

Universidades de Burgos, León y  
Valladolid

Máster universitario

## **Inteligencia de Negocio y Big Data en Entornos Seguros**



**Trabajo Fin de Máster**

**Algo muy largo y no sé que más  
puedo hacer aquí**

Presentado por José Miguel Ramírez Sanz  
en Universidad de Burgos — 2 de mayo  
de 2020

Tutor: Dr. José Francisco Díez Pastor  
Dr. Álvar Arnaiz Gonzalez





# Universidades de Burgos, León y Valladolid



## Máster universitario en Inteligencia de Negocio y Big Data en Entornos Seguros

D. nombre tutor, profesor del departamento de nombre departamento, área de nombre área.

Expone:

Que el alumno D. José Miguel Ramírez Sanz, con DNI 71303106R, ha realizado el Trabajo final de Máster en Inteligencia de Negocio y Big Data en Entornos Seguros titulado título de TFM.

Y que dicho trabajo ha sido realizado por el alumno bajo la dirección del que suscribe, en virtud de lo cual se autoriza su presentación y defensa.

En Burgos, 2 de mayo de 2020

Vº. Bº. del Tutor:

Dr. José Francisco Díez Pastor

Vº. Bº. del co-tutor:

Dr. Álvar Arnaiz Gonzalez





## **Resumen**

En este primer apartado se hace una **breve** presentación del tema que se aborda en el proyecto.

## **Descriptores**

Palabras separadas por comas que identifiquen el contenido del proyecto Ej: servidor web, buscador de vuelos, android ...

**Abstract**

A **brief** presentation of the topic addressed in the project.

**Keywords**

keywords separated by commas.

---

# Índice general

---

Índice general	III
Índice de figuras	v
Índice de tablas	vi
<b>Memoria</b>	<b>1</b>
<b>1. Introducción</b>	<b>3</b>
<b>2. Objetivos del proyecto</b>	<b>5</b>
2.1. Objetivos funcionales . . . . .	5
2.2. Objetivos técnicos . . . . .	5
2.3. Objetivos personales . . . . .	6
<b>3. Conceptos teóricos</b>	<b>7</b>
3.1. Secciones . . . . .	7
3.2. Referencias . . . . .	7
3.3. Imágenes . . . . .	8
3.4. Listas de items . . . . .	8
3.5. Tablas . . . . .	9
<b>4. Técnicas y herramientas</b>	<b>11</b>
4.1. Visión por computador ~ Detección de movimiento . . . . .	11
<b>5. Aspectos relevantes del desarrollo del proyecto</b>	<b>15</b>
5.1. Desarrollo FIS-HUBU . . . . .	15

5.2. Investigación algoritmos de visión por computador . . . . .	19
5.3. Investigación de <i>Detectron2</i> . . . . .	20
5.4. Cálculo de características . . . . .	26
<b>6. Trabajos relacionados</b>	<b>33</b>
<b>7. Conclusiones y Líneas de trabajo futuras</b>	<b>35</b>
 <b>Apéndices</b>	 <b>36</b>
<b>Apéndice A Plan de Proyecto Software</b>	<b>39</b>
A.1. Introducción . . . . .	39
A.2. Planificación temporal . . . . .	39
A.3. Estudio de viabilidad . . . . .	41
<b>Apéndice B Especificación de Requisitos</b>	<b>43</b>
B.1. Introducción . . . . .	43
B.2. Objetivos generales . . . . .	43
B.3. Catalogo de requisitos . . . . .	43
B.4. Especificación de requisitos . . . . .	43
<b>Apéndice C Especificación de diseño</b>	<b>45</b>
C.1. Introducción . . . . .	45
C.2. Diseño de datos . . . . .	45
C.3. Diseño procedimental . . . . .	45
C.4. Diseño arquitectónico . . . . .	45
<b>Apéndice D Documentación técnica de programación</b>	<b>47</b>
D.1. Introducción . . . . .	47
D.2. Estructura de directorios . . . . .	47
D.3. Manual del programador . . . . .	47
D.4. Compilación, instalación y ejecución del proyecto . . . . .	47
D.5. Pruebas del sistema . . . . .	47
<b>Apéndice E Documentación de usuario</b>	<b>49</b>
E.1. Introducción . . . . .	49
E.2. Requisitos de usuarios . . . . .	49
E.3. Instalación . . . . .	49
E.4. Manual del usuario . . . . .	49
<b>Bibliografía</b>	<b>51</b>

---

# Índice de figuras

---

3.1. Autómata para una expresión vacía . . . . .	8
4.2. Logo de Detectron2. . . . .	12
4.3. Logo de TensorFlow. . . . .	12
5.4. Menú principal de un responsable. . . . .	17
5.5. Menú de estadísticas. . . . .	17
5.6. Ejemplo de la evolución de un paciente vista por un responsable.	18
5.7. Menú principal de los pacientes. . . . .	18
5.8. Mejora del rendimiento en GB/s con NVlink [5]. . . . .	22
5.9. Modelo de tipo COCO Detection with Faster R-CNN, faster_rcnn_R_50_C4_1x.	23
5.10. Modelo de tipo COCO Detection with RetinaNet, retinanet_R_101_FPN_3x.	24
5.11. Modelo de tipo COCO Instance Segmentation Baselines with Mask R-CNN, mask_rcnn_R_50_DC5_3x. . . . .	25
5.12. Modelo de tipo COCO Person Keypoint Detection Baselines with Keypoint R-CNN, keypoint_rcnn_R_101_FPN_3x. . . . .	26
5.13. Modelo de tipo COCO Panoptic Segmentation Baselines with Panoptic FPN, panoptic_fpn_R_101_3x. . . . .	27
5.14. Modelo de tipo LVIS Instance Segmentation Baselines with Mask R-CNN, mask_rcnn_X_101_32x8d_FPN_1x. . . . .	28
5.15. Prueba con chaqueta. . . . .	29
5.16. Prueba con un objeto de por medio. . . . .	30
5.17. Prueba con <i>threshold</i> a 0.3. . . . .	31
5.18. Prueba con <i>threshold</i> a 0.99. . . . .	31
5.19. Puntos clave predichos con el modelo junto con la etiqueta con su orden. . . . .	32

---

## **Índice de tablas**

---

3.1. Herramientas y tecnologías utilizadas en cada parte del proyecto	10
5.2. Tabla con el estudio de los modelos de posición ordenado por ratio.	23

# **Memoria**



---

# **Introducción**

---

Descripción del contenido del trabajo y del estrucutra de la memoria y del resto de materiales entregados.



