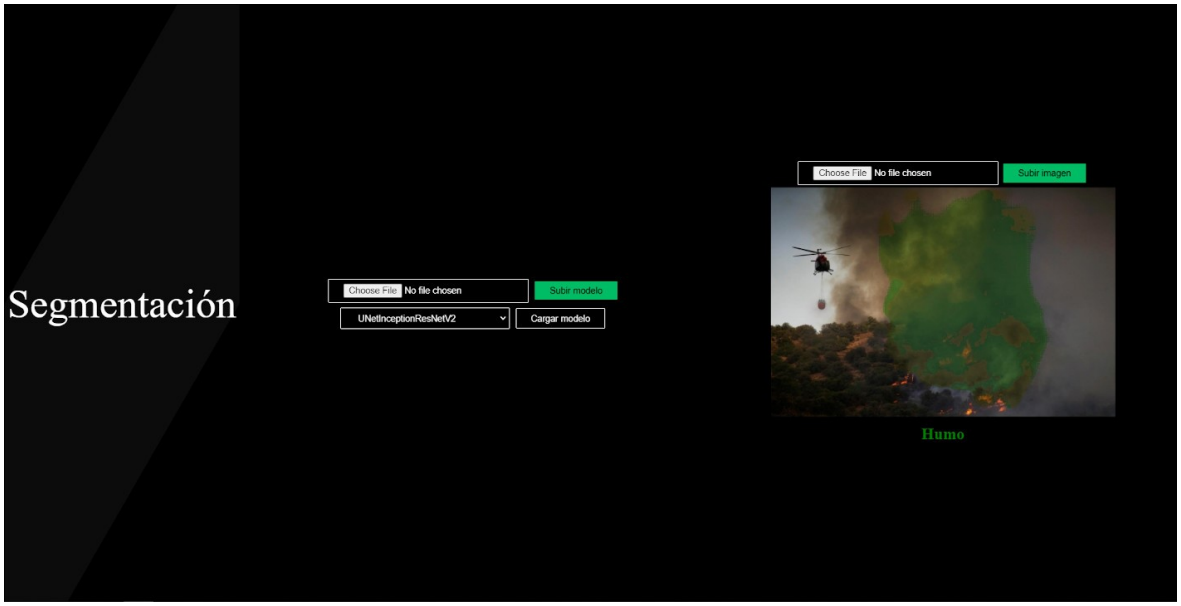


Figura 1.1: Imagen de la página principal de la aplicación web

- **Predicción**(Ver Fig. 5.22): está sección es la aplicación de la página web. Contiene la lógica de predicción tanto de segmentación como de clasificación. Tiene dos subsecciones: Clasificación y Segmentación. Cada sección nos permite cargar modelos previamente entrenados para la detección de humo. Los modelos predeterminados son modelos elegidos por el servidor, pero cada usuario podrá subir sus propios modelos. Una vez cargado un modelo se deberá subir una imagen que posteriormente será enviada al modelo y su resultado se visualizará en la web, ya sea clasificación con un etiquetado debajo de la imagen o segmentación con la creación de una imagen con el humo señalizado en verde.

Cabe destacar que está sección queda bloqueada una vez un usuario la está usando. A partir de la IP del cliente si está libre solo aceptará acciones de está IP hasta que lleve 5 o más minutos sin realizar ninguna acción (Subir modelo, subir imagen, cargar modelo, cambiar de método…). Una vez el usuario termine la sesión los modelos e imágenes que haya subido serán borrados. El listado 5.7 muestra como se estructura la Web en código y la lógica para controlar la IP del cliente.

(a) Clasificación



(b) Segmentación

Figura 1.2: Imagen de la interfaz de predicción de la aplicación web

- **Manual**: se muestra una guía de cómo utilizar la página web y sus características.
- **Instalación**: la sección proporciona un tutorial con la instalación de las dependencias necesarias para desplegar la página web en cualquier computador.
- **Documentación**: muestra sin previas descargas la documentación de este proyecto. En este capítulo se dará una visión general del proyecto. Se introducirá su contexto general, una motivación que se pretenderá cumplir, un resumen y una explicación de la estructura de este documento.

Listado 1.1: Código fuente de las funciones que representan cada dirección de la web

```
CurrentIP = ''
LastUsed = datetime.now()

# Página principal
def index(request):
    return render(request, 'index.html')


# Página de elección de predicción
def prediction(request):
    return render(request, 'prediccion.html')


# Página para la clasificación de imágenes
def classification(request):
    global CurrentIP
    ...
    ip = request.META['REMOTE_ADDR']
    if CurrentIP == ip:
        context = {}
        LastUsed = datetime.now()
        if request.method == 'POST':
            ...
        return render(request, 'clasificacion.html', context)
    else:
        if (datetime.now() - LastUsed).total_seconds() > 300:
            CurrentIP = ip
            ...
            return render(request, 'clasificacion.html', context)
        else:
            return render(request, 'prediccionOcupada.html')


# Página para la segmentación de imágenes
def segmentation(request):
    global CurrentIP
    ...
    ip = request.META['REMOTE_ADDR']
    if CurrentIP == ip:
        context = {}
        LastUsed = datetime.now()
        ...
        return render(request, 'segmentacion.html', context)
    else:
        if (datetime.now() - LastUsed).total_seconds() > 300:
            CurrentIP = ip
            ...
            return render(request, 'segmentacion.html', context)
        else:
            return render(request, 'prediccionOcupada.html')


# Página donde muestra la documentación
def documentation(request):
    return render(request, 'documentacion.html')

...
```