



Universidad de Jaén

Escuela Politécnica Superior de Jaén

MÉTODOS DE APRENDIZAJE PROFUNDO PARA LA DETECCIÓN DE POSES EN ANIMALES

Autor: Álvaro de Castro Escribano

Grado: Grado en Ingeniería Informática

Director: Antonio Jesús Rivera Rivas
María Dolores Pérez Godoy

Departamento del director: Informática

Fecha: junio de 2024

Licencia CC

CREA



UNIVERSIDAD DE JAÉN

D./D^a Antonio Jesús Rivera Rivas y D./D^a María Dolores Pérez Godoy, tutor(es) del Trabajo Fin de Grado titulado: **Métodos de aprendizaje profundo para la detección de pose en animales**, que presenta Álvaro de Castro Escribano, autoriza(n) su presentación para defensa y evaluación en la Escuela Politécnica Superior de Jaén.

Jaén, junio de 2024

El estudiante

Los tutores

Firmado por DE
CASTRO
ESCRIBANO
ALVARO -
77387041E el
día 25/06/2024
con un
certificado
emitido por AC

Álvaro de Castro Escribano

Antonio
Jesús
Rivera
Rivas
Fecha:
2024.06.
25
09:18:40
+02'00'

Antonio Jesús Rivera
Rivas

María Dolores
Pérez Godoy
Fecha:
2024.06.
25
09:59:04
+02'00'

María Dolores Pérez
Godoy

Agradecimientos

En primer lugar agradecer a Antonio Jesús Rivera Rivas y María Dolores Pérez Godoy por su gran implicación y ayuda en todo momento con este trabajo. Todos los materiales, recursos, conocimientos y consejos que han hecho posible este trabajo, y me han permitido desarrollarme no solo en lo educativo, pero también en lo profesional y en lo personal.

Agradecer también al grupo de investigación SIMIDAT y el equipo detrás que me han facilitado lo necesario para llevar a cabo esta investigación de la forma más fluida posible.

No me olvido de María José Fernández Abad y María Concepción Paredes Olay por confiar en nosotros la idea original que ha dado vida a este trabajo.

A mi familia por aportarme todo lo que necesito para estar dónde estoy, y estaré.

A mis compañeros de clase, las infinitas risas y momentos que han hecho que estos cuatro años hayan sido tan especiales.

Porque este trabajo marca el fin de una etapa, pero esto solo acaba de empezar. Por mirar hacia atrás y ver lo que hemos logrado. Por mirar hacia adelante y ver lo que lograremos.

Gracias a todos los que estuvieron, a los que están y a los que estarán.

Tabla de contenidos

1. INTRODUCCIÓN	1
1.1. Motivación	1
1.2. Objetivos y metodología	2
1.3. Estructura del documento	4
1.4. Glosario de términos	5
2. ANTECEDENTES	7
2.1. Inteligencia Artificial	7
2.2. Aprendizaje automático	9
2.2.1. Aprendizaje supervisado	10
2.2.2. Aprendizaje no supervisado	11
2.3. Aprendizaje profundo	11
2.3.1. Redes neuronales	12
2.4. Visión por computador	13
2.5. Estimación de poses	16
2.5.1. Aplicaciones de la estimación de poses	17
2.5.2. Estimación de poses en personas	19
2.5.3. Estimación de poses en animales	19

2.6. Recursos existentes	21
2.6.1. Tierpsy-Tracker y Open Worm Movement Database	21
2.6.2. WormPose	24
2.7. Modelos y arquitecturas existentes	25
2.7.1. Tensorflow Lite	25
2.7.2. YOLOv8	26
2.8. Conjuntos de datos existentes	27
3. OBJETIVOS	31
3.1. Objetivo general	31
3.2. Objetivos específicos	31
4. MATERIALES Y MÉTODOS	35
4.1. Entorno de experimentación	35
4.1.1. Especificaciones del sistema	35
4.2. Lenguajes de programación	36
4.2.1. Python	36
4.2.2. Bash	37
4.2.3. XML	37
4.2.4. YAML	38
4.3. Modelos para la experimentación	38
4.3.1. Formato YAML del conjunto de datos	39
4.3.2. Ejecución del modelo	40
4.4. Anotaciones	43
4.4.1. Anotación de vídeos ejemplo	44

4.4.2. Exportar anotaciones desde la plataforma	47
4.4.3. Formato de las anotaciones escogidas	51
4.5. Técnica de aumento para el conjunto de datos	56
4.6. Software para el tratamiento de anotaciones	63
4.6.1. Descarga y extracción de frames	63
4.6.2. Creación del dataset en formato correcto	64
4.6.3. Separación en entrenamiento, validación y test	65
4.6.4. Unión de varios conjuntos	66
4.6.5. Aumento del conjunto de datos	66
4.6.6. Visualización de anotaciones	66
4.6.7. Control de versiones	67
4.7. Creación de conjuntos de datos	68
4.7.1. Datasets utilizados	68
4.8. Estrategia de validación	70
4.8.1. Holdout validation	71
4.8.2. Validación externa	72
4.8.3. Comparación de los métodos de validación	73
4.9. Depuración de resultados y entrenamientos	75
4.9.1. Seguimiento de modelos y ejecuciones de entrenamiento	76
4.9.2. Visualización de resultados	79
4.10. Configuraciones de hiperparámetros	80
4.11. Métricas de validación	82
4.11.1. Intersección sobre unión (IoU)	83
4.11.2. Object Keypoint Similarity (OKS)	84

4.11.3. Precisión (Pr)	85
4.11.4. Recuperación (Re)	86
4.11.5. Precisión Media Promedio (mAP)	86
5. EXPERIMENTACIÓN Y RESULTADOS	87
5.1. Conjunto de datos sin aumentar	88
5.1.1. Modelo L	88
5.1.2. Modelo X	90
5.1.3. Modelo X-P6	91
5.2. Conjunto de datos aumentados	92
5.2.1. Modelo L	92
5.2.2. Modelo X	94
5.2.3. Modelo X-P6	96
5.3. Comparativa con la técnica de aumento de datos	97
5.3.1. Modelo L	97
5.3.2. Modelo X	99
5.4. Comparativa genérica de los mejores modelos	101
6. CONCLUSIONES	103
6.1. Posibles mejoras	105
6.2. Propuestas a futuro	108
6.2.1. Generación de datos sintéticos	108
Bibliografía	XI

Lista de figuras

1.1. Muestra del entorno de estudio de los sujetos	3
2.1. Diagrama representativo de la jerarquía del campo de conocimiento. Fuente: elaboración propia	8
2.2. Representación de la neurona artificial por McCulloch and Pitts, 1943. Fuente [1]	13
2.3. Representación de una ANN simple. Fuente [2]	14
2.4. Representación de una ANN profunda. Edición personal a partir de la fuente original [2]	14
2.5. Posibles puntos asociados a la estimación de pose humana	18
2.6. Posibles puntos asociados a la estimación de pose en un coche	19
2.7. Puntos asociados a la estimación de pose en el estudio	20
2.8. Fotograma individual original	23
2.9. Fotograma individual tras aplicar función umbral	23
4.1. Entrenamiento de una red <i>ResNet-50</i> , para 90 épocas. Servidor CPU: Dual Xeon E5-2699v4, 2.6GHz. Fuente [3]	36
4.2. Visualización de anotación con Roboflow	45
4.3. Visualización de la anotación frame 68. Anotación a mano.	46
4.4. Visualización de la anotación frame 247. Anotación interpolada.	47
4.5. Visualización de la anotación frame 547. Anotación a mano.	48

4.6. Comparación de anotación interpolada y manual	49
4.7. Formato <i>CVAT for video 1.1</i> de las anotaciones descargadas	51
4.8. Formato <i>Datamaro for video 1.0</i> de las anotaciones descargadas	51
4.9. Uso de la propiedad <i>keyframe</i> en la plataforma	55
4.10. Uso de la propiedad <i>outside</i> en la plataforma	55
4.11. Uso de la propiedad <i>occluded</i> en la plataforma	55
4.12. Posibles modificaciones para el proceso de aumento de datos. Fuente [4]	58
4.13. Imagen original a la que se le aplicarán las transformaciones	59
4.14. Comparación de la transformación cambio de perspectiva	59
4.15. Comparación de la transformación rotación afín.	60
4.16. Comparación de la transformación cambio de canal	60
4.17. Comparación de la transformación de modificación del valor hue y la saturación	60
4.18. Comparación de la transformación alternar valores de los canales RGB	61
4.19. Comparación de la transformación ecualizar el histograma de la imagen	61
4.20. Comparación de la transformación modificar el brillo y contraste de la imagen	61
4.21. Ejemplo 1 de transformación completa de la línea de aumento	62
4.22. Ejemplo 2 de transformación completa de la línea de aumento	62
4.23. Ejemplo 3 de transformación completa de la línea de aumento	63
4.24. Funcionamiento del visionado de anotaciones a partir de una imagen creada por técnicas de aumento de datos	67
4.25. Representación visual de la validación holdout. Fuente [5]	72
4.26. Datos de una serie temporal para ventas a lo largo del tiempo. Para extrapolarlo a nuestro problema deberíamos considerar las ventas como la posición del sujeto. Fuente [6]	74

4.27. Representación del conjunto de entrenamiento escogido de forma aleatoria. Fuente [6]	75
4.28. Pantalla principal presentada en un experimento de Comet. Métricas.	77
4.29. Algunos de los hiperparámetros mostrados por Comet	77
4.30. Ficheros registrados por Comet. Código fuente usado.	78
4.31. Salida por consola registrada por Comet.	78
4.32. Paquetes Python instalados registrados por Comet	79
4.33. Métricas del sistema registradas por Comet	80
4.34. Opción de reproducir experimento en Comet	80
4.35. Posibilidades para añadir un panel en Comet [7]	81
4.36. Representación de la intersección sobre la unión	84
4.37. Cálculo del IoU con umbral a 0.5	85
6.1. Anotaciones transformadas de la imagen ejemplo	107
6.2. Anotaciones transformadas de la imagen ejemplo y su corrección	107
6.3. Representación matemática de las distintas posturas de un gusano	109
6.4. Funcionamiento de la generación de imágenes sintéticas	109

Lista de tablas

1.1. Glosario de términos usados	6
2.1. Medidas de rendimiento de los modelos proporcionados para Tensor-Flow Lite. Fuente [8]	26
2.2. Características	27
2.3. Algunos de los conjuntos de datos públicos usados para la estimación de poses en animales en la actualidad. Fuente original modificada [9]	29
4.1. Descripción de los posibles argumentos en el entrenamiento del modelo	43
4.2. dataset rawDataset	69
4.3. dataset mainDataset	70
4.4. dataset wormTesting1-2. Solo frames concretos.	70
5.1. Métricas para distintas configuraciones de YoloV8l para 50 epochs. Datos sin aumentar. Validación del conjunto interno.	88
5.2. Métricas para distintas configuraciones de YoloV8l para 50 epochs. Datos sin aumentar. Validación del conjunto externo.	89
5.3. Métricas para distintas configuraciones de YoloV8l para 100 epochs. Datos sin aumentar. Validación del conjunto interno.	89
5.4. Métricas para distintas configuraciones de YoloV8l para 100 epochs. Datos sin aumentar. Validación del conjunto externo.	89

5.5. Métricas para distintas configuraciones de YoloV8x para 50 epochs. Datos sin aumentar. Validación del conjunto interno.	90
5.6. Métricas para distintas configuraciones de YoloV8x para 50 epochs. Datos sin aumentar. Validación del conjunto externo.	90
5.7. Métricas para distintas configuraciones de YoloV8x para 100 epochs. Datos sin aumentar. Validación del conjunto interno.	91
5.8. Métricas para distintas configuraciones de YoloV8x para 100 epochs. Datos sin aumentar. Validación del conjunto externo.	91
5.9. Métricas para distintas configuraciones de YoloV8xp6 para 100 epochs. Datos sin aumentar. Validación del conjunto externo.	92
5.10. Métricas para distintas configuraciones de YoloV8l para 50 epochs. Datos aumentados. Validación del conjunto interno.	93
5.11. Métricas para distintas configuraciones de YoloV8l para 50 epochs. Datos aumentados. Validación del conjunto externo.	93
5.12. Métricas para distintas configuraciones de YoloV8l para 100 epochs. Datos aumentados. Validación del conjunto interno.	93
5.13. Métricas para distintas configuraciones de YoloV8l para 100 epochs. Datos aumentados. Validación del conjunto externo.	94
5.14. Métricas para distintas configuraciones de YoloV8x para 50 epochs. Datos aumentados. Validación del conjunto interno.	94
5.15. Métricas para distintas configuraciones de YoloV8x para 50 epochs. Datos aumentados. Validación del conjunto externo.	94
5.16. Métricas para distintas configuraciones de YoloV8x para 100 epochs. Datos aumentados. Validación del conjunto interno.	95
5.17. Métricas para distintas configuraciones de YoloV8x para 100 epochs. Datos aumentados Validación del conjunto externo.	95
5.18. Métricas para distintas configuraciones de YoloV8xp6 para 50 epochs. Datos aumentados. Validación del conjunto externo. Falla la ejecución en la época 10.	96

5.19. Métricas para distintas configuraciones de YoloV8xp6 para 100 epochs. Datos aumentados. Validación del conjunto externo. Falla la ejecución en la época 10.	96
5.20. Comparativa de métricas para distintas configuraciones de YoloV8l para 50 epochs. Validación del conjunto interno.	97
5.21. Comparativa de métricas para distintas configuraciones de YoloV8l para 50 epochs. Validación del conjunto externo.	98
5.22. Comparativa de métricas para distintas configuraciones de YoloV8l para 100 epochs. Validación del conjunto interno.	98
5.23. Comparativa de métricas para distintas configuraciones de YoloV8l para 100 epochs. Validación del conjunto externo.	99
5.24. Comparativa de métricas para distintas configuraciones de YoloV8x para 50 epochs. Validación del conjunto interno.	99
5.25. Comparativa de métricas para distintas configuraciones de YoloV8x para 50 epochs. Validación del conjunto externo.	100
5.26. Comparativa de métricas para distintas configuraciones de YoloV8x para 100 epochs. Validación del conjunto interno.	100
5.27. Comparativa de métricas para distintas configuraciones de YoloV8x para 100 epochs. Validación del conjunto externo.	101
5.28. Comparación general de los mejores modelos obtenidos	101

Lista de listados de código

4.1. Formato del fichero de configuración del conjunto de datos	39
4.2. Formato anotaciones CVAT for video 1.1	47
4.3. Formato anotaciones Datumaro 1.0	49
4.4. Formato de las anotaciones 1	52
4.5. Formato de los metadatos del fichero de anotaciones	52
4.6. Etiquetas track del fichero de anotaciones	53
4.7. Formato de la anotación de caja delimitante	54
4.8. Formato de la anotación de punto	55
4.9. Transformaciones aplicadas en la línea de aumentación del conjunto de datos	57
4.10. Formato para la anotación de puntos en YoloV8. 2 dimensiones	65
4.11. Formato para la anotación de puntos en YoloV8. 3 dimensiones	65

Capítulo 1

INTRODUCCIÓN

En este trabajo, se abordará la tarea de detección de poses, concretamente en animales, haciendo uso de métodos de aprendizaje profundo.

1.1. Motivación

El planteamiento del presente trabajo surge a partir de un estudio dentro del departamento de psicología de la Universidad de Jaén. En este, se proporciona unos sujetos de análisis para los que se pide automatizar la detección de respuestas frente a cierto estímulo.

Los sujetos estudiados son lombrices, para las que se busca anotar la respuesta frente a un estímulo lumínico. Una vez el sujeto recibe este estímulo, podrá responder de diversas formas realizando movimientos diferentes, en función de otras condiciones propias del experimento realizado. Será esta respuesta realizada por los sujetos la que resulta de interés para la investigación en el departamento de psicología.

Esto supone un esfuerzo cognitivo considerablemente alto por parte de los anotadores en el estudio psicológico planteado, ya que requiere que estos mantengan una concentración y enfoque pleno en la tarea de observar a los sujetos de prueba. Ante esta situación se plantea la posibilidad de automatizar de algún modo esta tarea. De esta forma, se conseguirán avances a un mayor ritmo en una investigación dentro del campo de la psicología.

Si nos adentrásemos en el ámbito de aplicación más específico de un estudio como el que se presenta, deberíamos hablar de neurociencia. Dentro de este campo, la

posibilidad de estimar la pose de pequeños animales de laboratorio puede ser ampliamente usada para la identificación de fenotipos, capacidades genéticas, y modelado de enfermedades. Una correcta implementación de un sistema como el que se plantea podría permitir una alta escalabilidad de los estudios en comportamiento animal, permitiendo realizar las observaciones y conclusiones a un ritmo mucho mayor, en incluso de forma paralela para numerosos sujetos de pruebas. Esto sin duda alguna supondría un avance en los estudios conductuales de gran importancia.

Es por eso que plantea un gran interés permitir al especialista en el ámbito de la psicología centrarse en avances para el estudio de forma paralela al trabajo mecánico de anotar y estimar la pose de los sujetos de análisis, enfocando su tiempo y esfuerzo a tareas que sean de una faceta más intelectual y no mecánica como hasta la que ahora vienen llevando a través de la observación de las lombrices durante largas horas.

A priori, no se plantea ninguna restricción para resolver el problema, únicamente se busca el objetivo de obtener un sistema capaz de agilizar el proceso de anotación de la respuesta al estímulo por parte del sujeto.

1.2. Objetivos y metodología

Conocida la motivación para llevar a cabo un proyecto como el que se presenta, se plantea un objetivo claro, y es el de automatizar de algún modo la anotación estímulo-respuesta de los sujetos de prueba. Se detallará una explicación más en profundidad de los objetivos concretos en el [Capítulo 3](#).

Para alcanzar este objetivo, se necesita obtener una base de conocimiento suficiente, para lo cual se estudiará el campo de la estimación de poses, realizando un estudio de manera incremental dentro de la disciplina a lo largo del [Capítulo 2](#).

Es importante destacar que un problema con la naturaleza del planteado supone un reto mayor respecto a otros del ámbito de la estimación de poses. Y es que los sujetos para el análisis propuesto (lombrices), a diferencia de otros seres vivos como podrían ser personas, perros o gatos, no tienen bien diferenciadas las distintas partes de su cuerpo. La diferenciación de la cabeza, como será el objetivo de esta experimentación respecto a la cola ya supone un reto considerable para el anotador externo, incluso en una imagen estática. No solo el hecho de poder indicar la posición exacta de la cabeza supondrá un reto para sistema o humano, sino que también por la naturaleza del sujeto, estos realizan movimientos poco naturales para cualquier otro tipo de animal. Así, estos sujetos son capaces de retorcerse, se enrollan sobre sí mismos,

y llevan a cabo movimientos que son difíciles de predecir para un observador externo.

Otro aspecto que necesariamente se debe tener en cuenta a la hora de enfrentarse a este problema es el entorno de estudio de los sujetos, que como se ve en la [Figura 1.1](#), puede dar pie a confusión debido a la existencia de muchos objetos externos. Podríamos decir que no se trata de un entorno de estudio demasiado limpio, que sumado a una iluminación en ocasiones insuficiente, y oclusiones de los sujetos por elementos externos, de forma indudable supondrá un reto para cualquier sistema que intente detectar y estimar la pose de la lombriz.



Figura 1.1: Muestra del entorno de estudio de los sujetos

De forma obvia, será necesario hacer un estudio de los distintos modelos de aprendizaje profundo y resto de estado del arte para la tarea de estimación de poses, y realizar una comparación en base a cierto criterio y medidas de rendimiento oportunas que indiquen el correcto desempeño del modelo escogido. Por consiguiente, será necesario explorar las distintas medidas de rendimiento que indiquen cómo de bien funcionan los modelos y que nos permitan compararlos con suficiente criterio como para escoger uno, o tal vez más de uno, de los resultados obtenidos durante la experimentación.

Se realizará por tanto, un estudio sobre conjuntos de datos existentes en la bibliografía que nos permitan una predicción correcta para nuestra tarea. En caso de ser necesario, se analizarán las herramientas disponibles, y formas de anotación para las fuentes reales objeto de estudio proporcionadas, con el objetivo de crear un conjunto de datos aplicado a nuestro problema concreto. Con esto, surge la necesidad de conseguir que estos datos anotados se ajusten al formato correcto y dependiente del modelo que se utilizará para realizar las predicciones sobre los datos. Con este objetivo, se requerirá tratar estas anotaciones para la construcción de al menos un conjunto de datos suficientemente grande que permita el correcto entrenamiento del modelo

escogido.

Se compararán entonces distintos modelos existentes que nos permitan la predicción de pose y puntos clave como el requerido, entrenando un número determinado de modelos distintos para realizar la comparación de la evolución de estos modelos en relación a un mismo conjunto de datos. De esta forma, obtendremos qué configuración de hiperparámetros resulta más eficiente para obtener mejores resultados en el conjunto de datos manejado.

Una vez comparados distintos modelos existentes, y escogido el de mejor rendimiento frente a nuestro problema podemos explorar los resultados producidos para el problema real.

1.3. Estructura del documento

El presente trabajo está estructurado de la siguiente manera:

- En el [Capítulo 2](#) se describirán los antecedentes. De forma concreta se planteará una visión global de la situación actual de la estimación de poses en animales, siguiendo un enfoque incremental para entrar en contexto sobre **inteligencia artificial**, **aprendizaje automático** como supervisado y no supervisado, **aprendizaje profundo** y redes neuronales, **visión por computador** y la **estimación de poses**, con una visión a sus aplicaciones habituales, y la estimación de poses en personas. Nos centraremos después en la estimación de poses en animales, y qué tenemos a nuestra disposición para una correcta resolución de la tarea propuesta. Se estudiarán recursos existentes para tareas similares, así como modelos y arquitecturas habituales para problemas de naturaleza similar. Se hará también una revisión de los conjuntos de datos que en la actualidad se usan para diversas aplicaciones de estimación de poses en animales.
- En el [Capítulo 3](#) se relatarán de forma concreta los objetivos a seguir en este trabajo, una vez conocido el estado del arte actual y qué tenemos a nuestra disposición.
- En el [Capítulo 4](#) se plantearán los materiales y métodos de los que se hará uso para una correcta experimentación posterior. Se describirán aquellos puestos a nuestra disposición para un correcto desempeño del estudio, como el entorno de experimentación. Aquellos materiales usados como son el caso de lenguajes de programación o modelos existentes, y los métodos más correctos para

su ejecución y estudio. Las herramientas concretas usadas para la anotación de datos, así como aquellas que nos permitirán aumentar los datos existentes para crear un conjunto de datos mayor. Se describirán también las herramientas de elaboración propia para el tratamiento de anotaciones y conjuntos de datos. Serán descritos por su importancia los conjuntos de datos que se usarán para la experimentación, así como las estrategias de validación que nos permitirá la comparación posterior de resultados. Las herramientas concretas usadas para una correcta depuración y visualización de resultados durante la experimentación. Serán planteadas las configuraciones de hiperparámetros de las que se hará uso en la experimentación y las métricas que nos permitirán la comparación de resultados.

- En el [Capítulo 5](#) se mostrará el proceso de experimentación llevado a cabo, así como los resultados obtenidos. En primer lugar se compararán los resultados haciendo uso de la técnica de aumento de datos. Por último, tendrá lugar una comparación de los mejores modelos obtenidos, destacándose el que cuente con la configuración de hiperparámetros más eficiente a lo largo del proceso de experimentación.
- En el [Capítulo 6](#) y para finalizar, se detallarán las conclusiones extraídas durante la investigación que se ha llevado a cabo, así como propuestas a futuro que plantean la posibilidad de mejorar los resultados obtenidos en el presente trabajo.

1.4. Glosario de términos

Con el objetivo de facilitar la comprensión de la lectura del presente documento, con especial interés en aquel lector no familiarizado con el campo de estudio, se facilita a continuación un glosario de términos, con su respectivo significado en español e inglés, así como las siglas que comúnmente en inglés serán usadas a lo largo del documento.

Tratándose de un campo de amplio desarrollo en el idioma inglés, es habitual encontrar terminología con la que escasa vez nos referiremos a ella con su traducción directa en nuestro idioma. Es por esto que conocer la traducción directa a español puede ser de gran interés para un lector poco habituado no solo al campo de estudio, sino al idioma extranjero.

Término español	Término inglés	Siglas Inglés
Inteligencia Artificial	Artificial Intelligence	AI
Aprendizaje Automático	Machine Learning	ML
Aprendizaje Profundo	Deep Learning	DL
Visión por computador	Computer Vision	CV
Red Neuronal	Neural Newtork	NN
Red Neuronal Artificial	Artificial Neural Newtork	ANN
Red Neuronal Convolucional	Convolutional Neural Newtork	CNN
K-vecinos más cercanos	K-nearest neighbours	KNN
Máquina de Soporte Vectorial	Support Vector Machine	SVM
Sistema de desarrollo integrado	Integrated Development System	IDE
Unidad de Procesamiento Gráfico	Graphics Processing Unit	GPU
Unidad de Procesamiento Central	Central Processing Unit	CPU
Intersección sobre Unión	Intersection over Union	IoU

Tabla. 1.1: Glosario de términos usados

Capítulo 2

ANTECEDENTES

En el presente capítulo se detalla todo el proceso realizado para la adquisición de conocimientos necesarios para un correcto desarrollo de la investigación dentro del campo del aprendizaje profundo, pero más concretamente la visión por computador para la estimación de poses.

De forma secuencial y creciente se revisarán los distintos ámbitos que nos llevan como término final a la estimación de poses en animales, concretamente para los sujetos de prueba propuestos. De forma breve, se realizará una introducción a **inteligencia artificial**, para a continuación adentrarnos en una de sus vertientes, el **aprendizaje automático**. Posteriormente se detallará el campo del **aprendizaje profundo** para después enfocar la explicación en el caso concreto de la **visión por computador** y la **estimación de poses**, siendo este último el objeto del presente trabajo. La [Figura 2.1](#) representa una visión global de la jerarquía del ámbito en el que trabajamos.

2.1. Inteligencia Artificial

Tal y como define el que muchos conocen como el padre de la inteligencia artificial, John McCarthy: [\[10\]](#)

It is the science and engineering of making intelligent machines, especially intelligent computer programs. It is related to the similar task of using computers to understand human intelligence, but AI does not have to confine itself to methods that are biologically observable.

«Es la ciencia de hacer máquinas inteligentes. Está relacionado con la ta-

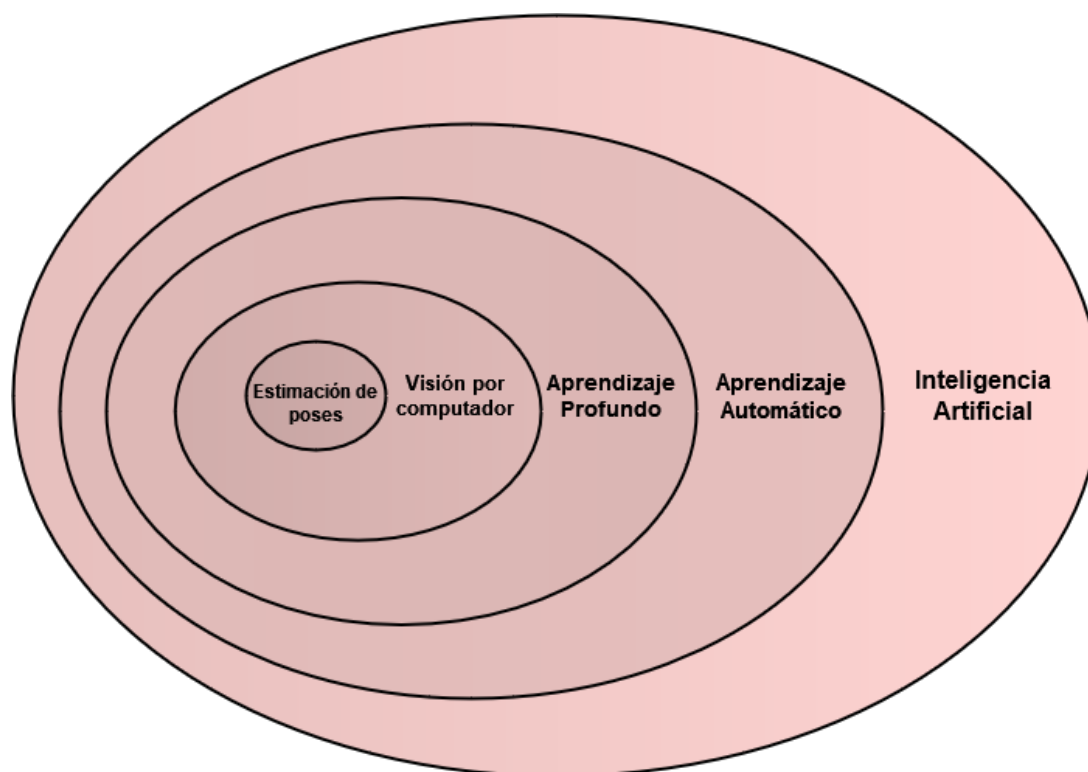


Figura 2.1: Diagrama representativo de la jerarquía del campo de conocimiento. Fuente: elaboración propia

rea similar de usar los ordenadores para entender la inteligencia humana, pero la IA no tiene por qué ceñirse a métodos que son biológicamente observables.»

Será en 1955 cuando el matemático John McCarthy usaría por primera vez el término de **inteligencia artificial** con el que propondría un proyecto de investigación en el que 10 personas investigarían acerca de este campo durante dos meses, en el verano de 1956 en la Universidad de Dartmouth de Hanover, New Hampshire. Planteará un objetivo claro para este proyecto: [11]

The study is to proceed on the basis of the conjecture that every aspect of learning or any other feature of intelligence can in principle be so precisely described that a machine can be made to simulate it.

«El estudio es para avanzar en la conjetura básica de que cada uno de los aspectos involucrados en el aprendizaje o cualquier otro aspecto de la inteligencia puede ser en principio tan precisamente descrito como para hacer que una máquina sea capaz de simularlo.»

A partir de aquí, y como resultado de este encuentro propuesto junto a otros matemáticos como Claude E. Shannon, Marvin L. Minsky y Nathaniel Rochester, se abrirían

nuevas ramas de estudios en relación a la propuesta original sobre la inteligencia artificial, y que cambiaría tanto como para llegar hasta lo que conocemos hoy día.

A sus comienzos, la disciplina de la inteligencia artificial se centra en el uso de algoritmos que aunque complejos, establecidos por el propio creador para un fin concreto. Sin embargo, para un gran número de problemas que los humanos realizamos con conocida simplicidad no existe, a priori, una forma sencilla de reflejarlo en un ordenador para que pueda resolverlo. Conocida esta complejidad, se levantará cierto interés en transmitir a la máquina técnicas de resolver estos problemas sin tener la necesidad de detallar de forma concreta los pasos a seguir por el algoritmo. Es con este objetivo, con el que surge el auge del ML.

2.2. Aprendizaje automático

Según A. L. Samuel [12]:

Machine Learning is the field of study that gives computers the ability to learn without being explicitly programmed

«El aprendizaje automático es el campo de estudio que da a los computadores la capacidad de aprender sin ser programados de manera explícita.»

Más tarde, Tom Mitchell usaría la siguiente definición para referirse al término [13]:

Any computer program that improves its performance at some task through experience.

«Cualquier ordenador que mejora su rendimiento en alguna tarea a partir de la experiencia.»

En lo que él mismo definiría como **aprender**:

*A computer program is said to **learn** from experience E with respect to some class of task T and performance measure P , if its performance at tasks in T , as measured by P , improves with experience E .*

«Un programa de computador se dirá que **aprende** de una experiencia E , con respecto a una tarea T y una medida de rendimiento R , si su rendimiento en T , medido por R , mejora con la experiencia E . »

Es claro entonces que el ML se basa en la realización de tareas a priori demasiado complejas para un planteamiento computacional tradicional o para el que no se conocen algoritmos sencillos que se puedan utilizar. De forma sencilla, podríamos centrar el aprendizaje automático para tareas tal vez existentes, pero cuya resolución requieran un gran número de reglas en un programa tradicional que puedan hacer que su funcionamiento sea peor, entornos en los que los datos sean cambiantes, con lo que un algoritmo de ML se podría adaptar, o para acercamientos a ciertos problemas complejos con gran cantidad de datos. [14]

Dentro del ML, podemos diferenciar diferentes formas de abordar los problemas concretos presentados: aprendizaje supervisado, aprendizaje no supervisado, aprendizaje semi-supervisado y aprendizaje por refuerzo entre otras.

2.2.1. Aprendizaje supervisado

El **aprendizaje supervisado** [15] usa un algoritmo de aprendizaje basado en ejemplos, es decir, cada dato tiene asociado consigo una etiqueta sobre la que el algoritmo entrenará, para ser capaz de predecir esta etiqueta en otro dato del mismo tipo no visto hasta el momento. Dentro del aprendizaje supervisado encontramos problemas de clasificación y regresión diferenciados por el tipo de resultado esperado [16].

Si el tipo de resultado deseado es un valor cualitativo dentro de un conjunto de clases, hablaremos de **clasificación**. Podríamos ejemplificar el problema de clasificación de la siguiente forma:

El sistema de aprendizaje trata de etiquetar (clasificar) una serie de vectores utilizando una entre varias categorías (clases). La base de conocimiento del sistema está formada por ejemplo de etiquetados anteriores [17].

Si el tipo de resultado deseado es un valor cuantitativo hablaremos de un problema de **regresión**.

Tendremos por tanto que en los problemas de clasificación se buscará asignar un valor cualitativo, generalmente denominado etiqueta, a los datos. Un ejemplo claro y bastante común es la clasificación multiclase de imágenes según el tipo de animal que aparece en ella. Serán clasificadas de forma independiente imágenes en las que aparezca un perro, un gato o un pájaro. Estas etiquetas están previamente predefinidas.

Por el contrario, en un problema de regresión no se tiene como objetivo asignar

una etiqueta a cada instancia del conjunto de datos, sino predecir un valor numérico basado en las características disponibles. Por ejemplo, la predicción del precio de un producto a lo largo del tiempo.

Existen distintos algoritmos aplicables para el caso de regresión, como son [18] **regresión lineal**, **regresión polinómica**, **KNN** o **SVM**. En relación a los problemas de clasificación tendremos algoritmos [18] como son **KNN**, **árboles de decisión**, **SVM** o **ANN**, que serán detalladas en la sección 2.3.1 por su importancia en el caso de nuestro problema concreto.

2.2.2. Aprendizaje no supervisado

A diferencia del aprendizaje supervisado detallado en la sección 2.2.1, podríamos entender de forma intuitiva el **aprendizaje no supervisado** [19] como un algoritmo de aprendizaje que no toma ejemplos etiquetados para aprender. De esta forma, no se tiene ningún tipo de información sobre categorías o clases de las que hablábamos en la sección anterior.

Este tipo de aprendizaje es comúnmente utilizado para casos en los que no se tiene amplia información sobre los datos a trabajar, constituyéndose así un tipo de aprendizaje por observación y descubrimiento del propio sistema. Habitualmente se realiza la agrupación de una serie de características que el algoritmo es capaz de reunir en un mismo concepto. Esta técnica de aprendizaje no supervisado denominada como *técnica de agrupamiento de casos o clúster* [18], segrega los datos en los grupos correspondientes teniendo en cuenta ciertas similitudes existentes entre los datos. Encontramos algunos algoritmos como **algoritmo simple de las K-medias** o **clase simple de maximización esperada**.

2.3. Aprendizaje profundo

El **aprendizaje profundo** [20] es una rama dentro del aprendizaje automático. Lo que caracteriza a este subconjunto es su estructura en base a una red neuronal que le permite 'aprender' a partir de un número grande de datos, algo no posible en una ANN clásica. Esta ANN intenta imitar la manera en la que el cerebro humano procesa la información, optimizando, y dando predicciones sobre los datos de entrada. El término **profundo** dentro del aprendizaje hace referencia a una metodología de aprendizaje de la red haciendo uso de diferentes niveles o **capas** a través de la cual se procesa la

información, lo que le permite aprender conceptos complejos sobre datos complejos. [20]

Los métodos de aprendizaje profundo permiten el aprendizaje a través de varios niveles de representación, obtenidos mediante la composición de funciones no lineales, conocidas como capas, y produciendo una serie de transformaciones al conjunto de entrada. La combinación de suficientes capas permite llegar a un nivel de abstracción más alto, en el que conceptos muy complejos pueden ser aprendidos. [21]

Los métodos de aprendizaje profundo son capaces de descubrir las estructuras y patrones contenidos en el conjunto de datos de entrenamiento mediante el uso del algoritmo de *backpropagation*, lo que le permite modificar los parámetros internos usados para calcular los parámetros de cada una de las capas. [22].