---

# **Objetivos del proyecto**

---

En este apartado se va a comentar los objetivos funcionales, técnicos y personales que se tienen en el desarrollo de este proyecto.

## **2.1. Objetivos funcionales**

En este subapartado se va a comentar los distintos objetivos que se querían cumplir con el desarrollo del proyecto:

- Desarrollar una aplicación web para poder realizar y evaluar la rehabilitaciones *online* de los pacientes con Parkinson.
- Desarrollar una aplicación web accesible y fácil de utilizar, sobre todo para los pacientes.
- Crear un herramienta capaz de comparar las rehabilitaciones hechas por los pacientes con los ejercicios bases.

## **2.2. Objetivos técnicos**

En este subapartado se va a comentar los distintos objetivos relacionados con las técnicas y herramientas que se quieren aprender y utilizar:

- Crear una aplicación web para realizar rehabilitaciones *online*.
- Crear una herramienta de comparación de ejercicios a partir de algoritmos de visión por computador y detección de movimientos.

- Utilizar el lenguaje de programación *Python* para la comparación de ejercicios.
- Realizar el desarrollo del proyecto utilizando un repositorio *Git*, en concreto en *GitHub* para poder controlar las tareas y las versiones del proyecto.
- Utilizar la extensión de *Git* llamada *ZenHub* para controlar el estado de las tareas y la temporalidad de estas.
- Seguir el modelo *SCRUM* para desarrollar el proyecto de forma incremental.

### **2.3. Objetivos personales**

Por último, se van a comentar los objetivos personales que se tienen en este proyecto, donde se encuentran objetivos desde probar conocimientos adquiridos en el máster como mejores algunas cualidades personales.

- Poder ayudar a las personas mayores con Parkinson para que puedan realizar las rehabilitaciones necesarias sin necesidad de salir de sus casas. Además, mejorando esta rehabilitación a partir de la comparación de los ejercicios realizados.
- Mejorar mis capacidades comunicativas y de exposición en las diversas presentaciones del proyecto a los responsables.
- Usar los conocimientos adquiridos durante la carrera y durante el máster.
- Mejorar mis conocimientos sobre visión por computador.
- Conocer los distintos algoritmos de comparación y clasificación de instancias para los datos obtenidos sobre los ejercicios.

---

# **Conceptos teóricos**

---

En aquellos proyectos que necesiten para su comprensión y desarrollo de unos conceptos teóricos de una determinada materia o de un determinado dominio de conocimiento, debe existir un apartado que sintetice dichos conceptos.

Algunos conceptos teóricos de L<sup>A</sup>T<sub>E</sub>X<sup>1</sup>.

## **3.1. Secciones**

Las secciones se incluyen con el comando section.

### **Subsecciones**

Además de secciones tenemos subsecciones.

### **Subsubsecciones**

Y subsecciones.

## **3.2. Referencias**

Las referencias se incluyen en el texto usando cite [7]. Para citar webs, artículos o libros [4].

---

<sup>1</sup>Créditos a los proyectos de Álvaro López Cantero: Configurador de Presupuestos y Roberto Izquierdo Amo: PLQuiz

### 3.3. Imágenes

Se pueden incluir imágenes con los comandos standard de L<sup>A</sup>T<sub>E</sub>X, pero esta plantilla dispone de comandos propios como por ejemplo el siguiente:



Figura 3.1: Autómata para una expresión vacía

### 3.4. Listas de items

Existen tres posibilidades:

- primer item.
  - segundo item.
1. primer item.
  2. segundo item.

**Primer item** más información sobre el primer item.

**Segundo item** más información sobre el segundo item.

■

### 3.5. Tablas

Igualmente se pueden usar los comandos específicos de L<sup>A</sup>T<sub>E</sub>Xo bien usar alguno de los comandos de la plantilla.

Herramientas	App	AngularJS	API REST	BD	Memoria
HTML5		X			
CSS3		X			
BOOTSTRAP		X			
JavaScript		X			
AngularJS		X			
Bower		X			
PHP			X		
Karma + Jasmine		X			
Slim framework			X		
Idiorm			X		
Composer			X		
JSON		X	X		
PhpStorm		X	X		
MySQL				X	
PhpMyAdmin				X	
Git + BitBucket	X		X	X	X
MikTeX					X
TeXMaker					X
Astah					X
Balsamiq Mockups	X				
VersionOne	X		X	X	X

Tabla 3.1: Herramientas y tecnologías utilizadas en cada parte del proyecto

---

## **Técnicas y herramientas**

---

En este apartado se van a comentar las distintas técnicas y herramientas utilizadas en el desarrollo del proyecto.

### **4.1. Visión por computador ~ Detección de movimiento**

La visión por computador, también llamada visión artificial, es la parte de la ciencia de la computación orientada a la recogida y al tratamiento de imágenes y vídeos [?].

Por la tipología de los datos con los que se trabaja en la visión por computador el procesamiento de éstos es muy costoso, este elevado coste computacional es causado en gran parte por la calidad de las imágenes y en el caso de los vídeos, además de la calidad de la imagen, por la cantidad de fotogramas por segundo con el que se ha grabado éste. **¿AÑADIR AQUI QUE POR ESO SE HA USADO GAMMA (TENDRÍA QUE EXPLICAR ANTES QUE ES GAMMA)?**

Una de las principales funciones que tiene la visión artificial es la detección de movimiento, la cual consiste en primero detectar la posición de cada una de las partes del cuerpo (depende del propio algoritmo la división que se quiera realizar sobre el cuerpo) para posteriormente realizar un seguimiento de estos elementos.

Existen distintos algoritmos de detección de movimiento, cada uno diferenciado por el uso de distintos lenguajes de programación y diferentes métodos de inteligencia artificial con los que entrenar el modelo. Como en este

proyecto se ha querido trabajar en *Python* se han investigado los siguientes algoritmos que están implementados en este lenguaje de programación.

