eman ta zabal zazu
Universidad del País Vasco
Euskal Herriko Unibertsitatea
BILBOKO INGENIARITZA ESKOLA
ESCUELA DE INGENIERÍA DE BILBAO

GRADO EN INGENIERÍA INFORMÁTICA DE GESTIÓN Y SISTEMAS DE INFORMACIÓN

TRABAJO FIN DE GRADO

INTERPOLACIÓN DE FOTOGRAMAS CON DEEP LEARNING

Alumno: Angulo Celada, Andoni

Director: Labaka Intxauspe, Gorka

Curso: 2018-2019

Fecha: Bilbao, 20 de julio de 2019

Interpolación de fotogramas con *deep learning*

Julio de 2019

Andoni Angulo Celada

DIRIGIDO POR GORKA LABAKA INTXAUSPE

Resumen

Castellano

Se ha desarrollado un sistema de aprendizaje automático para el procesamiento de imágenes, aplicado a vídeos. Concretamente, su funcionalidad es la interpolación de imágenes, que consiste en generar un fotograma intermedio entre dos fotogramas contiguos. Aplicando este proceso a un vídeo entero, se duplican los fotogramas, consiguiendo un efecto de cámara lenta. Para desarrollar un sistema de este tipo, se necesita un conjunto de entrenamiento, el cual ha sido creado a partir de vídeos descargados de YouTube, cuyos fotogramas han sido extraídos y seleccionados en base a un criterio. Con estos datos, se ha entrenado el sistema, construido con una arquitectura basada en *deep learning* (o aprendizaje profundo); concretamente, redes neuronales convolucionales. Por último, se ha evaluado el sistema en un conjunto de datos público, comprobando la calidad de los resultados y la mejora respecto a un sistema base.

Palabras clave: aprendizaje automático, *deep learning*, redes neuronales convolucionales, procesamiento de imagen y vídeo.

Euskara

Proiektu honetan irudiak prozesatzeko ikasketa automatikoko sistema bat garatu da, bideoei aplikatuta. Hain zuzen ere, irudien interpolaziorako sistema bat sortu da, elkarren ondoan dauden bi fotogrametatik tarteako fotograma sortzen duena. Prozesu hau bideo osoa bati aplikatuz, fotogramak bikoiztu egiten dira, kamera geldoaren efektua lortuz. Mota honetako sistema bat garatzeko entrenamendu-multzo bat behar da, proiektu honetan YouTubetik deskargatutako bideoetatik sortu dugu gure entrenamendu-multzoa, horretarako bideoetako fotogramak irizpide baten arabera erauzi eta aukeratu direlarik. Datu horiekin *deep learning* (edo ikasketa sakona) arkitektura bateko sistema bat entrenatu da; neurona-sare konoluzionalak, hain zuzen ere. Azkenik, sistema hori datu-multzo publiko batean ebaluatu da, emaitzen kalitatea eta oinarri-lerro sistema batekiko hobekuntza egiaztatuz.

Gako-hitzak: ikasketa automatikoa, *deep learning*, neurona-sare konboluzionalak, irudi eta bideoen prozesamendua.

English

This project presents the development of a machine learning system for image processing, which is applied to videos. In particular, its functionality is image interpolation, which consists of generating an intermediate frame between two contiguous frames. Applying this process to an entire video, its frames are duplicated, getting a slow-motion effect. To develop this kind of system, a training dataset is needed, which has been created from YouTube videos, whose frames have been extracted and selected based on a criterion. The system has been trained on this data, and built with a deep learning-based architecture; specifically, convolutional neural networks. Finally, the system has been evaluated on a public dataset, verifying the quality of the results and the improvement with respect to a baseline system.

Keywords: machine learning, deep learning, convolutional neural networks, image and video processing.

Índice general

Índice de figuras

Índice de tablas

1. Introducción

En este capítulo se hará una descripción del proyecto y una explicación de las razones de su elección.