2.3.1. Redes neuronales

Las **redes neuronales** [23] son dentro de la inteligencia artificial en general, y el aprendizaje profundo en particular, una de las técnicas más usadas y efectivas para resolver un problema de alta complejidad como el que se presentará en capítulos posteriores.

Dentro del ámbito de la inteligencia artificial, siempre se ha perseguido el objetivo de realizar tareas de forma similar a los humanos. Las ANN tienen este objetivo, el de simular de la forma más precisa posible el comportamiento de las redes neuronales presentes en el cerebro humano.

Las redes neuronales se forman por las llamadas *neuronas artificiales*, que se unen entre sí a través de conexiones que cuentan con *pesos sinápticos* asociados. Estos pesos en su conjunto representarán el conocimiento total de una ANN. Cada neurona será en realidad una función matemática del tipo mostrado en la [Figura 2.2](#) y definida por primera vez por McCulloch and Pitts en 1943 con la siguiente expresión:

$$y = \gamma\left(\sum_{i=1}^m w_i x_i + w_0\right)$$

Donde γ es la función de activación no lineal propuesta por McCulloch-Pitts, y conocida como la función de todo o nada.

Esta definición originaria de neurona pretende simular de forma concreta el funcionamiento de las neuronas biológicas en los cerebros humanos, mediante el uso

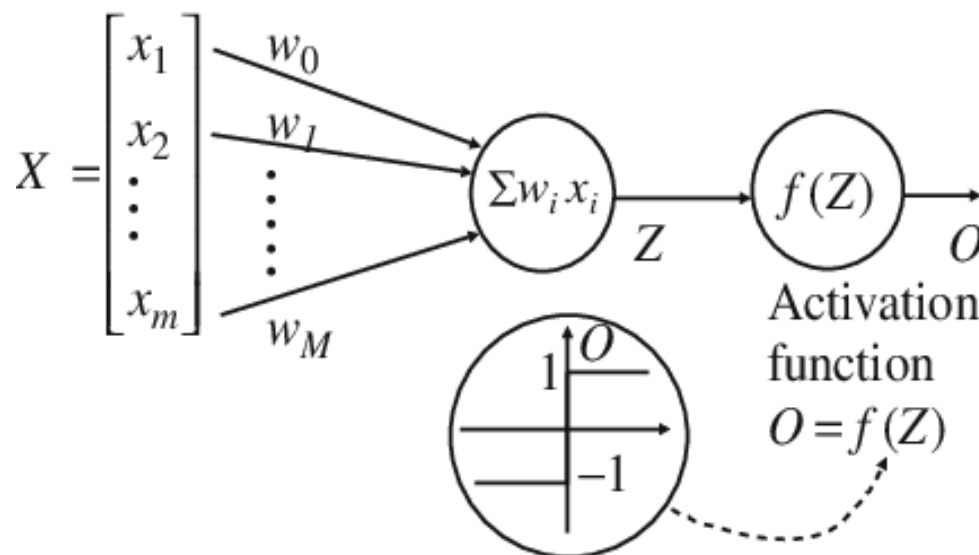


Figura 2.2: Representación de la neurona artificial por McCulloch and Pitts, 1943. Fuente [1]

de lógica proposicional para la realización de tareas complejas. Un conjunto de estas neuronas formará por tanto una capa, y un conjunto de capas formará una **red neuronal artificial**, que denominaremos simple si no contiene más de una capa oculta, y sigue la estructura de la [Figura 2.3](#).

Siguiendo este esquema, el trabajo en aprendizaje profundo se centra normalmente en **redes neuronales profundas**, que incluirán un número considerable de capas ocultas, cada una con un número determinado de neuronas que implementan funciones de activación características. Un ejemplo de estas redes se muestra en la [Figura 2.4](#) en la que se tendrán n capas ocultas.

Existen distintos tipos de redes neuronales profundas que se diferenciarán en la estructura de sus capas y las funciones de activación empleadas. Tendremos numerosas redes neuronales usadas en la actualidad [20], por ejemplo: **redes feedforward**, **redes convolucionales**, **redes generativas antagónicas**, **redes neuronales residuales** o **máquinas de estado líquido**.

2.4. Visión por computador

El comúnmente conocido como padre de la visión por computador, Larry Roberts, reconocería este nuevo campo de estudio en la década de 1970. Siendo también uno de los principales propulsores en el desarrollo de ARPANET [24], el ingeniero de computadores y eléctrico visionaba en la época un gran potencial en los ordenadores,

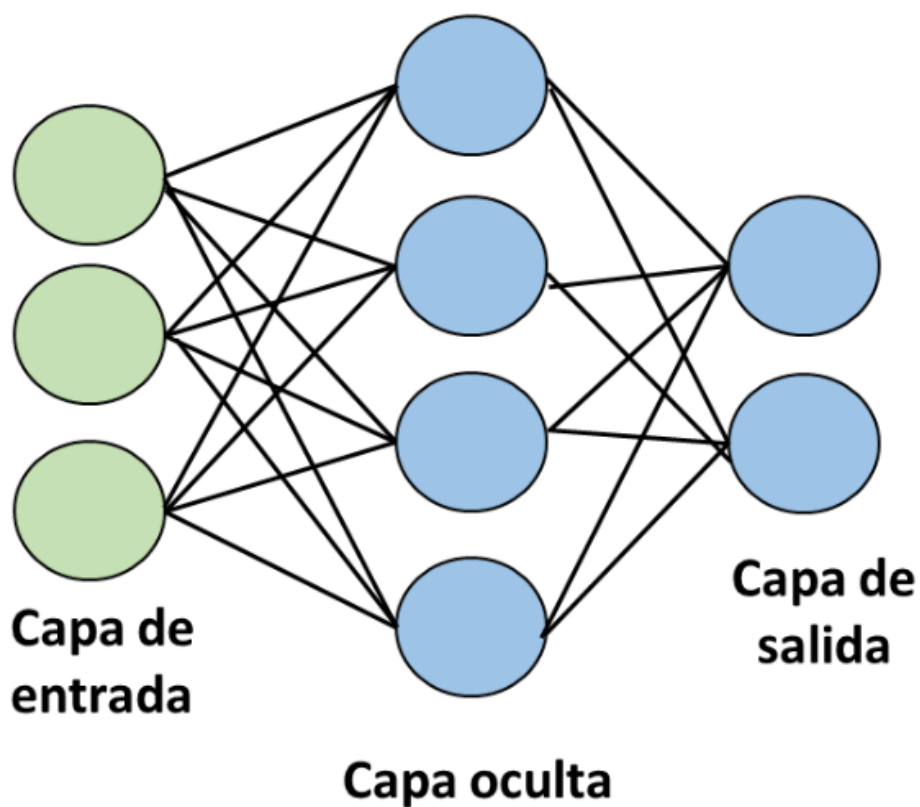


Figura 2.3: Representación de una ANN simple. Fuente [2]

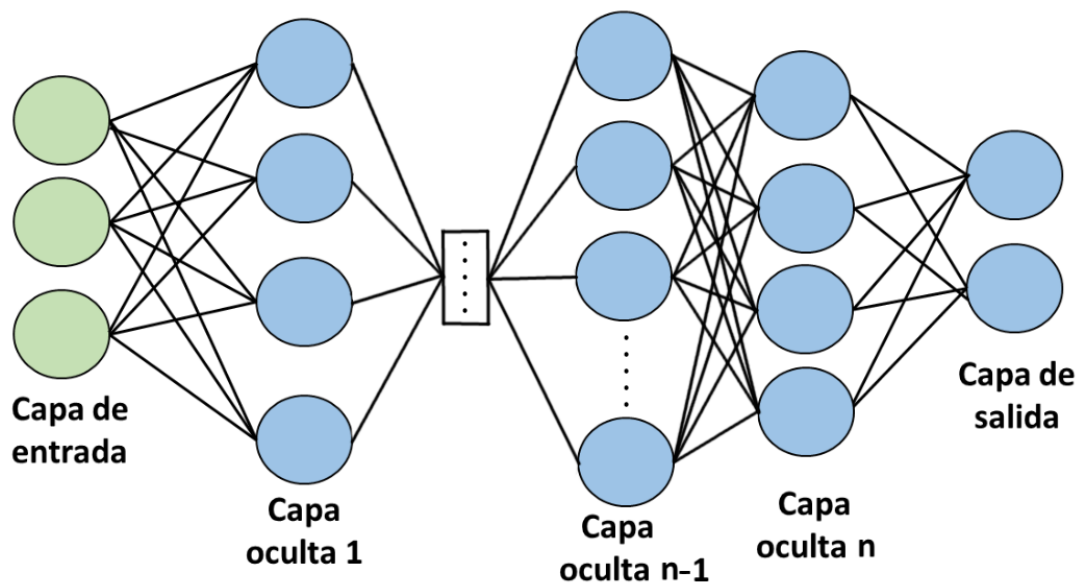


Figura 2.4: Representación de una ANN profunda. Edición personal a partir de la fuente original [2]

y creía firmemente que podrían ver e interpretar información visual, algo muy innovador en aquellos años.

Algunos investigadores de forma posterior definen la visión por computador de la siguiente forma:

Es una disciplina que desarrolla sistemas capaces de interpretar el contenido de escenas naturales.

(K.R. Castleman, 1996) *Digital Image Processing* [25]

Es una ciencia que desarrolla las bases teóricas y algorítmicas para obtener información sobre el mundo real a partir de una o varias imágenes.

(Haralick, Robert M. and Shapiro, Linda G., 1992) *Computer and Robot Vision* [26]

La visión por computador tiene un doble objetivo. Desde el punto de vista de ciencia biológica, la visión por computador busca crear modelos del sistema visual humano. Desde la perspectiva ingenieril, la visión por computador busca construir sistemas autónomos que puedan realizar algunas de las tareas de las que el sistema visual humano es capaz de llevar a cabo (incluso superarlo en algunos casos).

(T. S. Huang) *Computer Vision: Evolution and Promise* [27]

Se observan por tanto, unas definiciones de CV no demasiado concretas, y muy amplias. Por lo general, podríamos describir la **visión por computador** como diversas técnicas de procesamiento de imágenes, reconocimiento de patrones, computación gráfica entre otros para obtener medidas determinadas o características. Una tecnología que combina los ordenadores con cámaras de vídeo para adquirir, analizar e interpretar imágenes de una forma equivalente a la inspección humana. [28]

Sus límites y aplicaciones no están centradas en un único ámbito, sino que son diversas. A continuación vemos algunos de los tópicos considerados en la **European Conference on Computer Vision 2022 (ECVV)** [29] que tuvo lugar en Tel Aviv, Israel en el mes de octubre de 2022. En esta, vemos estudios en los que se habla de cómo la CV avanza hacia temáticas como:

- Reconocimiento de objetos.

- Clasificación de imágenes
- Procesamiento de imágenes
- Detección de objetos
- Segmentación semántica
- Estimación de postura humana
- Reconstrucción 3D
- Visión estéreo
- Fotografía computacional
- Redes neuronales
- Codificación de imágenes
- Reconstrucción de imágenes
- Reconocimiento de objetos
- Estimación de movimiento

En este estudio, como se podrá ver en las próximas secciones, la visión por computador será aplicada sobre las temáticas de **detección de objetos**, **redes neuronales** y **estimación de movimiento** de entre las discutidas en el más reciente ECCV.

El objetivo buscado, como ya relatado en la sección 1.2, será la de emular un anotador humano, que con la mayor exactitud posible, sea capaz de aproximar las anotaciones del estímulo-respuesta de los sujetos. Para conseguir esta tarea, se realizará el estudio de las posibilidades existentes buscando crear una solución al problema lo más óptima posible.

2.5. Estimación de poses

La **estimación de pose** a generado desde sus orígenes una gran expectación e interés. La posibilidad de hacer uso de distintas técnicas que permitan identificar y realizar un seguimiento de objetos y personas en tiempo real es algo que ofrece un número importante de posibilidades para distintas aplicaciones y campos.

Para entender la estimación de poses, es necesario que el lector conozca qué es la pose. La pose puede ser definida como el conjunto de puntos clave de un objeto que se alinean de una forma concreta, siendo esta habitualmente asociada a personas, aunque no limitándose a este ámbito de forma única.

La estimación de poses dentro del contexto de la CV es la tarea específica de detectar la posición u orientación de un objeto dentro de una imagen o conjunto de imágenes (un vídeo). De forma habitual para conseguirlo, se busca localizar o realizar el seguimiento de distintos puntos de un objeto concreto dentro de dichas imágenes. Podríamos ejemplificar este concepto entendiendo que por ejemplo dentro de la estimación de poses humana un punto clave sería el pie izquierdo o el hombro derecho, como podemos ver en la [Figura 2.5](#). Un punto clave dentro de la estimación de pose en objetos, aplicado a un coche sería por ejemplo la rueda delantera izquierda o el faro trasero derecho, como podemos ver en la [Figura 2.6](#). En nuestro caso concreto de estudio, buscaremos la pose dada por la cabeza de la lombriz al que se estudiará, tal como fue relatado en la sección 1.2, podemos ver en la [Figura 2.7](#) los puntos de los que se hará uso para estimar la pose del sujeto de pruebas.

Se encuentran los orígenes alrededor de las década de los 90, con diversos intentos de seguimiento de movimiento o reconocimiento de patrones faciales en humanos [\[30\]](#).

En la actualidad, el campo de la estimación de poses a avanzado impulsado por el simultáneo auge de las tecnologías en relación al DL, lo que ha llevado a la aparición de otras ramas menos tradicionales que la conocida estimación de poses humanas, como es la estimación de poses en animales y sus respectivas aplicaciones.

2.5.1. Aplicaciones de la estimación de poses

Diferentes campos se ven beneficiados con técnicas de estimación de poses para distintas finalidades, desde el estudio de la biomecánica y rehabilitación médica haciendo uso de técnicas de monitorización de la posición o ejecución de ejercicios [\[31\]](#), [\[32\]](#); rendimiento en actividades deportivas [\[33\]](#), [\[34\]](#); interacción persona-ordenador [\[35\]](#); robótica y modelado 3D para la generación de avatares en realidad virtual y aumentada [\[36\]](#); técnicas de animación 3D [\[37\]](#); vídeo vigilancia para la detección de armas [\[38\]](#) y seguridad ciudadana para acciones delictivas [\[39\]](#) entre otras muchas aplicaciones.

Si hablamos de la estimación de poses en animales, aplicaciones concretas bus-



Figura 2.5: Posibles puntos asociados a la estimación de pose humana

can aumentar el conocimiento sobre el comportamiento dentro del mundo animal en distintas especies [40], [41]; la preservación de especies protegidas y en peligro [42]; estudiar mecanismos de migración de especies salvajes [43]; el cuidado de animales del hogar [44] y entre otras muchas, la que plantea el objetivo del presente trabajo: la relacionada con el comportamiento animal con el campo de la neurociencia [45],[46], [47].



Figura 2.6: Posibles puntos asociados a la estimación de pose en un coche

2.5.2. Estimación de poses en personas

La estimación de poses en personas, o estimación de poses humana es la tarea que busca las técnicas de estimación de poses conocidas a sujetos humanos. El objetivo es predecir puntos concretos dentro de la estructura del cuerpo humano dentro de una imagen o secuencia de imágenes.

Existen un gran número de conjuntos de datos que nos permiten llevar a cabo tareas de estimación de poses humana, y lo que ha permitido en los últimos años un gran avance en este campo de estudio. Por lo general la estimación de poses en dos dimensiones ha tenido una mayor relevancia e interés de forma histórica, sin embargo, cada vez con más frecuencia encontraremos aplicaciones orientadas a la estimación de poses humanas en tres dimensiones [48].

2.5.3. Estimación de poses en animales

Como ya se ha comentado, siendo la tarea de la estimación de poses una de las tal vez no tan extendidas, la mayoría de aplicaciones y métodos existentes en el estado del arte actual hacen referencias a la estimación de poses humana, la cual ha estado evolucionando en los últimos años a un ritmo mayor que cualquier aplicación

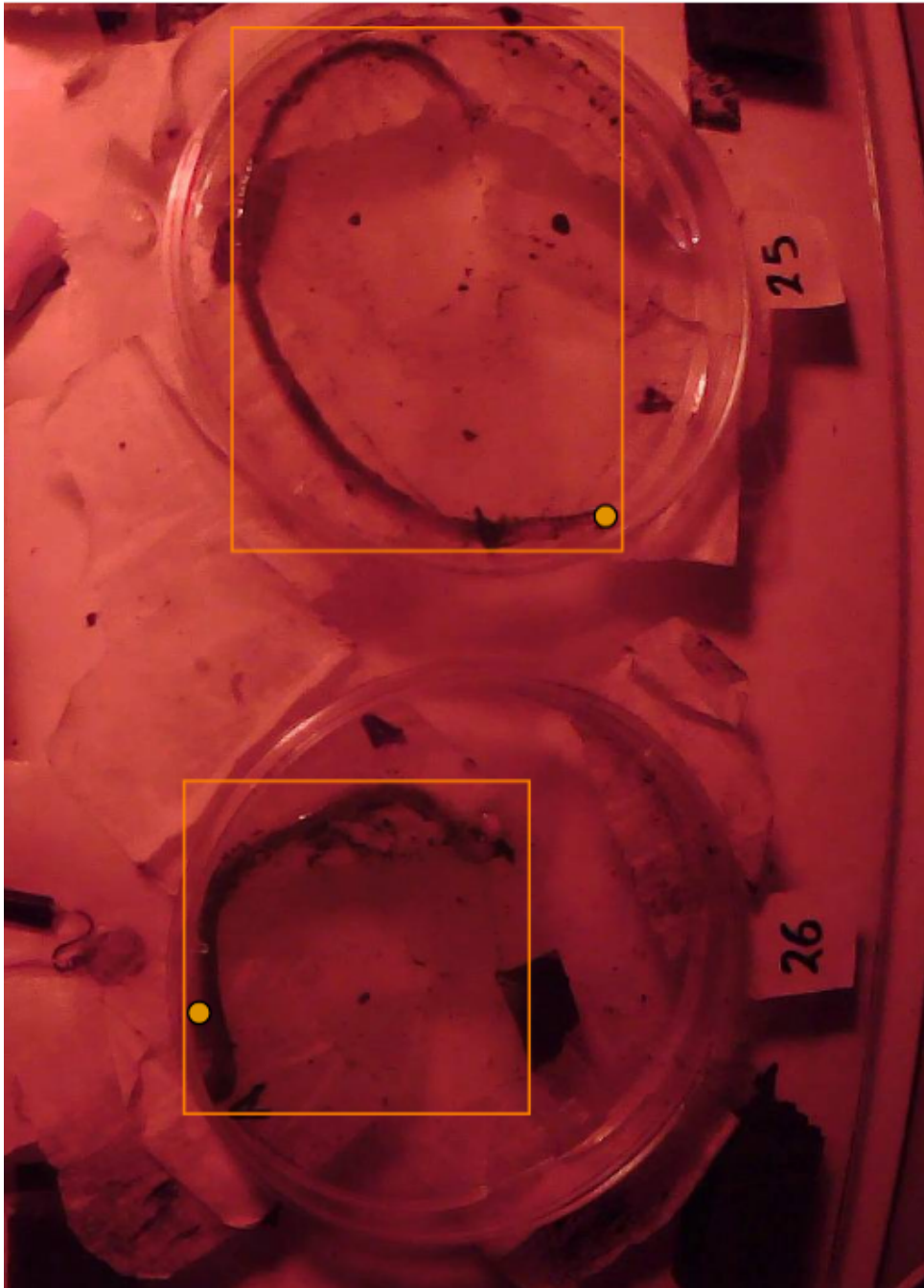


Figura 2.7: Puntos asociados a la estimación de pose en el estudio

en animales. Dentro de un estudio como sabemos orientado a la estimación de poses en animales, contaremos entonces con limitaciones y retos adicionales a los que nos podríamos encontrar en otras tareas del campo de estudio.

La escasa cantidad de herramientas existentes en comparación con otros campos de estudio hacen que no solo la adaptabilidad a nuestro problema concreto sea compleja, sino que en ocasiones será incluso no considerable. Esto será relatado en la sección [2.6](#).

Existe la imperiosa necesidad también para cualquier problema del estilo de contar con datos suficientes. La notable variedad entre especies animales complican aún más la tarea, y es que datos anotados para un ave, difícilmente servirán de mucha utilidad para tareas relacionadas con otros vivíparos o reptiles.

En este caso, podremos encontrar varios conjuntos de datos que por lo general son usados en los demás problemas de estimación de poses en animales, pero que en nuestro sujeto de prueba difícilmente aplica, por lo que la limitación es latente. Esto nos llevará a crear un conjunto de datos propio, entendida la limitación y el esfuerzo que esto supone.

2.6. Recursos existentes

Una vez conocemos el campo en el que nos encontramos trabajando, y conocidas las dificultades para un campo de aplicación no tan extendido como es la estimación de poses en animales, debemos ceñirnos a nuestra tarea concreta, y es la estimación de poses en lombrices.

Parece claro que si el campo de estudio en la estimación de poses en animales no está tan extendida como podría ser en personas, menos aún lo estará para un sujeto de análisis tan fuera de lo común como es el de este trabajo. Si bien es cierto que existen ciertos estudios en relación al *Caenorhabditis elegans* o comúnmente conocido como **C. Elegans**, una especie de nematodo de la familia *Rhabditidae*, ninguno será usable y extrapolable de forma directa a nuestro problema. Sin embargo, podemos analizar las formas de trabajo existentes para problemas de similar naturaleza. Analizaremos a continuación distintos proyectos y estudios que trabajan con *C. Elegans*.

2.6.1. Tierpsy-Tracker y Open Worm Movement Database

Tierpsy-Tracker se trata de un sistema que permite el seguimiento multi-animal, desarrollado en el *MRC Laboratory of Medical Sciences Phenomics Lab*, en la universidad Imperial de Londres. Principalmente su enfoque trata en el análisis del comportamiento del *Caenorhabditis elegans*. [49] [50].

Numerosos estudios hacen referencia al uso conjunto de la herramienta *Tierpsy-Tracker*, junto a un repositorio abierto enfocado en almacenar datos relativos a gu-

sanos, su movimiento y comportamiento. Este repositorio denominado **Open Worm Movement Database** [50] nace con un objetivo claro:

Animal behavior is increasingly being recorded in systematic imaging studies that generate large datasets. To maximize the usefulness of these data, there is a need for improved resources for analyzing and sharing behavioral data that will encourage reanalysis and methodological developments.

« El comportamiento animal está siendo grabado de forma creciente en estudios con largos conjuntos de datos. Para maximizar la usabilidad de estos datos, existe la necesidad de crear recursos mejorados que permitan analizar y compartir los datos sobre el comportamiento de los sujetos, lo que motivará para analizarlos y para realizar desarrollos en la metodología. » [51]

En relación al estudio con estos *Caenorhabditis elegans* en el mismo estudio se comenta:

For computational ethology to approach the level of maturity of other areas of bioinformatics, at least three challenges must be addressed: storing and accessing video files; defining flexible data formats to facilitate data sharing; and developing software to read, write, browse, and analyze the data. We have generated an open resource to begin addressing these challenges for Caenorhabditis elegans behavioral data.

« Para conseguir que la etología computacional alcance el nivel de madurez de otras áreas de la bioinformática, al menos se deben afrontar tres retos: almacenar y acceder a archivos de vídeo; definir tipos de datos flexibles para facilitar la compartición de datos; y desarrollar software para leer, escribir, buscar y analizar los datos. Con el objetivo de enfrentar estos problemas para los datos sobre el comportamiento del *Caenorhabditis elegans* hemos desarrollado este software »

Tenemos entonces una plataforma de código abierto para el estudio de un tipo concreto de gusanos, los *C. elegans*, aunque proporciona recursos y datos que podrían llegar a ser prometedores y que nos sirviesen en nuestro problema, el estudio de nuestros sujetos propuestos es tan concreto que no nos permitirá hacer uso de las herramientas proporcionadas, ya previamente probadas para nuestros datos.

El funcionamiento de *Tierpsy-Tracker* se basa en una compresión del vídeo, en la que el algoritmo es capaz de identificar partículas oscuras en un entorno más claro

[52]. Hará uso de un umbral adaptativo [53] que filtrará las partículas por tamaño. Tras probar el software, además de una costosa implementación inicial, vemos que es complicado adaptarlo a nuestro problema pues es un entorno de pruebas muy dispar con el que se plantea en los vídeos usados en la documentación.

Podemos ver un ejemplo claro de la [Figura 2.8](#) tras aplicarle la función umbral en la [Figura 2.9](#). Es complicado poder distinguir de esta forma y de forma clara las lombrices.

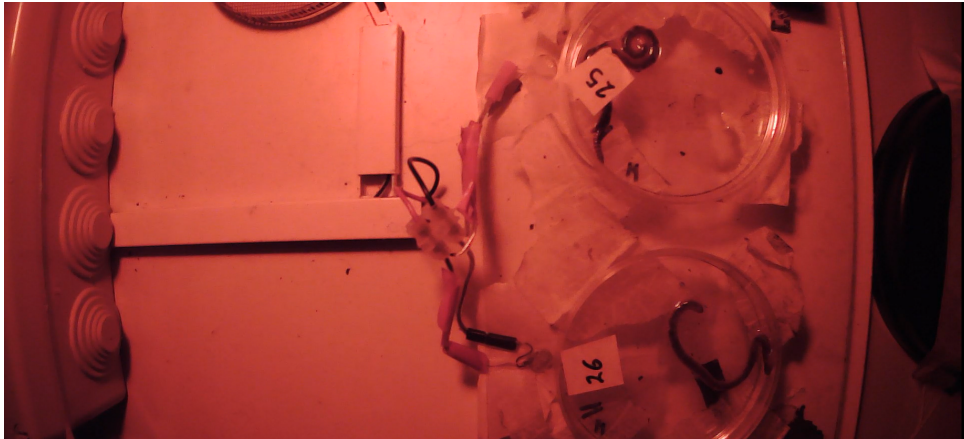


Figura 2.8: Fotograma individual original



Figura 2.9: Fotograma individual tras aplicar función umbral

Vemos como dado el entorno propuesto para nuestro problema, que como se relataba en la sección [1.2](#) no es ideal, es complicada la tarea de hacer uso de la herramienta. Aunque en principio prometedora, *Tierpsy-Tracker* va a ser difícilmente extrapolable a nuestro problema concreto, al menos sin un mayor esfuerzo del que sería usar algún otro método posible, como ha sido mostrado, y bajo la experiencia propia.

Por otro lado, en el análisis de estos estudios encontramos una clara forma de

trabajo, y es que, los dos artículos encontrados están redactados por el mismo grupo de personas, los cuales son los desarrolladores de *Tierpsy-Tracker* y colaboradores en el proyecto *Open Worm Movement Database*. No será de extrañar entonces que lo presentado en los estudios en los que nos basamos para esta investigación sea una forma de trabajo interna y propia del problema tratado por ellos mismos, adaptado para sus necesidades.

2.6.2. WormPose

WormPose es un sistema creado en Python para la estimación en 2D para *C. elegans*, incluyendo distintas posiciones de situaciones de estos seres. Presenta una mejora al problema que no alcanzaría a solucionar el estudio previo sobre *Tierpsy-Tracker*. Involucra el uso de imágenes sintéticas para el entrenamiento de una CNN con datos no reales pero que simulan al sujeto en diversas situaciones como superpuesto y enrollado en sí mismo, evitando así la necesidad de un anotador externo.

WormPose [54] pone a disposición los datos usados, sin embargo el formato de los datos es a priori desconocido y propio del sistema, por lo que la tarea de familiarizarse con los mismos, y poder usarlos sería compleja, y con poca poca esperanza de que realmente sirvan para nuestro problema concreto. Los datos usados para el sistema de WormPose provienen de la base de datos relatada en la sección anterior **Open Worm Movement Database**, que en el momento del estudio contendrá 305 especies distintas de *Caenorhabditis elegans* [55] vistas a microscopio sumando un total de 9203 experimentos. Además a estos se sumarán durante la investigación 386 tipos más de *C. elegans* hasta llegar a un total de 14874 experimentos individuales para el seguimiento de los individuos.

Dado el material de estudio para nuestro problema, la naturaleza es totalmente diferente a cualquiera de las existentes para *C. Elegans* y cualquier conjunto de datos en los que estos se utilicen serán por tanto inservible.

Sí que es cierto que este estudio plantea una cuestión no antes considerada, y es la elaboración de un conjunto de datos sintético que se podría implementar en el conjunto de entrenamiento usado, para ampliar la variedad de situaciones en las que podremos alimentar el modelo.

Aunque efectivo, este acercamiento al problema no será implementado, y será únicamente planteado en la sección 6.2 como una propuesta de mejora.

2.7. Modelos y arquitecturas existentes

Se han vistos algunos de los recursos existentes en el estado del arte actual en lo relacionado a la estimación de poses para sujetos de prueba similares a los planteados en este estudio. Sin embargo, como se ha discutido en la sección anterior, difícilmente serán usables en un problema como el planteado.

Es por ello, que se plantean a continuación, un poco a más bajo nivel, qué herramientas, y por consiguiente, modelos concretos, existen en la actualidad para tareas relacionadas con la estimación de poses. Serán planteadas arquitecturas y modelos que pueden ser usados tanto para la pose en personas como en animales. [56]

Una vez conocidos algunos de los métodos y arquitecturas existentes para estimar poses más habituales, se escogerá la que más conveniente pueda ser para nuestro problema. De este modo, se seguirá una forma de trabajo muy habitual en el campo y es la **transferencia de conocimiento** o *transfer learning* [57], mediante la cual se hará uso de un modelo existente para reducir el tiempo de entrenamiento de forma considerable. No solo se reducirá el tiempo de entrenamiento si hacemos uso de un modelo ya preentrenado, sino que es habitual una mejora significativa en el rendimiento [58].

2.7.1. Tensorflow Lite

Tensowflow Lite [8] es un framework para problemas de aprendizaje profundo y de código abierto. Proporciona una serie de herramientas para numerosos tipos de problemas dentro del DL, junto a los modelos preentrenados concretos. Tendremos tareas de CV como la clasificación de imágenes, detección de objetos, estimación de poses, segmentación de imágenes, etc. Muchas otras tareas dentro del procesamiento del lenguaje natural están disponibles también.

Es parte de **Tensorflow**, una librería desarrollada por Google para tareas de ML en 2015, cuya gran facilidad de creación e implementación de modelos supone un gran atractivo tanto a desarrolladores principiantes como expertos.

El objetivo inicial de esta versión *lite* de Tensorflow es su ejecución en dispositivos de bajas prestaciones, algo habitual en aplicaciones reales de estimación de poses.

El framework proporciona una implementación de dos modelos para la estimación de poses:

- *MoveNet*: el modelo más reciente de los presentados, con dos versiones posibles: *Lightning* y *Thunder*
- *PoseNet*: el modelo de la generación anterior.

En la [Tabla 2.1](#) se presentan las métricas de rendimiento presentadas en la documentación oficial. Los valores mAP se calculan a partir de un subconjunto del conjunto COCO. Los valores de latencia han sido los mínimos presentados y obtenidos con el cómputo en un sistema *Pixel 5 - GPU*. No se especifican más detalles en la documentación oficial.

Modelo	Tamaño (MB)	mAP	Latencia (ms)
MoveNet.Thunder (FP16 quantized)	12.6MB	72.0	45ms
MoveNet.Thunder (INT8 quantized)	7.1MB	68.9	52ms
MoveNet.Lightning (FP16 quantized)	4.8MB	63.0	25ms
MoveNet.Lightning (INT8 quantized)	2.9MB	57.4	28ms
PoseNet(MobileNetV1 backbone, FP32)	13.3MB	45.6	40ms

Tabla. 2.1: Medidas de rendimiento de los modelos proporcionados para TensorFlow Lite. Fuente [8]

2.7.2. YOLOv8

YoloV8 [59] siglas para el nombre completo *You Only Look Once* es uno de los frameworks más potentes y conocidos para aplicaciones de visión por computador.

La empresa a cargo de YOLO, *Ultralytics*, pone a disposición de los desarrolladores innumerables herramientas relacionadas con la IA, permitiendo el desarrollo de modelos de forma accesible, eficiente de entrenar y fácil de desplegar.

La versión 8 de este modelo, se trata de una versión mejorada, haciendo uso de las últimas tecnologías, y estado del arte actual, además de la implementación de las mejoras respecto a las anteriores versiones de YOLO. Además, ahora proporciona una robusta implementación para la tarea de estimación de poses, algo que en versiones anteriores [60] parecía no funcionar tan bien.

Otro punto muy a favor del uso del framework de Ultralytics es su extensa y detallada documentación en esta nueva versión, lo que permite una gran facilidad y flexi-

bilidad en la implementación de cualquier modelo, entrenamiento de uno propio, o el uso de cualquier herramienta disponible.

Existen en la actualidad las versiones 9 y 10 de Yolo, que implementan mejoras en los métodos usados, pero que estando en fase experimental y no tan documentada, serán excluidas del estudio.

Veremos entonces los modelos YoloV8 puestos a disposición para la tarea de estimación de poses.

Estos modelos se encuentran preentrenados sobre el conjunto de datos COCO [61]. En la [Tabla 2.2](#) vemos las medidas de rendimiento presentadas para los modelos facilitados por Ultralytics.

Modelo	Tamaño(px)	mAP50-95	mAP50	Velocidad TensorRT (ms)	Parámetros (M)
YOLOv8n-pose	640	50.4	80.1	1.18	3.3
YOLOv8s-pose	640	60.0	86.2	1.42	11.6
YOLOv8m-pose	640	65.0	88.8	2.00	26.4
YOLOv8l-pose	640	67.6	90.0	2.59	44.4
YOLOv8x-pose	640	69.2	90.2	3.73	69.4
YOLOv8x-pose-p6	1280	71.6	91.2	10.04	99.1

Tabla. 2.2: Características más significativas de los modelos utilizados.

El tamaño en píxeles es el tamaño de entrada imágenes usadas. Los valores de mAPval50-95 y mAP50 corresponden a la validación sobre de un modelo concreto sobre el conjunto de datos de COCO Keypoint val2017 [61] y a las métricas que se detallarán en la sección 4.11. La velocidad es la media computada sobre las imágenes de validación de COCO usando una instancia de Amazon EC2 P4d [62]. Los parámetros es el número de parámetros en millones que tiene el modelo.

2.8. Conjuntos de datos existentes

Es habitual encontrar el desarrollo de modelos para tareas de estimación de poses en animales, acompañado con conjuntos de datos específicos de los que se ha hecho uso para el caso concreto dada la diferente naturaleza del tipo de animal estudiado, y

las notables diferencias entre especies.

En el caso del tipo de anotaciones, no encontramos una por defecto utilizada en las investigaciones, sino que cada estudio hará uso de las que le sean más convenientes y fáciles. Existen algunos estudios que intentan crear una adaptación para distintas y que de modo general sirva para múltiples experimentaciones [63], incluso en formatos conocidos para otras tareas de estimación de pose como COCO [64], pero con resultados muy dispares dependiendo de las especies utilizada.

En la tabla 2.3 se presentan diferentes conjuntos de datos que dan el estado del arte más actual para la tarea de estimación de poses en animales recogidos en 2022. [9].

Como se puede observar, ninguno de la gran cantidad de conjunto de datos más usados en el estado del arte actual incluye alguna referencia a reptiles del tipo que estudiaremos en este trabajo, por ello no se detallará en mayor profundidad los conjuntos de datos presentados, pues como es lógico, no nos serán válidos.

Todos los factores mencionados influyen en la dificultad de encontrar una solución lo más generalista posible, y llevará irremediablemente a crear la propia para la investigación que se plantea.