## Detectron

Detectron es un proyecto de inteligencia artificial de *Facebook* centrado en la detección de objetos y personas con el uso de algoritmos de *deep learning* [3]. El proyecto se apoya en una de las librerías de inteligencia artificial más usadas en *Python*, *PyTorch*.

El proyecto cuenta ya con una segunda versión llamada Detectron2, figura 4.2, en la cual se han mejorado y añadido más modelos [8].



Figura 4.2: Logo de Detectron2.

## PoseNet

PoseNet es un proyecto de *Google* orientado únicamente al seguimiento del movimiento de las personas centrando su análisis en los puntos claves del cuerpo humano. El proyecto se basa en la librería de inteligencia artificial de la propia compañía *Google* llamada *TensorFlow*, figura 4.3 [1, 6].



Figura 4.3: Logo de TensorFlow.

#### *4.1. VISIÓN POR COMPUTADOR ~ DETECCIÓN DE MOVIMIENTO*

##### **TF-Pose-Estimator**



---

# **Aspectos relevantes del desarrollo del proyecto**

---

En este apartado se va a comentar, a manera de resumen temporal, el desarrollo del proyecto. Es en este apartado donde se comentarán las opciones y decisiones tomadas, los problemas surgidos y todos los aspectos importantes. Se ha considerado que la mejor forma de organizar este apartado es en secciones donde se comenta cada apartado del desarrollo.

## **5.1. Desarrollo FIS-HUBU**

FIS-HUBU es una aplicación web que permite realizar vídeo llamadas entre responsables y pacientes con Parkinson para realizar rehabilitaciones de manera *online*, es decir, sin la necesidad de desplazarse hasta la consulta o el hospital. Esta aplicación, que ha sido desarrollada junto con mi compañero José Luis Garrido Labrador, permite por parte del responsable observar y evaluar la evolución del estado de un paciente, esta evolución también es visible para el paciente que puede ver su propio progreso.

La aplicación puede dividirse en distintas partes que serán comentadas a continuación.

### **Vídeo llamadas**

El punto principal de la aplicación, y por ende su principal uso, son las vídeo llamadas entre pacientes y responsables o terapeutas que permitan sustituir la rehabilitaciones presenciales en consulta por rehabilitaciones *online*, permitiendo así que estás se puedan dar más a menudo y puedan

ser accesibles para un mayor número de personas, sobre todo para aquellos pacientes que no se pueden desplazar.

Primero se realizó una investigación sobre las principales plataformas de vídeo llamadas. Lo que se estaba buscando de estas plataformas era:

- Creación de llamadas de manera sencilla y automática. Si es posible a partir de url.
- Vídeo llamada estable sin necesidad de una gran conexión.
- Plataforma que permita grabar la cámara de los pacientes.
- Plataforma gratuita.

Dentro de las plataformas que se investigaron están las más conocidas aplicaciones de este tipo como puede ser *Skype*, pero al final se decidió utilizar *Jitsi* ya que proporciona todas las necesidades anteriormente comentadas, y además permite en un futuro poder crear un servidor propio donde poder modificar parámetros como la calidad de las llamadas.

## Responsable

Parte de la aplicación donde los responsables pueden realizar las siguientes tareas:

- Iniciar un vídeo llamada con un paciente.
- Evaluar la evolución de los pacientes.
- Modificar las evaluaciones de los pacientes.
- Comprobar la evolución de los pacientes.

La interfaz de la aplicación para tipo de usuario es sencilla y clara, como se puede ver en el menú principal, figura 5.4, o en el menú de estadísticas, figuras 5.5 y 5.6.



Figura 5.4: Menú principal de un responsable.



Figura 5.5: Menú de estadísticas.

## Paciente

Los pacientes que van a utilizar la aplicación, son pacientes mayores con Parkinson. Esto se ha tenido muy en cuenta tanto en el diseño como en la creación de la parte de la aplicación orientada en los pacientes, se ha intentado que esta parte sea lo más accesible posible, para que todos los pacientes puedan manejarse bien con la aplicación y así poder sacarle el mayor provecho.

Para poder realizar una aplicación lo más accesible posible primero se ha de saber la forma que van a tener los pacientes de interactuar con ésta. En este caso los pacientes van a utilizar un mando de SNES<sup>2</sup> con botones de colores, es por ello que se ha aprovechado estos colores para poder mostrar en la interfaz de la aplicación que botón han de pulsar para realizar que acción. Además, se ha creado un botón de ayuda que carga una página donde se puede ver la acción que realiza cada botón del mando. Un ejemplo de la interfaz de esta aplicación se puede ver en la figura 5.7.

---

<sup>2</sup>SNES: *Super Nintendo Entertainment System*



Figura 5.6: Ejemplo de la evolución de un paciente vista por un responsable.

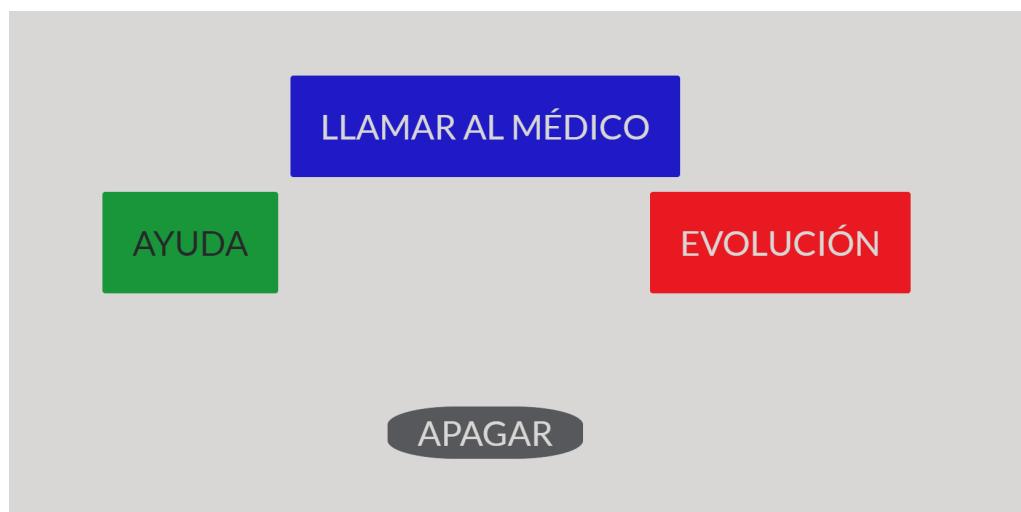


Figura 5.7: Menú principal de los pacientes.