1.1 Descripción

Este proyecto presenta el desarrollo de un modelo de aprendizaje automático, concretamente, *deep learning*. El aprendizaje profundo o *deep learning* es un tipo de aprendizaje automático, actualmente en gran auge. La particularidad de estos algoritmos consiste en que aprenden sin necesidad de que un operador humano especifique formalmente su conocimiento, sino que lo acumulan en base a la práctica y construyen conceptos a partir de unos más simples. Es esta jerarquía de conceptos a la que hace alusión el adjetivo "profundo" [4, página 1].

La funcionalidad del modelo es la de interpolar fotogramas en un vídeo; esto es, generar un fotograma artificial entre cada uno de ellos. Para lograrlo, se usa una red neuronal, que toma un par de fotogramas y genera uno intermedio, intentando aproximar lo que se vería entre los dos instantes de tiempo. Con esto, se consigue un efecto de cámara lenta, ya que se obtienen el doble de fotogramas de los que tenía el vídeo inicialmente. Alternativamente, puede ser usado para aumentar la tasa de fotogramas de un vídeo, pasando, por ejemplo, de 30 FPS (fotogramas por segundo) a 60. De esta manera, la visualización del vídeo es más fluida.

1.2 Motivación

Entre las distintas posibilidades se ha decidido realizar un sistema de interpolación de fotogramas, porque en la vida de una persona hay muchos momentos memorables que grabar a cámara lenta, como la primera vez que un bebé anda, un paseo en bici, o un festival de música. Pero, aunque existen teléfonos capaces de grabar a 240 FPS, no todo el mundo tiene acceso a ellos; y además, para grabar a mayores frecuencias se necesitan cámaras profesionales. Por esta razón, es muy útil poder generar vídeos a cámara lenta a partir de vídeos grabados. [10]

A parte de esto, la gran capacidad del *deep learning* para resolver problemas en imágenes y vídeos [20] ha sido determinante para elegir este trabajo, ya que esta tecnología es muy prometedora. Tanto es así, que la inteligencia artificial será una de las causantes de la cuarta revolución industrial [31]. Por ello, tengo interés en aprender a desarrollar sistemas de este tipo, y quizás, continuar una carrera profesional en este ámbito.

2. Planteamiento inicial

En este capítulo se presentará el planteamiento inicial del proyecto, que comprende los objetivos, el alcance, la planificación temporal, las herramientas, la gestión de riesgos y la evaluación económica. Estos apartados, pero en especial, la planificación temporal (y por tanto, la evaluación económica), son susceptibles de sufrir modificaciones a lo largo de la ejecución del proyecto, debido a contratiempos en esta o a fallos en la planificación.

2.1 Objetivos

El objetivo principal de este trabajo es el desarrollo de un modelo de *deep learning* para aplicar a vídeos un efecto de cámara lenta. Concretamente, se pretende que el sistema sea capaz de interpolar fotogramas intermedios por cada par de estos. A continuación, se exponen desglosados los objetivos del proyecto.

Personales

- **Aprender más sobre redes neuronales e inteligencia artificial:** profundizar en el aprendizaje de redes neuronales y el procesamiento de imágenes con *deep learning*, ya que durante la carrera solamente se han visto las bases de estos.
- **Ganar experiencia en el desarrollo de proyectos de investigación:** al tratarse de un proyecto de investigación en el área de la inteligencia artificial, se busca adquirir agilidad siguiendo la metodología empleada en este tipo de proyectos.

De *software*

- **Conseguir un modelo con una calidad aceptable:** consiste en que la versión final de la aplicación sea capaz de producir buenos resultados, para su posterior uso personal. Este objetivo comprende tanto la calidad del conjunto de entrenamiento como la del modelo y su entrenamiento. El cumplimiento de este objetivo se verificará mediante las pruebas realizadas en el capítulo de Evaluación (7).
- **Escribir buen código:** significa que el código implementado sea entendible para otras personas, fácilmente reutilizable, robusto frente a fallos, flexible y modular. Estas características

se comprobarán al término del proyecto, y el cumplimiento del objetivo se reflejará en el capítulo de Conclusiones (8).