Año	Conjunto de datos	Clases de animales	Tamaño	Tipo de datos	Anotación	Referencia primer uso
2021	AP-10K Dataset	23 familias de animales y 54 especies	10K imágenes etiquetadas, 50K imágenes no etiquetadas	Imágenes 2D	17 puntos de referencia (pose 2D)	[64]
2021	Horse-10 Dataset (Rogers et al., 2021)	Caballos	8.1K imágenes	Imágenes 2D	22 puntos de referencia	[65]
2021	AcinoSet (Daniel et al., 2021)	Guepardos	119K fotogramas	Vídeo multivista, imágenes 2D	20 puntos de referencia (pose 2D), Predicción de pose 3D	[41]
2020	StanfordExtra (Benjamin et al., 2020)	Perros	12K imágenes	Imágenes 2D	20 puntos de referencia (pose 2D)	[44]
2020	Synthetic Animal Dataset (Mu et al., 2019)	Sabueso sintético, Tigre, Caballo, Oveja, Elefante	50K	Imágenes 2D	18 puntos de referencia (pose 2D), Segmentación	[66]
2019	ATRW dataset (Jianguo et al., 2019)	Tigres de Amur	8K imágenes	Imágenes 2D	15 puntos de referencia (pose 2D)	[67]
2019	Synthetic Grevy's Zebra Dataset (Silvia et al., 2019)	Cebra de Grevy sintética	13K imágenes	Imágenes 2D	28 puntos de referencia (pose 2D)	[42]
2019	Animal Pose (Jinkun et al., 2019)	Perros, Gatos, Vacas, Caballos, Ovejas, otros 7 categorías	710 MB	Imágenes 2D	20 puntos de referencia (pose 2D), Cajas delimitadoras	[63]
2018	BADJA (Biggs et al., 2018a)	Osos, Camellos, Vacas, Perros, Caballos, Tigres, Gatos	2.8 GB	Fotogramas 2D de vídeo	20 puntos de referencia (pose 2D), Segmentación	[68]
2017	Digital Life 3D (Duncan et al., 2017)	Rinoceronte, Lagarto, Delfín	N/A	Malla 3D (con textura)	No	[69]
2016	TigDog (Pero et al., 2016)	Perros, Caballos, Tigres	7.5 GB	Vídeo 2D	19 puntos de referencia (pose 2D)	[70]
2014	COCO (Lin et al., 2014a)	Gato, Perro, Caballo, Oveja, Vaca, Elefante, Oso, Jirafa, Cebra	18GB	Imágenes 2D	Caja delimitadora, Segmentación, Etiqueta de clase	[71]
2012	Poselets (Bourdev, 2012)	Perros, Gatos, caballos, Ovejas, Vacas	1.6GB	Imágenes 2D	16 puntos de referencia (pose 2D), Caja delimitadora	[72]
2011	Stanford Dogs (Aditya et al., 2011)	Perros (120 razas, 20K)	2.8GB	Imágenes 2D	Etiquetas de clase, Caja delimitadora	[73]
2009	Non-rigid world (Alexander et al., 2009)	Gatos, Perros, Lobos, Caballos, Leones, Gorilas	24MB	Malla 3D (sin textura)	No	[74]

Tabla. 2.3: Algunos de los conjuntos de datos públicos usados para la estimación de poses en animales en la actualidad. Fuente original modificada [9]

Capítulo 3

OBJETIVOS

En este capítulo, se detallan los objetivos generales planteados en este trabajo, así como los objetivos concretos que se perseguirán durante la realización del mismo.

3.1. Objetivo general

El objetivo general marcado para esta memoria es el estudio y aplicación de métodos de aprendizaje profundo que permitan la detección de poses en animales.

Este objetivo general implica la creación de un conjunto de datos que nos permita llevar a cabo la investigación. A continuación, y decidido el modelo o modelos que se usarán, se llevará a cabo el entrenamiento de los mismos para obtener los mejores resultados bajo distintas condiciones. Por último, será crucial la comparación de los resultados obtenidos para sacar las conclusiones necesarias sobre los modelos de aprendizaje profundo más óptimos para la estimación de poses en animales.

3.2. Objetivos específicos

Planteado el objetivo general con el que se da lugar a esta investigación, se detallan en esta sección los objetivos concretos y tangibles que se buscarán en este estudio.

Obtención de una visión general del campo de la ciencia de datos en general y el aprendizaje profundo en particular

Dado un conocimiento previo mínimo en el campo, en primer lugar se buscará obtener la visión concreta más completa posible de la ciencia de datos en general, y el aprendizaje profundo en particular. Para ello, se recorrerá el campo de estudio completo desde sus orígenes para situarnos en el estado del arte actual.

Adquisición de los conceptos implicados en el área de la identificación de poses

Presentándose un estudio relacionado con una rama concreta del aprendizaje profundo, la estimación de pose, se buscará adquirir de forma concreta los conceptos en lo que a esta tarea se refiere para abordar de forma correcta el problema planteado.

Conocimiento de los modelos habituales de aprendizaje profundo para la identificación de poses en animales

Conocidas las bases teóricas de la estimación de pose, deberá ser estudiado por tanto, los modelos de aprendizaje profundo que habitualmente se usan para esta tarea. Con ello, se abordarán las distintas posibilidades existentes, obteniendo un conocimiento concreto de características y funcionamiento de distintos modelos.

Adaptación, preparación y ejecución de modelos de aprendizaje profundo para la identificación de poses en animales

Una vez estudiados los recursos existentes para un problema como el que se plantea (Sección 2.6), y dada la inexistencia de un conjunto de datos consistente que sea extrapolable a nuestro problema, se planteará la creación de un conjunto de datos válido, y de la mejor calidad posible para entrenar los correspondientes modelos.

Para ello, se hará uso de las fuentes proporcionadas para el estudio concreto. Esta será una serie de vídeos correspondientes a distintos sujetos de estudio en condiciones por lo general similares. A partir de estos vídeos se creará el conjunto de anotaciones relativo, y se llevarán a cabo las acciones necesarias para obtener un conjunto de datos en el formato correcto para su posterior uso en el entrenamiento de modelos.

Sabiendo que es necesaria una adaptación previa de los modelos, así como los

datos necesarios para cada problema, será obligado realizar un proceso metódico de adaptación y preparación de modelos existentes, previos a la ejecución de los mismos.

Vistos los modelos y arquitecturas que en la actualidad se suelen utilizar para una tarea como la estimación de poses (Sección 2.7), y escogido sobre el que se experimentará en este trabajo, debemos realizar los entrenamientos correspondientes a partir del conjunto de datos tratado, y las configuraciones de hiperparámetros consideradas para la obtención de resultados.

Se buscará obtener un espacio de búsqueda lo más amplio posible, para poder sacar conclusiones consistentes, por lo que se entrenarán distintos modelos con distintas configuraciones de hiperparámetros, y distintas modificaciones al conjunto de datos y validación, algo que será esencial en el rendimiento obtenido durante el entrenamiento y sus resultados.

Análisis de resultados y conclusiones

Una vez realizado el proceso de experimentación, es de crucial importancia analizar los resultados obtenidos, y sacar conclusiones de los mismos, comparando distintas configuraciones posibles estudiadas y destacando las más fructíferas en el proceso de investigación realizado en base a cierto criterio y métricas.

Este será el último objetivo de esta investigación, en el que se espera que el proceso realizado durante el trabajo nos permita obtener unas conclusiones sólidas sobre la experimentación, permitiendo exponer al lector el mejor modelo de aprendizaje profundo obtenido para la tarea planteada.

Capítulo 4

MATERIALES Y MÉTODOS

4.1. Entorno de experimentación

Desde un primer momento se comenzará a trabajar en el sistema propio, sin embargo conocidas las limitaciones que supone el entrenamiento de modelos y la experimentación en un sistema de tan pocos recursos se comenzará a hacer uso de recursos cedidos por el grupo de investigación **SIMIDAT**. Concretamente se hará uso de una de las máquinas del **Instituto Andaluz Interuniversitario en Data Science and Computational Intelligence (DaSCI)** ubicada en el **Centro de Estudios Avanzados en Tecnologías de la Información y Comunicación (CEATIC)**.

Estas máquinas utilizadas serán del tipo **NVIDIA DGX-1** un servidor especializado para trabajar en problemas relacionados con la ciencia de datos en DL.

4.1.1. Especificaciones del sistema

El sistema **NVIDIA DGX-1** [3] usado plantea una configuración hardware integral para su uso en problemas de AI y DL involucrando un total de 8 NVIDIA Tesla V100 con 32GB de VRAM cada una, y sumando un total de 40960 núcleos CUDA. En la [Figura 4.1](#) vemos el rendimiento de la máquina usada frente a alguna otra máquina trivial con el mismo número de GPUs y frente a una máquina que hace únicamente uso de CPU, lo que nos da una buena idea del gran rendimiento que nos proporciona la máquina utilizada.

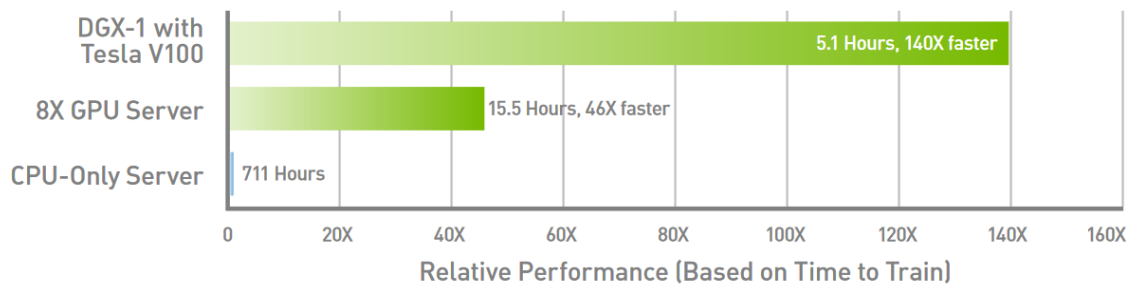


Figura 4.1: Entrenamiento de una red *ResNet-50*, para 90 épocas. Servidor CPU: Dual Xeon E5-2699v4, 2.6GHz. Fuente [3]

4.2. Lenguajes de programación

Durante diversos puntos a lo largo del presente estudio, será necesario hacer uso de lenguajes de programación para poder realizar ciertas tareas como el tratamiento de anotaciones. Será por esto de gran interés conocer qué herramientas y lenguajes se han utilizado, teniendo en cuenta experiencia previa en estas, lo que permitirá facilitar la tarea de escribir código, y la facilidad y flexibilidad que aporte el propio lenguaje para tareas propias del campo de estudio.

Se presentan a continuación los distintos lenguajes de programación usados y sobre los que se ha escrito código del relatado en la sección 4.6. Como es obvio, no se tendrán en cuenta aquellos lenguajes sobre los que no se haya trabajado directamente o aquellos cuyo uso venga inherente de la utilización de ciertas herramientas. Es decir, no se entrará a bajo nivel en la construcción y programación de herramientas utilizadas.

4.2.1. Python

El lenguaje de programación Python es ampliamente conocido en el ámbito de la ciencia de datos. No son pocas las aplicaciones que a diario hacen uso de este lenguaje por su facilidad de aprendizaje para usuarios noveles y por su amplio soporte y disponibilidad de librerías y módulos específicos enfocados en la ciencia de datos.

A lo largo de este proyecto, se ha hecho uso de numerosas librerías, y aunque no se entrará en detalle pues esto es detallado en las aplicaciones de código directamente, es importante mencionar algunas de las librerías más comúnmente usadas para aplicaciones del estilo y no nativas de Python. Estas son:

- **OpenCV**: se trata de una librería de código abierto ampliamente usada que incluye más de 2500 algoritmos optimizados para numerosas tareas de CV y ML. [75]
- **Matplotlib**: es una librería muy útil para la creación de visualizaciones de datos de forma dinámica, o estática para los datos y estadísticas. [76]
- **Albumentations**: es una librería rápida y flexible que permite el aumento de imágenes. Es ampliamente usada en la industria y en diversos trabajos de investigación en DL y competiciones de ML. [4]
- **NumPy**: es una librería especializada para el cálculo numérico y el análisis de datos, ofreciendo una gran rapidez frente a estructuras de datos tradicionales. [77]

4.2.2. Bash

El Bourne Again SHell, conocido como **bash** es un intérprete de comandos que funciona como puente entre el usuario y el kernel de Unix. Bash permite a los usuarios interactuar y controlar el sistema operativo haciendo uso de una serie de comandos de texto, además de ofrecer un lenguaje de scripting completo y robusto.

Es importante destacar que trabajamos en el entorno de experimentación que se relata en la sección 4.1, y que por tanto no contamos con una interfaz gráfica tradicional. Esto, y sumado a que ya se tenían ciertos conocimientos sobre el scripting en bash y la línea de órdenes de linux para el tratamiento de ficheros, y el uso de comandos propios del sistema operativo. Se hace uso de bash para tareas sencillas como la descarga automatizada de los vídeos que formarán el conjunto de entrenamiento (sección 4.6.1) o la unificación de forma automatizada de diferentes directorios con muchos frames y sus anotaciones (sección 4.6.4).

4.2.3. XML

El lenguaje de marcado extensivo en español o *extensive markup language*, comúnmente conocido por su acrónimo **XML** es un lenguaje de marcado que define un conjunto de reglas para la codificación de un documento. Se trata de una especificación de la **W3C**[78] como un lenguaje de propósito general. El hecho de ser de propósito general implica que no tiene etiquetas predefinidas, y estas deben ser definidas por el programador.

Aunque no se suele considerar un lenguaje de programación en sí mismo, será incluido en esta sección ya que el XML desempeña un papel fundamental en el desarrollo de software al proporcionar una forma estándar de estructurar y intercambiar datos. Esto lo hace relevante en numerosas ocasiones para los desarrolladores de software, y de hecho lo será para nuestra tarea, tal y como se estudiará en la sección [4.4.3](#).

4.2.4. YAML

Conocido como *YAML Ain't Markup Language* [\[79\]](#), se trata de un lenguaje de serialización de datos fácilmente legible, simple y que se puede usar en conjunto con cualquier lenguaje de programación, que suele ser utilizado en archivos de configuración, en nuestro caso será así y se relatará en la sección [4.3.1](#).

De la misma forma que XML, aunque no se suele considerar un lenguaje de programación en sí mismo, se incluirá en esta sección pues su uso, basado en sangrías, pares clave-valor y convenciones intuitivas hace que se puedan expresar estructuras de datos complejas en un formato sencillo para el programador. [\[80\]](#)

4.3. Modelos para la experimentación

Dado el estudio de algunos de los distintos modelos que en la actualidad existen para la estimación de poses visto en la sección [2.7](#), se presenta a continuación la elección que se utilizará para la experimentación en este estudio.

Se hará uso del modelo Yolo en su versión 8: **YoloV8**, por su gran facilidad de uso y eficiencia para un problema como el presentado. El hecho de contar con una extensa documentación, y numerosa literatura relativa al uso de esta herramienta, supone un punto muy a favor para la decisión de hacer uso de este modelo. Al no contar con experiencia previa en problemas de este tipo, el hecho de disponer una gran base documental es importante para el correcto desarrollo del trabajo.

Como punto de partida de nuestra experimentación, contaremos con tres de los modelos proporcionados por Ultralytics [\[59\]](#) para YOLOv8 para la tarea de estimación de poses. Se escogerán los siguientes tres modelos de entre los existentes para la tarea de estimación de poses, y relatados en la sección [2.7.2](#): **YOLOv8l-pose**, **YOLOv8x-pose**, **YOLOv8x-pose-p6**

Cabe destacar que aunque la capacidad de procesamiento de modelos mayores, en principio debería suponer unos mejores resultados, hay otros factores que se tendrán en cuenta la hora de escoger un modelo, como es la velocidad que el mismo tiene o la adaptabilidad a los recursos de los que se disponen para realizar la investigación. Es importante tener en cuenta esto en el siguiente desarrollo de la investigación, pues en el caso del modelo *YOLOv8x-pose-p6* que con gran capacidad de procesamiento y formado por un gran número de parámetros, quedará en algunos casos descartado del estudio por su gran requerimiento en recursos del sistema y memoria gráfica, haciendo que fallen algunos entrenamientos.

4.3.1. Formato YAML del conjunto de datos

Para lanzar el entrenamiento del modelo será necesario indicar el conjunto de datos a usar para la ejecución. Dentro del framework de Ultralytics esto se hace a partir de un fichero del tipo YAML con la siguiente estructura [81]:

```
1 # Train/val/test sets as 1) dir: path/to/imgs, 2) file: path/to/imgs.txt, or 3) list: [path/to/imgs1,
  path/to/imgs2, ...]
2 path: ../datasets/coco8-pose # dataset root dir
3 train: images/train # train images (relative to 'path') 4 images
4 val: images/val # val images (relative to 'path') 4 images
5 test: # test images (optional)
6
7 # Keypoints
8 kpt_shape: [17, 3] # number of keypoints, number of dims (2 for x,y or 3 for x,y,visible)
9 flip_idx: [0, 2, 1, 4, 3, 6, 5, 8, 7, 10, 9, 12, 11, 14, 13, 16, 15]
10
11 # Classes dictionary
12 names:
13   0: person
```

Listado 4.1: Formato del fichero de configuración del conjunto de datos

En este fichero de configuración, el cual habrá que adaptar a nuestro conjunto de datos concreto y que será configurado en la sección 4.6.2.

Para ello, tenemos que tener en cuenta que `train` y `val` deberán contener los directorios que contienen las imágenes de los conjuntos de entrenamiento y validación respectivamente. El valor `names` es un diccionario de los nombres de las clases presentes, en nuestro caso, para el sujeto de prueba será un único valor y corresponderá a la lombriz.

Contaremos también como el parámetro `kpt_shape` el cual indicará el formato de la anotación de puntos clave. Esto es, el primer valor numérico corresponderá con el

número de puntos clave anotados, en nuestro caso concreto será 1 correspondiente a la cabeza. El segundo valor numérico corresponderá con el número de dimensiones que podrá ser dimensión 2 o 3. Se relata en qué consiste el valor de la dimensión en la sección [4.6.2](#).

De forma opcional, se podría configurar `flip_idx` si tuviésemos varios puntos que fuesen simétricos entre sí. Al no ser el caso ignoraremos este campo en el fichero final.

4.3.2. Ejecución del modelo

Una vez se tiene el fichero de configuración YAML correspondiente a los datos que usará el modelo para su entrenamiento y validación, es importante conocer la configuración necesaria para el entrenamiento, en la que se incluirán los diferentes hiperparámetros y configuraciones, que influirán en el rendimiento del mismo y velocidad de entrenamiento entre otros muchos factores.

Dada la importancia de encontrar una correcta configuración de hiperparámetros para un correcto desempeño del modelo entrenado, y como detallado en la sección [4.10](#) serán estos los que se modifiquen y comparen a lo largo de la experimentación del presente trabajo, unido a modificaciones en el conjunto de datos, con el fin final de encontrar la mejor configuración posible.

A continuación se describen algunas de las distintas configuraciones que pueden ser indicadas al entrenar un modelo del tipo **YOLO** haciendo uso del framework de Ultralytics. Dado que la posible configuración tiene muchas distintas posibilidades, se incluirán únicamente en la [Tabla 4.3.2](#) las más relevantes, quedando las demás configuraciones en su valor por defecto.

Cabe destacar que aunque durante el proceso de experimentación de este trabajo no se documentarán las distintas modificaciones de todos los hiperparámetros que en la tabla se reflejan, sí que se han realizado pruebas con todos o la mayoría de ellos. En este trabajo nos centraremos únicamente en obtener los mejores resultados modificando los hiperparámetros que se describirán en la sección [4.10](#).

Otras configuraciones necesarias para el entrenamiento de modelos, entre las que podemos incluir por ejemplo `device` para indicar sobre qué unidades gráficas se repartirá la carga computacional para entrenar el modelo, `project` para indicar el nombre del directorio del proyecto o `name` para el nombre de la ejecución del modelo, son

interesantes para la organización del experimentador durante el proceso de investigación. Sin embargo son triviales y no influyentes en el desempeño del modelo, por lo que no se describirán en este trabajo.

Argumento	Por defecto	Descripción
<code>model</code>	None	Especifica el archivo del modelo para el entrenamiento. Acepta una ruta a un modelo preentrenado <code>.pt</code> o un archivo de configuración <code>.yaml</code> . Esencial para definir la estructura del modelo o inicializar pesos.
<code>data</code>	None	Ruta al archivo de configuración del conjunto de datos (por ejemplo, <code>coco8.yaml</code>). Detallado en la sección 4.3.1
<code>epochs</code>	100	Número total de épocas de entrenamiento.
<code>patience</code>	100	Número de épocas a esperar sin mejora en las métricas de validación antes de detener el entrenamiento. Ayuda a prevenir el sobreajuste deteniendo el entrenamiento cuando el rendimiento se estanca.
<code>batch</code>	16	Tamaño del lote para el entrenamiento. <code>AutoBatch</code> (<code>batch=-1</code>) ajusta dinámicamente el tamaño del lote basado en la disponibilidad de memoria GPU.
<code>imgsz</code>	640	Tamaño de imagen objetivo para el entrenamiento. Todas las imágenes se redimensionan a esta dimensión antes de ser introducidas en el modelo. Afecta la precisión del modelo y la complejidad computacional.
<code>save</code>	True	Habilita el guardado de puntos de control del entrenamiento y los pesos finales del modelo. Útil para reanudar el entrenamiento o la implementación del modelo.
<code>device</code>	None	Especifica el dispositivo computacional para el entrenamiento: una sola GPU (<code>device=0</code>), múltiples GPUs (<code>device=0,1</code>), CPU (<code>device=cpu</code>), o MPS para Apple silicon (<code>device=mps</code>). Nos será de gran utilidad para un entorno de experimentación como el que se utiliza, y detallado en la sección 4.1
<code>project</code>	None	Nombre del directorio del proyecto donde se guardan los resultados del entrenamiento. Permite un almacenamiento organizado de diferentes experimentos.

Argumento	Por defecto	Descripción
name	None	Nombre de la ejecución de entrenamiento. Utilizado para crear un subdirectorio dentro de la carpeta del proyecto, donde se almacenan los registros y resultados del entrenamiento.
exist_ok	False	Si es True, permite sobrescribir un directorio de proyecto/nombre existente. Útil para la experimentación iterativa sin necesidad de borrar manualmente salidas anteriores.
pretrained	True	Determina si se inicia el entrenamiento desde un modelo preentrenado. Puede ser un valor booleano o una ruta de cadena a un modelo específico del cual cargar los pesos. Mejora la eficiencia del entrenamiento y el rendimiento del modelo.
optimizer	'auto'	Elección del optimizador para el entrenamiento. Las opciones incluyen SGD, Adam, AdamW, NAdam, RAdam, RMSProp, etc., o auto para selección automática basada en la configuración del modelo. De forma habitual, se hará uso de su valor por defecto.
verbose	False	Habilita la salida detallada durante el entrenamiento, proporcionando registros detallados y actualizaciones de progreso. Útil para la depuración y el monitoreo cercano del proceso de entrenamiento.
close_mosaic	10	Desactiva la augmentación de datos de mosaico en las últimas N épocas para estabilizar el entrenamiento antes de completarlo. Configurar a 0 desactiva esta función.
resume	False	Reanuda el entrenamiento desde el último punto de control guardado. Carga automáticamente los pesos del modelo, el estado del optimizador y el conteo de épocas, continuando el entrenamiento sin problemas.
freeze	None	Congela las primeras N capas del modelo o capas específicas por índice, reduciendo el número de parámetros entrenables. Útil para ajuste fino o aprendizaje por transferencia.

Argumento	Por defecto	Descripción
lr0	0.01	Tasa de aprendizaje inicial (es decir, SGD=1E-2, Adam=1E-3). Ajustar este valor es crucial para la convergencia del modelo y la tasa de aprendizaje óptima puede variar según el modelo y los datos.
lrf	0.01	Factor final de tasa de aprendizaje para los programadores de tasa de aprendizaje. Dicta la tasa de aprendizaje mínima al final del entrenamiento.
warmup_epochs	3.0	Número de épocas de calentamiento con tasa de aprendizaje linealmente incrementada desde 0 hasta lr0, ayudando a un entrenamiento estable inicial.
warmup_momentum	0.8	Impulso inicial durante el calentamiento, ajustado para proporcionar una estabilidad inicial en la convergencia del modelo.

Tabla. 4.1: Descripción de los posibles argumentos en el entrenamiento del modelo

4.4. Anotaciones

Uno de los principales problemas en el campo de visión por computador en particular, o el aprendizaje profundo en general, es la anotación de datos para entrenar el modelo con el que se requiera trabajar. En nuestro caso concreto, dada la naturaleza del problema es prácticamente inexistente cualquier dataset en lo relativo a lombrices. Sí podemos encontrar algunos recursos como TierpsyTracker, que trabajan con datos relativos a *Caenorhabditis Elegans* (*C.elegans*) como ya se ha estudiado en la sección 2.6, y que no es extrapolable a nuestro campo de estudio dada el tipo de anotación, el objetivo del problema [82] y el comportamiento del sujeto.

Debida a la escasez de datos disponibles, la principal y única fuente de datos anotados utilizados serán los creados por el experimentador a partir de los vídeos de los sujetos de prueba proporcionados. La creación de los datasets utilizados, así como su detallada composición será concretada más adelante.

En esta sección, se describirá el proceso seguido para obtener un conjunto de anotaciones, válido y suficiente para la investigación, estudiando distintas posibilidades de anotación existentes así como los distintos formatos posibles de los que nos podemos

valer para generar el conjunto de datos con el que entrenar el posterior modelo.

4.4.1. Anotación de vídeos ejemplo

Como punto de partida para la investigación, se buscan distintas herramientas de anotación que faciliten el trabajo para anotar grandes volúmenes de imágenes. De entre todas las existentes, destacaremos dos: **Roboflow** y **CVAT**, por su facilidad de uso y aprendizaje considerando el hecho de no tener experiencia en una tarea similar.

A priori, ambas herramientas ponen a disposición del usuario distintas formas de anotación que pueden ser útiles para afrontar nuestro problema, y sin una mayor idea de cómo se afrontará el caso concreto para entrenar el modelo escogido, se decide probar ambas herramientas.

Anotación con Roboflow

Roboflow es un conjunto de herramientas para diversos proyectos de ML y CV. Una de las herramientas más interesantes que nos proporciona es *Roboflow Annotate*, para la anotación de imágenes y vídeos. Es por esto, y por su aparente facilidad de uso que se decide probarla.

Desde un principio, se hará diversas pruebas de la plataforma realizando anotaciones de diferentes tipos para una familiarización con la plataforma, sin embargo tras descubrir un poco su funcionamiento, nos encontramos con el problema de que en el momento de la creación del dataset, Roboflow no proporciona una forma sencilla de anotación de Keypoints, cómo indican en su plataforma con las siguientes palabras:

Currently, we only support Object Detection (Bounding Box) and Single-Class Classification projects. We're prioritizing based on user demand. You can log your support and request this feature by adding your vote.

(Mensaje mostrado en la plataforma de Roboflow)

Se decide usar el método desarrollado en uno de los pocos artículos encontrados al respecto [83]. Este consiste en anotar cajas limítrofes para el cuerpo y para la cabeza como se aprecia en la [Figura 4.2](#). Con estos datos anotados, exportaremos el dataset para utilizar un script que convertirá las cajas en puntos. Tras explorar esta posibilidad,

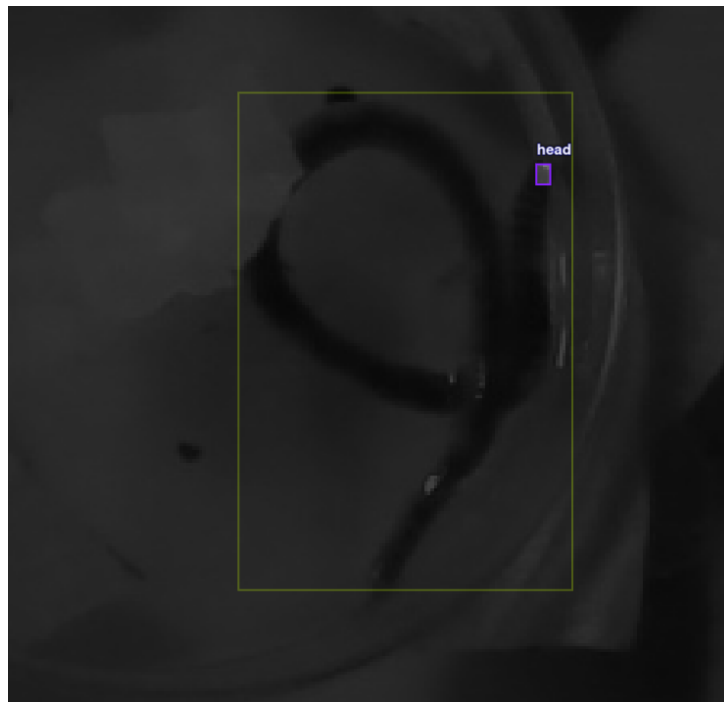


Figura 4.2: Visualización de anotación con Roboflow

y desarrollar un pequeño conjunto de datos, se comprueba su ineficiencia y falta de exactitud en las transformaciones, lo que llevaría a dificultades en el trabajo posterior. Es por esto que se decide dejar a un lado este método para centrarnos en el siguiente.

Anotación con CVAT

CVAT es una plataforma de código abierto para la anotación de imágenes y vídeos, por lo que proporciona todas las herramientas necesarias para nuestra tarea.

Una de las características más importantes en relación a las anotaciones con esta herramienta es la posibilidad de anotar mediante interpolación lineal. De esta forma, podemos aproximar de forma significativa anotaciones que se mueven, en frames sucesivos dentro del mismo vídeo. Esto es, si anotamos de forma adecuada los frames 1 y 5 dentro del vídeo, para los frames 2, 3 y 4 se realizará una interpolación de las coordenadas entre el frame anterior y posterior, aproximando de forma aceptablemente precisa la posición del keypoint, y por tanto, ahorrando una cantidad de trabajo considerable. Para el uso de esto de la forma más adecuada posible, hay que tener en cuenta, y de forma obvia, que cuantos menos frames haya entre ambos puntos usados para la interpolación, más exacta será el punto interpolado.

Esto se puede ver de forma sencilla en la [Figura 4.3](#), [Figura 4.4](#) y [Figura 4.5](#). La

primera y la última imagen corresponden a frames anotados de forma manual, la [Figura 4.4](#) corresponde a uno de los frames intermedios interpolados por la función matemática, y que como vemos, aunque se acerca, no es anotado con exactitud. Para el sujeto 26 pues no se mueve de forma lineal, algo que sí hace el sujeto 25.

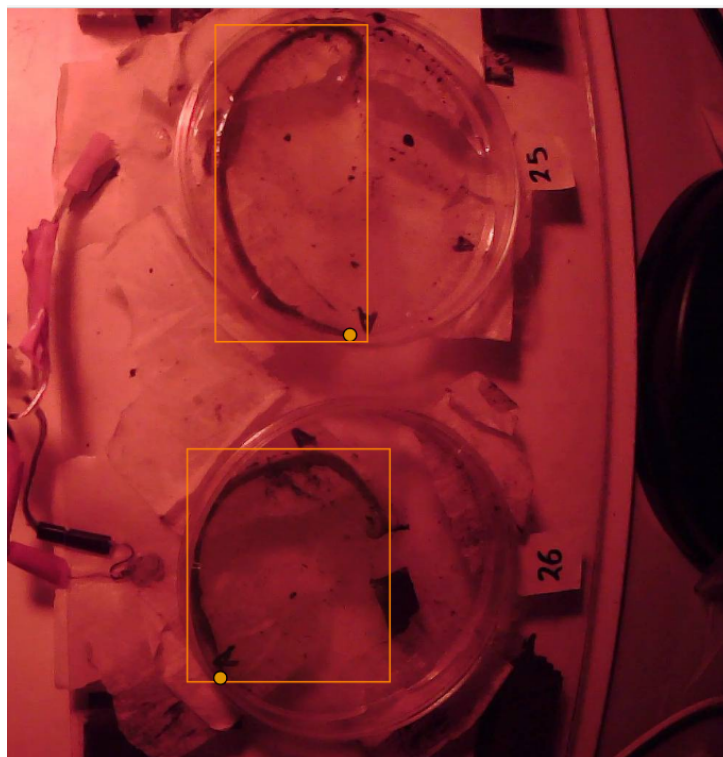


Figura 4.3: Visualización de la anotación frame 68. Anotación a mano.

Vemos de forma ampliada en la [Figura 4.6](#), cómo debería aproximarse la anotación a la interpolada, para el sujeto 26. En verde la anotación más precisa posible, mientras que en amarillo vemos la anotación real que se interpola.

Entendemos entonces, que aunque facilita el trabajo de anotación de forma significativa, si se quieren anotaciones de la mayor precisión posible han de corregirse gran parte de estas interpolaciones a mano, lo que incrementa de forma abismal el tiempo que llevará el proceso de anotación de un único vídeo.

En nuestro caso, es habitual que el sujeto de estudio se mueva de forma no lineal, por lo que en muchos caso la interpolación falla y hay que anotar un número de frames mayor de lo que sería habitual para otro problema. Sin embargo, tras ponerlo en práctica es cierto que facilita mucho el trabajo de anotación respecto a la herramienta utilizada antes, es por esto que se decide usar *CVAT* como la herramienta principal para crear anotaciones a pesar de sus complicaciones detalladas a continuación.

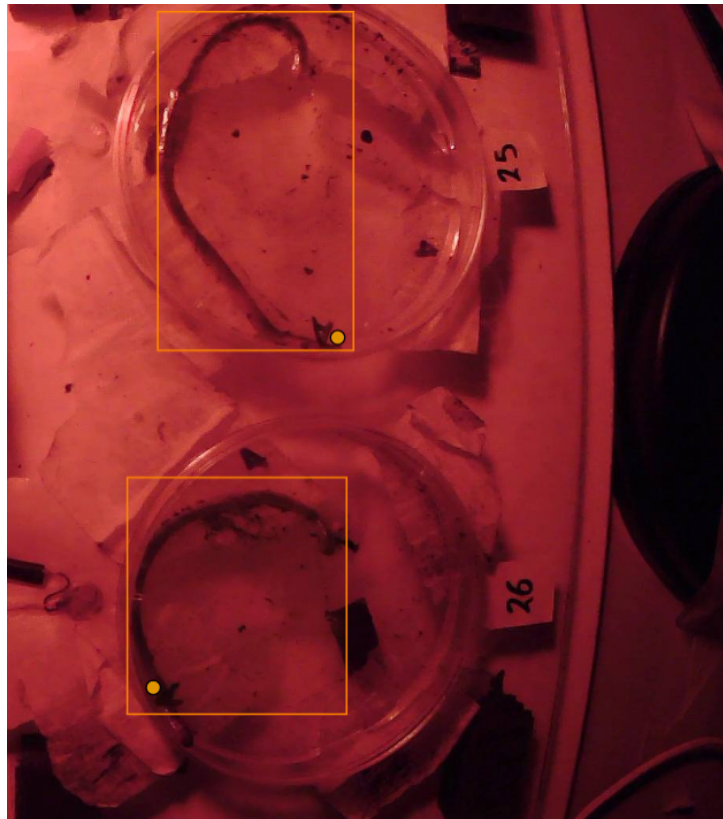


Figura 4.4: Visualización de la anotación frame 247. Anotación interpolada.

4.4.2. Exportar anotaciones desde la plataforma

Uno de los principales problemas relacionados con el uso de la herramienta de anotación escogida es el formato que este produce. En nuestro caso, y como se usará *CVAT* según lo relatado en la sección 4.4.1, se nos permitirá únicamente exportar los datos anotados de forma completa en dos formatos específicos. Esto es por la naturaleza de las anotaciones realizadas, concretamente para los puntos, y dado un tipo de problema dentro del aprendizaje profundo como la estimación de pose, la cual no está tan ampliamente extendida, es difícil encontrar demasiado soporte para tareas tan concretas como la nuestra, y sobre todo para anotaciones creadas sobre vídeos[84].