## Dispositivos necesarios

Para poder utilizar la aplicación se necesitan distintos dispositivos, que dependiendo del tipo de usuario (responsable o paciente) son unos u otros.

Para los responsables lo único que se necesita es un ordenador con conexión a *Internet* y una cámara conectada a éste.

Por otro lado, los pacientes se presupone que no disponen de ningún dispositivo capaz de cargar la aplicación, es por ello que a cada paciente se le proporciona:

- MSI
- CAMARA

## Conexión

Como ya se ha comentado, el principal uso de la aplicación es las rehabilitaciones *online* para pacientes, mayoritariamente de tercera edad, que no se pueden desplazar a las consultas. Muchas de estas personas mayores viven en lugares donde no se tiene contratada ninguna línea de *Internet*, es por ello que además del desarrollo de la aplicación se contrató una serie de *routers 4G* para poder proporcionar conexión a los dispositivos necesarios.

## 5.2. Investigación algoritmos de visión por computador

Tras haber desarrollado la versión inicial, donde en un futuro se quiere añadir lo resultados de este proyecto, se pasó a realizar la primera investigación sobre los distintos algoritmos de visión por computar capaces de detectar y seguir los movimientos de una persona.

De estos algoritmos se necesita:

- Posibilidad de cargar un modelo existente o crear un modelo capaz de detectar a la persona que sale en la imagen.
- Posibilidad de cargar un modelo existente o crear un modelo capaz de detectar la posición de la persona.
- Que el procesado de nuevos fotogramas para detectar a la persona y su posición se realice en poco tiempo.
- Que la salida del procesado de los fotogramas pueda servir para una posterior comparación con el ejercicio base.

Teniendo todos estos factores en cuenta se estudió qué herramientas se pueden usar para esta fase del trabajo. Se buscó herramientas programadas

en *Python* para poder conectarse bien con el resto del proyecto y porque es uno de los lenguajes con los cuales se tiene más soltura, tanto por parte del alumno como por parte de los tutores para resolver dudas y ayudar en los problemas. Las herramientas encontradas fuera:

- *TF-Pose-Estimator.*
- *PoseNet.*
- *Detectron2.*

De cada una de estas herramientas se realizó una investigación y experimentación para probar si cumplían las necesidades requeridas. Al finalizar esta etapa, la única herramienta que permitía realizar todas las necesidades era *Detectron2*. Además, permite con un modelo ya creado por los propios desarrolladores realizar las tareas de predicción de elementos y de la posición de la persona.

Cabe destacar uno de los grandes problemas que surgió en esta fase, y fue justamente con *Detectron2*, la herramienta elegida. El problema era que las versiones de *CUDA* y de *PyTorch* no eran compatible. El problema se agrandó al estar trabajando en un *workstation* de la universidad como es *Gamma* que utilizan otros investigadores y también por la compleja estructura del propio computador. Tras hablar con el administrador de la computadora, que es uno de los tutores de este trabajo, el doctor Álvar Arnaiz Gonzalez, se pudo arreglar el problema actualizando la versión de los *drivers* de *CUDA*, y una vez actualizados descargando la versión compatible de *PyTorch*.

### 5.3. Investigación de *Detectron2*

La fase de investigación de *Detectron2* fue una de las más importantes, y por ende, más costosas en tiempo. Fue en esta fase donde se investigaron las distintas formas que tiene la herramienta para crear o importar modelos con los que poder trabajar. Una vez se seleccionó el modelo se realizó otro estudio para ver que configuración era la más correcta en el problema tratado.

Al ser una de las fases en las que más tiempo se ha dedicado, también es una de las fases donde surgieron más problemas tanto en la creación de los modelos, como en la visualización y el almacenamiento de los resultados obtenidos. Todos estos problemas serán comentados en este apartado.

## Selección del modelo

Tras haber elegido a *Detectron2* como herramienta de visión por computador para capturar detectar a los pacientes y obtener de estos sus posiciones, elegir qué modelo, dentro de las posibilidades de la herramienta, utilizar para realizar estas tareas.

*Detectron2* permite dos formas de obtener un modelo:

- Crear un modelo propio a partir de una gran cantidad de datos.
- Importación de modelos ya entrenados.

Ante la falta de datos para la creación de un modelo que diese buenos resultados, y al comprobar en la fase anterior que con la importación de modelos existentes se podían obtener buenos resultados la investigación se centró en los modelos ya existentes realizados por los creados de la herramienta. Estos modelos de redes neuronales fueron entrenados en servidores *Big Basin* de *Facebook*, sucesores de los servidores *Big Sur*, ambos orientados al uso de arquitecturas con varias GPUs potentes para el entrenamiento y uso de algoritmos de inteligencia artificial [2]. En concreto, estos modelos fueron entrenados con 8 Nvidia Tesla V100 con *NVLink*, una tecnología que mejorar las interconexiones entre GPUs proporcionando mayor ancho de banda, haciendo el sistema más escalable [5]. La mejora con otras generaciones de comunicación *inter-GPU* se puede ver en la figura 5.8.

Los modelos creados en *Detectron2* se diferencian en los siguientes tipos:

- COCO Detection with Faster R-CNN (Figura 5.9): modelos que predice los objetos y personas de la imagen.
- COCO Detection with RetinaNet (Figura 5.10): modelos que detecta objetos y persona. Como se puede ver en la imagen de ejemplo la interpretación no es buena aun utilizando un *threshold* alto.
- COCO Detection with RPN and Fast R-CNN: modelos que no se han conseguido probar ya que no siguen la misma estructura que el resto de modelos.
- COCO Instance Segmentation Baselines with Mask R-CNN (Figura 5.11): modelos de detección de objetos y persona, con delimitación en el área ocupada.