Académicos

- **Entregar el proyecto en las convocatorias de junio o julio:** es imprescindible haber finalizado el proyecto y entregarlo, como máximo, en la convocatoria de julio, para poder cursar un máster el siguiente curso. Se verificará su cumplimiento el 19 de julio de 2019, fecha límite para iniciar los trámites de entrega.
- **Obtener una nota mínima de "Notable":** conseguir una buena nota en el trabajo, para utilizarlo como carta de presentación en el mundo laboral y para mejorar la nota media del expediente académico.

2.2 Alcance

En esta sección se expondrán las tareas necesarias para conseguir los objetivos mencionados anteriormente. En la figura 2.1 se presenta el diagrama EDT (Estructura de Descomposición del Trabajo), que contiene todas las tareas, a su vez divididas en subtareas. En las tablas subsiguientes se describe cada una, junto con su duración estimada.

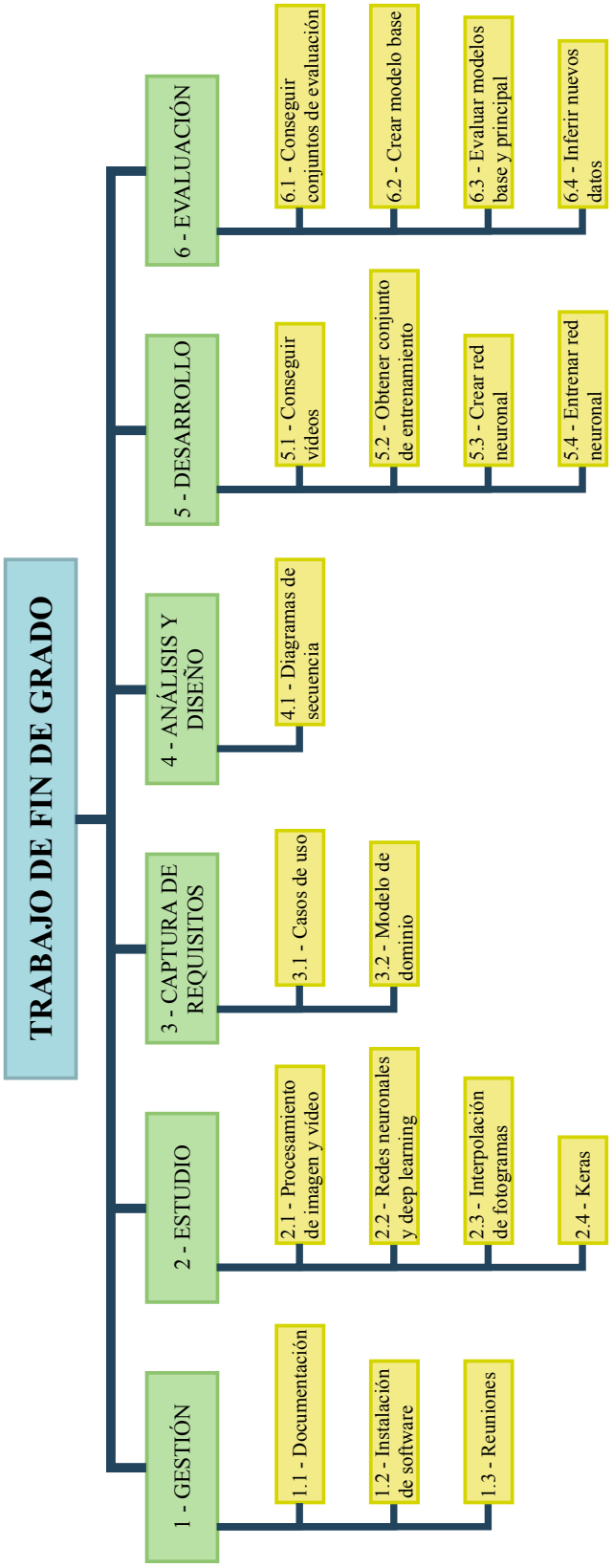


Figura 2.1: Diagrama EDT del proyecto.