Encontramos entonces los siguientes formatos para exportar los datos creados:

- *CVAT for video 1.1*. Se trata de un formato de exportación propio de la plataforma. Su estructura será la presentada en la figura Figura 4.7 [85]. El formato del archivo de anotaciones será el siguiente:

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <annotations>
3 ...
4 <track id="Number: idofthetrack(doesn't have any special meeting)" label="String:
```

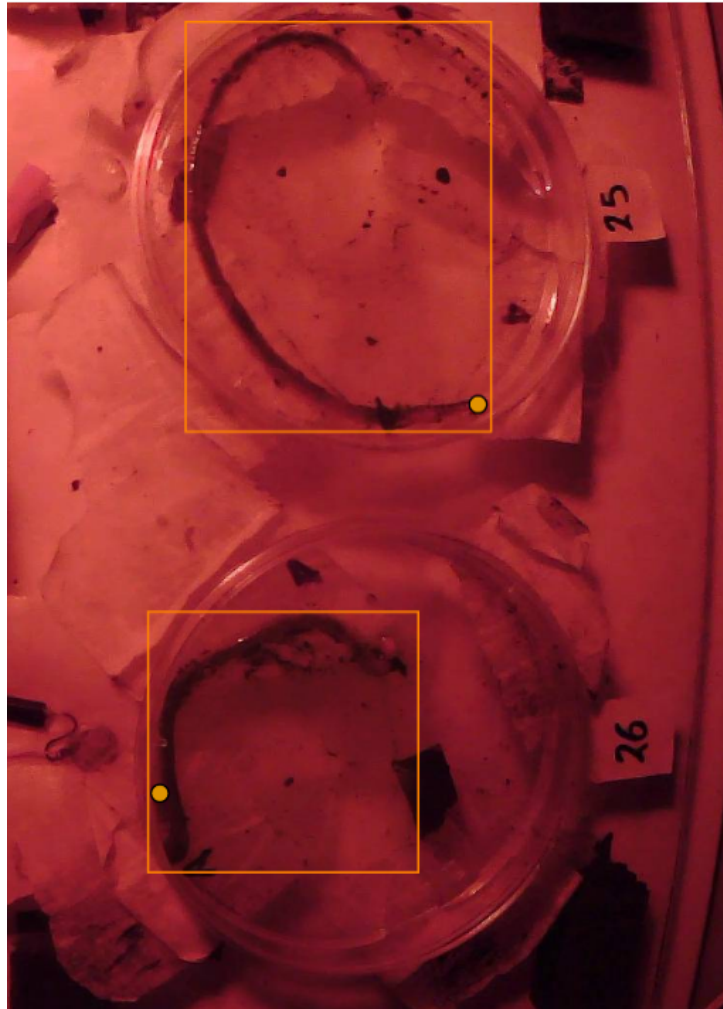


Figura 4.5: Visualización de la anotación frame 547. Anotación a mano.

```

    theassociatedlabel"source="manualorauto">
5 <boxframe="Number:frame"xtl="Number:float"ytl="Number:float"xbr="Number:float"ybr="
  Number:float"outside="Number:0-False,1-True"occluded="Number:0-False,1-True"
  keyframe="Number:0-False,1-True">
6 <attributename="String:anattributename">String:theattributevalue</attribute>
7 ...
8 </box>
9 <polygonframe="Number:frame"points="x0,y0;x1,y1;..."outside="Number:0-False,1-True"
  occluded="Number:0-False,1-True"keyframe="Number:0-False,1-True">
10 <attributename="String:anattributename">String:theattributevalue</attribute>
11 </polygon>
12 <polylineframe="Number:frame"points="x0,y0;x1,y1;..."outside="Number:0-False,1-True"
  "occluded="Number:0-False,1-True"keyframe="Number:0-False,1-True">
13 <attributename="String:anattributename">String:theattributevalue</attribute>
14 </polyline>
15 <pointsframe="Number:frame"points="x0,y0;x1,y1;..."outside="Number:0-False,1-True"
  occluded="Number:0-False,1-True"keyframe="Number:0-False,1-True">
16 <attributename="String:anattributename">String:theattributevalue</attribute>
17 </points>
18 <maskframe="Number:frame"outside="Number:0-False,1-True"occluded="Number:0-False,1-
  True"rle="RLEmask"left="Number:leftcoordinateoftheimagewherehemaskbegins"top="
  Number:topcoordinateoftheimagewherehemaskbegins"width="Number:widthofthemask"
  height="Number:heightofthemask"z_order="Number:z-orderoftheobject">

```

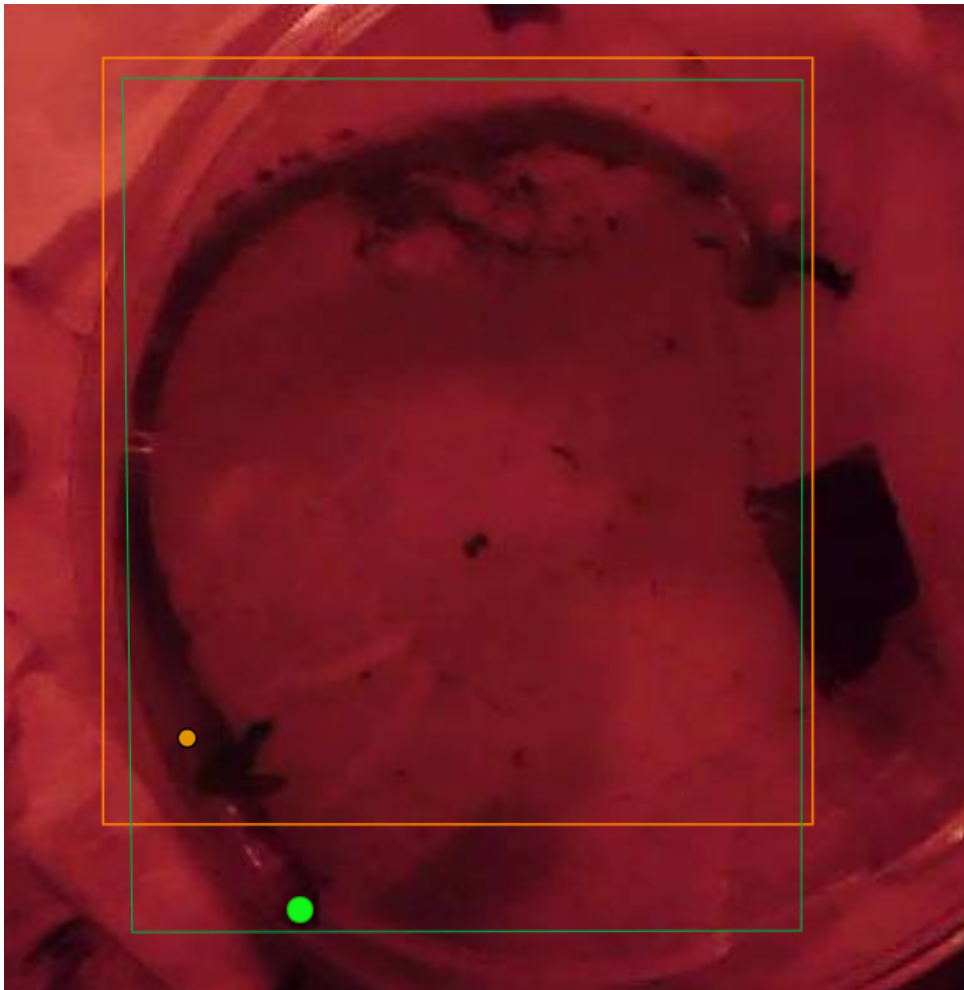


Figura 4.6: Comparación de anotación interpolada y manual

```

19 </mask>
20 ...
21 </track>
22 <trackid="Number:idofthetrack(doesn'thaveanyspecialmeeting)"label="String:
    theassociatedlabel"source="manualorauto">
23 <skeletonframe="Number:frame"keyframe="Number:0-False,1-True">
24 <pointslabel="String:theassociatedlabel"outside="Number:0-False,1-True"occluded="
    Number:0-False,1-True"keyframe="Number:0-False,1-True"points="x0,y0;x1,y1">
25 </points>
26 ...
27 </skeleton>
28 ...
29 </track>
30 ...
31 </annotations>

```

Listado 4.2: Formato anotaciones CVAT for video 1.1

- *Datumaro 1.0*. Es en realidad un framework para el manejo de conjunto de datos [86]. La estructura es la representada en [Figura 4.8](#), en el que cada uno de los archivos con extensión *.json* tendrá el siguiente formato:

```
1 {
2   "info": {},
3   "categories": {
4     "label": {
5       "labels": [
6         {
7           "name": "label_0",
8           "parent": "",
9           "attributes": []
10        },
11        {
12          "name": "label_1",
13          "parent": "",
14          "attributes": []
15        }
16      ],
17      "attributes": []
18    }
19  },
20  "items": [
21    {
22      "id": "img1",
23      "annotations": [
24        {
25          "id": 1,
26          "type": "bbox",
27          "attributes": {},
28          "group": 1,
29          "label_id": 0,
30          "z_order": 0,
31          "bbox": [1.0, 2.0, 3.0, 4.0]
32        },
33        {
34          "id": 2,
35          "type": "mask",
36          "attributes": {},
37          "group": 1,
38          "label_id": 0,
39          "rle": {
40            "counts": "d0d0:F\\0",
41            "size": [10, 10]
42          },
43          "z_order": 0
44        }
45      ]
46    }
47  ]
48 }
```

Listado 4.3: Formato anotaciones Daturaro 1.0

Desde un primer momento este tipo de anotación parece ser interesante, ya que permite la conversión de datos para modelos *YOLO*, sin embargo, tras investigar en profundidad, únicamente será compatible para cajas delimitadoras (**bounding boxes**), por lo que a la hora de la verdad se descarta por no ser posible la conversión de las anotaciones de puntos.

```
taskname.zip/  
├─ images/  
│   ├── frame_000000.png  
│   └── frame_000001.png  
└─ annotations.xml
```

Figura 4.7: Formato *CVAT for video 1.1* de las anotaciones descargadas

```
<archive_name>.zip/  
├─ annotations/  
│   ├── subset1.json # descripción completa de las clases y todos los  
│   │   │   objetos del dataset  
│   └── subset2.json # descripción completa de las clases y todos los  
│       │   objetos del dataset
```

Figura 4.8: Formato *Datamaro for video 1.0* de las anotaciones descargadas

Es importante comentar que la posibilidad de exportar en uno de los formatos más ampliamente usados para tareas como la nuestra, el formato *COCO* y concretamente en su versión para la anotación de puntos. Conocida la existencia de un recurso proporcionado por los desarrolladores de *YOLO* el cual permite realizar una conversión automática de anotaciones en formato *COCO* hacia el formato requerido para entrenar un modelo *YOLO*: *COCO2YOLO* [87]. Sin embargo este formato está pensado para esqueletos completos, y sobre todo centrado en la tarea de estimación de poses en humanos, un problema al que ya se realiza un acercamiento en las secciones 2.5.2 y 2.5.3.

Escogido entonces el formato en el que se exportarán los datos, y dado que no existe a priori otra opción, se deberá realizar una conversión de los datos en formato *CVAT for video 1.1* exportados a un formato que permita entrenar el modelo posteriormente. En primera instancia, el proceso de conversión del mismo parece trivial, sin embargo, va tomando complejidad conforme se avanza en el mismo.

4.4.3. Formato de las anotaciones escogidas

Con el objetivo de acercar al lector hacia un mayor entendimiento del formato concreto de las anotaciones que se van a tratar, y que de forma obvia han de ser entendidas para la creación de un software que permita su correcto tratamiento y posterior formateo para el entrenamiento de los modelos, se presenta esta sección.

Aunque la vista general del fichero *xml* exportado ya se ha realizado en la sección anterior, con el objetivo de que el lector pueda alcanzar un mayor entendimiento del

proceso realizado para el tratamiento de anotaciones que se realizará en la sección 4.6, a continuación se describe de forma detallada la información contenida y la cual tendrá que ser tratada para la creación de un dataset.

En primer lugar, dentro del fichero encontraremos una etiqueta `annotations` la cual contendrá toda la información:

```
1 <annotations>
2 ...
3 </annotations>
```

Listado 4.4: Formato de las anotaciones 1

En el siguiente nivel, encontraremos en primer lugar una etiqueta `version` con el identificador de la versión del tipo de anotación escogida; una etiqueta `meta`, con los metadatos asociados al fichero de anotación como nombre del trabajo de anotación, fecha y hora de creación y de última modificación, el propietario, el número de frames, el tamaño de dichos frames, etc:

```
1 <meta>
2   <task>
3     <id>351059</id>
4     <name>habsujetos39y40.mp4_annotation</name>
5     <size>3300</size>
6     <mode>interpolation</mode>
7     <overlap>5</overlap>
8     <bugtracker></bugtracker>
9     <created>2023-10-29 13:45:19.411450+00:00</created>
10    <updated>2023-10-29 15:24:16.188275+00:00</updated>
11    <subset>default</subset>
12    <start_frame>0</start_frame>
13    <stop_frame>3299</stop_frame>
14    <frame_filter></frame_filter>
15    <segments>
16      <segment>
17        <id>382727</id>
18        <start>0</start>
19        <stop>3299</stop>
20        <url>https://app.cvat.ai/api/jobs/382397</url>
21      </segment>
22    </segments>
23    <owner>
24      <username>ace00029</username>
25      <email>ace00029@red.ujaen.es</email>
26    </owner>
27    <assignee></assignee>
28    <labels>
29      <label>
30        <name>worm_box</name>
31        <color>#f98911</color>
32        <type>rectangle</type>
33        <attributes>
34          </attributes>
35      </label>
```

```

36     <label>
37         <name>head</name>
38         <color>#ba9109</color>
39         <type>points</type>
40         <attributes>
41         </attributes>
42     </label>
43 </labels>
44 <original_size>
45     <width>1920</width>
46     <height>1080</height>
47 </original_size>
48 <source>habsujetos39y40.mp4</source>
49 </task>
50 <dumped>2024-05-30 15:33:52.014961+00:00</dumped>
51 </meta>

```

Listado 4.5: Formato de los metadatos del fichero de anotaciones

La información contenida en esta etiqueta es de gran importancia para la organización y automatización de diversas fuentes de datos a lo largo de la experimentación, lo que nos permitirá identificar de forma sencilla a qué video pertenece el fichero de anotación con el que estamos trabajando. De la misma forma, uno de los campos más importantes es `original_size` el cual nos proporciona las dimensiones de las imágenes usadas, que serán necesarias para el posterior tratamiento.

A continuación, encontraremos ya las etiquetas correspondientes a las anotaciones:

```

1  ...
2  <track id="0" label="worm_box" source="manual">
3      ...
4  </track>
5  <track id="1" label="worm_box" source="manual">
6      ...
7  </track>
8  <track id="2" label="head" source="manual">
9      ...
10 </track>
11 <track id="3" label="head" source="manual">
12     ...
13 </track>
14 </annotations>

```

Listado 4.6: Etiquetas track del fichero de anotaciones

Como se puede apreciar, contaremos con 4 `track` diferentes, los cuales habrá que tratar en el futuro. Cada una de estas etiquetas corresponderá a una anotación concreta sobre un sujeto concreto. Por ejemplo el track con `id=0` y cuyo `label` corresponde con `worm_box` corresponde con las anotaciones realizadas para la caja delimitante para una de las lombrices. Gracias a la herramienta relatada en la sección 4.6.2 podremos

identificar a cuál de los dos sujetos corresponderá. De la misma forma ocurrirá con el track correspondiente a las anotaciones de tipo `head`.

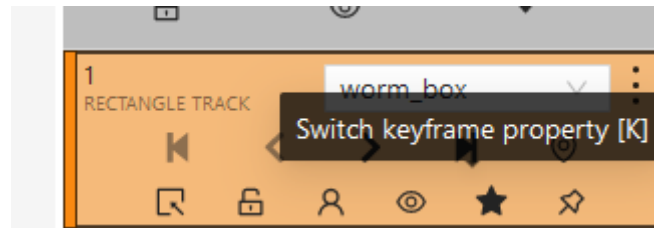
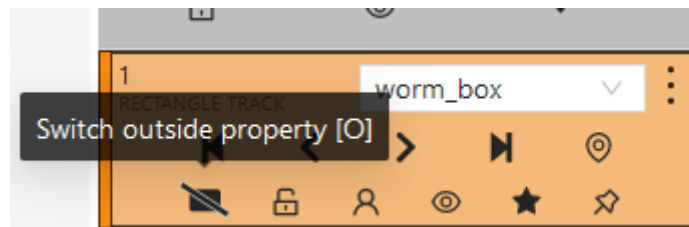
A continuación se detalla el formato genérico de las anotaciones de tipo caja delimitante que nos encontraremos:

```
1 <box frame="22" keyframe="1" outside="0" occluded="0" xtl="1220.00" ytl="171.87" xbr="
  1457.33" ybr="390.67" z_order="0"> </box>
2 <box frame="23" keyframe="0" outside="0" occluded="0" xtl="1220.37" ytl="171.87" xbr="
  1457.33" ybr="391.55" z_order="0"> </box>
3 <box frame="24" keyframe="0" outside="0" occluded="0" xtl="1220.73" ytl="171.87" xbr="
  1457.33" ybr="392.43" z_order="0"> </box>
4 <box frame="25" keyframe="0" outside="0" occluded="0" xtl="1221.10" ytl="171.87" xbr="
  1457.33" ybr="393.31" z_order="0"> </box>
5 ....
```

Listado 4.7: Formato de la anotación de caja delimitante

Podemos encontrar las anotaciones generadas para los 4 primeros frames. Es importante destacar que las anotaciones comienzan en el frame 22 pues es el primero anotado en el vídeo. De forma general, encontraremos que cada anotación está incluida en una etiqueta del tipo `box` cuyos atributos son los siguientes:

- `frame`: Número de frame dentro del vídeo.
- `keyframe`: Booleano que indica si el frame está marcado como *keyframe* o frame importante. De forma automática serán marcados como importantes aquellos frames anotados manualmente y no interpolados. En la [Figura 4.9](#) se aprecia cómo se vería esto en la plataforma de anotación.
- `outside`: Booleano que indica si el frame está marcado como fuera de la imagen. Por lo general no consideraremos esta propiedad pues los sujetos se encontrarán siempre dentro de la imagen analizada. En la [Figura 4.10](#) se aprecia cómo se vería esto en la plataforma de anotación.
- `occluded`: Booleano que indica si el frame está marcado como ocluido o tapado por otro objeto. Al tratarse de un entorno de pruebas controlado, el sujeto no es esta oculto en su totalidad en ningún momento, por lo que esta propiedad no se usará a menudo. En la [Figura 4.11](#) se aprecia cómo se vería esto en la plataforma de anotación.
- `xtl`: Coordenada X_1 de la caja.
- `ytl`: Coordenada Y_1 de la caja.
- `xbr`: Coordenada X_2 de la caja.

Figura 4.9: Uso de la propiedad *keyframe* en la plataformaFigura 4.10: Uso de la propiedad *outside* en la plataforma

- *ybr*: Coordenada Y_2 de la caja.
- *z_order*: Orden en el eje Z. No será considerado para nuestro problema.

A continuación se detalla el formato genérico de las anotaciones de tipo punto que nos encontraremos:

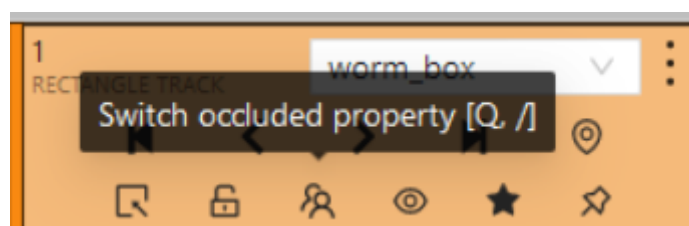
```

1 <points frame="22" keyframe="1" outside="0" occluded="0" points="1225.33,302.67" z_order=
  "0"> </points>
2 <points frame="23" keyframe="0" outside="0" occluded="0" points="1226.49,304.29" z_order=
  "0"> </points>
3 <points frame="24" keyframe="0" outside="0" occluded="0" points="1227.65,305.91" z_order=
  "0"> </points>
4 <points frame="25" keyframe="0" outside="0" occluded="0" points="1228.81,307.54" z_order=
  "0"> </points>
5 ....

```

Listado 4.8: Formato de la anotación de punto

De forma general, encontraremos que cada anotación está incluida en una etiqueta del tipo *points* cuyos atributos son los siguientes:

Figura 4.11: Uso de la propiedad *occluded* en la plataforma

- **frame**: Número de frame dentro del vídeo.
- **keyframe**: Booleano que indica si el frame está marcado como *keyframe* o frame importante. De forma automática serán marcados como importantes aquellos frames anotados manualmente y no interpolados.
- **outside**: Booleano que indica si el frame está marcado como fuera de la imagen. Por lo general no consideraremos esta propiedad pues los sujetos se encontrarán siempre dentro de la imagen analizada.
- **occluded**: Booleano que indica si el frame está marcado como ocluido o tapado por otro objeto. Al tratarse de un entorno de pruebas controlado, el sujeto no es esta oculto en su totalidad en ningún momento, por lo que esta propiedad no se usará a menudo.
- **points**: Coordenadas X,Y del punto dentro de la imagen.
- **z_order**: Orden en el eje Z. No será considerado para nuestro problema.

Una vez conocemos el formato y las características concretas de las anotaciones exportadas, podemos proceder a su tratamiento para obtener un formato válido que nos permita entrenar el modelo deseado.

4.5. Técnica de aumento para el conjunto de datos

El rendimiento de un modelo de aprendizaje profunda viene dado en gran medida por el número de datos disponibles para su entrenamiento. De forma concreta, los algoritmos de visión por computador basan su funcionamiento y gran rendimiento en que tienen disponible conjuntos de datos lo suficientemente amplios como para ser capaces de generalizar.

En casos concretos en los que el conjunto de datos inicial es reducido, el proceso de aumento de datos juega un papel crucial para obtener un modelo que presente un desempeño correcto.

El **aumento de datos**, conocido en inglés como *data augmentation* es un conjunto de técnicas usadas para incrementar el tamaño y la calidad del conjunto de datos, manteniendo las anotaciones originales. [88]

Si consideramos el caso concreto de estudio, parece obvia la necesidad de un conjunto de datos mayor del que a priori podríamos tener, pues las anotaciones serán de

elaboración propia, lo que supone un gran esfuerzo, pero sobre todo una gran cantidad de tiempo en una tarea trivial. Por motivos de eficiencia, se anotará un conjunto reducido de frames, para el que después se aplicarán técnicas de aumento de datos para hacerlo crecer, y contar con un mayor número de casos posibles que permitan al modelo generalizar de la mejor forma posible.

Se usará la librería *Albumentations* [4], que aplicará una línea de aumentación para frames y anotaciones de forma simultánea, aplicando una serie de transformaciones a las imágenes, y haciendo que las anotaciones originales se correspondan con las modificadas.

Algunos ejemplos de las aumentaciones usadas son transformaciones afines, cambios en la perspectiva, giros en la imagen, cambio en el canal de color o cambio en el brillo y contraste de la imagen. Se puede ver de forma clara en la [Figura 4.12](#) de la documentación oficial.

De forma concreta, el código usado y presente en la sección [4.6.5](#) contiene las siguientes transformaciones:

```
1 transformObject = A.Compose([
2     A.Perspective(p=1),
3     A.Affine(rotate=[-360, 360], rotate_method="ellipse", p=1),
4     A.OneOf([
5         A.ChannelShuffle(p=0.1),
6         A.HueSaturationValue(p=0.5),
7         A.RGBShift(r_shift_limit=[-20, 80], p=0.7)
8     ], p=0.8),
9     A.OneOf([
10        A.Equalize(p=0.3),
11        A.RandomBrightnessContrast(p=0.5)
12    ], p=0.4)
13 ]
14 ]
```

Listado 4.9: Transformaciones aplicadas en la línea de aumentación del conjunto de datos

En la que vemos que la función `Compose` aplicará las transformaciones contenidas en su interior a la imagen y anotaciones. La probabilidad de que se aplique cada transformación viene dada por el parámetro `p=PROBABILIDAD`.

Con el objetivo de que el lector pueda visualizar algunos ejemplos de las transformaciones aplicadas individualmente en la línea de aumento de datos, se presenta en la [Figura 4.13](#) una imagen original del conjunto de datos. Se presentará la imagen original seguida de la imagen con la transformación aplicada.



Figura 4.12: Posibles modificaciones para el proceso de aumento de datos. Fuente [4]

Encontraremos que se aplicarán las siguientes transformaciones:

- **Cambio en la perspectiva.** Probabilidad de 1. Vemos en la [Figura 4.14](#) la comparación.
- **Transformación afín** del tipo rotación en forma de elipse entre -360° y 360° . Probabilidad de 1. Vemos en la [Figura 4.15](#) la comparación
- Una de las siguientes. Probabilidad 0.8:
 - **Cambio en el canal de la imagen.** Probabilidad 0.1. Vemos en la [Figura 4.16](#) la comparación.

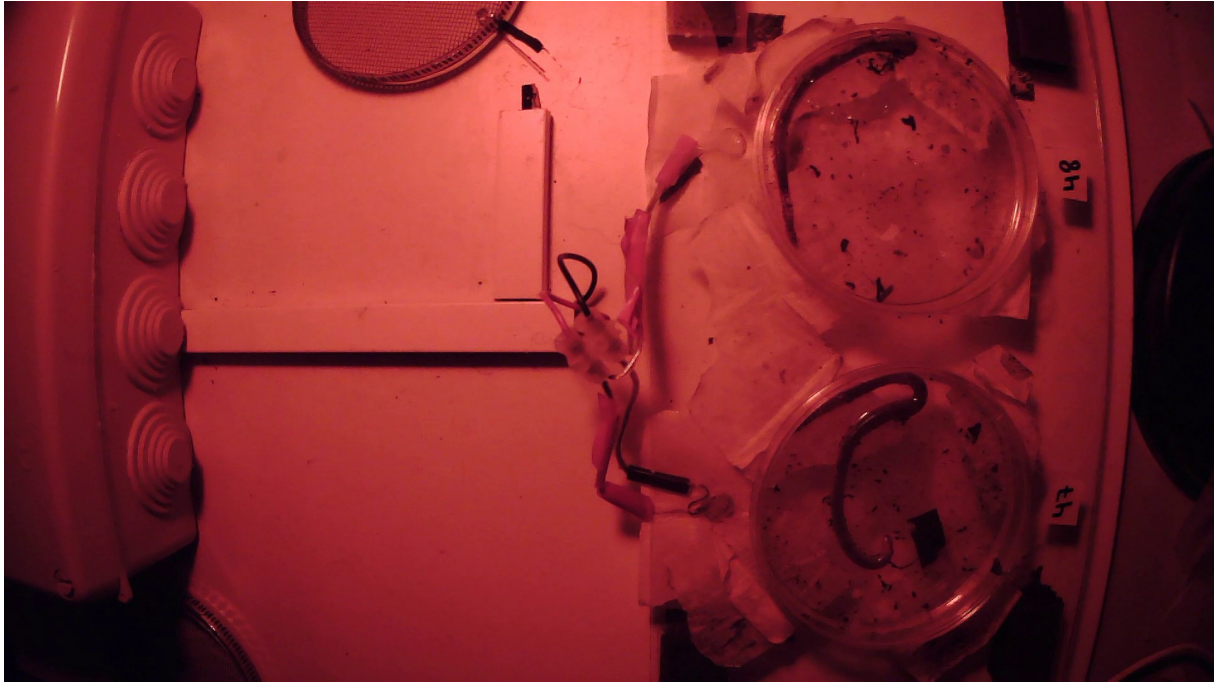
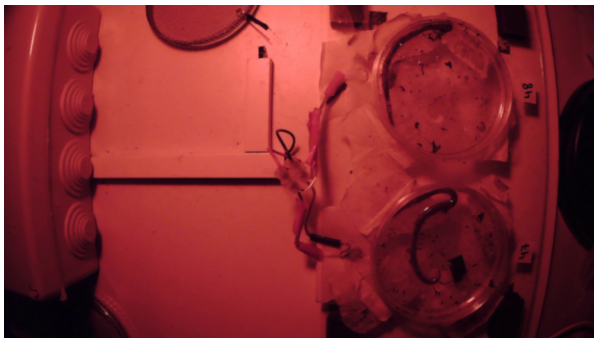


Figura 4.13: Imagen original a la que se le aplicarán las transformaciones



Original



Transformado

Figura 4.14: Comparación de la transformación cambio de perspectiva

- **Modificar el valor de hue y la saturación** [89]. Probabilidad 0.5. Vemos en la [Figura 4.17](#) la comparación.
- **Alternar valores de los canales RGB**. Probabilidad 0.7. Vemos en la [Figura 4.18](#) la comparación.
- Una de las siguientes. Probabilidad 0.4:
 - **Ecualizar el histograma de la imagen** [90]. Probabilidad 0.3. Vemos en la [Figura 4.19](#) la comparación.
 - **Modificar el brillo y el contraste de la imagen**. Probabilidad 0.5. Vemos en la [Figura 4.20](#) la comparación.

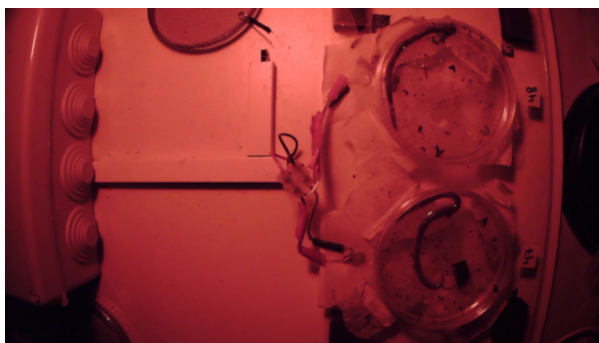
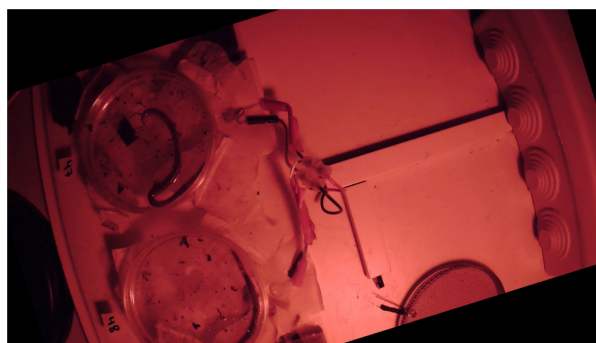
**Original****Transformado**

Figura 4.15: Comparación de la transformación rotación afín.

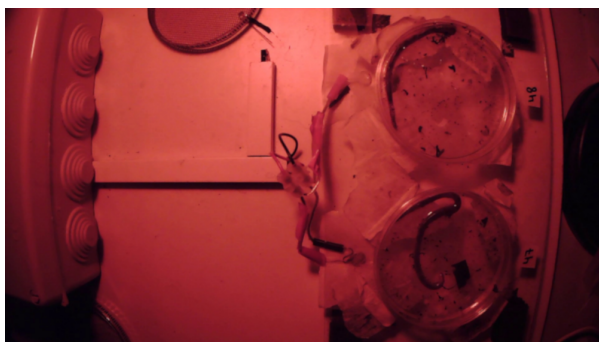
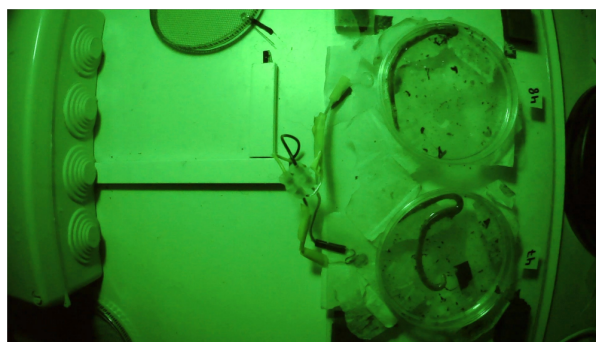
**Original****Transformado**

Figura 4.16: Comparación de la transformación cambio de canal

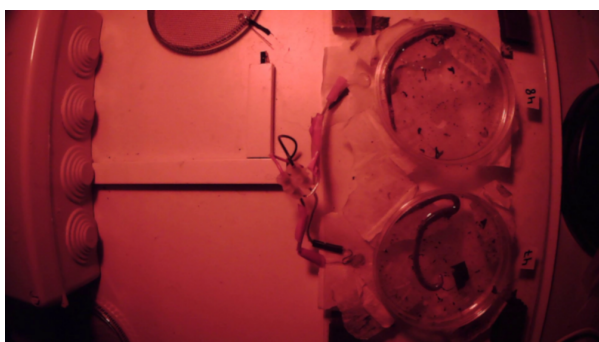
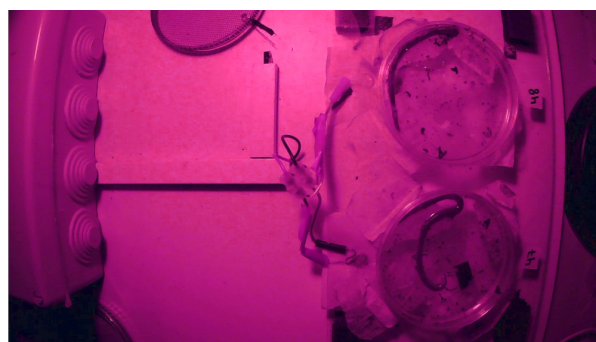
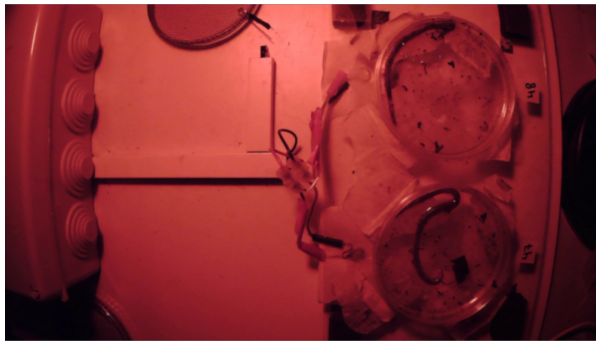
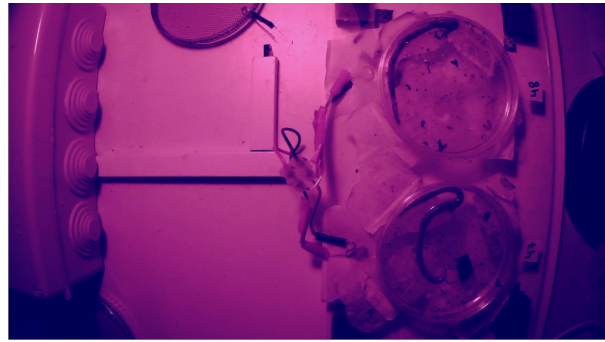
**Original****Transformado**

Figura 4.17: Comparación de la transformación de modificación del valor hue y la saturación

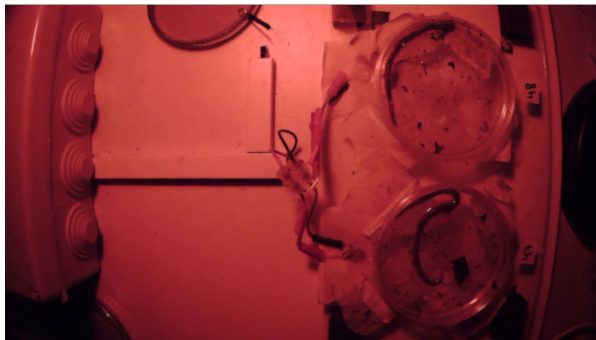


Original

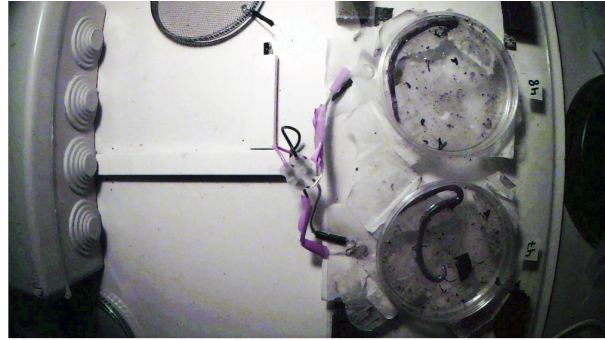


Transformado

Figura 4.18: Comparación de la transformación alternar valores de los canales RGB

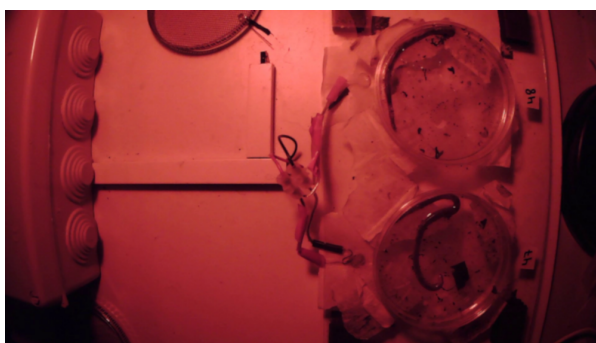


Original

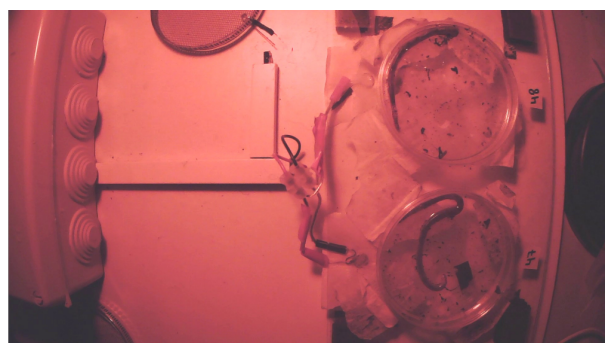


Transformado

Figura 4.19: Comparación de la transformación ecualizar el histograma de la imagen



Original



Transformado

Figura 4.20: Comparación de la transformación modificar el brillo y contraste de la imagen

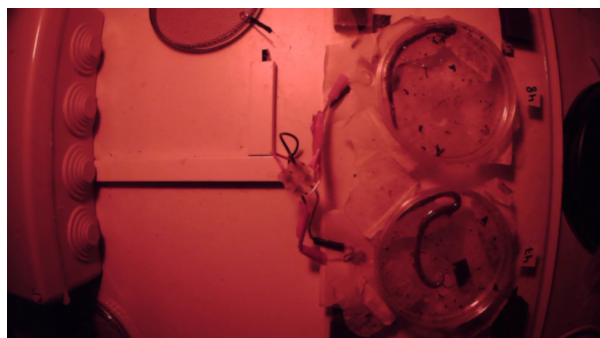
**Original****Transformado**

Figura 4.21: Ejemplo 1 de transformación completa de la línea de aumento

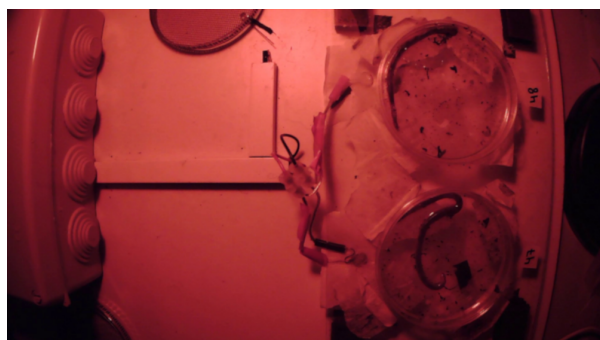
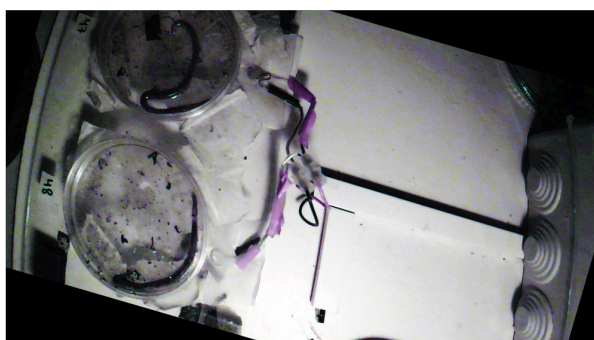
**Original****Transformado**

Figura 4.22: Ejemplo 2 de transformación completa de la línea de aumento

En las imágenes [4.21](#), [4.22](#) y [4.23](#) vemos el caso real de transformaciones aplicadas en una ejecución aleatoria del software. En este caso se aplicarán las transformaciones según la probabilidad establecida para cada una. Hay que destacar también que se aumenta en un factor de 3, por lo que se generarán 3 nuevas imágenes a priori diferentes.

El fundamento usado para aplicar cada probabilidad y tipo de modificación será no realizar transformaciones demasiado complejas que puedan modificar por completo la imagen original, pero para que sean suficientemente diferentes.

También se considera el hecho de que transformaciones más agresivas sucedan con poca probabilidad, lo que permitirá incluir algunos casos más dispares, pero no de forma demasiado frecuente, lo que evitará en gran medida el sobreentrenamiento del modelo, mejorando su capacidad de generalización.

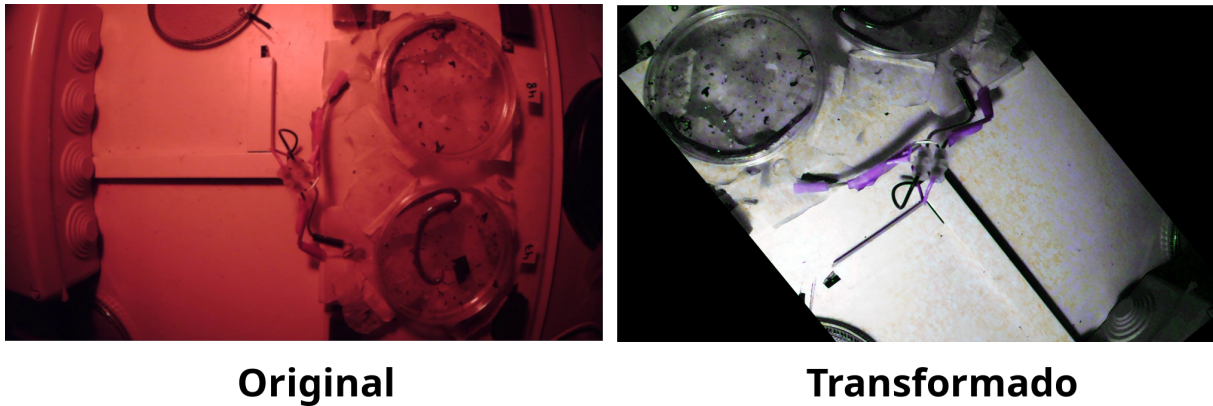


Figura 4.23: Ejemplo 3 de transformación completa de la línea de aumento

4.6. Software para el tratamiento de anotaciones

Con el objetivo de poder utilizar las anotaciones que se han hecho para el entrenamiento de un modelo, es necesario un conjunto de herramientas, a priori no existentes, que nos permitirán realizar todo el proceso una vez las anotaciones han sido exportadas de la plataforma app.cvat.ai hasta su posterior formato que requiere entrenar con YOLO. Este conjunto de herramientas serán escritas en Python, y adjuntadas al presente proyecto, además del repositorio completo creado en Github.

En la sección 4.2 se puede encontrar el conjunto de lenguajes de programación de los que se hará uso durante el proceso de creación de el conjunto de herramientas que a continuación se describen.

4.6.1. Descarga y extracción de frames

Dado que la tarea es realizada en unos servidores remotos, que no presentan interfaz gráfica, con el objetivo de agilizar el proceso de descarga de vídeos, y extracción de frames, se crean las dos herramientas que a continuación se presentan:

- **downloadDriveVideo:** En este caso por su sencillez se programará un script shell (sh) que tomará como argumentos el identificador del vídeo publicado en drive, y hecho accesible mediante enlace, y el directorio de salida para guardar dicho vídeo. El presente script descargará el vídeo correspondiente al servidor para su posterior uso.
- **extractFrames:** se usará la librería de Python `cv2` [75] la cual nos permite de forma fácil tomar cada frame de un vídeo, y guardarlos en un directorio con-

creto. Como es obvio, con el objetivo de contar con cierta correspondencia entre el vídeo y cada frame, estos se guardarán con un nombre significativo "frame_x" siendo x el número de frame dentro del vídeo.

4.6.2. Creación del dataset en formato correcto

El formato usado por app.cvat.ai es un formato propio [85] con un lenguaje de marcado xml, por lo que se usará la api XML de ElementTree *xml.etree.ElementTree* [91] de Python para tratar las anotaciones.

Tenemos el programa *xmlToYolo*, que se compone de dos partes diferenciadas: parsear el archivo exportado desde la plataforma quedándonos únicamente con aquello que nos servirá para la elaboración del dataset (función *parseXML()*), y la elaboración del dataset en el formato adecuado (función *setUpYoloDataset()*).

A modo de mejora, y con el objetivo de crear los dataset que más tarde se detallarán, se utilizará el parámetro *onlyKeyFrames* para hacer que el programa solo coja aquellos frames que en la anotación se marcaron como frames destacados, o en su defecto aquellos anotados de forma manual y no interpolado (ver sección 4.4.3). Esta forma de crear los conjuntos de datos es necesario pues como ya se ha detallado en el apartado 4.4.1, un conjunto de datos creado con todos los frames del vídeo, la mayor parte de ellos será anotado por interpolación, lo que supone calidad de anotaciones en ocasiones reducida.

De este modo, *xmlToYolo* tomará como parámetros el archivo xml de anotaciones, y el directorio dónde se encuentran los frames creados antes. Se establecerá una referencia de cada anotación con su respectivo frame, siguiendo los requerimientos de YOLOv8 para el conjunto de datos que a continuación se detalla:

- Un archivo de texto por imagen: cada imagen en el conjunto de datos tendrá el correspondiente archivo .txt con el mismo nombre que el archivo de imagen, pero extensión.txt.
- Cada archivo de texto tendrá una fila por objeto: cada fila corresponderá a una instancia del objeto anotado en la imagen. Esto es, si en cada imagen hay dos sujetos cuyos puntos queremos anotar, nuestros archivo de texto tendrán dos filas cada uno en caso de que ambos sujetos tengan al menos un punto anotado visible.
- Información para cada línea:

- **Índice de la clase del objeto anotado:** Un entero que representa la clase del objeto anotada. En nuestro caso, solo tenemos un tipo de objeto, por lo que siempre será el número 0, que corresponde con la clase `Worm` marcada en el archivo correspondiente.
- **Coordenadas del centro del objeto:** las coordenadas x e y del centro del objeto normalizadas entre 0 y 1.
- **Ancho y alto del objeto:** La anchura y altura del objeto normalizada entre 0 y 1.
- **Coordenadas para los puntos del objeto:** Los puntos del objeto normalizados entre 0 y 1.

A continuación se presenta un ejemplo del tipo de anotación que se debe seguir para la tarea de estimación de poses en YoloV8 [81]:

- Para formato con dimensión=2:

```
1 <class-index> <x> <y> <width> <height> <px1> <py1> <px2> <py2> ... <pxn> <pyn>
```

Listado 4.10: Formato para la anotación de puntos en YoloV8. 2 dimensiones

- Para formato con dimensión=3:

```
1 <class-index> <x> <y> <width> <height> <px1> <py1> <p1-visibility> <px2> <py2> <p2-visibility> <pxn> <pyn> <p2-visibility>
```

Listado 4.11: Formato para la anotación de puntos en YoloV8. 3 dimensiones

En el formato indicado, `<class-index>` es el índice de la clase del objeto, `<x>` `<y>` `<width>` `<height>` son las coordenadas de la caja (*bounding box*), y `<px1>` `<py1>` `<p1-visibility>` `<px2>` `<py2>` `<p2-visibility>` `<pxn>` `<pyn>` `<p2-visibility>` son cada punto y sus coordenadas, además del parámetro de visibilidad, que tomará 0 si no es visible y 1 si es visible.

4.6.3. Separación en entrenamiento, validación y test

Es clara la necesidad de separar el conjunto de datos completo en distintos subconjuntos, con el objetivo de tener separados datos de entrenamiento y validación. El estado del arte presenta numerosas formas que han sido estudiadas para la separación del conjunto total de datos en distintos subconjuntos [92]. Como primer acercamiento,

el código desarrollado realizará la separación de forma aleatoria en tres subconjuntos diferentes: entrenamiento, validación y test.

splitDataset tomará los porcentajes de entrenamiento y validación que deseamos darle al modelo, y separará de forma aleatoria el total de los frames y anotaciones. Un mayor acercamiento a esto se detallará en la sección 4.8.

4.6.4. Unión de varios conjuntos

Puesto que cada conjunto de los creados hasta ahora corresponde a un único vídeo, se nos plantea la necesidad de unir varios conjuntos en uno, de forma que este nuevo conjunto unificado nos pudiese servir para entrenar el modelo. Con este objetivo se crea ***joinDatasets***, un shell script que tomando como parámetros ambos directorios que corresponden a dos conjuntos de datos diferentes, copia cada archivo a un nuevo directorio que corresponderá con el nuevo dataset.

4.6.5. Aumento del conjunto de datos

Dentro de cualquier problema de aprendizaje profundo, es clara la necesidad de aumentar el conjunto de datos con datos "sintéticos" de forma que el modelo presente una mayor abanico de casos sobre los que aprender. Con este objetivo, se crea ***dataAugmentation***, un código que nos permitirá dado un conjunto de datos completo y sus anotaciones, además de un coeficiente de aumento n , crear un dataset n veces mayor que el original.

Se ha detallado de forma concreta el proceso de aumento como técnica concreta en la sección 4.5.

4.6.6. Visualización de anotaciones

A modo de comprobación, sobre todo para corregir posibles errores en el proceso de aumento de datos, se crea el programa ***visualizeAnotations***.

Ya que en el momento, no se conocen herramientas que permitan esta funcionalidad, es necesario crear este software que nos permita visualizar a partir de una imagen concreta, y el archivo de texto correspondiente a las anotaciones, las anotaciones superpuestas en la imagen.

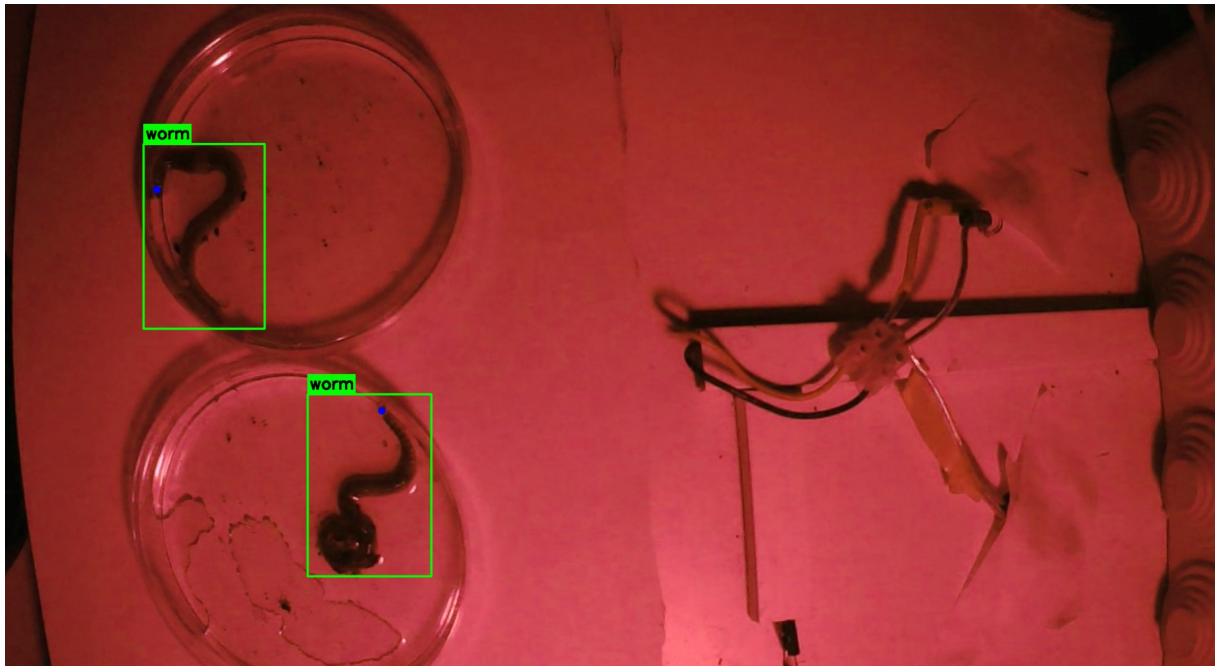


Figura 4.24: Funcionamiento del visionado de anotaciones a partir de una imagen creada por técnicas de aumento de datos

El funcionamiento es simple, el programa tomará como parámetro ambos archivos, y haciendo uso de la librería *OpenCV* de Python mostrará la imagen concreta, superponiendo las anotaciones relativas a la caja limítrofe y al punto clave de la cabeza.

Podemos ver en la [Figura 4.24](#) un ejemplo del funcionamiento de este software para una de las imágenes sintéticas creadas mediante aumento.

4.6.7. Control de versiones

Una vez creados las distintas piezas de software de las que se hará uso para el tratamiento de datos, y como cualquier pieza de software creada, se requerirá un sistema de control de versiones que permita implementar modificaciones, correcciones o mejoras al código implementado.

Por experiencia previa, facilidad, flexibilidad y robustez, se hará uso de Git, y concretamente Github para almacenar y realizar el control de versiones del código implementado. Se creará un repositorio específico y en el que se almacenará todo el código usado [93]. Esto será muy útil también dado el entorno de experimentación, en el que únicamente tendremos que clonar el repositorio existente para hacer uso de las herramientas que se programen en un entorno con interfaz gráfica y haciendo uso de un IDE.

De este modo, el código relativo al tratamiento de anotaciones relatado en la sección 4.6 estará organizado en el repositorio creado con dicho propósito, permitiendo de este modo una gran escalabilidad y disponibilidad independientemente de la plataforma en la que se requiera el software implementado.

4.7. Creación de conjuntos de datos

En esta sección, se profundizará en los aspectos relativos a la obtención de los conjuntos de datos necesarios para el entrenamiento de un modelo en cualquiera de las tareas de aprendizaje profundo.

Como ya se ha comentado anteriormente, y una vez conocemos la naturaleza de nuestro problema, parece obvia la necesidad de la utilización de un conjunto de datos a partir de los sujetos de análisis a estudiar, será por esto que los dataset utilizados serán creados a partir de los vídeos proporcionados por las investigadoras del departamento de psicología. Estos vídeos corresponderán a grabaciones estáticas de los sujetos de estudio durante un periodo de tiempo concreto.

4.7.1. Datasets utilizados

A continuación se definirán distintos datasets utilizados para el entrenamiento de modelos y características propias de los mismos. Los dataset que aquí se plantean quedan libres de aquellos frames que por diversos motivos pudieran ser mal anotados en su momento. Esto incluye frames parcialmente anotados, en el que únicamente alguna de las dos instancias del sujeto anotado fuese anotada o estuviesen mal anotados. El motivo de esto es prevenir fallos en el dataset y mejorar la calidad de las poca anotaciones que tenemos.

En la [Tabla 4.2](#) se presenta la forma que tiene el conjunto de datos original, previo a realizar el aumento descrito en la sección 4.5. La división de las imágenes es realizada con el programa creado ***splitDataset*** detallado en la sección 4.6.3.

En la [Tabla 4.3](#) se presenta la forma que tiene el conjunto de datos obtenido tras el proceso de aumento y su posterior división. El factor de aumento usado será de 3, por lo que a partir de cada frame del conjunto original se tendrán ahora 3 imágenes diferentes.

La fuente usada para ambos conjunto de datos será uno de los vídeos proporcionados para la experimentación, con nombre *habsujetos47y48.mp4* y corresponderá con una grabación de los sujetos 47 y 48 durante un total de 3397 frames.

En la [Tabla 4.4](#) se presenta un tercer conjunto de datos creado a partir únicamente de ciertos frames anotados de forma manual y no interpolados, de un conjunto más amplio de vídeos. En la columna *Frames* se refleja el número total de frames con los que cuenta el vídeo, y en la columna *Usados* se refleja únicamente el número de frames de dicho video que ha sido usado, y por tanto, anotado manualmente. Este conjunto de datos planteado, como ya se verá posteriormente, será usado para una validación externa del modelo (Sección [4.8.2](#)).

El proceso para la obtención de este último conjunto de datos difiere algo del resto, y es que como relatado en la sección [4.6.2](#), el programa ***xmlToYolo*** tomará un parámetro opcional *onlyKeyFrames*, el cual hará que el conjunto de datos se cree a partir de los frames anotados manualmente, y que en la plataforma se indica como *Key Frame*. A partir de la obtención de cada conjunto reducido de diferentes vídeos, se usará el programa ***joinDatasets*** para unirlos todos y obtener un conjunto de datos total de 203 frames diferentes.

Como es obvio, sería conveniente contar con más frames en este último conjunto, así como en cualquier conjunto un mayor número de datos nos permitiría en gran medida mejorar los resultados de la experimentación. Sin embargo, esto supondría una excesiva carga de trabajo en una tarea simple como la de anotar cada una de las imágenes, pero que requiere de mucho tiempo y estrés cognitivo. Es conveniente recordar la motivación de este proyecto relatada en la sección [1.1](#), y es de reducir al máximo un esfuerzo cognitivo como el que se plantea con el análisis de los sujetos, por lo que no tendría lógica ninguna para el experimentador, someterse a dicho esfuerzo cognitivo para una tarea como la de anotar un número tan elevado de frames de forma manual.

Fuente usada	Frames	Factor augmentación	Total	División imágenes
habsujetos47y48.mp4	3397	*1	3397	Test: 169 Entrenamiento: 2888 Validación:340

Tabla. 4.2: dataset rawDataset

Fuente usada	Frames	Factor augmentación	Total	División imágenes
habsujetos47y48.mp4	3397	*3	10191	Test: 509 Entrenamiento: 8663 Validación:1009

Tabla. 4.3: dataset mainDataset

Fuente usada	Frames	Usados	Total
habsujetos47y48.mp4	3397	101	
habsujetos45y46.mp4	3292	21	
habsujetos31y32.mp4	3409	15	
habsujetos37y38.mp4	3508	11	
habsujetos39y40.mp4	3300	13	
habsujetos43y44.mp4	3321	13	
Rehabsujetos25y26.mp4	2747	9	
sinluz.mp4	11383	20	
			203

Tabla. 4.4: dataset wormTesting1-2. Solo frames concretos.

4.8. Estrategia de validación

De forma ideal, las métricas de validación que obtendremos de nuestros modelos, nos ayudarán a elegir el mejor y nos indicará también cómo de bien actuará dicho modelo para datos no vistos anteriormente, y esto es el objetivo final del entrenamiento de modelos profundos. El objetivo final de esta experimentación tal y como relatado en el [Capítulo 3](#) será obtener el modelo que mejor actúe frente a datos no vistos anteriormente, para ello debemos buscar la forma de comparar distintos modelos entre sí.

Existen numerosas formas y métodos para la validación de un modelo [\[5\]](#). Dado los dos conjuntos de datos iniciales ([4.2](#) y [4.3](#)) sobre los que realizaremos las pruebas y que será la principal fuente de experimentación, realizaremos un primer acercamiento para ver el comportamiento cada modelo una vez entrenado gracias al conjunto de validación presente en cada uno de los conjunto de datos, y que será relatado en la sección [4.8.1](#).

A modo de prueba, y para ver cómo se comportarían los modelos que entrenare-

mos posteriormente, contamos con un tercer conjunto de datos(4.4). El motivo para la utilización de este último dataset para la realización de pruebas que podríamos considerar como más allá de la experimentación, es incluir una mayor variedad de casos diferentes. De este modo se expondrá al modelo a un conjunto, que aunque reducido, reflejará de mejor forma datos que encontrará en casos reales u otros datos no vistos anteriormente. Este proceso de validación para un conjunto externo será relatado en la sección 4.8.2.

4.8.1. Holdout validation

El funcionamiento de este método de validación es el siguiente:

1. En primer lugar se divide el conjunto de datos disponibles en dos subconjuntos: entrenamiento y validación. El subconjunto de validación, con el que evaluaremos el funcionamiento del modelo, debe estar formado por datos no vistos por el modelo anteriormente.
2. Una vez tenemos el conjunto de validación separado, escogemos un algoritmo de aprendizaje apropiado. Es importante aclarar que el valor de los hiperparámetros (*Hyperparameter Values*) que aparece en la Figura 4.25 son los correspondientes al algoritmo que escojamos, y por tanto debemos especificarlos de forma manual a la hora de comenzar el entrenamiento del modelo. El modelo resultante no aprenderá nada de estos hiperparámetros, por lo tanto escoger el mejor conjunto de estos es una tarea independiente, y que será realizada más adelante.
3. Contando ya con un modelo entrenado sobre nuestro conjunto de entrenamiento, debemos resolver el problema de cómo de bien funciona este, pudiendo así comparar entre los distintos modelos. Es aquí dónde utilizaremos el conjunto de validación antes separado. Al tratar con datos que el modelo no ha visto previamente, los resultados no deberían estar condicionados, y por tanto, ser considerados como una buena estimación de cómo de bien funcionará el modelo para cualquier dato.
4. Finalmente, conociendo la capacidad de generalización del modelo, podríamos ajustar los hiperparámetros correspondientes si es necesario, y volver a entrenarlo sobre el conjunto completo de datos para obtener el modelo final.

Cabe destacar que en la figura presentada, procedente de una fuente externa, se hará uso del subconjunto denominado como *test*. Este subconjunto será el equivalente

referenciado como el conjunto de validación en la explicación, como también suele ocurrir en la bibliografía.

Durante la experimentación, se hará referencia a este método de validación como **validación interna**, o al conjunto de datos usado para esta validación como **conjunto interno**.

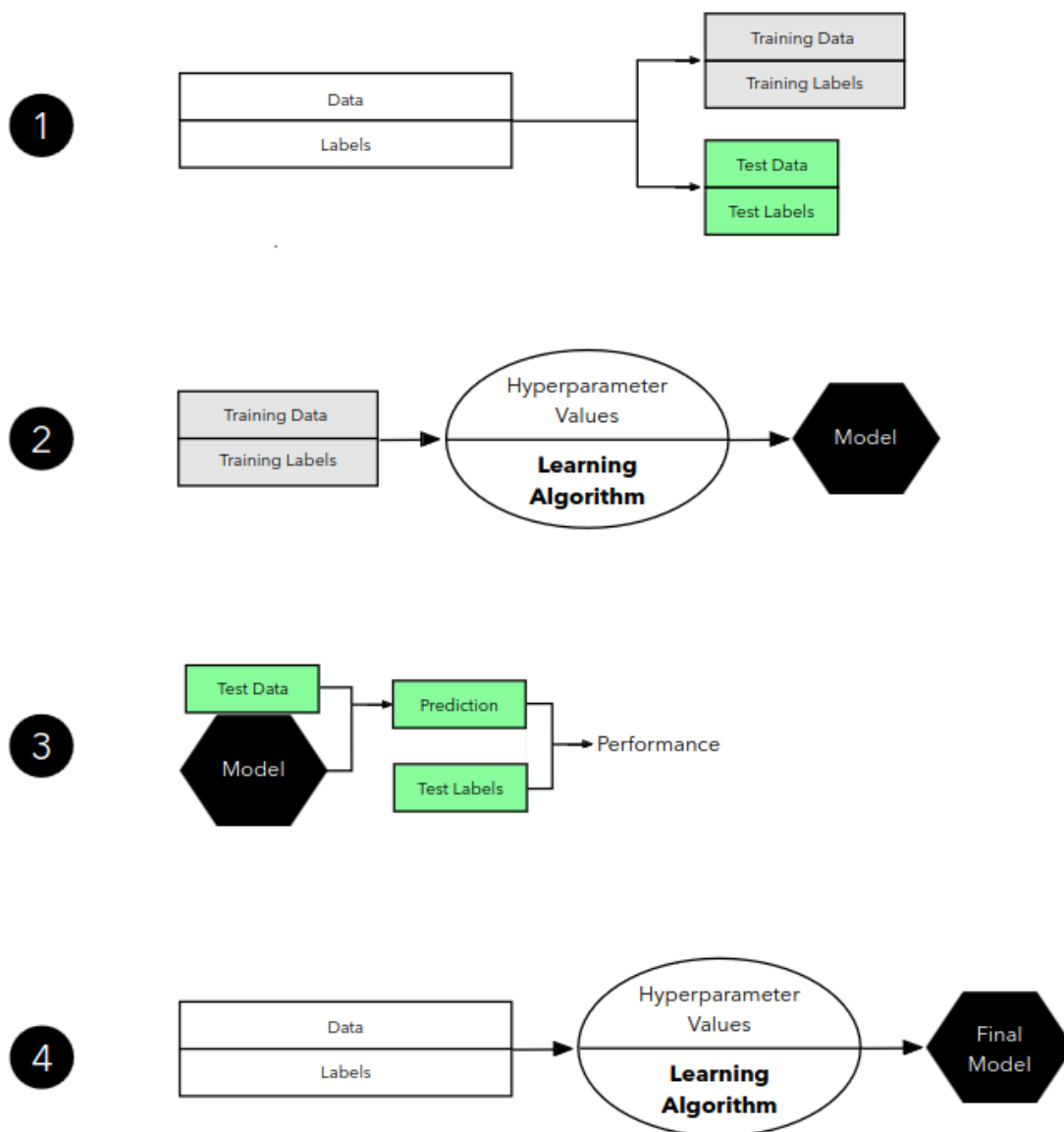


Figura 4.25: Representación visual de la validación holdout. Fuente [5]

4.8.2. Validación externa

Dado el objetivo inicial de este proyecto, el de crear un modelo que permita la anotación estímulo respuesta de forma autónoma, se entiende la necesidad de crear un

modelo que sea capaz de generalizar para todos los casos posibles proporcionados.

Con esta visión, se decide hacer uso de un conjunto externo, y no el perteneciente al mismo vídeo sobre el que se obtendrá el conjunto de entrenamiento, sino otros vídeos independientes, en otras circunstancias de entorno y en ocasiones de luz. De esta forma se intenta poner las cosas un poco más difíciles al modelo, que por el contrario estaría realizando la validación sobre un conjunto como el reflejado en la sección previa [4.8.1](#).

Esta alternativa se plantea posteriormente de realizar los primeros entrenamientos de forma convencional, y que como podremos ver en el [Capítulo 5](#), se consigue un modelo que de forma sencilla obtendrá unas métricas de validación con los mejores resultados, por ejemplo en la [Tabla 5.10](#).

Se entiende que este planteamiento, supondrá que el modelo se validará conforme a unos datos más reales en un entorno funcional, dado que el objetivo de su creación es poder usarlo para todos los casos similares, pero no iguales.

Planteadas estas visiones, se creará el conjunto de datos presentado en la [tabla 4.4](#), el cual estará compuesto de frames anotados de forma manual y exacta, para todos los vídeos proporcionados.

4.8.3. Comparación de los métodos de validación

Hay que tener en cuenta la naturaleza del problema a resolver. De forma habitual, para diversos problemas de aprendizaje profundo, y cuando se tiene un conjunto de datos, este se separa en tres subconjuntos: entrenamiento, validación y/o test. Existen diversos métodos y fundamentos para la separación del conjunto de datos original de manera aleatoria en estos subconjuntos como visto en la [sección 4.8.1](#) para la validación Holdout.

Sin embargo, teniendo en cuenta la naturaleza de nuestro problema, esta no parece ser la mejor solución, pues usar imágenes(frames) escogidas de forma aleatoria dentro del mismo vídeo sobre el que el modelo entrena, hará que con casi total certeza el modelo haya entrenado sobre las imágenes inmediatamente anterior y posterior a la que estamos validando, debida la inherente correlación de los datos. Esto, por tanto, parece ilógico para intentar obtener resultados de validación significativos del buen desempeño del modelo con datos no vistos, y que es un requisito para el problema planteado.

Podríamos considerar nuestro conjunto de datos como una serie temporal, en la que buscamos obtener una posición concreta (cabeza) dentro de la imagen concreta a lo largo del tiempo [6] y un subconjunto de validación escogido aleatoriamente entre el conjunto completo, daría lugar a pequeños huecos dentro de esa serie temporal de forma similar a la representada en la serie temporal de ventas a lo largo de un periodo de tiempo mostrado en la [Figura 4.26](#).

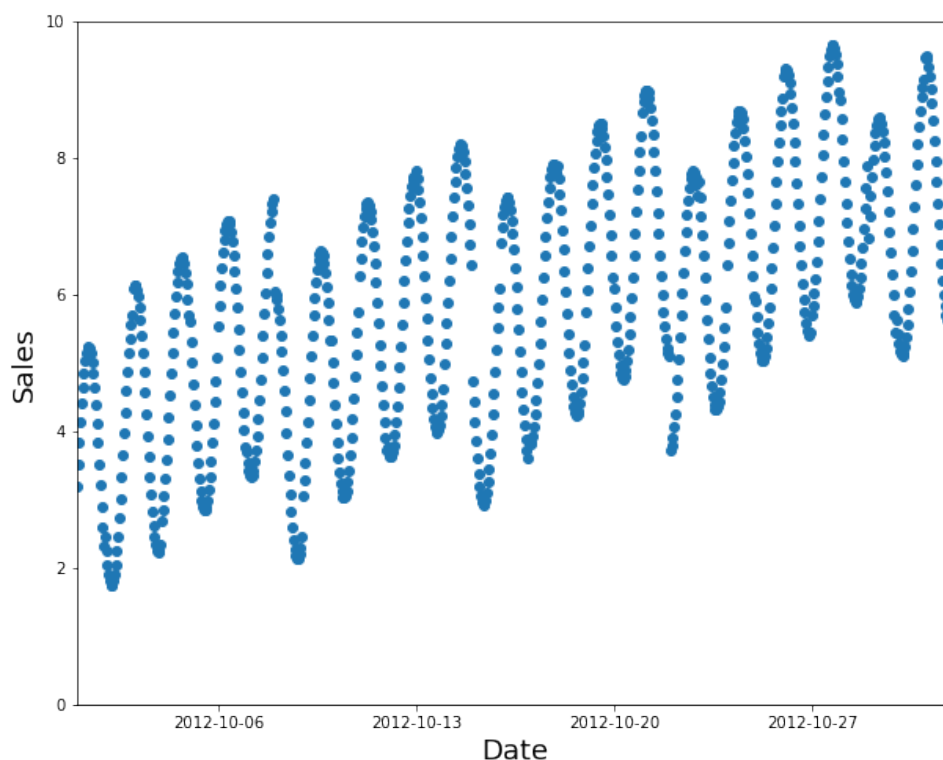


Figura 4.26: Datos de una serie temporal para ventas a lo largo del tiempo. Para extrapolarlo a nuestro problema deberíamos considerar las ventas como la posición del sujeto. Fuente [6]

Por tanto, la elección de un subconjunto de validación de forma aleatoria dentro del conjunto total, daría como resultado un conjunto de entrenamiento como reflejada la [Figura 4.27](#), por lo que sería demasiado fácil para nuestro modelo rellenar lo huecos restantes, o para nuestro problema, localizar con acertada precisión la posición del sujeto. Siguiendo este razonamiento, parecería lógico esperar unas métricas de validación considerablemente buenas, y en muchos casos cercanas a la perfección, algo que será estudiado en el [Capítulo 5](#).

Es por esto que no solo se validarán los distintos modelos con el conjunto de validación propio del conjunto de datos sobre el que entrena, al que llamaremos **conjunto de datos interno**, como sería lo habitual, sino que la experimentación irá un paso más allá, y se realizarán pruebas de validación de los modelos que obtendremos con un **conjunto de datos externo**, como mencionado en la sección [4.8](#).

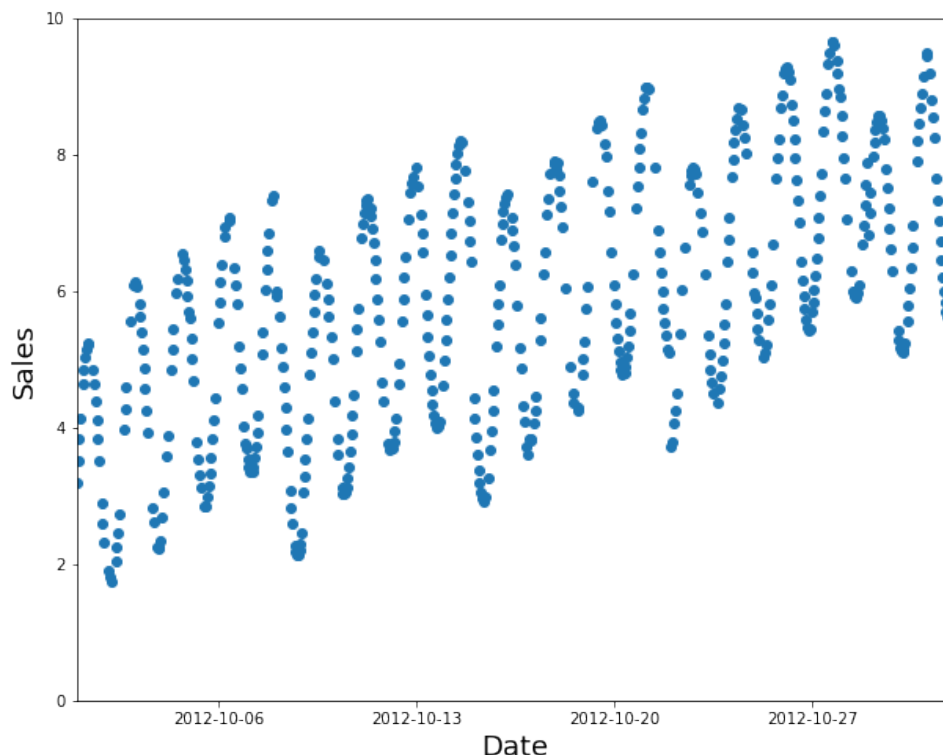


Figura 4.27: Representación del conjunto de entrenamiento escogido de forma aleatoria. Fuente [6]

4.9. Depuración de resultados y entrenamientos

En una experimentación como la planteada, en la que numerosos modelos serán entrenados, es de vital importancia conseguir un correcto seguimiento de los mismos, que de forma ordenada nos permita entender los resultados. Hay que tener en cuenta también el entorno de experimentación usado, y es que al tratarse de un sistema sin interfaz gráfica, en el que lanzaremos numerosos modelos de forma simultanea para entrenar, complica la presentación de resultados formateada y clara para el investigador.

Es por esto, que se decide hacer uso de una herramienta que permita de forma sencilla llevar el seguimiento, administrar, visualizar y optimizar los distintos entrenamientos de modelos llevados a cabo. La herramienta utilizada para la depuración será **Comet** [94].

Esta herramienta, puede ser utilizada con gran sencillez. Únicamente mediante la generación de una clave API, ya se podrá implementar a través de un simple script en python de la siguiente forma:

```
15| import comet_ml
```

```
16 from ultralytics import YOLO
17 import os
18
19 os.environ["COMET_API_KEY"] = API_KEY
20 comet_ml.init(NOMBRE_PROYECTO)
21
22 model = YOLO(MODELO)
23 results = model.train(
24     data=FICHERO_DATASET,
25     project=NOMBRE_PROYECTO_YOLO,
26     batch=N_BATCH,
27     save_json=True,
28     epochs=N_EPOCH,
29 )
```

Listado 4.12: Integración de Comet en Python

Para una mejor comprensión del funcionamiento por parte del lector, en el anterior código tenemos los siguientes parámetros:

1. **API_KEY**: corresponde con la clave API generada para nuestra cuenta de *Comet*.
2. **NOMBRE_PROYECTO**: corresponde con el nombre de proyecto que se usará en nuestra cuenta de *Comet*.
3. **MODELO**: nombre del modelo que deseamos entrenar. Se usará normalmente uno de los vistos en la sección 2.7.
4. **FICHERO_DATASET**: Fichero usado para referenciar al conjunto de datos. Relatado en la sección 4.3.2.
5. **NOMBRE_PROYECTO_YOLO**: nombre interno del proyecto con el que lo identificará YOLO.
6. **N_BATCH**: número de batch usados para el entrenamiento.
7. **N_EPOCH**: número de épocas usadas para el entrenamiento.

Encontramos un gran número de funcionalidades existentes en Comet. A continuación se detallan algunas de las más interesantes e involucradas en este proyecto.

4.9.1. Seguimiento de modelos y ejecuciones de entrenamiento

Lanzar un experimento con Comet [95] nos permite interactuar con el mismo casi en su totalidad. Podremos de forma sencilla analizar el rendimiento del mismo en

tiempo real, lo que supone una gran ventaja. En el momento que se lance el entrenamiento del modelo, el framework de Comet empezará a registrar de forma automática métricas (Figura 4.28) e hiperparámetros (Figura 4.29), el fichero de código usado (Figura 4.30), información relativa al repositorio git si es el caso, salida por consola (Figura 4.30), paquetes de python instalados (Figura 4.30) y métricas del sistema (Figura 4.33).

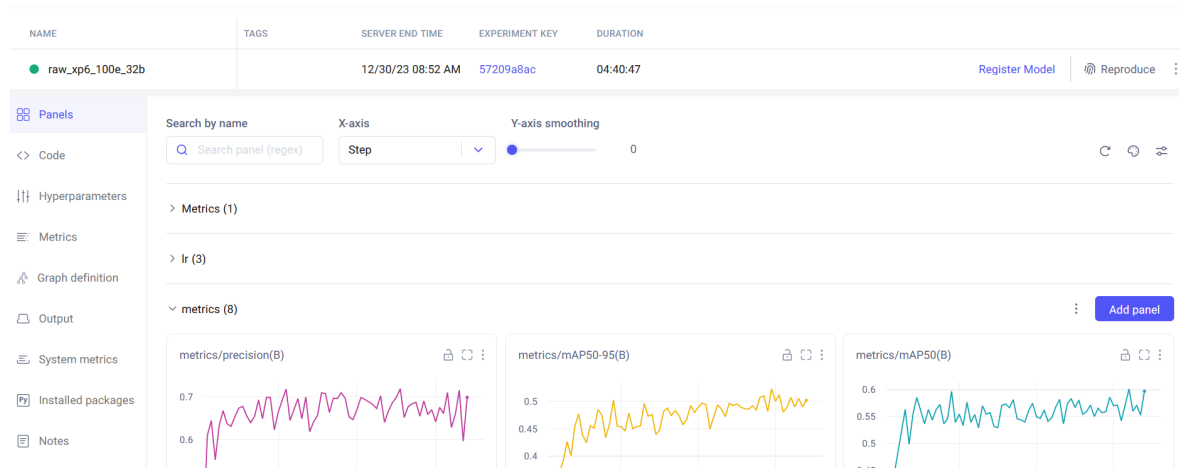


Figura 4.28: Pantalla principal presentada en un experimento de Comet. Métricas.

KEY NAME	RAW_XP6_100E_32B
agnostic_nms	false
amp	true
augment	false
batch	32
box	7.5
boxes	true
cache	false
cfg	null

Figura 4.29: Algunos de los hiperparámetros mostrados por Comet

The screenshot shows the Comet UI interface. On the left, there's a sidebar with 'Assets' and 'Artifacts' tabs. Below them is a search bar and a list of file names: 'confusion-matrices', 'images', 'models', 'others', and 'source-code'. The 'source-code' folder is expanded, showing two files: '/home/alvaro/trainYolov8/r...' and '/home/alvaro/conda/envs/...'. On the right, there's a code editor showing Python code for training a YOLOv8 model. Below the code, there's a metadata section with fields like 'File Name', 'Last Modified', 'Path', and 'Files Size'. At the bottom right, there are 'Download' and 'View' buttons.

Figura 4.30: Ficheros registrados por Comet. Código fuente usado.

The screenshot shows the Comet UI interface with the 'Output' tab selected. The console output displays training progress, including metrics like 'Class', 'Images', 'Instances', 'Box(P)', 'R', 'mAP50', and 'mAP50-95'. It also shows the completion of 68 epochs and the saving of the best model. The output is truncated, with a 'Download full log' link provided. The interface includes a sidebar with various panels like 'Code', 'Hyperparameters', 'Metrics', 'Graph definition', 'Output', 'System metrics', 'Installed packages', 'Notes', 'Graphics', 'Audio', 'Text', 'Confusion matrix', and 'Histograms'.

Figura 4.31: Salida por consola registrada por Comet.

Además, una interesante funcionalidad es que permite la descarga de cualquier fichero, ya sea de código o de tipo log, para cualquier fin que pueda ser necesario, garantizando una reproducibilidad total del experimento realizado. Esto podría hacerse de forma manual recreando la configuración si fuese necesario, sin embargo, Comet ya nos facilita este trabajo con la opción **Reproduce**, la cual nos indica la información necesaria para volver a lanzar el mismo experimento si fuese necesario como puede verse en la [Figura 4.34](#).

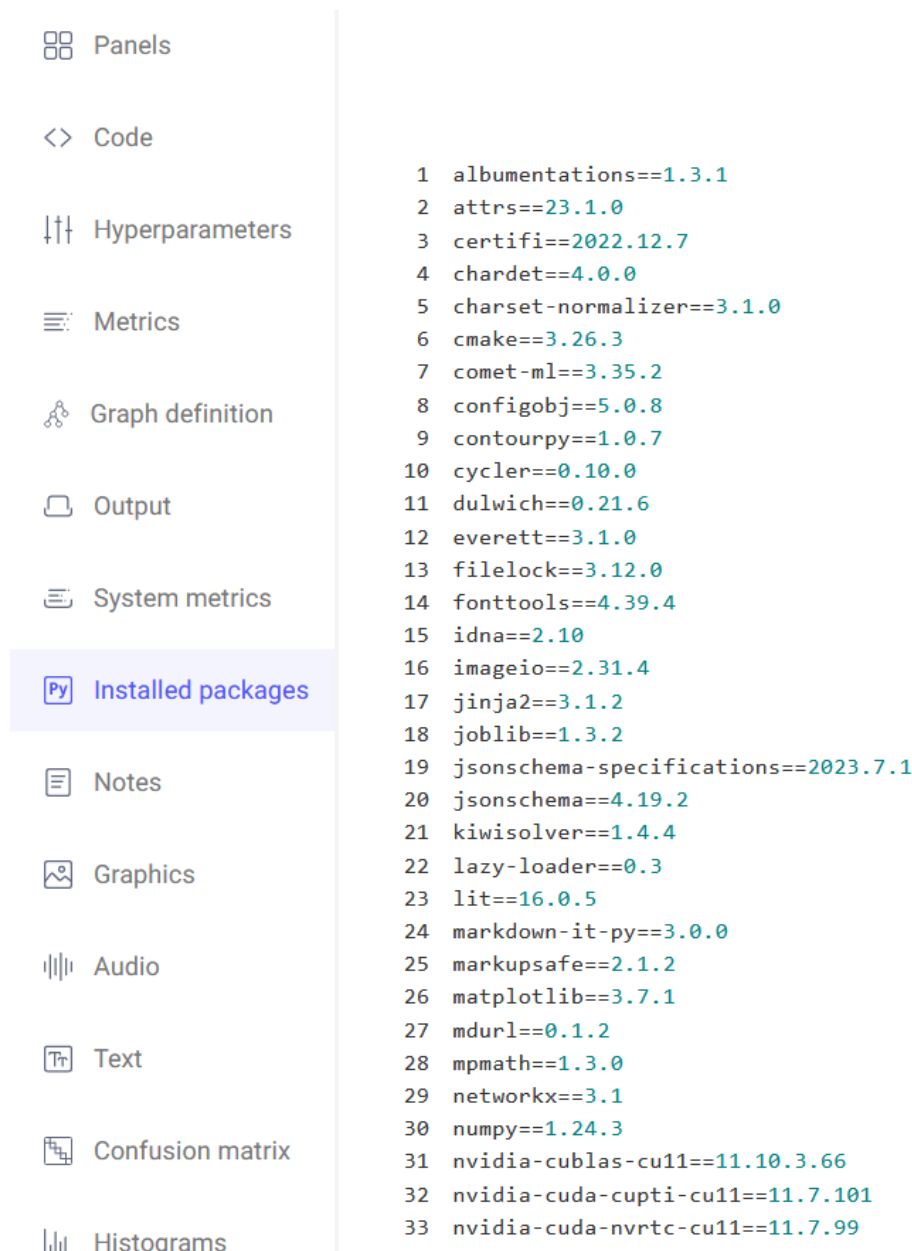


Figura 4.32: Paquetes Python instalados registrados por Comet

4.9.2. Visualización de resultados

Los resultados son representados en *paneles* [96] de muchos tipos distintos. Podemos usar paneles ya incluidos en Comet o usar alguno creado por la comunidad. Una opción muy interesante es crear y usar los propios del espacio de trabajo, lo que nos permite tener un espacio colaborativo dentro de la organización en el que otro usuario del mismo proyecto pueda crear paneles concretos para su uso dentro del proyecto.

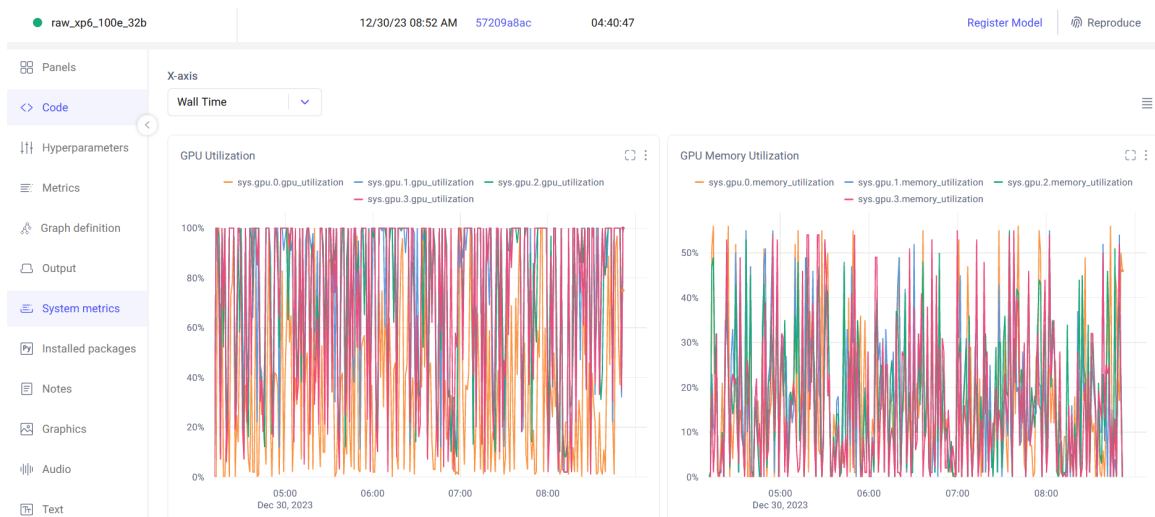


Figura 4.33: Métricas del sistema registradas por Comet

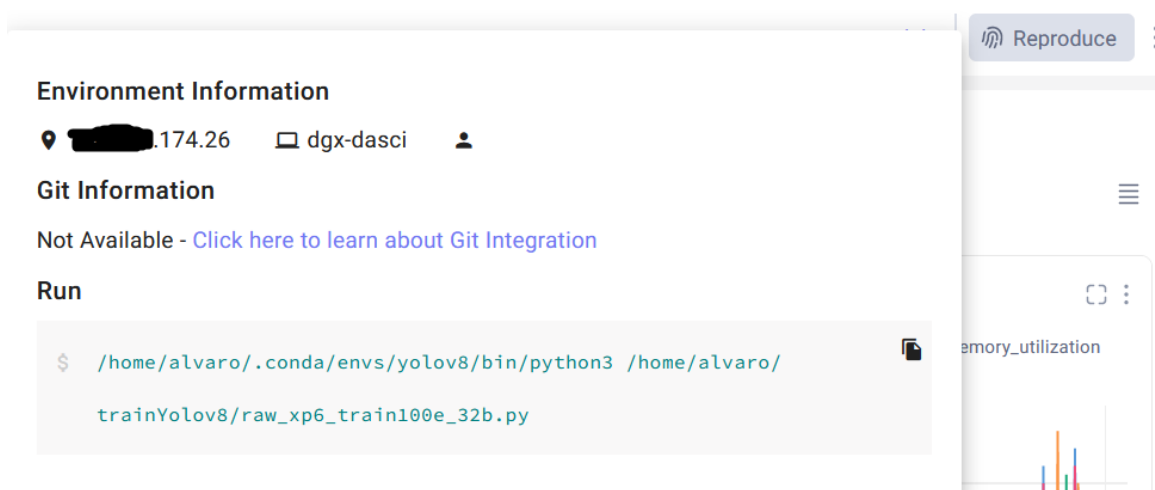


Figura 4.34: Opción de reproducir experimento en Comet

El uso de paneles permite crear visualizaciones concretas, herramientas, reportes o métodos de exportación concretos y totalmente personalizados. Podemos ver en la [Figura 4.35](#) la pantalla que se presenta al añadir un nuevo panel.

4.10. Configuraciones de hiperparámetros

Para contar con un amplio abanico de estudio inicial, se aplicarán distintas configuraciones de hiper-parámetros. En este punto de partida se alterarán únicamente dos hiperparámetros comunes: el número de epochs y el tamaño de batch. En los siguien-

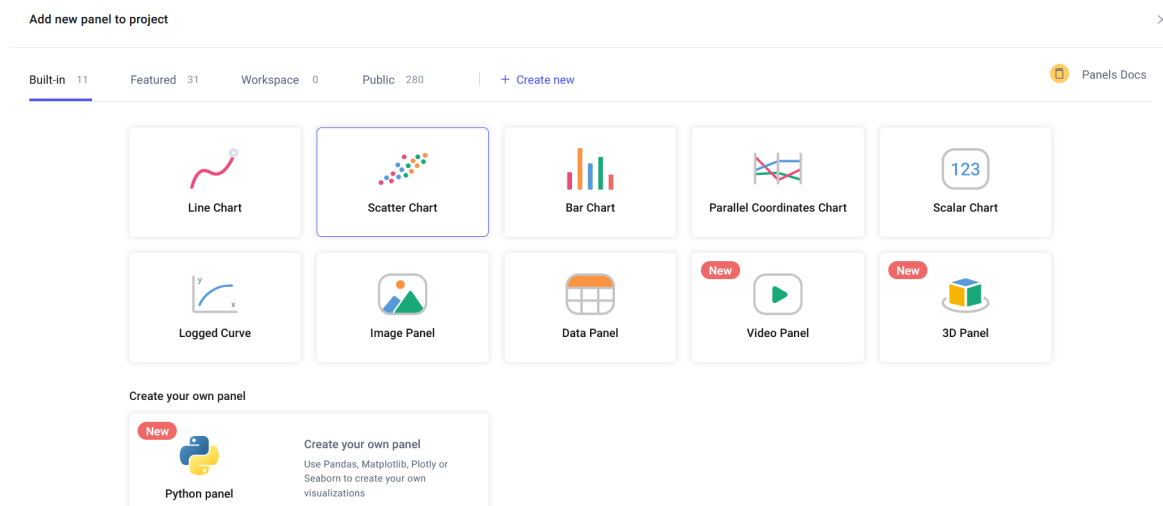


Figura 4.35: Posibilidades para añadir un panel en Comet [7]

tes pasos de la investigación se expondrán otras configuraciones que podrían resultar producir mejores resultados en el problema a tratar. A continuación se presenta la definición formal de estos hiperparámetros para un mejor entendimiento del lector en el proceso de experimentación:

- **Epoch:** es una iteración del ciclo de entrenamiento completo en el conjunto de datos de entrenamiento. Será por tanto el número de veces que se actualizan los pesos de la red.
- **Batch:** es el número de datos que se introducen en la red a cada época para el entrenamiento.

Se aplicarán entonces las siguientes configuraciones de hiper-parámetros para cada uno de los modelos con el objetivo de encontrar el conjunto que mejor resultados produce respecto a las métricas estudiadas:

- 50 epochs
 - Tamaño de batch 16
 - Tamaño de batch 32
 - Tamaño de batch 64
- 100 epochs
 - Tamaño de batch 16

- Tamaño de batch 32
- Tamaño de batch 64

Siguiendo las diferentes configuraciones planteadas, y dado que se realizará el estudio de numerosos modelos, para los que habrá que seguir también el entrenamiento, cada uno de los modelos es nombrado siguiendo un patrón, en base a los datos sobre los que entrenará, el modelo base y la configuración de hiperparámetros planteada.

Se detalla a continuación para evitar cualquier posible confusión del lector, cómo identificar de forma sencilla cada uno de los modelos resultantes del entrenamiento, y que en las siguientes secciones se relatará:

- **Entrenado o no sobre datos aumentados.** Aquellos modelos cuyo conjunto de datos no haya sido aumentado, se identificará con el prefijo **raw**. Vemos por ejemplo **raw_150e_16b** como el primero de los modelos que aparece.
- **Modelo usado.** La primera letra a continuación corresponderá al tipo de modelo base usado. Estos podrán ser:
 - **l** para *YOLOv8l-pose*
 - **x** para *YOLOv8x-pose*
 - **xp6_** para *YOLOv8x-pose-p6*
- **Número de epochs.** El primero de los dos números encontrados en el nombre corresponde al número de épocas o epochs sobre los que el modelo entrenará. Este número aparecerá seguido de la letra **e** como indicador de **epoch**.
- **Tamaño de batch.** El segundo de los números que aparece en el nombre corresponderá con el tamaño de batch utilizado para dicho entrenamiento. Este número aparecerá seguido de la letra **b** como indicador de **batch**.

Siguiendo lo relatado, entendemos de forma fácil por qué el modelo más abajo nombrado como **raw_150e_16b** corresponde con el modelo base *YOLOv8l-pose* entrenado sobre datos no aumentados, durante 50 épocas y un tamaño de batch de 16.

4.11. Métricas de validación

Las métricas de validación son uno de los puntos esenciales en cualquier problema dentro del ML para conocer cómo de bien está funcionando el modelo que se entrena,

posibles mejoras, o fallos que se pudiesen estar comentiendo durante su entrenamiento.

En nuestro caso, las métricas nos permitirán conocer la efectividad de nuestro modelo para localizar a los sujetos dentro de la imagen, así como la posibilidad de poder detectar la cabeza de cada uno de ellos. Esto es crucial para conocer si las técnicas que se aplicarán, como el aumento de datos, o la modificación de hiperparámetros está resultando efectiva para obtener el mejor modelo posible.

Es importante que el lector entienda que en el caso de aplicación, se están realizando dos tareas distintas y reconocidas dentro del campo de CV. La primera tarea es la de **detección**, para la cual se hará uso de las anotaciones de cajas limítrofes o *bounding boxes*. Para una detección se le asociará una clase, lo que involucra también la tarea de **clasificación**. La segunda tarea, y la que en principio nos afecta en mayor medida es la de **estimación de pose**, para la cual se hará uso de los puntos anotados o *keypoints*. Para una correcta tarea de estimación de pose es necesaria la previa detección del objeto, en nuestro caso la lombriz, para después poder encontrar los puntos claves requeridos. Posteriormente, estos puntos servirán para la tarea superior de estimar la pose, o seguir el movimiento del sujeto.

Conocido esto, parece lógico intuir que para el estudio de una correcta estimación de pose, se requiere la optimización de las métricas relativas a la caja limítrofe, y a los puntos. Por ello, son las métricas que se evaluarán en este estudio, y las que se usarán para las comparaciones en la experimentación. Cabe destacar que el conjunto de métricas será el mismo, pero aplicado por separado a las cajas limítrofes y a los puntos, por lo que la descripción que a continuación se plantea será genérica, y no aplicada a ninguna de las dos características medibles en concreto. En el caso de la métrica aplicada a la caja, se usará la letra **B** identificándola. En el caso de la métrica para los puntos, se usará la letra **P** identificándola.

Las métricas reflejadas en las tablas en el [Capítulo 5](#) estarán abreviadas por disposición de los resultados, sin embargo a continuación se describirá cada una de ellas.

4.11.1. Intersección sobre unión (IoU)

Aunque no será una medida de rendimiento que se evaluará de forma directa en los modelos, es importante para el lector conocerla, ya que será la base para entender las demás medidas.

La intersección sobre la unión marca el solapamiento de la caja delimitadora predicha y la real. Cuanto mayor sea el IoU, más se parecerá la predicción a la anotación real. En la [Figura 4.36](#) [97] vemos un ejemplo claro del funcionamiento.

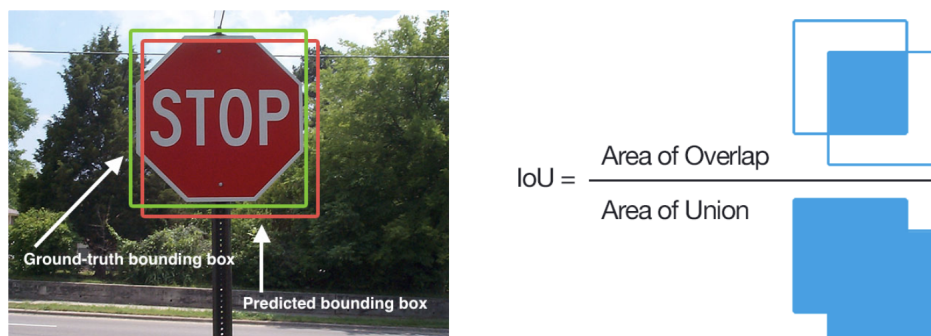


Figura 4.36: Representación de la intersección sobre la unión

4.11.2. Object Keypoint Similarity (OKS)

Vista la medida de intersección sobre unión de la sección 4.11.1 parece afectar únicamente a las cajas limítrofes. Esto es cierto, sin embargo, la métrica análoga de la que se hace uso para el cálculo de las posteriores métricas en puntos claves es **Object Keypoint Similarity**, comúnmente conocido como **OKS**. El uso de esta métrica es la forma de facto para calcular las predicciones de puntos con las anotaciones reales. Sigue la siguiente fórmula:

$$OKS = \exp\left(-\frac{d_i^2}{2s^2k_i^2}\right)$$

Donde:

- d_i es la distancia euclídea entre el punto real y el anotado.
- s escala: la raíz cuadrada del area segmentada del objeto.
- k constante para el punto i .

Es importante destacar que a pesar de no mostrarse de forma concreta dentro de la documentación de YoloV8, pues solo se muestra de forma concreta el cálculo de las métricas para cajas limítrofes, si analizamos el código fuente del que hace uso, se puede ver de forma sencilla como la denominada *IoU* de la que hablábamos para las cajas limítrofes, será en realidad el análogo de la métrica *OKS*. [98]

Por motivo de generalización, aunque sabiendo que no es la forma más correcta de denominarla, en la descripción de las métricas que siguen se hará uso del término *Intersección sobre unión* o en su defecto *IoU* para incluir también la métrica de *OXS* cuando hablemos de puntos. De esta forma se detalla las métricas de las que se hará uso para evaluar el modelo de una forma genérica, independientemente de afectar a cajas limitantes, o puntos.

4.11.3. Precisión (Pr)

Es la precisión del objeto detectado, indicando cuantas instancias detectadas son correctas, estableciendo la capacidad de evitar falsos positivos. Es la proporción que cuantifica la proporción de positivos que son realmente positivos. Sigue la siguiente formula:

$$Precision = \frac{VP}{VP + FP}$$

Siendo *VP* los verdaderos positivos y *FP* los falsos positivos.

Es necesario saber la precisión se calculará según un umbral dado, que hará que la detección se clasifique como FP o TP. Vemos en la [Figura 4.37](#) como se clasificarían distintas predicciones si se establece el umbral de IoU a 0.5. En nuestro caso y por

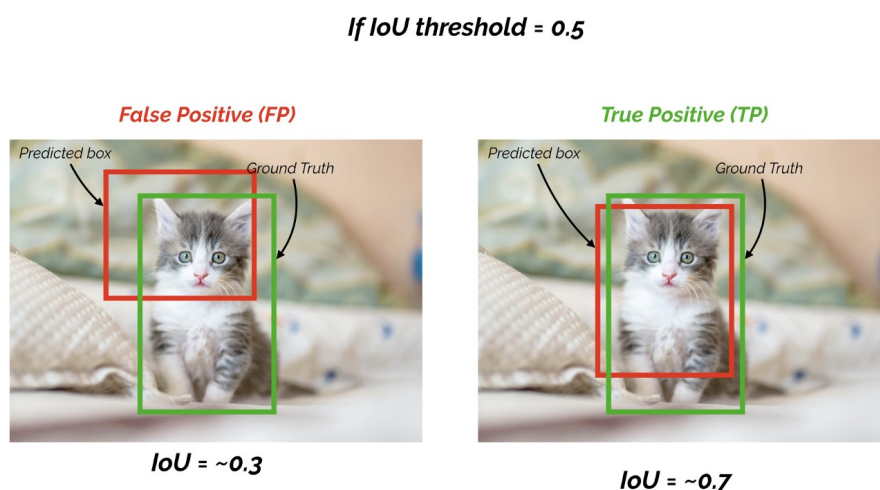


Figura 4.37: Cálculo del IoU con umbral a 0.5

defecto, durante la validación del modelo este valor se establece a 0.6.

4.11.4. Recuperación (Re)

Indica la capacidad de detectar una instancia relevante, entre todas las instancias relevantes, estableciendo la capacidad de recuperar todos los positivos de una clase. Sigue la siguiente fórmula:

$$Recuperacion = \frac{VP}{VP + FN}$$

Siendo *VP* los verdaderos positivos y *FN* los falsos negativos.

4.11.5. Precisión Media Promedio (mAP)

Es una de las medidas de rendimiento más usadas en la comunidad de investigación de CV. Su formula se basa en la matriz de confusión, intersección sobre unión, recuperación y precisión [97].

Conociendo la precisión media (AP), que es el área bajo la curva precisión-recuperación, un valor que engloba tanto el rendimiento de precisión como la tasa de recuperación, la métrica **mAP** amplía el concepto de AP, calculando los valores medios de precisión para múltiples clases. Tendremos dos mAP diferentes:

- **mAP50**. Precisión media promedio con umbral intersección sobre unión de 0.50. Es una medida de la precisión del modelo considerando sólo las detecciones consideradas fáciles.
- **mAP50-95**. La media de la precisión media calculada con distintos umbrales de intersección sobre unión, que van desde 0.50 a 0.95. Esto permite obtener una visión global del rendimiento del modelo en distintos niveles de dificultad de detección.

Capítulo 5

EXPERIMENTACIÓN Y RESULTADOS

A continuación, se desarrollará la experimentación que se lleva a cabo, presentando los distintos resultados obtenidos en la misma.

Se entrenarán los distintos modelos con varias configuraciones de hiper-parámetros detallados en la sección 4.10, sobre el conjunto de datos 4.2. Como mejora a esta experimentación inicial, se compararán con los mismos modelos, pero entrenados ahora sobre el dataset 4.3 con datos aumentados.

Se presentarán en este capítulo los resultados obtenidos tras el entrenamiento de cada uno de los modelos. Se obtendrán en primer lugar los resultados correspondientes al desempeño del modelo sobre el conjunto de validación del propio conjunto de datos. A modo de mejora en la experimentación, y ateniéndonos a un caso más real, se realizará una posterior validación sobre el conjunto de datos 4.4 siguiendo lo relatado en la sección 4.8 y 4.8.3.

Es necesario destacar al lector que aquellas tablas de resultados presentadas, en las que no aparezcan valores numéricos será por la imposibilidad de realizar la ejecución correspondiente. Por lo general, esto será a causa de falta de recursos del sistema. Por ello, quedarán fuera del estudio realizado. De cualquier modo se reflejará en las tablas correspondientes el hecho de que se ha llegado a probar dichas configuraciones aunque sin éxito.

5.1. Conjunto de datos sin aumentar

Partiendo de lo relatado en la sección anterior, se realiza el entrenamiento de los distintos modelos, cada uno con una diferente configuración de hiper-parámetros, y utilizando como conjunto de entrenamiento el conjunto de datos sin aumentar *rawDataset* detallado en la Tabla 4.2.

Con el objetivo de facilitar al lector un primer acercamiento a la interpretación de estas tablas, se marcarán en color rojo aquella métrica de entre las configuraciones de hiperparámetros que tenga el resultado más bajo respecto a las demás configuraciones. Por el contrario, la mejor métrica de entre las distintas configuraciones de hiperparámetros será marcada en verde. En el caso de no haber diferencia entre las métricas resultantes de distintas configuraciones de hiperparámetros, no se marcarán en ningún color.

Tras los resultados mostrados en las tablas, el lector podrá ver cuál de las configuraciones se plantea a priori más favorable según las métricas mostradas. Esto servirá para la comparación posterior de las mejores configuraciones para los modelos planteados en la sección 5.4.

A continuación se presentan los resultados obtenidos por cada uno de los modelos.

5.1.1. Modelo L

50 epochs

A continuación vemos las tablas 5.1 y 5.2, correspondientes a las validaciones de un conjunto interno al conjunto de datos y externo respectivamente. Métricas correspondientes al entrenamiento de los modelos YoloV8l configurado para 50 epochs y sin aumentar.

Modelo	Pr(B)	Re(B)	mAP50(B)	mAP50-95(B)	Pr(P)	Re(P)	mAP50(P)	mAP50-95(P)
raw_150e_16b	1	1	0.995	0.995	1	1	0.995	0.995
raw_150e_32b	1	1	0.995	0.995	1	1	0.995	0.995
raw_150e_64b	1	1	0.995	0.995	1	1	0.995	0.995

Tabla. 5.1: Métricas para distintas configuraciones de YoloV8l para 50 epochs. Datos sin aumentar. Validación del conjunto interno.

Modelo	Pr(B)	Re(B)	mAP50(B)	mAP50-95(B)	Pr(P)	Re(P)	mAP50(P)	mAP50-95(P)
raw_l50e_16b	0.689	0.674	0.647	0.55	0.7	0.844	0.726	0.662
raw_l50e_32b	0.638	0.706	0.634	0.546	0.723	0.8	0.705	0.659
raw_l50e_64b	0.657	0.665	0.639	0.543	0.656	0.876	0.725	0.667

Tabla. 5.2: Métricas para distintas configuraciones de YoloV8l para 50 epochs. Datos sin aumentar. Validación del conjunto externo.

Vemos en la tabla 5.1 correspondiente a la validación interna unos resultados prácticamente óptimos, y con poco margen de mejora aún sin haber implementado ninguna técnica como el aumento de datos. Todas las configuraciones producen los mismos resultados.

Vemos en la tabla 5.2 correspondiente a la validación de un conjunto externo, como el modelo con menor tamaño de batch parece ser en principio el que mejor resultados produce: **raw_l50e_16b**.

100 epochs

A continuación vemos las tablas 5.3 y 5.4, correspondientes a las validaciones de un conjunto interno al conjunto de datos y externo respectivamente. Métricas correspondientes al entrenamiento de los modelos YoloV8l configurado para 100 epochs y sin aumentar.

Modelo	Pr(B)	Re(B)	mAP50(B)	mAP50-95(B)	Pr(P)	Re(P)	mAP50(P)	mAP50-95(P)
raw_l100e_16b	1	1	0.995	0.994	1	1	0.995	0.995
raw_l100e_32b	1	1	0.995	0.994	1	1	0.995	0.995
raw_l100e_64b	1	1	0.995	0.995	1	1	0.995	0.995

Tabla. 5.3: Métricas para distintas configuraciones de YoloV8l para 100 epochs. Datos sin aumentar. Validación del conjunto interno.

Modelo	Pr(B)	Re(B)	mAP50(B)	mAP50-95(B)	Pr(P)	Re(P)	mAP50(P)	mAP50-95(P)
raw_l100e_16b	0.688	0.706	0.686	0.581	0.728	0.756	0.741	0.679
raw_l100e_64b	0.678	0.712	0.646	0.558	0.731	0.768	0.692	0.644
raw_l100e_32b	0.704	0.624	0.603	0.506	0.759	0.753	0.719	0.659

Tabla. 5.4: Métricas para distintas configuraciones de YoloV8l para 100 epochs. Datos sin aumentar. Validación del conjunto externo.

Vemos en la tabla 5.3 correspondiente a la validación interna unos resultados prácticamente óptimos, y con poco margen de mejora aún sin haber implementado ninguna

técnica como el aumento de datos. Las dos primeras configuraciones bajan un poco el resultado en la métrica *mAP50-95* para la caja, pero es prácticamente igual.

Vemos en la tabla 5.4 como al igual que en las configuraciones para 50 épocas, el que mejor resultados por lo general suele dar es la configuración con tamaño de batch de 16: **raw_l100e_16b**.

5.1.2. Modelo X

50 epochs

A continuación vemos las tablas 5.5 y 5.6, correspondientes a las validaciones de un conjunto interno al conjunto de datos y externo respectivamente. Métricas correspondientes al entrenamiento de los modelos YoloV8x configurado para 50 epochs y sin aumentar.

Modelo	Pr(B)	Re(B)	mAP50(B)	mAP50-95(B)	Pr(P)	Re(P)	mAP50(P)	mAP50-95(P)
raw_x50e_16b	1	1	0.995	0.995	1	1	0.995	0.995
raw_x50e_32b	1	1	0.995	0.988	1	1	0.995	0.995
raw_x50e_64b	1	1	0.995	0.994	1	1	0.995	0.995

Tabla. 5.5: Métricas para distintas configuraciones de YoloV8x para 50 epochs. Datos sin aumentar. Validación del conjunto interno.

Modelo	Pr(B)	Re(B)	mAP50(B)	mAP50-95(B)	Pr(P)	Re(P)	mAP50(P)	mAP50-95(P)
raw_x50e_16b	0.687	0.69	0.621	0.526	0.697	0.8	0.695	0.637
raw_x50e_32b	0.678	0.706	0.67	0.539	0.732	0.806	0.751	0.672
raw_x50e_64b	0.708	0.644	0.655	0.537	0.731	0.726	0.729	0.672

Tabla. 5.6: Métricas para distintas configuraciones de YoloV8x para 50 epochs. Datos sin aumentar. Validación del conjunto externo.

Vemos en la tabla 5.5 correspondiente a la validación interna unos resultados prácticamente óptimos, y con poco margen de mejora aún sin haber implementado ninguna técnica como el aumento de datos. La configuración con tamaño de batch 32 baja un poco el resultado en la métrica *mAP50-95* para la caja. La configuración superior parece ser la de tamaño de batch menor: **raw_x50e_16b**.

Vemos en la tabla 5.6 que en este modelo entrenado sobre 50 épocas, el tamaño de batch 32 parece ser el que mejor resultado da por lo general: **raw_x50e_32b**.

100 epochs

A continuación vemos las tablas 5.7 y 5.8, correspondientes a las validaciones de un conjunto interno al conjunto de datos y externo respectivamente. Métricas correspondientes al entrenamiento de los modelos YoloV8x configurado para 100 epochs y sin aumentar.

Modelo	Pr(B)	Re(B)	mAP50(B)	mAP50-95(B)	Pr(P)	Re(P)	mAP50(P)	mAP50-95(P)
raw_x100e_16b	1	1	0.995	0.993	1	1	0.995	0.995
raw_x100e_32b	1	1	0.995	0.995	1	1	0.995	0.995
raw_x100e_64b	1	1	0.995	0.994	1	1	0.995	0.995

Tabla. 5.7: Métricas para distintas configuraciones de YoloV8x para 100 epochs. Datos sin aumentar. Validación del conjunto interno.

Modelo	Pr(B)	Re(B)	mAP50(B)	mAP50-95(B)	Pr(P)	Re(P)	mAP50(P)	mAP50-95(P)
raw_x100e_16b	0.689	0.657	0.634	0.528	0.742	0.719	0.723	0.66
raw_x100e_32b	0.681	0.703	0.644	0.521	0.746	0.776	0.725	0.667
raw_x100e_64b	0.673	0.671	0.642	0.536	0.642	0.885	0.73	0.659

Tabla. 5.8: Métricas para distintas configuraciones de YoloV8x para 100 epochs. Datos sin aumentar. Validación del conjunto externo.

Vemos en la tabla 5.7 correspondiente a la validación interna unos resultados prácticamente óptimos, y con poco margen de mejora aún sin haber implementado ninguna técnica como el aumento de datos. La configuración con tamaño de batch 16 baja un poco el resultado en la métrica *mAP50-95* para la caja, aunque prácticamente podríamos considerar el mismo resultado.

Vemos en la tabla 5.8 que al igual que en la configuración previa con 50 épocas, esta configuración de 100 épocas parece funcionar mejor un tamaño de batch de 32: **raw_x100e_32b**.

5.1.3. Modelo X-P6

50 epochs

Durante la experimentación no es posible realizar ninguna ejecución para 50 épocas pues todos los intentos realizados fallan por sobrecarga del sistema. El resto de la experimentación quedará excluida para este modelo y esta configuración.

100 epochs

A continuación vemos la tabla 5.9, correspondientes a las validaciones de un conjunto externo al conjunto de datos. Métricas correspondiente al entrenamiento de los modelos YoloV8xp6 configurado para 100 epochs y sin aumentar. Solo es fructífero el entrenamiento de uno de los modelos.

El resto de la experimentación quedará excluida para este modelo y esta configuración.

Modelo	Pr(B)	Re(B)	mAP50(B)	mAP50-95(B)	Pr(P)	Re(P)	mAP50(P)	mAP50-95(P)
raw_xp6_100e_16b	—	—	—	—	—	—	—	—
raw_xp6_100e_32b	0.699	0.621	0.597	0.502	0.747	0.685	0.688	0.632
raw_xp6_100e_64b	—	—	—	—	—	—	—	—

Tabla. 5.9: Métricas para distintas configuraciones de YoloV8xp6 para 100 epochs. Datos sin aumentar. Validación del conjunto externo.

5.2. Conjunto de datos aumentados

Como mejora del entrenamiento de los modelos con un conjunto de datos básico, se utilizará el conjunto de datos aumentado *mainDataset* detallado en la Tabla 4.3. A continuación se presentan los resultados por cada uno de los modelos.

5.2.1. Modelo L

50 epochs

A continuación vemos las tablas 5.10 y 5.11, con las métricas correspondientes al entrenamiento de los modelos YoloV8l configurado para 50 epochs con datos aumentados.

Vemos en la tabla 5.10 correspondiente a la validación interna unos resultados prácticamente óptimos, aunque con pequeñas variaciones en la métrica *mAP50-95* para la caja. La configuración con tamaño de batch 16 parece funcionar un poco peor con el uso de datos aumentados. Los dos restantes son los que mejor funcionan: **50e_32b** o **50e_64b**.

Modelo	Pr(B)	Re(B)	mAP50(B)	mAP50-95(B)	Pr(P)	Re(P)	mAP50(P)	mAP50-95(P)
I50e_16b	1	1	0.995	0.938	1	1	0.995	0.995
I50e_32b	1	1	0.995	0.94	1	1	0.995	0.995
I50e_64b	1	1	0.995	0.947	1	1	0.995	0.995

Tabla. 5.10: Métricas para distintas configuraciones de YoloV8l para 50 epochs. Datos aumentados. Validación del conjunto interno.

Modelo	Pr(B)	Re(B)	mAP50(B)	mAP50-95(B)	Pr(P)	Re(P)	mAP50(P)	mAP50-95(P)