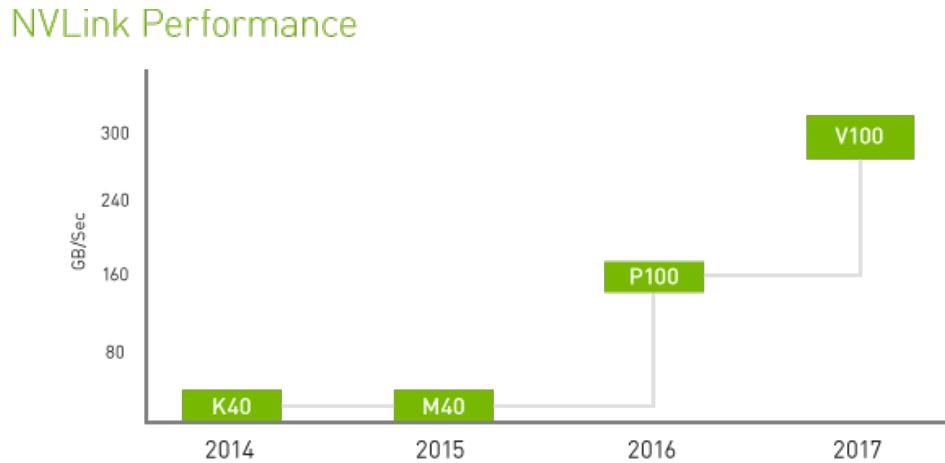


Figura 5.8: Mejora del rendimiento en GB/s con NVlink [5].

- COCO Person Keypoint Detection Baselines with Keypoint R-CNN (Figura 5.12): modelos que detectan a las posiciones y unos puntos claves de ellas, estos puntos son nariz, ojos, orejas, hombros, codos, muñecas, cadera, rodilla y tobillos. Como se puede ver en la imagen de ejemplo, estos modelos predicen bastante bien la posición pero a veces detectan otras cosas como personas, por lo que se posteriormente se vio necesario un estudio sobre el *threshold*.
- COCO Panoptic Segmentation Baselines with Panoptic FPN (Figura 5.13): modelos que segmentación como COCO Instance Segmentation Baselines with Mask R-CNN, obtienen resultados muy parecidos.
- LVIS Instance Segmentation Baselines with Mask R-CNN (Figura 5.14): modelos orientados a la detección de objetos, necesitan un *threshold* bajo para detectar diversos objetos.
- Por último se encuentran modelos como Cityscapes & Pascal VOC Baselines, otras configuraciones y algoritmos de la primera versión de la herramienta *Detectron*. Estos modelos son muy parecidos a los ya mostrados.

Con el estudio de todos los modelos existentes en *Detectron2* se ha comprobado como los únicos modelos que sirven para el problema planteado



Figura 5.9: Modelo de tipo COCO Detection with Faster R-CNN, faster\_rcnn\_R\_50\_C4\_1x.

son COCO Person Keypoint Detection Baselines with Keypoint R-CNN. Dentro de este tipo de modelos existen un total de 4 modelos distintos, para la elección del mejor de estos modelos se realizó otro estudio esta vez comprobando los tiempos de ejecución, ya que todos los modelos de este tipo realizan unas buenas predicciones (principalmente porque el problema es sencillo, al tener a la persona en el centro de la imagen con el mínimo de objetos de por medio).

Para realizar este estudio se analizó el tiempo de procesado e impresión de varios vídeos con cada uno de los modelos. Para ello se calculó estos valores para los 4 posibles modelos con un total de 7 vídeos. De este estudio se obtuvieron los resultados que se pueden ver en la tabla 5.2.

Modelo	T. Carga (s)	T. Procesamiento (s)	N. Fotogramas	Frecuencia
keypoint_rcnn_R_50_FPN_3x	9.783479	236.774504	1989	0.119042
keypoint_rcnn_R_50_FPN_1x	8.562254	237.289880	1989	0.119301
keypoint_rcnn_R_101_FPN_3x	13.017239	279.327051	1989	0.140436
keypoint_rcnn_X_101_32x8d_FPN_3x	18.106832	422.024270	1989	0.212179

Tabla 5.2: Tabla con el estudio de los modelos de posición ordenado por ratio.

Como se pueden observar, los datos obtenidos para los modelos son muy parecidos, sobre todo para los dos con mejor frecuencia. Tras observar

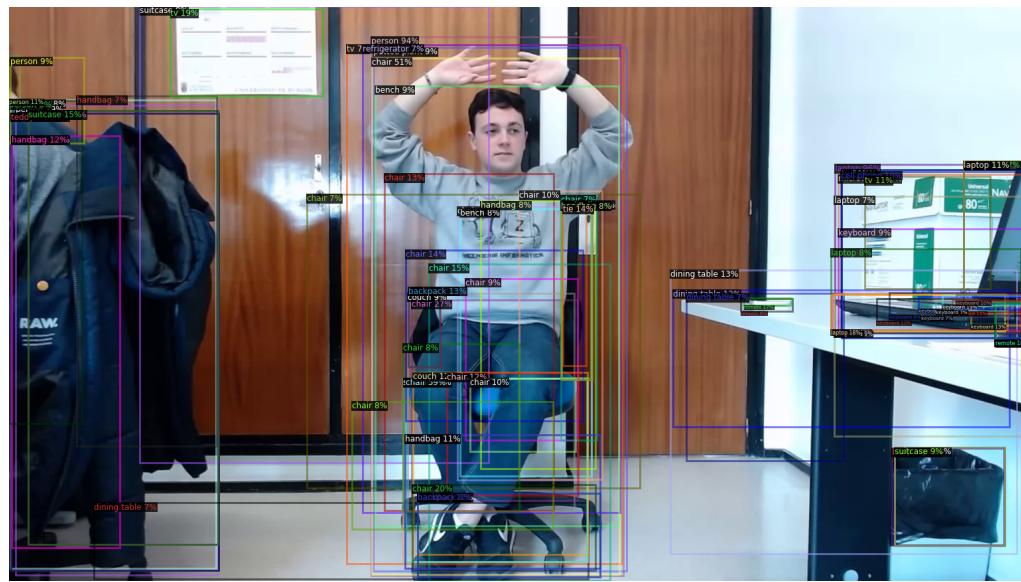


Figura 5.10: Modelo de tipo COCO Detection with RetinaNet, retina-net\_R\_101\_FPN\_3x.