1 - GESTIÓN
1.1 - Documentación
Duración 100 horas. **Descripción** Redacción completa de la memoria del proyecto. **Entradas** Conocimiento de todas las áreas del trabajo. **Salidas** Memoria del proyecto. **Recursos necesarios** Editor de textos LaTeXy de diagramas. **Precedencias** Ninguna.

Tabla 2.1: Descripción de la tarea “Documentación”.

1 - GESTIÓN
1.2 - Instalación de *software*
Duración 5 horas. **Descripción** Instalación y puesta en marcha de las librerías de código y controladores necesarios para implementar el sistema. **Entradas** Ninguna. **Salidas** Librerías de código y controladores instalados en el ordenador y listos para su funcionamiento. **Recursos necesarios** Ordenador con conexión a internet y guías de instalación. **Precedencias** Ninguna.

Tabla 2.2: Descripción de la tarea “Instalación de *software*”.

1 - GESTIÓN
1.3 - Reuniones
Duración 25 horas. **Descripción** Reuniones bisemanales con el tutor del proyecto, para guiar y hacer un seguimiento del proyecto. **Entradas** Progreso del proyecto respecto a la anterior reunión. **Salidas** Orientación y dudas resueltas. **Recursos necesarios** Lugar de reunión y correo electrónico. **Precedencias** Ninguna.

Tabla 2.3: Descripción de la tarea “Reuniones”.

2 - ESTUDIO
2.1 - Procesamiento de imagen y vídeo
Duración 25 horas. **Descripción** Formación en el procesamiento de imágenes y vídeos, con y sin aprendizaje automático. **Entradas** Libros, artículos científicos y tutoriales. **Salidas** Conocimientos sobre el procesamiento de imagen y vídeo. **Recursos necesarios** Ordenador con acceso a internet y cliente VPN para acceder a artículos de pago con la licencia de la UPV/EHU. **Precedencias** Ninguna.

Tabla 2.4: Descripción de la tarea “Procesamiento de imagen y vídeo”.

2 - ESTUDIO
2.2 - Redes neuronales y *deep learning*
Duración 50 horas. **Descripción** Formación en el funcionamiento de redes neuronales profundas. **Entradas** Libros, artículos científicos y tutoriales. **Salidas** Conocimientos básicos sobre *deep learning*. **Recursos necesarios** Ordenador con acceso a internet y cliente VPN para acceder a artículos de pago con la licencia de la UPV/EHU. **Precedencias** Ninguna.

Tabla 2.5: Descripción de la tarea "Redes neuronales y *deep learning*".

2 - ESTUDIO
2.3 - Interpolación de fotogramas
Duración 25 horas. **Descripción** Formación en el estado del arte de la interpolación de fotogramas con *deep learning*. **Entradas** Libros y artículos científicos. **Salidas** Conocimientos sobre interpolación de fotogramas con *deep learning*. **Recursos necesarios** Ordenador con acceso a internet y cliente VPN para acceder a artículos de pago con la licencia de la UPV/EHU. **Precedencias** Tarea 2.2, "Redes neuronales y *deep learning*".

Tabla 2.6: Descripción de la tarea "Interpolación de fotogramas".

2 - ESTUDIO
2.4 - Keras
Duración 10 horas. **Descripción** Formación en el funcionamiento básico de la librería de *software* para *deep learning* “Keras”. **Entradas** Documentación extraída de la página web de “Keras” [24]. **Salidas** Conocimientos sobre el manejo de la librería “Keras”. **Recursos necesarios** Ordenador con acceso a internet y tarjeta gráfica. **Precedencias** Tarea 2.2, “Redes neuronales y *deep learning*”.

Tabla 2.7: Descripción de la tarea “Keras”.

3 - CAPTURA DE REQUISITOS
3.1 - Casos de uso
Duración 2 horas. **Descripción** Desarrollo del modelo de casos de uso, diagrama que muestra las relaciones existentes entre el sistema y los distintos roles bajo los que se puede usar, denominados “casos de uso”. **Entradas** Información básica sobre el alcance del proyecto. **Salidas** Diagrama del modelo de casos de uso. **Recursos necesarios** Editor de diagramas. **Precedencias** Ninguna.