I50e_16b	0.753	0.794	0.694	0.505	0.787	0.926	0.783	0.737
I50e_32b	0.759	0.807	0.713	0.536	0.783	0.893	0.792	0.720
I50e_64b	0.731	0.838	0.687	0.528	0.803	0.946	0.805	0.760

Tabla. 5.11: Métricas para distintas configuraciones de YoloV8l para 50 epochs. Datos aumentados. Validación del conjunto externo.

En la tabla 5.11 vemos como con el aumento de datos, y para una configuración de 50 épocas, el tamaño de batch 32 y 64 parecen funcionar mejor. Si nos fijamos más en las métricas relativas a la pose, el tamaño de batch 64 es superior: **I50e_64b**.

100 epochs

A continuación vemos las tablas 5.12 y 5.13, con las métricas correspondientes al entrenamiento de los modelos YoloV8l configurado para 100 epochs con datos aumentados.

Modelo	Pr(B)	Re(B)	mAP50(B)	mAP50-95(B)	Pr(P)	Re(P)	mAP50(P)	mAP50-95(P)
I100e_16b	1	1	0.995	0.937	1	1	0.995	0.995
I100e_32b	1	1	0.995	0.966	1	1	0.995	0.995
I100e_64b	1	1	0.995	0.949	1	1	0.995	0.995

Tabla. 5.12: Métricas para distintas configuraciones de YoloV8l para 100 epochs. Datos aumentados. Validación del conjunto interno.

Vemos en la tabla 5.12 correspondiente a la validación interna unos resultados prácticamente óptimos, aunque con pequeñas variaciones en la métrica *mAP50-95* para la caja. La configuración con tamaño de batch 16 parece funcionar un poco peor con el uso de datos aumentados. El que mejor resultados da es para tamaño de batch 32: **I100e_32b**

En la tabla 5.13 podemos encontrar que para 100 épocas, al igual que con 50 épocas, el uso de datos aumentados beneficia a las configuraciones de batch 32 y 64.

Modelo	Pr(B)	Re(B)	mAP50(B)	mAP50-95(B)	Pr(P)	Re(P)	mAP50(P)	mAP50-95(P)
l100e_16b	0.752	0.756	0.699	0.527	0.749	0.874	0.771	0.726
l100e_32b	0.715	0.826	0.711	0.544	0.776	0.895	0.775	0.734
l100e_64b	0.715	0.789	0.673	0.523	0.777	0.894	0.778	0.728

Tabla. 5.13: Métricas para distintas configuraciones de YoloV8l para 100 epochs. Datos aumentados. Validación del conjunto externo.

En este caso, el tamaño de batch 32 parece ser ligeramente superior: **l100e_32b**.

5.2.2. Modelo X

50 epochs

A continuación vemos las tablas 5.14 y 5.15, con las métricas correspondientes al entrenamiento de los modelos YoloV8x configurado para 50 epochs con datos aumentados.

Modelo	Pr(B)	Re(B)	mAP50(B)	mAP50-95(B)	Pr(P)	Re(P)	mAP50(P)	mAP50-95(P)
x50e_16b	1	1	0.995	0.946	1	1	0.995	0.995
x50e_32b	1	1	0.995	0.891	1	1	0.995	0.994
x50e_64b	1	1	0.995	0.935	1	1	0.995	0.995

Tabla. 5.14: Métricas para distintas configuraciones de YoloV8x para 50 epochs. Datos aumentados. Validación del conjunto interno.

Modelo	Pr(B)	Re(B)	mAP50(B)	mAP50-95(B)	Pr(P)	Re(P)	mAP50(P)	mAP50-95(P)
x50e_16b	0.761	0.768	0.688	0.511	0.756	0.921	0.765	0.718
x50e_32b	0.726	0.786	0.703	0.487	0.769	0.844	0.761	0.723
x50e_64b	0.724	0.785	0.695	0.511	0.754	0.875	0.771	0.721

Tabla. 5.15: Métricas para distintas configuraciones de YoloV8x para 50 epochs. Datos aumentados. Validación del conjunto externo.

Como podemos ver en la tabla 5.14 correspondiente a la validación interna, el caso del aumento de datos para este modelo parece perjudicar a sus resultados relativos a la métrica *mAP50-95* para la caja. La configuración de tamaño de batch 32 baja su rendimiento. El que mejor funciona es el tamaño de batch 16: **x50e_16b**

En la tabla 5.15 referente al conjunto de validación externo para datos aumentados y 50 épocas, parece menos claro cuál de las configuraciones aparenta un mejor

rendimiento. Podríamos considerar las configuraciones de tamaño de batch 16 y 32: **x50e_16b** o **x50e_32b**.

100 epochs

A continuación vemos las tablas 5.16 y 5.17, con las métricas correspondientes al entrenamiento de los modelos YoloV8x configurado para 100 epochs con datos aumentados.

Modelo	Pr(B)	Re(B)	mAP50(B)	mAP50-95(B)	Pr(P)	Re(P)	mAP50(P)	mAP50-95(P)
x100e_16b	1	1	0.995	0.991	1	1	0.995	0.995
x100e_32b	1	1	0.995	0.990	1	1	0.995	0.995
x100e_64b	1	1	0.995	0.989	1	1	0.995	0.995

Tabla. 5.16: Métricas para distintas configuraciones de YoloV8x para 100 epochs. Datos aumentados. Validación del conjunto interno.

Modelo	Pr(B)	Re(B)	mAP50(B)	mAP50-95(B)	Pr(P)	Re(P)	mAP50(P)	mAP50-95(P)
x100e_16b	0.684	0.721	0.644	0.505	0.694	0.947	0.754	0.697
x100e_32b	0.719	0.782	0.684	0.543	0.780	0.900	0.792	0.752
x100e_64b	—	—	—	—	—	—	—	—

Tabla. 5.17: Métricas para distintas configuraciones de YoloV8x para 100 epochs. Datos aumentados Validación del conjunto externo.

Vemos en la tabla 5.16 correspondiente a la validación interna unos resultados prácticamente óptimos, aunque con pequeñas variaciones en la métrica *mAP50-95* para la caja. La configuración con tamaño de batch 64 parece funcionar un poco peor con el uso de datos aumentados. Sin embargo, todos los resultados podríamos considerarlos como óptimos.

En la tabla 5.17 vemos de forma clara como la configuración de tamaño de batch 32 para 100 épocas y datos aumentados es superior a la única otra de tamaño de batch 16: **x100e_32b**.

5.2.3. Modelo X-P6

50 epochs

A continuación vemos la tabla 5.18, con las métricas correspondientes al entrenamiento de los modelos YoloV8xp6 configurado para 50 epochs con datos aumentados.

Vemos cómo solo es posible el registro de un modelo, y que además fallará durante la ejecución. El resto de la experimentación quedará excluida para este modelo y esta configuración.

Modelo	Pr(B)	Re(B)	mAP50(B)	mAP50-95(B)	Pr(P)	Re(P)	mAP50(P)	mAP50-95(P)
raw_xp6_50e_16b	0.663	0.744	0.571	0.348	0.784	0.942	0.794	0.747
raw_xp6_50e_32b	—	—	—	—	—	—	—	—
raw_xp6_50e_64b	—	—	—	—	—	—	—	—

Tabla. 5.18: Métricas para distintas configuraciones de YoloV8xp6 para 50 epochs. Datos aumentados. Validación del conjunto externo. Falla la ejecución en la época 10.

100 epochs

A continuación vemos la tabla 5.19, con las métricas correspondientes al entrenamiento de los modelos YoloV8xp6 configurado para 100 epochs con datos aumentados.

Vemos como solo es posible el registro de un modelo, y que además fallará durante la ejecución. El resto de la experimentación quedará excluida para este modelo y esta configuración.

Modelo	Pr(B)	Re(B)	mAP50(B)	mAP50-95(B)	Pr(P)	Re(P)	mAP50(P)	mAP50-95(P)
raw_xp6_100e_16b	0.724	0.773	0.657	0.414	0.753	0.915	0.786	0.726
raw_xp6_100e_32b	—	—	—	—	—	—	—	—
raw_xp6_100e_64b	—	—	—	—	—	—	—	—

Tabla. 5.19: Métricas para distintas configuraciones de YoloV8xp6 para 100 epochs. Datos aumentados. Validación del conjunto externo. Falla la ejecución en la época 10.

5.3. Comparativa con la técnica de aumento de datos

A continuación se presentan las comparaciones de cada modelo entrenado con y sin aumento de datos, con el objetivo de ver en qué medida y cómo de efectiva es esta técnica para mejorar los resultados de nuestro modelo en su aplicación.

Es importante destacar, como detallado en la sección 4.10, que aquellos modelos cuyo nombre se le aplica el prefijo *raw* son aquellos entrenados sin datos aumentados, haciendo así referencia a un entrenamiento en crudo o simple.

Con el objetivo de facilitar al lector un primer acercamiento a esta comparativa de forma sencilla, se marcarán en color rojo aquellas medidas de rendimiento que empeoren al implementar el aumento de datos. Por el contrario, se marcarán en verde aquellas medidas que mejoren con el aumento de datos.

5.3.1. Modelo L

50 epochs

A continuación vemos la comparativa en la tabla 5.20, con las métricas correspondientes al entrenamiento de los modelos YoloV8l configurado para 50 epochs con datos sin aumento y con aumento para la validación del conjunto interno.

Modelo	Pr(B)	Re(B)	mAP50(B)	mAP50-95(B)	Pr(P)	Re(P)	mAP50(P)	mAP50-95(P)
raw_l50e_16b	1	1	0.995	0.995	1	1	0.995	0.995
l50e_16b	1	1	0.995	0.938	1	1	0.995	0.995
raw_l50e_32b	1	1	0.995	0.995	1	1	0.995	0.995
l50e_32b	1	1	0.995	0.94	1	1	0.995	0.995
raw_l50e_64b	1	1	0.995	0.995	1	1	0.995	0.995
l50e_64b	1	1	0.995	0.947	1	1	0.995	0.995

Tabla. 5.20: Comparativa de métricas para distintas configuraciones de YoloV8l para 50 epochs. Validación del conjunto interno.

Podemos observar en la comparativa de la tabla 5.20 cómo los resultados son iguales para todas las métricas exceptuando para la métrica de *mAP50-95* de la caja, la cual empeora.

A continuación vemos la comparativa en la tabla 5.21, con las métricas correspondientes al entrenamiento de los modelos YoloV8l configurado para 50 epochs con

datos sin aumento y con aumento para la validación del conjunto externo.

Modelo	Pr(B)	Re(B)	mAP50(B)	mAP50-95(B)	Pr(P)	Re(P)	mAP50(P)	mAP50-95(P)
raw_l50e_16b	0.689	0.674	0.647	0.55	0.7	0.844	0.726	0.662
l50e_16b	0.753	0.794	0.694	0.505	0.787	0.926	0.783	0.737
raw_l50e_32b	0.638	0.706	0.634	0.546	0.723	0.8	0.705	0.659
l50e_32b	0.759	0.807	0.713	0.536	0.783	0.893	0.792	0.720
raw_l50e_64b	0.657	0.665	0.639	0.543	0.656	0.876	0.725	0.667
l50e_64b	0.731	0.838	0.687	0.528	0.803	0.946	0.805	0.760

Tabla. 5.21: Comparativa de métricas para distintas configuraciones de YoloV8l para 50 epochs. Validación del conjunto externo.

En el caso de la tabla 5.21 sí que el aumento de datos supone una mejora significativa en las métricas. Del mismo modo, la métrica de *mAP50-95* de la caja empeora ligeramente.

100 epochs

A continuación vemos la comparativa en la tabla 5.22, con las métricas correspondientes al entrenamiento de los modelos YoloV8l configurado para 100 epochs con datos sin aumento y con aumento para la validación del conjunto interno.

Modelo	Pr(B)	Re(B)	mAP50(B)	mAP50-95(B)	Pr(P)	Re(P)	mAP50(P)	mAP50-95(P)
raw_l100e_16b	1	1	0.995	0.994	1	1	0.995	0.995
l100e_16b	1	1	0.995	0.937	1	1	0.995	0.995
raw_l100e_32b	1	1	0.995	0.994	1	1	0.995	0.995
l100e_32b	1	1	0.995	0.966	1	1	0.995	0.995
raw_l100e_64b	1	1	0.995	0.995	1	1	0.995	0.995
l100e_64b	1	1	0.995	0.949	1	1	0.995	0.995

Tabla. 5.22: Comparativa de métricas para distintas configuraciones de YoloV8l para 100 epochs. Validación del conjunto interno.

Podemos observar en la comparativa de la tabla 5.22 cómo los resultados son iguales para todas las métricas exceptuando para la métrica de *mAP50-95* de la caja, la cual empeora.

A continuación vemos la comparativa en la tabla 5.23, con las métricas correspondientes al entrenamiento de los modelos YoloV8l configurado para 100 epochs con datos sin aumento y con aumento para la validación del conjunto externo.

Modelo	Pr(B)	Re(B)	mAP50(B)	mAP50-95(B)	Pr(P)	Re(P)	mAP50(P)	mAP50-95(P)
raw_l100e_16b	0.688	0.706	0.686	0.581	0.728	0.756	0.741	0.679
l100e_16b	0.752	0.756	0.699	0.527	0.749	0.874	0.771	0.726
raw_l100e_32b	0.704	0.624	0.603	0.506	0.759	0.753	0.719	0.659
l100e_32b	0.715	0.826	0.711	0.544	0.776	0.895	0.775	0.734
raw_l100e_64b	0.678	0.712	0.646	0.558	0.731	0.768	0.692	0.644
l100e_64b	0.715	0.789	0.673	0.523	0.777	0.894	0.778	0.728

Tabla. 5.23: Comparativa de métricas para distintas configuraciones de YoloV8l para 100 epochs. Validación del conjunto externo.

Del mismo modo que para las configuraciones de 50 épocas, en la tabla 5.23 vemos como todas las métricas mejoran de forma significativa exceptuando la métrica de *mAP50-95* de la caja, la cual empeora.

5.3.2. Modelo X

50 epochs

A continuación vemos la comparativa en la tabla 5.24, con las métricas correspondientes al entrenamiento de los modelos YoloV8x configurado para 50 epochs con datos sin aumento y con aumento para la validación del conjunto interno.

Modelo	Pr(B)	Re(B)	mAP50(B)	mAP50-95(B)	Pr(P)	Re(P)	mAP50(P)	mAP50-95(P)
raw_x50e_16b	1	1	0.995	0.995	1	1	0.995	0.995
x50e_16b	1	1	0.995	0.946	1	1	0.995	0.995
raw_x50e_32b	1	1	0.995	0.988	1	1	0.995	0.995
x50e_32b	1	1	0.995	0.891	1	1	0.995	0.994
raw_x50e_64b	1	1	0.995	0.994	1	1	0.995	0.995
x50e_64b	1	1	0.995	0.935	1	1	0.995	0.995

Tabla. 5.24: Comparativa de métricas para distintas configuraciones de YoloV8x para 50 epochs. Validación del conjunto interno.

Podemos observar en la comparativa de la tabla 5.24 cómo los resultados son iguales para todas las métricas exceptuando para la métrica de *mAP50-95* de la caja, la cual empeora. Además, encontramos un ligero empeoramiento en la análoga para la pose, aunque es prácticamente igual.

A continuación vemos la comparativa en la tabla 5.25, con las métricas correspondientes al entrenamiento de los modelos YoloV8x configurado para 50 epochs con datos sin aumento y con aumento para la validación del conjunto externo.

Modelo	Pr(B)	Re(B)	mAP50(B)	mAP50-95(B)	Pr(P)	Re(P)	mAP50(P)	mAP50-95(P)
raw_x50e_16b	0.687	0.69	0.621	0.526	0.697	0.8	0.695	0.637
x50e_16b	0.761	0.768	0.688	0.511	0.756	0.921	0.765	0.718
raw_x50e_32b	0.678	0.706	0.67	0.539	0.732	0.806	0.751	0.672
x50e_32b	0.726	0.786	0.703	0.487	0.769	0.844	0.761	0.723
raw_x50e_64b	0.708	0.644	0.655	0.537	0.731	0.726	0.729	0.672
x50e_64b	0.724	0.785	0.695	0.511	0.754	0.875	0.771	0.721

Tabla. 5.25: Comparativa de métricas para distintas configuraciones de YoloV8x para 50 epochs. Validación del conjunto externo.

Del mismo modo que la misma configuración para el modelo anterior, con la validación de un conjunto externo el aumento de datos supone una mejora significativa en todas las métricas, exceptuando la métrica de *mAP50-95* de la caja.

100 epochs

A continuación vemos la comparativa en la tabla 5.26, con las métricas correspondientes al entrenamiento de los modelos YoloV8x configurado para 100 epochs con datos sin aumento y con aumento para la validación del conjunto interno.

Modelo	Pr(B)	Re(B)	mAP50(B)	mAP50-95(B)	Pr(P)	Re(P)	mAP50(P)	mAP50-95(P)
raw_x100e_16b	1	1	0.995	0.993	1	1	0.995	0.995
x100e_16b	1	1	0.995	0.991	1	1	0.995	0.995
raw_x100e_32b	1	1	0.995	0.995	1	1	0.995	0.995
x100e_32b	1	1	0.995	0.990	1	1	0.995	0.995
raw_x100e_64b	1	1	0.995	0.994	1	1	0.995	0.995
x100e_64b	1	1	0.995	0.989	1	1	0.995	0.995

Tabla. 5.26: Comparativa de métricas para distintas configuraciones de YoloV8x para 100 epochs. Validación del conjunto interno.

En este caso, podemos ver en la tabla 5.26 como vuelve a empeorar la métrica de *mAP50-95* de la caja, aunque de forma muy ligera en este caso.

A continuación vemos la comparativa en la tabla 5.27, con las métricas correspondientes al entrenamiento de los modelos YoloV8x configurado para 100 epochs con datos sin aumento y con aumento para la validación del conjunto externo.

En el caso de la tabla 5.27 las configuraciones para 100 épocas se ven mejoradas por lo general con el aumento de datos. Únicamente dos valores parecen empeorar, pero muy ligeramente.

Modelo	Pr(B)	Re(B)	mAP50(B)	mAP50-95(B)	Pr(P)	Re(P)	mAP50(P)	mAP50-95(P)
raw_x100e_16b	0.689	0.657	0.634	0.528	0.742	0.719	0.723	0.66
x100e_16b	0.684	0.721	0.644	0.505	0.694	0.947	0.754	0.697
raw_x100e_32b	0.681	0.703	0.644	0.521	0.746	0.776	0.725	0.667
x100e_32b	0.719	0.782	0.684	0.543	0.780	0.900	0.792	0.752
raw_x100e_64b	0.673	0.671	0.642	0.536	0.642	0.885	0.73	0.659
x100e_64b	-	-	-	-	-	-	-	-

Tabla. 5.27: Comparativa de métricas para distintas configuraciones de YoloV8x para 100 epochs. Validación del conjunto externo.

Parece claro como la técnica de aumento de datos parece beneficiar en los resultados obtenidos de forma significativa. En la sección 6.1 de las conclusiones se detallará el motivo por el que la métrica *mAP50-95* relativa a la caja parece sin embargo empeorar con el aumento de datos.

5.4. Comparativa genérica de los mejores modelos

Considerado que la técnica de aumento de datos mejora por lo general de forma significativa los resultados del modelo entrenado, a continuación se presentan las comparaciones de los mejores modelos obtenidos en la sección 5.2 para cada configuración de épocas. En caso de tener dos resultados similares o no tener una superioridad notable, se incluirán ambos.

Se ha visto que las métricas para la validación del conjunto externo no producen una variabilidad significativa, ya que por lo general son casi óptimas. Es por esto que en esta sección se compararán únicamente aquellos modelos que hayan sido validados con un conjunto externo, lo que nos dará una mejor idea de qué configuración es a priori la más óptima en base a las métricas de validación presentadas.

Modelo	Pr(B)	Re(B)	mAP50(B)	mAP50-95(B)	Pr(P)	Re(P)	mAP50(P)	mAP50-95(P)
l50e_32b	0.759	0.807	0.713	0.536	0.783	0.893	0.792	0.720
l50e_64b	0.731	0.838	0.687	0.528	0.803	0.946	0.805	0.760
l100e_32b	0.715	0.826	0.711	0.544	0.776	0.895	0.775	0.734
x50e_16b	0.761	0.768	0.688	0.511	0.756	0.921	0.765	0.718
x50e_32b	0.726	0.786	0.703	0.487	0.769	0.844	0.761	0.723
x100e_32b	0.719	0.782	0.684	0.543	0.780	0.900	0.792	0.752

Tabla. 5.28: Comparación general de los mejores modelos obtenidos

Se presenta la tabla 5.28 en la que podemos ver los modelos con mejor desempeño según las métricas de validación.

Parece sencillo ver a partir de la tabla [5.28](#) cómo el modelo ***150e_64b***, entrenado durante **50 épocas** y un tamaño de **batch de 64**, destaca por lo general sobre los demás. Considerando sobre todo las métricas relativas a la pose, tiene unos resultados bastante superiores comparados con modelos y otras configuraciones.

Capítulo 6

CONCLUSIONES

A continuación se presentan las conclusiones extraídas tras la elaboración del presente trabajo en su totalidad, y finalizada la experimentación. Se plantearán también al finalizar, en la sección [6.1](#) posibles mejoras que se han encontrado en el desarrollo propuesto y que no han sido implementadas en el trabajo actual. Como propuestas de una futura continuación de este estudio, se describirán algunas técnicas en la sección [6.2](#) estudiadas ligeramente durante el proceso de investigación, pero que no han sido implantadas por suponer una complejidad excesiva en este trabajo.

A lo largo de este documento, el lector ha podido adentrarse dentro del mundo de la estimación de pose, para lo que ha sido necesario estudiar en profundidad todo lo relativo a la disciplina subyacente, incluyendo el aprendizaje profundo. Pasando a su vez por la faceta histórica de la misma, y de todo lo que ella involucra, como es la disciplina de la inteligencia artificial, el aprendizaje automático, y la visión por computador.

Todo este recorrido ha permitido obtener el estado del arte actual de la tarea que aplica en este trabajo, la estimación de poses. Dentro de esta rama, que aunque no tan conocida, plantea un interés creciente en la actualidad, se han considerado las aplicaciones más comunes. Nos encontramos entonces en un punto en el que la estimación de poses en humanos abarca la mayor parte de la atención científica actual.

La estimación de poses en animales, aunque plantea una gran trayectoria en el futuro, en la actualidad sigue siendo una gran desconocida comparada con la estimación de poses humana. Es por esto, que se ha realizado un cuidadoso estudio de todo aquello a nuestra disposición para una correcta experimentación del problema planteado. Tras el análisis no fructífero del uso de ciertas herramientas existentes para estimar la pose en sujetos de prueba similares al propuesto, se han estudiado modelos y arquitecturas

que nos permitan llevar a cabo la tarea desde un nivel más bajo.

El uso de un modelo de aprendizaje profundo que nos permita la estimación de poses, presenta un gran potencial aplicable para solucionar el problema planteado. Es por esto que tras el estudio de algunos de los existentes, se decide hacer uso de diversos modelos **YoloV8**.

Es sabido que la calidad de los resultados del entrenamiento de modelos de aprendizaje profundo, reside en gran parte en los datos de entrenamiento. Por ello, se ha creado un conjunto de datos a partir de una serie de vídeos proporcionados para el estudio planteado.

Se plantearán dos técnicas de validación de los modelos resultantes tras el entrenamiento, las cuales son descritas en la sección 4.8 para que el lector pueda entender la motivación de algo que parece estar fuera de lo habitual en una experimentación de este tipo.

En lo relativo a la experimentación, originalmente se plantea el entrenamiento de 3 modelos YoloV8, con distintas características (Sección 4.3). A cada uno de los modelos se le aplicarán 6 configuraciones diferentes de hiperparámetros, lo que supone un total de 18 modelos.

Posteriormente, se plantearán otros 18 modelos que entrenarán sobre los datos aumentados, lo que nos permitirá comparar las mejoras obtenidas con esta técnica. Esto supondrá ahora un total de 36 modelos.

Teniendo en cuenta que se usarán dos formas de validación de los modelos resultantes, de forma teórica contaremos al final de la experimentación con 72 conjuntos de métricas a comparar. Sin embargo, como se detalla en la sección correspondiente, algunos de los entrenamientos realizados fallan por escasez de recursos o inestabilidad del sistema, por lo que en la práctica se han comparado menos resultados, concretamente 50.

Cabe destacar que el proceso de entrenamiento de modelos ha sido seguido con una de las herramientas más potentes en la actualidad para el registro de entrenamiento de modelos de aprendizaje profundo, **Comet**.

Dado que para el tipo de validación interna planteada, los resultados son prácticamente óptimos, tal y como se anticipaba en la sección 4.8.3, se hace el estudio de las métricas de la validación externa. A diferencia de lo que en primer lugar se planteaba, aquellos modelos más sencillos y con un tamaño de batch menor, parecen funcionar por lo general mejor antes de aplicar aumento de datos. Aquellos modelos con tamaño

de batch 16 o 32 son los que mejor funcionan.

Sin embargo es interesante ver cómo tras aplicar la técnica de aumento de datos, aunque el desempeño de todos los modelos mejora, ahora parecen destacar por lo general aquellos con un tamaño de batch mayor, de 32 o 64.

Confirmando lo que se acaba de comentar, al hacer una comparativa general de los mejores resultados obtenidos durante toda la experimentación, vemos que destaca de forma clara el modelo denominado **I50e_64b**, el modelo más simple de los planteados originalmente, entrenado sobre 50 épocas, y con un tamaño de batch de 64, el mayor de los considerados.

Analizando los resultados obtenidos por el mejor de los modelos, vemos que sobre todo los relativos a la pose son bastante altos en comparación con los planteados originalmente. Entendemos también que la técnica de aumento de datos supone una mejora significativa en los resultados obtenidos, y latentes en la comparativa planteada. No hay que descuidar sin embargo las métricas relativas a la caja limítrofe, las cuales de forma obvia vemos son considerablemente peores. Este rendimiento más bajo viene dado por lo que se planteará en la próxima sección, y que se debería mejorar de cara a obtener unos mejores resultados.

Se ha podido probar en este trabajo que los métodos de aprendizaje profundo, son un requisito indispensable para la tarea concreta de estimación de poses en animales. Es importante también destacar la importancia de estudios como el planteado para el avance en otras disciplinas y ámbitos. En el caso de este trabajo, motivado en beneficio de una investigación propia de la psicología y la neurociencia. Sin embargo, este y otros trabajos demuestran que estamos trabajando en una disciplina prácticamente sin límites en lo que a posibilidades se refiere, por lo que con el suficiente esfuerzo, recursos y dedicación, los avances para la sociedad pueden ser incalculables.

6.1. Posibles mejoras

Una vez realizado el proceso de experimentación, y tras analizar los resultados, surge una cuestión no considerada anteriormente. Si observamos la comparativa de resultados obtenidos con la técnica de aumento de datos en la sección [5.3](#), vemos que una de las métricas relativas a la caja limítrofe está empeorando al usar datos aumentados.

Desde un primer momento no se considera demasiado relevante, pues el resto de

métricas parecen mejorar como esperado. Sin embargo, intentando buscar el motivo para este empeoramiento se descubre un problema en el proceso de aumento de los datos.

Como se ha visto en la sección 4.5, se aplican diversas transformaciones a imágenes y anotaciones con el fin de obtener un conjunto de datos aumentado, y que incluya casos variados. El problema aparece cuando estudiando en profundidad el proceso de transformación llevado a cabo, se descubre que en aquellas transformaciones que implican movimiento de la imagen, como rotaciones o transformaciones afines, el movimiento de la caja lleva consigo el movimiento por igual de la anotación. Vemos un ejemplo claro en la Figura 6.1, en la que vemos cómo se haría de forma real la transformación imagen-anotación. Sin embargo, cuando hablamos de anotaciones de tipo caja limítrofe, la transformación de las anotaciones no coincide de forma exacta con la transformación de la imagen. Lo habitual es que el modelo sea capaz de detectar el sujeto con una caja en la orientación adecuada, como aparece en el recuadro naranja de la Figura 6.2, lo que sería la transformación ideal.

De este modo, estamos alimentando al modelo con unas anotaciones en ocasiones no lo más exactas posibles. A causa de esto, si recordamos en qué consiste la métrica *mAP50-95* parecería lógico afirmar que no es extraño empeore.

Recordamos que en dicha métrica, se aplican distintos umbrales de IoU, por lo que para aquellos umbrales más bajos como puede ser 50, no será demasiado problema. Por el contrario, para aquellos umbrales más elevados, la predicción hecha por el modelo en lo relativo a la caja debe coincidir en gran medida, y si la caja de la anotación real tiene pequeñas variaciones, evidentemente no lo hará.

Aunque no es un factor determinante, la solución de este problema a tiempo podría maximizar el rendimiento en las métricas relativas a la caja limítrofe.

De esta forma, un modelo que funcionase a la perfección haría una predicción de la caja como aparece el recuadro naranja en la Figura 6.2, sin embargo, la predicción será probablemente marcada como errónea pues la anotación real es la que aparece verde y la IoU tendrá un valor menor que el umbral. Esto hará que empeore el valor de la métrica *mAP50-95*.