estos datos se decidió elegir al modelo keypoint\_rcnn\_R\_50\_FPN\_3x (es además el modelo con el que se hicieron las primera pruebas y con el que se decidió a *Detectron2* como herramienta para detectar el movimiento) para utilizarlo en el resto del proyecto. Además, en este estudio se ha podido comprobar que los modelos trabajan correctamente sea cual sea el tipo de ropa que lleve la persona o si pasa un objeto por medio de la imagen (normalmente la posición es correcta, pero como se comentará más adelante a veces algunos fotogramas no son correctos), como se pueden ver en la figuras 5.15 y 5.16.

### Definición del *threshold*

Como se ha comentado en el apartado anterior, es necesario definir un *threshold*, que es un valor entre 0 y 1 que define la sensibilidad del modelo al detectar, es decir, con valores cercanos a 0 el modelo tiende a detectar más elementos (muchos de ellos erróneos) y cuanto más cercano a 1 menos sensible es, solo detectando los elementos más claros. Sobre este valor se ha realizado un estudio de posibles valores, estos valores probados han sido 0.3, 0.5, 0.75 y 0.99.

Con valores bajos del *threshold* se observa que se detectan cosas que no son personas, o se detectan personas que no salen completas en la imagen,



Figura 5.11: Modelo de tipo COCO Instance Segmentation Baselines with Mask R-CNN, mask\_rcnn\_R\_50\_DC5\_3x.

como se puede observar en el figura 5.17. Esto ocurre con los valores 0.3, 0.5 y 0.75 que dan la misma salida. Sin embargo, con el valor 0.99, como se puede ver en la figura 5.18, se obtienen muy buenos resultados detectando únicamente a la persona que aparece en el centro de la imagen, con la posición exacta en todos los puntos.

## Problemas surgidos

Como se ha comentado a lo largo del apartado, han surgido distintos problemas que se pueden resumir en:

- Problema con los modelos COCO Detection with RPN and Fast R-CNN que al no tener la misma estructura que el resto de modelos de Detectron2 no se ha podido probar.
- Se intentó almacenar los vídeos procesados pero por problemas con OpenCV no se pudo guardar en esta etapa.



Figura 5.12: Modelo de tipo COCO Person Keypoint Detection Baselines with Keypoint R-CNN, keypoint\_rcnn\_R\_101\_FPN\_3x.

## 5.4. Cálculo de características

Una vez se seleccionó el modelo que se iba a utilizar durante todo el proyecto y se determinó el parámetro *threshold* de los modelos, el siguiente paso es la interpretación de las salidas del modelo. Y después de haber interpretado con el modelo de *Detectron2* una posición ser capaces de calcular características que definan de la mejor manera los datos.

### Interpretación de las predicciones

Si se analiza la salida tras predecir un imagen con el predictor creado (DefaultPredictor) se observa que devuelve un diccionario con una única entrada llamada “instances” donde se almacena toda la información. Dentro de la instancia existen los siguientes apartados:

- *pred\_boxes*: Boxes, estructura propia de *Dectectron2* donde se almacenan límites de donde entiende que está la persona en un *tensor*, si se ha detectado más de una persona este Boxes estará creado por varios *tensors*.
- *scores*: *tensor* con las probabilidades de que los elementos sean personas. Es este el valor que se compara con el *threshold* para ver si se toma o



Figura 5.13: Modelo de tipo COCO Panoptic Segmentation Baselines with Panoptic FPN, panoptic\_fpn\_R\_101\_3x.

no como una persona. Si existe más de un valor estos están ordenados de mayor a menos, este es el orden que se sigue en el resto de valores.

- *pred\_classes*: *tensor* con el índice de la clase, en este caso todos los elementos son 0 ya que solo detecta personas, que se identifican con este índice.
- *pred\_keypoints*: *tensor* con todos los puntos clave de cada elemento (persona) detectada. De cada elemento detectado, si la predicción ha sido correcta, se obtienen un total de 17 puntos, con sus valores x e y. Después de investigar el posicionamiento de estos puntos se puede decir, como se ve en la figura 5.19, que los puntos representan (teniendo en cuenta que se han detectado todos los puntos y que la persona está mirando de frente al objetivo):
  - 0: nariz.
  - 1 y 2: ojo izquierdo y derecho.
  - 3 y 4: oreja izquierda y derecha.
  - 5 y 6: hombro izquierdo y derecho.
  - 7 y 8: codo izquierdo y derecho.
  - 9 y 10: muñeca izquierda y derecha.



Figura 5.14: Modelo de tipo LVIS Instance Segmentation Baselines with Mask R-CNN, mask\_rcnn\_X\_101\_32x8d\_FPN\_1x.

- 11 y 12: parte izquierda de la cadera y derecha.
- 13 y 14: rodilla izquierda y derecha.
- 15 y 16: tobillo izquierdo y derecho.

## Clase de posición

Una vez se conoce la salida de las predicciones se puede usar esta para obtener más información de la posición de la persona, que en futuras etapas fue usada para la comparación de posiciones. Por ello se creó una clase que almacena todos los puntos detectados además de realizar cálculos para extraer más características.

Para poder obtener más información sobre los puntos se crearon una serie de funciones que permiten el cálculo de:

- Cálculo de puntos medios. Este cálculo se realiza para obtener el punto inferior del cuello que se calcula como el punto medio de la recta que une los dos hombros y se utiliza también para calcular el punto central de la cadera a partir de su punto izquierdo y derecho.



Figura 5.15: Prueba con chaqueta.

- Cálculo de la distancia entre puntos. Este punto se ha realiza sobre todos los pares de puntos unidos. Se pueden utilizar para intuir profundidades.
- Cálculo de los ángulos en grados. Este cálculo se ha realizado en todas las combinaciones posibles (conjunto de 3 puntos consecutivos), ya que se cree que son las características que más información van a aportar debido a que no dependen de ningún otro valor como puede ser la profundidad a la que está el paciente, su altura...