Tabla 2.8: Descripción de la tarea “Casos de uso”.

3 - CAPTURA DE REQUISITOS
3.2 - Modelo de dominio
Duración 2 horas. **Descripción** Creación del modelo de dominio del proyecto, diagrama que contiene objetos y datos del sistema y sus relaciones. **Entradas** Diagrama de casos de uso, así como información sobre el alcance del proyecto. **Salidas** Diagrama del modelo de dominio. **Recursos necesarios** Editor de diagramas. **Precedencias** Tarea 3.1, "Casos de uso".

Tabla 2.9: Descripción de la tarea "Modelo de dominio".

4 - ANÁLISIS Y DISEÑO
4.1 - Diagramas de secuencia
Duración 7 horas. **Descripción** Creación de los diagramas de secuencia, que indican el flujo de eventos entre los objetos del sistema. **Entradas** Diagrama de casos de uso, además de información sobre las funcionalidades del sistema. **Salidas** Diagramas de secuencia. **Recursos necesarios** Editor de diagramas. **Precedencias** Tareas 3.1, "Casos de uso" y 3.2, "Modelo de dominio".

Tabla 2.10: Descripción de la tarea "Diagramas de secuencia".

5 - DESARROLLO
5.1 - Conseguir vídeos
Duración 5 horas. **Descripción** Descarga y filtrado de los vídeos necesarios de donde se extraerán los fotogramas. **Entradas** Información sobre conjuntos de entrenamiento de proyectos similares. **Salidas** Conjunto de vídeos. **Recursos necesarios** Entorno de desarrollo *Python* y librería de código para descargar vídeos. **Precedencias** Ninguna.

Tabla 2.11: Descripción de la tarea “Conseguir vídeos”.

5 - DESARROLLO
5.2 - Obtener conjunto de entrenamiento
Duración 100 horas. **Descripción** Extracción, filtrado y tratamiento de los fotogramas, que formarán el conjunto de entrenamiento. **Entradas** Conjunto de vídeos de los que se extraerán los fotogramas y conocimientos sobre el procesamiento de imagen y vídeo. **Salidas** Conjunto de entrenamiento final. **Recursos necesarios** Entorno de desarrollo *Python* y librerías para el procesamiento de imágenes y vídeos. **Precedencias** Tareas 2.1, “Procesamiento de imagen y vídeo”; 5.1, “Conseguir vídeos” y 4.1, “Diagramas de secuencia”.

Tabla 2.12: Descripción de la tarea “Obtener conjunto de entrenamiento”.

5 - DESARROLLO
5.3 - Crear red neuronal
Duración 80 horas. **Descripción** Creación de la arquitectura de la red neuronal. **Entradas** Conocimientos sobre *deep learning*, interpolación de fotogramas y el manejo de la librería "Keras". **Salidas** Arquitectura de la red neuronal programada en código. **Recursos necesarios** Entorno de desarrollo *Python*, librería "Keras" y servidor personal de *Dropbox*. **Precedencias** Tareas 2.2, "Redes neuronales y *deep learning*"; 2.3, "Interpolación de fotogramas"; 2.4, "Keras" y 4.1, "Diagramas de secuencia".

Tabla 2.13: Descripción de la tarea "Crear red neuronal".

5 - DESARROLLO
5.4 - Entrenar red neuronal
Duración 40 horas. **Descripción** Creación del entrenamiento de la red neuronal y su ejecución. **Entradas** Arquitectura de la red neuronal programada en código. **Salidas** Entrenamientos de la red neuronal realizados. **Recursos necesarios** Entorno de desarrollo *Python*, librería "Keras", tarjeta gráfica y servidor personal de *Dropbox*. **Precedencias** Tarea 5.3, "Crear red neuronal" y 4.1, "Diagramas de secuencia".

Tabla 2.14: Descripción de la tarea "Entrenar red neuronal".