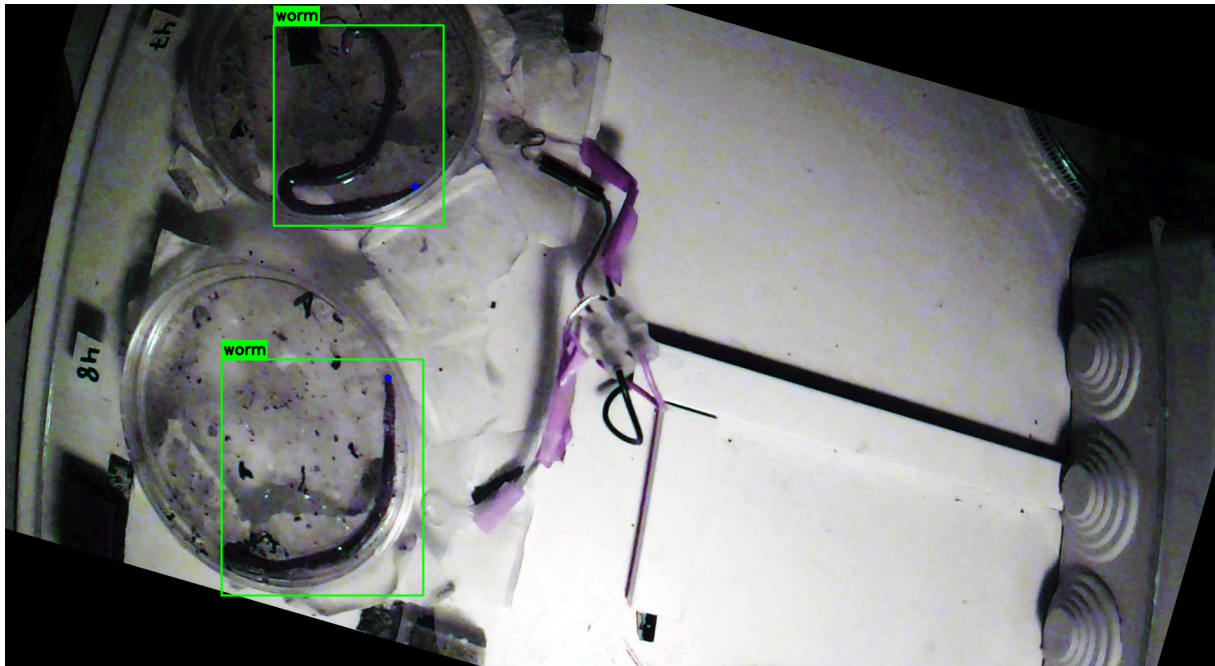


Figura 6.1: Anotaciones transformadas de la imagen ejemplo

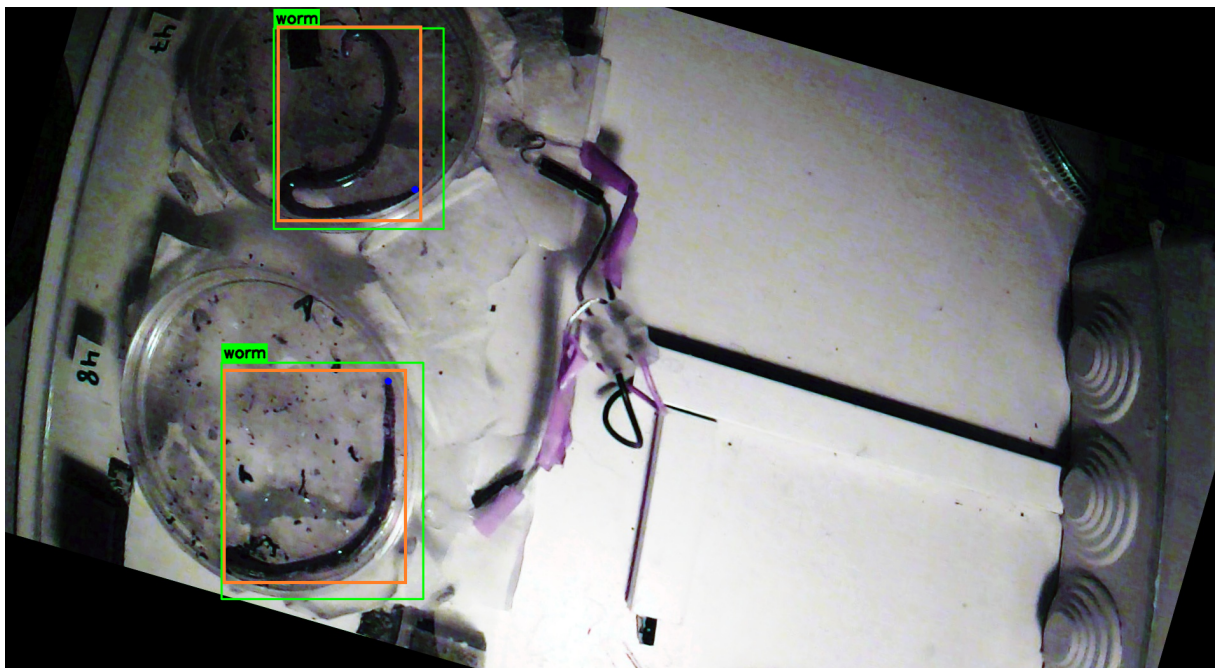


Figura 6.2: Anotaciones transformadas de la imagen ejemplo y su corrección

6.2. Propuestas a futuro

Como propuesta de mejora del presente trabajo, a continuación se describirán diversas técnicas que mejorarán los resultados obtenidos, y que podrían ser implementados a futuro.

6.2.1. Generación de datos sintéticos

El estudio relacionado con la herramienta *WormPose* [54] plantea una forma de trabajo interesante, y que aplicada al problema desarrollado en este trabajo, podría suponer unas mejoras significativas en los resultados. Dado que la misma herramienta no podría ser utilizada de forma directa para nuestro problema, y sería necesaria en cierto modo adaptarla, no encontrará el lector en esta sección nada relacionado con el código concreto del sistema, sino más bien un acercamiento al fundamento teórico y matemático de del estudio planteado, pudiendo así entender su funcionamiento y posible aplicación a nuestro problema.

Al igual que en nuestro problema, el nematodo *C. elegans* puede tomar distintas formas que complican la identificación de la espina dorsal, como podemos observar en la [Figura 6.3](#). En este mismo artículo se hace referencia a algo en lo que se viene insistiendo a lo largo del presente estudio y es:

While human annotators can quickly pinpoint the extremities of the worm body, other landmarks are less obvious. In some recordings, it is even difficult to distinguish the worm head from the tail, which makes the labeling error-prone and imprecise. Furthermore, the labeling is specific to the recording conditions and can be hard to generalize across changes in resolution, organism size, background, illumination, and to rare posture configurations not specifically isolated.

« Mientras anotadores humanos pueden de forma rápida señalar las extremidades del cuerpo del gusano, otros puntos no son tan obvios. En algunas grabaciones, es incluso difícil distinguir entre la cabeza del gusano y la cola, lo que hace que la anotación sea imprecisa y se propicien fallos. Además, la anotación es específica para las condiciones concretas, y puede ser difícil generalizar cuando se producen cambios en el tamaño del gusano, el fondo, la iluminación y posturas extrañas no asiladas. »

Es por esto, que nace *WormPose*, un algoritmo que estima la pose de los *C. ele-*

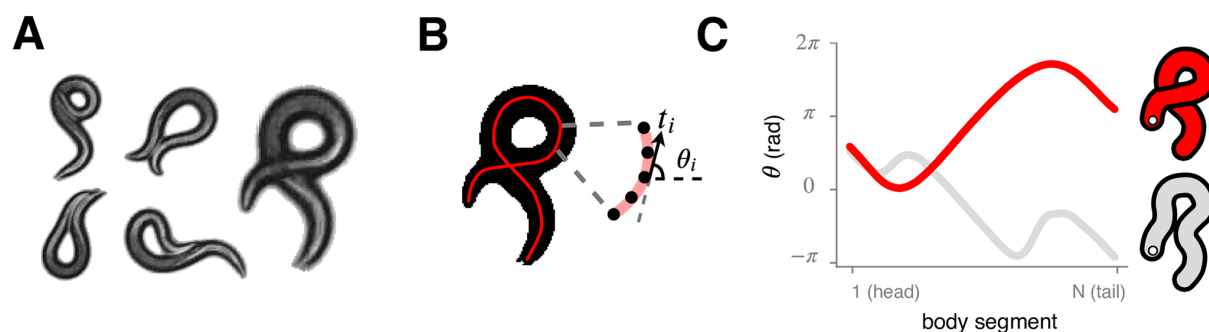


Figura 6.3: Representación matemática de las distintas posturas de un gusano

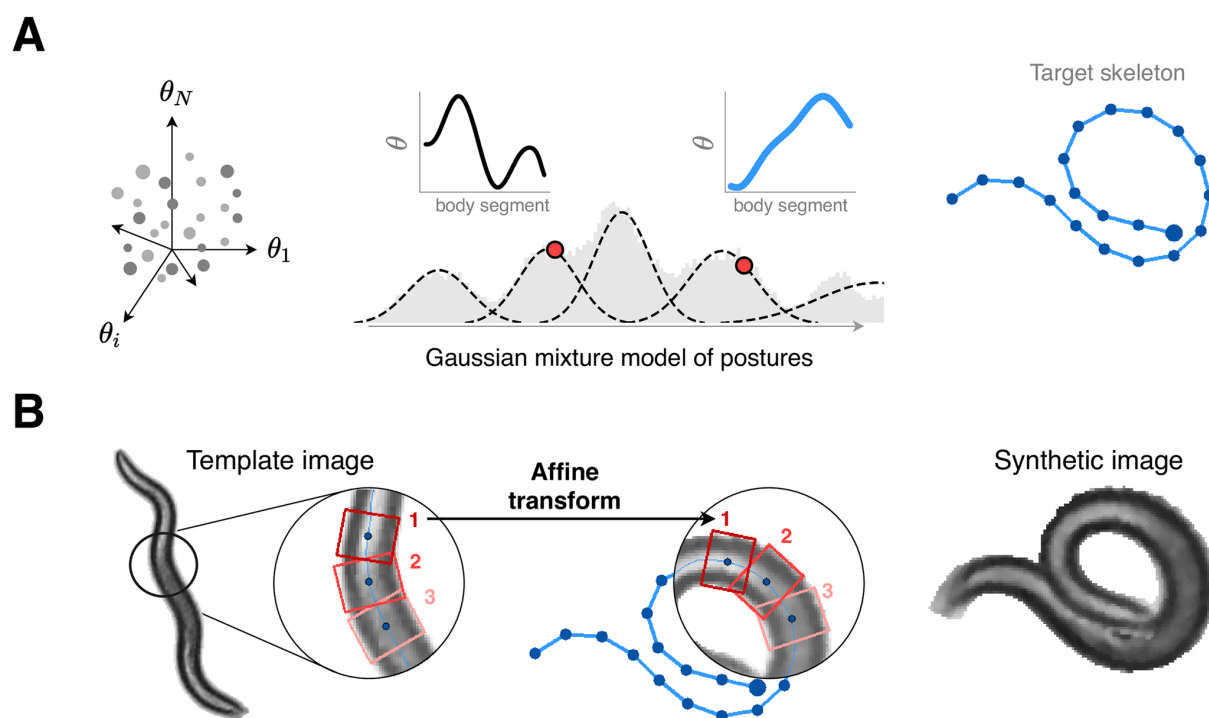


Figura 6.4: Funcionamiento de la generación de imágenes sintéticas

gans con dos ventajas principales: un modelo generativo que es capaz de crear imágenes sintéticas y realistas de gusanos, y que serán usadas para el entrenamiento de la NN. La siguiente ventaja que plantea *WormPose* es una CNN que transforma las imágenes de gusanos en su curva central, permitiendo a partir de aquí crear gusanos sintéticos en muchas más posiciones posibles, permitiendo al modelo generalizar mejor.

Se realizará de forma obvia un procesamiento de las imágenes reales, con el objetivo de obtener los mejores resultados en las imágenes sintéticas creadas. En el estudio se plantea una combinación de un modelo generativo de diferentes posturas en los gusanos, que unido a texturas procedentes de las imágenes reales, creará las imágenes sintéticas. Podemos ver el funcionamiento en la [Figura 6.4 \[99\]](#).

(A) En primer lugar se modela un espacio dimensional de las posturas del gusano con una mezcla Gaussiana que será construida a partir de imágenes previamente analizadas. Para cada una de las nuevas posturas generadas, se le aplicará una orientación global, escogida entre 0 y 2π , y se asignará la cabeza de forma aleatoria en uno de los extremos. El nuevo esqueleto generado será generado a partir de las coordenadas de los ángulos resultantes. (B) La imagen real correspondiente al gusano se dividirá a partir de su esqueleto en pequeños rectángulos, que serán asignados al correspondiente esqueleto generado, creando así una imagen sintética como se aprecia en la imagen de abajo a la derecha.

Bibliografía

- [1] Diego Andina, D. Pham, D. Andina, Antonio Vega-Corona, Juan Seijas, and J. Torres-García. Neural Networks Historical Review. February 2007. ISBN 978-0-387-37450-5. doi: 10.1007/0-387-37452-3_2.
- [2] Redacción KeepCoding. ¿Qué es una red neuronal en Deep Learning?, August 2022. URL <https://keepcoding.io/blog/red-neuronal-en-deep-learning/>. Section: Blog.
- [3] NVIDIA DGX-1: instrumento esencial para la investigación de IA, . URL <https://www.nvidia.com/es-es/data-center/dgx-1/>.
- [4] Alexander Buslaev, Vladimir I. Iglovikov, Eugene Khvedchenya, Alex Parinov, Mikhail Druzhinin, and Alexandr A. Kalinin. Albumentations: Fast and flexible image augmentations. *Information*, 11(2), 2020. ISSN 2078-2489. doi: 10.3390/info11020125. URL <https://www.mdpi.com/2078-2489/11/2/125>.
- [5] Sebastian Raschka. Model Evaluation, Model Selection, and Algorithm Selection in Machine Learning, November 2020. URL <http://arxiv.org/abs/1811.12808>. arXiv:1811.12808 [cs, stat].
- [6] Rachel Thomas. fast.ai - How (and why) to create a good validation set, November 2017. URL <https://www.fast.ai/posts/2017-11-13-validation-sets.html>.
- [7] Overview - Comet Docs, . URL <https://www.comet.com/docs/v2/api-and-sdk/javascript-panels-sdk/overview/>.
- [8] Estimación de poses | TensorFlow Lite. URL https://www.tensorflow.org/lite/examples/pose_estimation/overview?hl=es.
- [9] Le Jiang, Caleb Lee, Divyang Teotia, and Sarah Ostadabbas. Animal pose estimation: A closer look at the state-of-the-art, existing gaps and opportunities. *Computer Vision and Image Understanding*, 222:103483, 2022. ISSN 1077-3142. doi: <https://doi.org/10.1016/j.cviu.2022.103483>. URL <https://www.sciencedirect.com/science/article/pii/S1077314222000893>.

- [10] John McCarthy et al. What is artificial intelligence. 2007.
- [11] John McCarthy, Marvin L. Minsky, Nathaniel Rochester, and Claude E. Shannon. A Proposal for the Dartmouth Summer Research Project on Artificial Intelligence, August 31, 1955. *AI Magazine*, 27(4):12, December 2006. doi: 10.1609/aimag.v27i4.1904. URL <https://ojs.aaai.org/aimagazine/index.php/aimagazine/article/view/1904>.
- [12] A. L. Samuel. Some Studies in Machine Learning Using the Game of Checkers. *IBM Journal of Research and Development*, 3(3):210–229, July 1959. ISSN 0018-8646. doi: 10.1147/rd.33.0210. URL <https://ieeexplore.ieee.org/document/5392560>. Conference Name: IBM Journal of Research and Development.
- [13] Tom M. Mitchell. Does Machine Learning Really Work? *AI Magazine*, 18(3):11–11, September 1997. ISSN 2371-9621. doi: 10.1609/aimag.v18i3.1303. URL <https://ojs.aaai.org/aimagazine/index.php/aimagazine/article/view/1303>. Number: 3.
- [14] A. Géron. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*. ISBN 9781492032649.
- [15] Pádraig Cunningham, Matthieu Cord, and Sarah Jane Delany. Supervised Learning. In Matthieu Cord and Pádraig Cunningham, editors, *Machine Learning Techniques for Multimedia: Case Studies on Organization and Retrieval*, pages 21–49. Springer, Berlin, Heidelberg, 2008. ISBN 978-3-540-75171-7. doi: 10.1007/978-3-540-75171-7_2. URL https://doi.org/10.1007/978-3-540-75171-7_2.
- [16] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. Overview of Supervised Learning. In Trevor Hastie, Robert Tibshirani, and Jerome Friedman, editors, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, pages 9–41. Springer, New York, NY, 2009. ISBN 978-0-387-84858-7. doi: 10.1007/978-0-387-84858-7_2. URL https://doi.org/10.1007/978-0-387-84858-7_2.
- [17] Manuel Roberto Tolozano Benítez. *Técnicas de aprendizaje supervisado y no supervisado para el aprendizaje automatizado de computadoras*. Instituto Superior Tecnológico Bolivariano, 2015. ISBN 978-9942-17-011-8.
- [18] Lior Rokach and Oded Maimon. Supervised Learning. In Oded Maimon and Lior Rokach, editors, *Data Mining and Knowledge Discovery Handbook*, pages 133–147. Springer US, Boston, MA, 2010. ISBN 978-0-387-09823-4. doi: 10.1007/978-0-387-09823-4_8. URL https://doi.org/10.1007/978-0-387-09823-4_8.

- [19] Zoubin Ghahramani. Unsupervised Learning. In Olivier Bousquet, Ulrike von Luxburg, and Gunnar Rätsch, editors, *Advanced Lectures on Machine Learning: ML Summer Schools 2003, Canberra, Australia, February 2 - 14, 2003, Tübingen, Germany, August 4 - 16, 2003, Revised Lectures*, pages 72–112. Springer, Berlin, Heidelberg, 2004. ISBN 978-3-540-28650-9. doi: 10.1007/978-3-540-28650-9_5. URL https://doi.org/10.1007/978-3-540-28650-9_5.
- [20] Iqbal H. Sarker. Deep Learning: A Comprehensive Overview on Techniques, Taxonomy, Applications and Research Directions. *SN Computer Science*, 2(6): 420, August 2021. ISSN 2661-8907. doi: 10.1007/s42979-021-00815-1. URL <https://doi.org/10.1007/s42979-021-00815-1>.
- [21] Xing Hao, Guigang Zhang, and Shang Ma. Deep learning. *International Journal of Semantic Computing*, 10(03):417–439, 2016. doi: 10.1142/S1793351X16500045. URL <https://doi.org/10.1142/S1793351X16500045>.
- [22] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521 (7553):436–444, May 2015. ISSN 1476-4687. doi: 10.1038/nature14539. URL <https://www.nature.com/articles/nature14539>. Publisher: Nature Publishing Group.
- [23] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117, 2015. ISSN 0893-6080. doi: <https://doi.org/10.1016/j.neunet.2014.09.003>. URL <https://www.sciencedirect.com/science/article/pii/S0893608014002135>.
- [24] George Strawn. Masterminds of the arpanet. *IT Professional*, 16(3):66–68, 2014. doi: 10.1109/MITP.2014.32.
- [25] Kenneth R. Castleman. *Digital image processing*. Prentice Hall Press, USA, 1996. ISBN 0132114674.
- [26] Robert M. Haralick and Linda G. Shapiro. *Computer and Robot Vision*. Addison-Wesley Longman Publishing Co., Inc., USA, 1st edition, 1992. ISBN 0201569434.
- [27] T S Huang. Computer Vision: Evolution and Promise.
- [28] Lessing Arturo Cárdenas Díaz and Rina Margarita Tirado Ballestas. Visión por computador /. <http://biblioteca.utb.edu.co/notas/tesis/0028381.pdf>, 2004. URL <https://repositorio.utb.edu.co/handle/20.500.12585/3166>. Accepted: 2019-10-18T19:00:08Z Artwork Medium: 1 Cd-Rom Interview Medium: 1 Cd-Rom Publisher: Universidad Tecnológica de Bolívar.

- [29] Shai Avidan, Gabriel Brostow, Moustapha Cissé, Giovanni Maria Farinella, and Tal Hassner, editors. *Computer Vision – ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part I*, volume 13661 of *Lecture Notes in Computer Science*. Springer Nature Switzerland, Cham, 2022. ISBN 978-3-031-19768-0 978-3-031-19769-7. doi: 10.1007/978-3-031-19769-7. URL <https://link.springer.com/10.1007/978-3-031-19769-7>.
- [30] Guangzheng Yang and Thomas S Huang. Human face detection in a complex background. *Pattern recognition*, 27(1):53–63, 1994.
- [31] Giuseppe Airò Farulla, Daniele Pianu, Marco Cempini, Mario Cortese, Ludovico O. Russo, Marco Indaco, Roberto Nerino, Antonio Chimienti, Calogero M. Oddo, and Nicola Vitiello. Vision-Based Pose Estimation for Robot-Mediated Hand Telerehabilitation. *Sensors*, 16(2):208, February 2016. ISSN 1424-8220. doi: 10.3390/s16020208. URL <https://www.mdpi.com/1424-8220/16/2/208>. Number: 2 Publisher: Multidisciplinary Digital Publishing Institute.
- [32] Kenny Chen, Paolo Gabriel, Abdulwahab Alasfour, Chenghao Gong, Werner K Doyle, Orrin Devinsky, Daniel Friedman, Patricia Dugan, Lucia Melloni, Thomas Thesen, et al. Patient-specific pose estimation in clinical environments. *IEEE journal of translational engineering in health and medicine*, 6:1–11, 2018.
- [33] Jihye Hwang, Sungheon Park, and Nojun Kwak. Athlete pose estimation by a global-local network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 58–65, 2017.
- [34] Steven Chen and Richard R. Yang. Pose trainer: Correcting exercise posture using pose estimation, 2020.
- [35] Ngoc-Hoang Nguyen, Tran-Dac-Thinh Phan, Guee-Sang Lee, Soo-Hyung Kim, and Hyung-Jeong Yang. Gesture recognition based on 3d human pose estimation and body part segmentation for rgb data input. *Applied Sciences*, 10(18):6188, 2020.
- [36] Štěpán Obdržálek, Gregorij Kurillo, Jay Han, Ted Abresch, and Ruzena Bajcsy. Real-time human pose detection and tracking for tele-rehabilitation in virtual reality. In *Medicine Meets Virtual Reality 19*, pages 320–324. IOS Press, 2012.
- [37] Chung-Yi Weng, Brian Curless, and Ira Kemelmacher-Shlizerman. Photo wake-up: 3d character animation from a single photo. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5908–5917, 2019.

- [38] Alberto Lamas, Siham Tabik, Antonio Cano Montes, Francisco Pérez-Hernández, Jorge García, Roberto Olmos, and Francisco Herrera. Human pose estimation for mitigating false negatives in weapon detection in video-surveillance. *Neuro-computing*, 489:488–503, 2022. ISSN 0925-2312. doi: <https://doi.org/10.1016/j.neucom.2021.12.059>. URL <https://www.sciencedirect.com/science/article/pii/S0925231221019159>.
- [39] Dimitris Tsiktsiris, Nikolaos Dimitriou, Antonios Lalas, Minas Dasygenis, Konstantinos Votis, and Dimitrios Tzovaras. Real-time abnormal event detection for enhanced security in autonomous shuttles mobility infrastructures. *Sensors*, 20(17):4943, 2020.
- [40] Tracy L Anderson and Max Donath. Animal behavior as a paradigm for developing robot autonomy. *Robotics and autonomous systems*, 6(1-2):145–168, 1990.
- [41] Daniel Joska, Liam Clark, Naoya Muramatsu, Ricardo Jericevich, Fred Nicolls, Alexander Mathis, Mackenzie W. Mathis, and Amir Patel. Acinuset: A 3d pose estimation dataset and baseline models for cheetahs in the wild. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, May 2021. doi: 10.1109/icra48506.2021.9561338. URL <http://dx.doi.org/10.1109/ICRA48506.2021.9561338>.
- [42] Silvia Zuffi, Angjoo Kanazawa, Tanya Berger-Wolf, and Michael J. Black. Three-d safari: Learning to estimate zebra pose, shape, and texture from images in the wild", 2019.
- [43] Silke Bauer and Marcel Klaassen. Mechanistic models of animal migration behaviour—their diversity, structure and use. *Journal of Animal Ecology*, 82(3): 498–508, 2013.
- [44] Benjamin Biggs, Oliver Boyne, James Charles, Andrew Fitzgibbon, and Roberto Cipolla. Who Left the Dogs Out? 3D Animal Reconstruction with Expectation Maximization in the Loop. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision – ECCV 2020*, pages 195–211, Cham, 2020. Springer International Publishing. ISBN 978-3-030-58621-8.
- [45] Sandeep Robert Datta, David J Anderson, Kristin Branson, Pietro Perona, and Andrew Leifer. Computational neuroethology: a call to action. *Neuron*, 104(1): 11–24, 2019.
- [46] Mackenzie Weygandt Mathis and Alexander Mathis. Deep learning tools for the measurement of animal behavior in neuroscience. *Current opinion in neurobiology*, 60:1–11, 2020.

- [47] Talmo D. Pereira, Joshua W. Shaevitz, and Mala Murthy. Quantifying behavior to understand the brain. *Nature Neuroscience*, 23(12):1537–1549, December 2020. ISSN 1546-1726. doi: 10.1038/s41593-020-00734-z.
- [48] Jinbao Wang, Shujie Tan, Xiantong Zhen, Shuo Xu, Feng Zheng, Zhenyu He, and Ling Shao. Deep 3D human pose estimation: A review. *Computer Vision and Image Understanding*, 210:103225, 2021. ISSN 1077-3142. doi: <https://doi.org/10.1016/j.cviu.2021.103225>. URL <https://www.sciencedirect.com/science/article/pii/S1077314221000692>.
- [49] Avelino Javier, Lidia Ripoll-Sánchez, and André E.X. Brown. Powerful and interpretable behavioural features for quantitative phenotyping of *Caenorhabditis elegans*, sep 2018. URL <https://royalsocietypublishing.org/doi/10.1098/rstb.2017.0375>. Publisher: Royal Society.
- [50] Avelino Javier, Michael Currie, Chee Wai Lee, Jim Hokanson, Kezhi Li, Céline N. Martineau, Eviatar Yemini, Laura J. Grundy, Chris Li, QueeLim Ch'ng, William R. Schafer, Ellen A. A. Nollen, Rex Kerr, and André E. X. Brown. An open-source platform for analyzing and sharing worm-behavior data. *Nature Methods*, 15(9): 645–646, September 2018. ISSN 1548-7105. doi: 10.1038/s41592-018-0112-1. URL <https://www.nature.com/articles/s41592-018-0112-1>. Publisher: Nature Publishing Group.
- [51] Alex Gomez-Marin, Joseph J. Paton, Adam R. Kampff, Rui M. Costa, and Zachary F. Mainen. Big behavioral data: psychology, ethology and the foundations of neuroscience. *Nature Neuroscience*, 17(11):1455–1462, November 2014. ISSN 1546-1726. doi: 10.1038/nn.3812. URL <https://www.nature.com/articles/nn.3812>. Publisher: Nature Publishing Group.
- [52] Tierpsy Tracker, August 2023. URL <https://github.com/Tierpsy/tierpsy-tracker>. original-date: 2019-12-16T16:45:59Z.
- [53] Miscellaneous Image Transformations — OpenCV 3.0.0-dev documentation. URL https://docs.opencv.org/3.0-beta/modules/imgproc/doc/miscellaneous_transformations.html.
- [54] Laetitia Hebert, Tosif Ahamed, Antonio C. Costa, Liam O'Shaughnessy, and Greg J. Stephens. WormPose: Image synthesis and convolutional networks for pose estimation in *C. elegans*. *PLOS Computational Biology*, 17(4), April 2021. ISSN 1553-7358. doi: 10.1371/journal.pcbi.1008914. URL <https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1008914>. Publisher: Public Library of Science.

- [55] Eviatar Yemini, Tadas Jucikas, Laura J. Grundy, André E. X. Brown, and William R. Schafer. A database of *Caenorhabditis elegans* behavioral phenotypes. *Nature Methods*, 10(9):877–879, September 2013. ISSN 1548-7105. doi: 10.1038/nmeth.2560. URL <https://www.nature.com/articles/nmeth.2560>. Publisher: Nature Publishing Group.
- [56] Elisha Odemakinde. Human Pose Estimation with Deep Learning - Ultimate Overview in 2024, December 2023. URL <https://viso.ai/deep-learning/pose-estimation-ultimate-overview/>.
- [57] Karl Weiss, Taghi M. Khoshgoftaar, and DingDing Wang. A survey of transfer learning. *Journal of Big Data*, 3(1):9, May 2016. ISSN 2196-1115. doi: 10.1186/s40537-016-0043-6. URL <https://doi.org/10.1186/s40537-016-0043-6>.
- [58] Lisa Torrey and Jude Shavlik. Transfer Learning. In *Handbook of Research on Machine Learning Applications and Trends: Algorithms, Methods, and Techniques*, pages 242–264. IGI Global, 2010. ISBN 978-1-60566-766-9. doi: 10.4018/978-1-60566-766-9.ch011. URL <https://www.igi-global.com/chapter/transfer-learning/www.igi-global.com/chapter/transfer-learning/36988>.
- [59] Ultralytics. Home, . URL <https://docs.ultralytics.com/>.
- [60] Kin-Yiu Wong. WongKinYiu/yolov7, June 2024. URL <https://github.com/WongKinYiu/yolov7>. original-date: 2022-07-06T15:14:06Z.
- [61] COCO - Common Objects in Context. URL <https://cocodataset.org/#home>.
- [62] Amazon EC2 P4d Instances – AWS. URL <https://aws.amazon.com/ec2/instance-types/p4/>.
- [63] Jinkun Cao, Hongyang Tang, Hao-Shu Fang, Xiaoyong Shen, Cewu Lu, and Yu-Wing Tai. Cross-domain adaptation for animal pose estimation, 2019.
- [64] Hang Yu, Yufei Xu, Jing Zhang, Wei Zhao, Ziyu Guan, and Dacheng Tao. Ap-10k: A benchmark for animal pose estimation in the wild, 2021.
- [65] Alexander Mathis, Pranav Mamidanna, Kevin M. Cury, Taiga Abe, Venkatesh N. Murthy, Mackenzie Weygandt Mathis, and Matthias Bethge. DeepLabCut: markerless pose estimation of user-defined body parts with deep learning. *Nature Neuroscience*, 21(9):1281–1289, September 2018. ISSN 1546-1726. doi: 10.1038/s41593-018-0209-y. URL <https://www.nature.com/articles/s41593-018-0209-y>. Publisher: Nature Publishing Group.

- [66] Jiteng Mu, Weichao Qiu, Gregory D. Hager, and Alan L. Yuille. Learning From Synthetic Animals. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12383–12392, Seattle, WA, USA, June 2020. IEEE. ISBN 978-1-72817-168-5. doi: 10.1109/CVPR42600.2020.01240. URL <https://ieeexplore.ieee.org/document/9157335/>.
- [67] Shuyuan Li, Jianguo Li, Hanlin Tang, Rui Qian, and Weiyao Lin. Atrw: A benchmark for amur tiger re-identification in the wild. In *Proceedings of the 28th ACM International Conference on Multimedia, MM '20*. ACM, October 2020. doi: 10.1145/3394171.3413569. URL <http://dx.doi.org/10.1145/3394171.3413569>.
- [68] Benjamin Biggs, Thomas Roddick, Andrew Fitzgibbon, and Roberto Cipolla. Creatures Great and SMAL: Recovering the Shape and Motion of Animals from Video. In C.V. Jawahar, Hongdong Li, Greg Mori, and Konrad Schindler, editors, *Computer Vision – ACCV 2018*, pages 3–19, Cham, 2019. Springer International Publishing. ISBN 978-3-030-20873-8. doi: 10.1007/978-3-030-20873-8_1.
- [69] 3D Models. URL <http://digitallife3d.org/3d-model>.
- [70] Luca Del Pero, Susanna Ricco, Rahul Sukthankar, and Vittorio Ferrari. Articulated motion discovery using pairs of trajectories, 2015.
- [71] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: Common Objects in Context. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision – ECCV 2014*, pages 740–755, Cham, 2014. Springer International Publishing. ISBN 978-3-319-10602-1. doi: 10.1007/978-3-319-10602-1_48.
- [72] Poselets, . URL <https://www2.eecs.berkeley.edu/Research/Projects/CS/vision/shape/poselets/>.
- [73] Aditya Khosla, Nityananda Jayadevaprakash, Bangpeng Yao, and Li Fei-Fei. Novel dataset for fine-grained image categorization : Stanford dogs. 2012. URL <https://api.semanticscholar.org/CorpusID:3181866>.
- [74] Alexander M. Bronstein, Michael M. Bronstein, and Ron Kimmel. Efficient computation of isometry[U+2010]invariant distances between surfaces. *SIAM Journal on Scientific Computing*, 28(5):1812–1836, 2006. doi: 10.1137/050639296. URL <https://doi.org/10.1137/050639296>.
- [75] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

- [76] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007. doi: 10.1109/MCSE.2007.55.
- [77] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020. doi: 10.1038/s41586-020-2649-2. URL <https://doi.org/10.1038/s41586-020-2649-2>.
- [78] Tim Bray, Paoli Jean, C.M Sperberg-McQueen, Eve Maler, and François Yergeau. Extensible Markup Language (XML) 1.0 (Fifth Edition), November 2008. URL <https://www.w3.org/TR/xml/>.
- [79] The Official YAML Web Site. URL <https://yaml.org/>.
- [80] ¿Qué es YAML? | IBM, April 2024. URL <https://www.ibm.com/mx-es/topics/yaml>.
- [81] Ultralytics. Pose Estimation Datasets Overview, . URL <https://docs.ultralytics.com/datasets/pose>.
- [82] Siyu Serena Ding, Linus J Schumacher, Avelino E Javier, Robert G Endres, and André EX Brown. Shared behavioral mechanisms underlie *C. elegans* aggregation and swarming. *eLife*, 8:e43318, April 2019. ISSN 2050-084X. doi: 10.7554/eLife.43318. URL <https://doi.org/10.7554/eLife.43318>. Publisher: eLife Sciences Publications, Ltd.
- [83] Alex P. How to Annotate Keypoints Using Roboflow, June 2023. URL <https://medium.com/@alexppppp/how-to-annotate-keypoints-using-roboflow-9bc2aa8915cd>.
- [84] Export annotations and data from CVAT, . URL <https://docs.cvat.ai/docs/manual/advanced/formats/>.
- [85] CVAT for image, . URL <https://opencv.github.io/docs/manual/advanced/formats/format-cvat/>. Section: docs.
- [86] cvat-ai/datumaro, April 2024. URL <https://github.com/cvat-ai/datumaro>. original-date: 2022-07-19T18:33:29Z.
- [87] Glenn Jocher. ultralytics/COCO2YOLO: Improvements, May 2019. URL <https://doi.org/10.5281/zenodo.2738323>.

- [88] Cherry Khosla and Baljit Singh Saini. Enhancing performance of deep learning models with different data augmentation techniques: A survey. In *2020 International Conference on Intelligent Engineering and Management (ICIEM)*, pages 79–85, 2020. doi: 10.1109/ICIEM48762.2020.9160048.
- [89] Martin Loesdau, Sébastien Chabrier, and Alban Gabillon. Hue and Saturation in the RGB Color Space. In Abderrahim Elmoataz, Olivier Lezoray, Fathallah Nouboud, and Driss Mammass, editors, *Image and Signal Processing*, pages 203–212, Cham, 2014. Springer International Publishing. ISBN 978-3-319-07998-1. doi: 10.1007/978-3-319-07998-1_23.
- [90] R Dorothy, RM Joany, R Joseph Rathish, S Santhana Prabha, Susai Rajendran, and ST Joseph. Image enhancement by histogram equalization. *International Journal of Nano Corrosion Science and Engineering*, 2(4):21–30, 2015.
- [91] xml.etree.ElementTree — La API XML de ElementTree. URL <https://docs.python.org/3/library/xml.etree.elementtree.html>.
- [92] Zuzana Reitermanova et al. Data splitting. In *WDS*, volume 10, pages 31–36. Matfyzpress Prague, 2010.
- [93] Alvaro de Castro. alvarodcastro/WormPoseTools, October 2023. URL <https://github.com/alvarodcastro/WormPoseTools>. original-date: 2023-10-20T10:37:36Z.
- [94] Docs Home - Comet Docs, . URL <https://www.comet.com/docs/v2/>.
- [95] Create & Run an Experiment - Comet Docs. URL <https://www.comet.com/docs/v2/guides/experiment-management/create-an-experiment/>.
- [96] Overview - Comet Docs. URL <https://www.comet.com/docs/v2/guides/comet-ui/experiment-management/visualizations/overview/>.
- [97] Mean Average Precision (mAP) Explained: Everything You Need to Know. URL <https://www.v7labs.com/blog/mean-average-precision>, <https://www.v7labs.com/blog/mean-average-precision>.
- [98] ultralytics/ultralytics/utils/metrics.py at main · ultralytics/ultralytics. URL <https://github.com/ultralytics/ultralytics/blob/main/ultralytics/utils/metrics.py>.
- [99] WormPose: Image synthesis and convolutional networks for pose estimation in *C. elegans*. *PLOS Computational Biology*, 17(4):e1008914, April 2021. ISSN 1553-7358. doi: 10.1371/journal.pcbi.1008914. URL <https://journals.plos>.

[org/ploscompbiol/article?id=10.1371/journal.pcbi.1008914](https://doi.org/10.1371/journal.pcbi.1008914). Publisher: Public Library of Science.