Como se comentó en apartados anteriores, una vez se tienen los puntos se ha de realizar una serie de comprobaciones para saber si las posiciones son válidas:

- Si existe más de una persona detectada se selecciona la primera, ya que es la que tiene un mayor *score* y por lo tanto el modelo cree que puede ser con mayor probabilidad una persona.
- Si una predicción no dispone de todos los puntos no se tiene en cuenta. En la práctica con las pruebas que se han realizado esto solo pasa cuando se pasa un objeto por ciertos puntos del cuerpo, cuando ocurre estos hay algunos fotogramas, muy pocos, que no detectan todos los puntos.

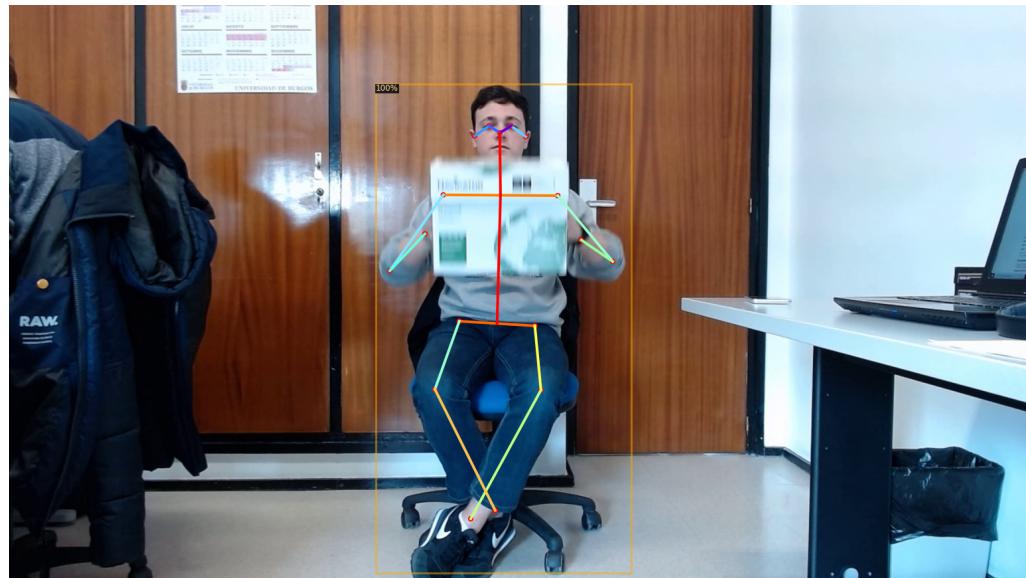


Figura 5.16: Prueba con un objeto de por medio.

- Se compara el fotograma actual con un conjunto de fotogramas anteriores, si la diferencia entre ellos son muy grandes se descarta el fotograma ya que se entiende que el modelo no ha procesado bien la imagen. Este suceso pasa también cuando se pasa un objeto por alguna parte del cuerpo, ya que cuando esto ocurre a veces algunos fotogramas dan predicciones erróneas que se pueden detectar con una simple comparación con una ventana deslizante de posiciones.



Figura 5.17: Prueba con *threshold* a 0.3.



Figura 5.18: Prueba con *threshold* a 0.99.

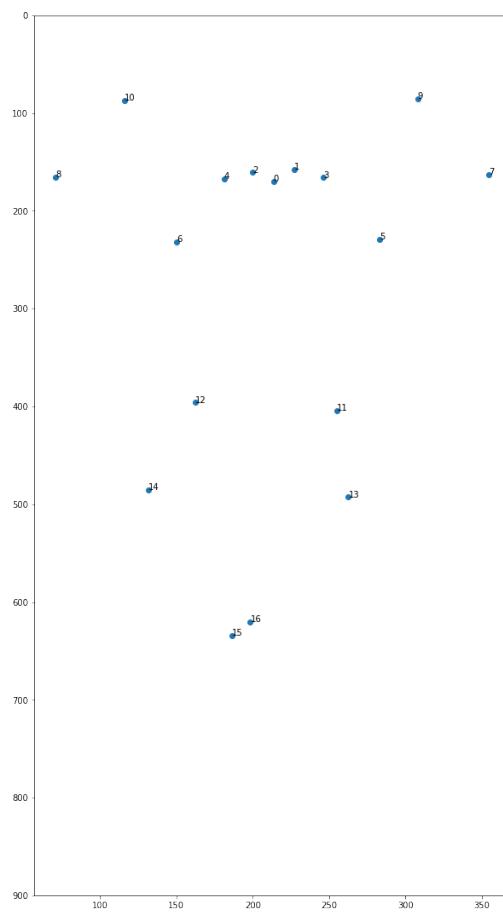


Figura 5.19: Puntos clave predichos con el modelo junto con la etiqueta con su orden.

---

## **Trabajos relacionados**

---

Este apartado sería parecido a un estado del arte de una tesis o tesina. En un trabajo final de máster no parece tan obligada su presencia, aunque se puede dejar a juicio del tutor el incluir un pequeño resumen comentado de los trabajos y proyectos ya realizados en el campo del proyecto en curso.



---

## **Conclusiones y Líneas de trabajo futuras**

---

Todo proyecto debe incluir las conclusiones que se derivan de su desarrollo. Éstas pueden ser de diferente índole, dependiendo de la tipología del proyecto, pero normalmente van a estar presentes un conjunto de conclusiones relacionadas con los resultados del proyecto y un conjunto de conclusiones técnicas. Además, resulta muy útil realizar un informe crítico indicando cómo se puede mejorar el proyecto, o cómo se puede continuar trabajando en la línea del proyecto realizado.



# Apéndices



## *Apéndice A*

---

# **Plan de Proyecto Software**

---

## **A.1. Introducción**

La planificación de un proyecto es una fase esencial en desarrollo de éste, ya que permite comprobar y guiar el proyecto según las necesidades actuales y futuras.