6 - EVALUACIÓN
6.1 - Conseguir conjuntos de evaluación
Duración 2 horas. **Descripción** Descarga de varios conjuntos con los que evaluar el sistema. **Entradas** Información sobre conjuntos de evaluación de proyectos similares. **Salidas** Conjuntos de vídeos. **Recursos necesarios** Ordenador con acceso a internet. **Precedencias** Ninguna.

Tabla 2.15: Descripción de la tarea “Conseguir conjuntos de evaluación”.

6 - EVALUACIÓN
6.2 - Crear modelo base
Duración 5 horas. **Descripción** Creación del modelo base con el que se comparará el modelo principal. **Entradas** Información sobre interpolación de fotogramas. **Salidas** Modelo base. **Recursos necesarios** Entorno de desarrollo *Python*, librerías para el procesamiento de imágenes y vídeos y servidor personal de *Dropbox*. **Precedencias** Tarea 2.1, “Procesamiento de imagen y vídeo”.

Tabla 2.16: Descripción de la tarea “Crear modelo base”.

6 - EVALUACIÓN
6.3 - Evaluar modelos base y principal
Duración 10 horas. **Descripción** Evaluar tanto el modelo base como el principal, con los conjuntos de evaluación obtenidos previamente. **Entradas** Modelos base y principal y conjuntos de evaluación. **Salidas** Resultados de las evaluaciones. **Recursos necesarios** Entorno de desarrollo *Python*, librería "Keras", librerías para el procesamiento de imágenes y vídeos y servidor personal de *Dropbox*. **Precedencias** Tareas 5.4, "Entrenar red neuronal"; 6.1, "Conseguir conjuntos de evaluación"; 6.2, "Crear modelo base" y 4.1, "Diagramas de secuencia".

Tabla 2.17: Descripción de la tarea "Evaluar modelo base y principal".

6 - EVALUACIÓN
6.4 - Inferir nuevos datos
Duración 2 horas. **Descripción** Inferir nuevos fotogramas en vídeos de ejemplo. **Entradas** Modelo principal y vídeos de evaluación. **Salidas** Nuevos vídeos con fotogramas interpolados. **Recursos necesarios** Entorno de desarrollo *Python*, librería "Keras", librerías para el procesamiento de imágenes y vídeos y servidor personal de *Dropbox*. **Precedencias** Tareas 5.4, "Entrenar red neuronal"; 6.1, "Conseguir conjuntos de evaluación" y 4.1, "Diagramas de secuencia".

Tabla 2.18: Descripción de la tarea "Inferir nuevos datos".

Tarea	Subtarea	Duración (horas)
GESTIÓN	Documentación	100
	Instalación de *software*	5
	Reuniones	25
ESTUDIO	Procesamiento de imagen y vídeo	25
	Redes neuronales y *deep learning*	50
	Interpolación de fotogramas	25
	Keras	10
CAPTURA DE REQUISITOS	Casos de uso	2
	Modelo de dominio	2
ANÁLISIS Y DISEÑO	Diagramas de secuencia	7
DESARROLLO	Conseguir vídeos	5
	Obtener conjunto de entrenamiento	100
	Crear red neuronal	80
	Entrenar red neuronal	40
EVALUACIÓN	Conseguir conjuntos de evaluación	2
	Crear modelo base	5
	Evaluar modelos base y principal	10
	Inferir nuevos datos	2
TOTAL		495

Tabla 2.19: Duración total de las tareas.

2.3 Planificación temporal

Después de identificar cada tarea y su duración hay que distribuirlas en el tiempo, mediante una planificación temporal. Esta se ha realizado mediante un diagrama de *Gantt* (Figura 2.2), que, mediante rectángulos, indica el tiempo de ejecución de cada tarea.