Dentro de este plan para el desarrollo del proyecto se pueden diferenciar dos puntos sobre los que se puede enfocar este estudio:

- **Temporal:** enfoque en el que se analiza la evolución del proyecto en el tiempo. Al utilizar una metodología SCRUM, se ha dividido el desarrollo en *sprints* de 2 semanas.
- **Viabilidad:** comprobación de la adecuación del proyecto tanto desde el ámbito económico como desde el ámbito legal.

## **A.2. Planificación temporal**

Como ya se ha comentado, la planificación temporal se ha dividido en distintos *sprints* de 2 semanas cada uno, en los cuales se realizaron distintas tareas y reuniones. Cabe destacar que antes de realizar el primer *sprint* se desarrolló la aplicación FIS sobre la cual será implementado en el futuro el proyecto desarrollado.

## Sprint0

En este *sprint* se creó la aplicación para el Hospital Universitario de Burgos llamada FIS-HUBU. Esta aplicación, como se ha visto en el apartado 5.1, sirve para realizar vídeo llamadas entre médicos y los pacientes con Parkinson en las cuales se realizan rehabilitaciones para mejorar su calidad de vida. Como se ha explicado, esta aplicación cuenta con dos partes, una para los médicos o responsables y otra para los pacientes. Las tareas realizadas en este *sprint* son las siguientes:

- Investigación sobre plataformas de vídeo llamadas.
- Creación de la estructura base de la página web.
- Diseño e implementación de la base de datos.
- Creación del login por usuario, guardando una *cookie* para mantener la sesión.
- *Backend* de las páginas de los pacientes, donde se incluye la creación de las llamadas, el cálculo de la evolución a partir de los datos en la base de datos, la grabación de la cámara del paciente y subida a un servidor propio de la universidad...
- *Backend* de las páginas de los responsables.

Tras desarrollar la aplicación se empezaron con las tareas de explotación de la aplicación, en donde se cogieron los dispositivos para los pacientes (AQUI ESCRIBIR EL MODELO DEL BICHO Y LA CAMARA) y se les instalaron todo el *software* necesario para funcionar correctamente, y con ello poder grabar los vídeos y subirlos a un servidor propio. Debido a la aparición del COVID-19 y el posterior confinamiento sufrido y las consiguientes medidas de seguridad no se pudo empezar a instalar los dispositivos en las casas de los pacientes.

## Sprint1: 17/02/2020 - 26/02/2020

Este es el primer *sprint* real del desarrollo de la parte del proyecto de visión artificial, en el se realizaron las primeras tareas de creación del repositorio en *GitHub* y la creación de la estructura del mismo. Además, se creó el documento *LaTeX* a partir de la plantilla de la asignatura. Las tareas más relevantes en este primer *sprint* fueron las relacionadas con la configuración de *Gamma* y los primeros pasos en la investigación de

algoritmos de visión por computador y detección de movimiento, como son *Detectron2* y *PoseNet*.

### ***Sprint2: 27/02/2020 - 11/03/2020***

En este *sprint* se realizó una tarea fundamental dentro del desarrollo del proyecto, que es la investigación y elección del algoritmo de visión artificial para la obtención de la postura, que se usará en los fotogramas de los vídeos recogidos con la aplicación FIS-HUBU y compararlos con otros vídeos base para saber la exactitud del ejercicio. Esta tarea de investigación de los distintos algoritmos dio como resultado que el *Dectectron2* (apartado 4.1) es el algoritmo que más se amoldaba al problema del proyecto.

Además, en este *sprint* se realizaron las tareas de documentación necesarias sobre los algoritmos candidatos y sobre la selección del mejor.

### ***Sprint3: 12/03/2020 - 17/04/2020***

En este tercer *sprint* se profundizó en la investigación de los distintos algoritmos y modelos de *Detectron2*. En esta investigación se encontró un modelo ya entrenado capaz de obtener la posición de la persona fotograma a fotograma de una forma extraordinariamente precisa.

Una vez se encontró el modelo se pasó a interpretar la salida del modelo, para poder ajustarlo (principalmente para que solo detecte la persona que mejor identifica, normalmente el centro de la pantalla) y obtener datos para su posterior tratamiento y comparación.

Por último, en este *sprint* se realizaron diversas tareas de documentación.

## **A.3. Estudio de viabilidad**

### **Viabilidad económica**

### **Viabilidad legal**



*Apéndice B*

---

## **Especificación de Requisitos**

---

- B.1. Introducción
- B.2. Objetivos generales
- B.3. Catalogo de requisitos
- B.4. Especificación de requisitos



*Apéndice C*

---

## **Especificación de diseño**

---

- C.1. Introducción
- C.2. Diseño de datos
- C.3. Diseño procedimental
- C.4. Diseño arquitectónico



*Apéndice D*

---

## **Documentación técnica de programación**

---

- D.1. Introducción**
- D.2. Estructura de directorios**
- D.3. Manual del programador**
- D.4. Compilación, instalación y ejecución  
del proyecto**
- D.5. Pruebas del sistema**



*Apéndice E*

---

## **Documentación de usuario**

---

- E.1. Introducción**
- E.2. Requisitos de usuarios**
- E.3. Instalación**
- E.4. Manual del usuario**



---

## Bibliografía

---

- [1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.
- [2] Facebook Engineering. Introducing big basin: Our next-generation ai hardware. <https://engineering.fb.com/data-center-engineering/introducing-big-basin-our-next-generation-ai-hardware/>, mar 2017.
- [3] Ross Girshick, Ilija Radosavovic, Georgia Gkioxari, Piotr Dollár, and Kaiming He. Detectron. <https://github.com/facebookresearch/detectron>, 2018.
- [4] John R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, 1992.
- [5] Nvidia. Nvlink y nvswitch: Los elementos fundamentales de la comunicación multi-gpu avanzada. <https://www.nvidia.com/es-es/data-center/nvlink/>.

- [6] TensorFlow. Pose estimation. [https://www.tensorflow.org/lite/models/pose\\_estimation/overview](https://www.tensorflow.org/lite/models/pose_estimation/overview), feb 2020.
- [7] Wikipedia. Latex — wikipedia, la enciclopedia libre, 2015. [Internet; descargado 30-septiembre-2015].
- [8] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. <https://github.com/facebookresearch/detectron2>, 2019.