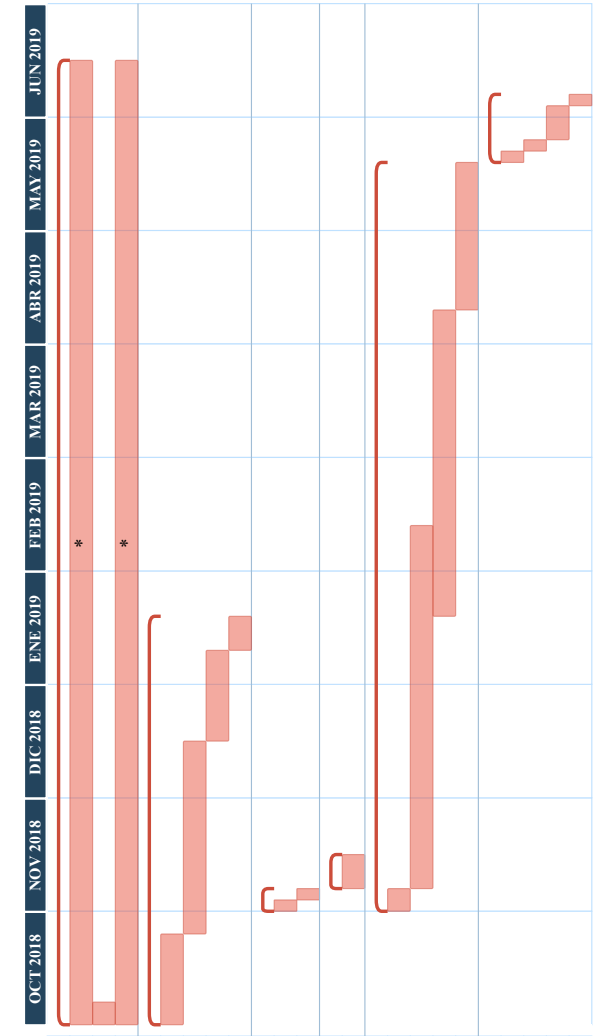
2.4 Herramientas

A continuación, se listan las herramientas necesarias para la realización del proyecto:

- **Ordenador personal:** ordenador del alumno, donde se llevan a cabo las tareas. Dispone de conexión a internet y una tarjeta gráfica suficientemente potente para entrenar la red neuronal del proyecto.
- ***Python*:** lenguaje de programación utilizado en el proyecto [28].
- ***PyCharm*:** entorno de desarrollo *Python*, donde se programa todo el código del sistema [36].
- ***Dropbox*:** servidor de archivos en la nube, donde se aloja el código [27].
- ***Overleaf*:** editor de textos LaTeX, mediante el cual se redacta el proyecto [34].
- ***Draw.io*:** editor de diagramas versátil y fácil de usar [26].
- ***Visual Paradigm*:** editor de diagramas para *Windows* [38].
- ***Online Charts*:** herramienta de diseño de gráficos para la representación de datos [32].
- ***Cisco AnyConnect*:** cliente VPN, usado para acceder a artículos de pago con la licencia de la UPV/EHU [25].
- ***Keras*:** librería de código *Python* para el desarrollo de modelos de *deep learning* [24].

2.5 Gestión de riesgos

En esta sección se describirán los riesgos asociados a la ejecución del proyecto. Por cada uno, además de su descripción, se especifica su prevención, plan de contingencia e impacto. Este último se calcula multiplicando la magnitud del riesgo (tiempo que se retrasaría el proyecto si ocurriera dicho riesgo) por la probabilidad de que ocurra.



TAREA	COMIENZO	DURACIÓN (horas)	FIN
1 GESTIÓN	**1 OCT 2018**	**130**	**16 JUN 2019**
1.1 Documentación	1 OCT	100	16 JUN
1.2 Instalación de software	1 OCT	5	5 OCT
1.3 Reuniones	1 OCT	25	16 JUN
2 ESTUDIO	**1 OCT 2018**	**110**	**18 ENE 2019**
2.1 Procesamiento de imagen y vídeo	1 OCT	25	25 OCT
2.2 Redes neuronales y deep learning	26 OCT	50	14 DIC
2.3 Interpolación de fotogramas	15 DIC	25	8 ENE
2.4 Keras	9 ENE	10	18 ENE
3 CAPTURA DE REQUISITOS	**1 NOV 2018**	**4**	**4 NOV 2018**
3.1 Casos de uso	1 NOV	2	2 NOV
3.2 Modelo de dominio	3 NOV	2	4 NOV
4 ANÁLISIS Y DISEÑO	**5 NOV 2018**	**7**	**11 NOV 2018**
4.1 Diagramas de secuencia	5 NOV	7	11 NOV
5 DESARROLLO	**1 NOV 2018**	**225**	**18 MAY 2019**
5.1 Conseguir vídeos	1 NOV	5	5 NOV
5.2 Obtener conjunto de entrenamiento	6 NOV	100	13 FEB
5.3 Crear red neuronal	19 ENE	80	8 ABR
5.4 Entrenar red neuronal	9 ABR	40	18 MAY
6 EVALUACIÓN	**19 MAY 2019**	**19**	**6 JUN 2019**
6.1 Conseguir conjuntos de evaluación	19 MAY	2	20 MAY
6.2 Crear modelo base	21 MAY	5	25 MAY
6.3 Evaluar modelos base y principal	26 MAY	10	4 JUN
6.4 Inferir nuevos datos	5 JUN	2	6 JUN

* La tarea se realiza de manera intermitente a lo largo de su periodo de ejecución.

Figura 2.2: Diagrama de *Gantt* del proyecto.

1 - PÉRDIDA DE LA DOCUMENTACIÓN
Descripción La documentación está alojada en el servidor de *Overleaf*, pero puede que, debido a un error, este deje de funcionar y el documento desaparezca. **Prevención** Realizar copias de seguridad en el dispositivo local o disco externo regularmente. **Plan de contingencia** Restaurar la última copia de seguridad guardada. **Probabilidad** Baja. **Impacto** Alto (más de una semana).

Tabla 2.20: Descripción del riesgo “Pérdida de la documentación”.

2 - PÉRDIDA DEL CÓDIGO
Descripción El código del sistema está alojado en el servidor personal de *Dropbox*, por lo que puede que deje de funcionar y el código desaparezca. **Prevención** Realizar copias de seguridad en el dispositivo local o disco externo regularmente. **Plan de contingencia** Restaurar la última copia de seguridad guardada. **Probabilidad** Baja. **Impacto** Alto (más de una semana).

Tabla 2.21: Descripción del riesgo “Pérdida del código”.

3 - DAÑOS EN EL ORDENADOR PERSONAL
Descripción El ordenador personal puede sufrir daños que le incapaciten para funcionar temporal o permanentemente. Es especialmente importante porque contiene la tarjeta gráfica donde se entrena el modelo. **Prevención** Tener precaución al utilizar y manipular en ordenador. **Plan de contingencia** Sustituir el ordenador principal por uno portátil hasta que esté operativo. **Probabilidad** Baja. **Impacto** Moderado (menos de una semana).

Tabla 2.22: Descripción del riesgo “Daños en el ordenador personal”.

4 - FALLO EN LA CONEXIÓN A INTERNET
Descripción La conexión a internet por cable puede verse interrumpida temporalmente, impidiendo la realización de tareas que lo requieren. **Prevención** Revisar el rúter periódicamente y disponer de un adaptador wifi como alternativa. **Plan de contingencia** Utilizar la señal wifi hasta que la conexión por cable se haya restablecido. **Probabilidad** Baja. **Impacto** Muy bajo (menos de un día).

Tabla 2.23: Descripción del riesgo “Fallo en la conexión a internet”.

5 - TARJETA GRÁFICA NO SUFICIENTEMENTE POTENTE
Descripción Dado el gran número de cómputos que se deben hacer para entrenar una red neuronal profunda, la unidad de procesamiento de gráficos debe ser lo suficientemente potente para poderlos llevar a cabo. Debido a las limitaciones en cuanto potencia computacional, puede que la tarjeta gráfica del ordenador personal sea incapaz de realizar el entrenamiento en un tiempo prudencial. **Prevención** Disponer de servicios en la nube donde realizar el entrenamiento, tales como *Amazon Web Services* [23] o *Microsoft Azure* [30]. **Plan de contingencia** Poner en marcha los servicios de entrenamiento en la nube. **Probabilidad** Baja. **Impacto** Bajo (más de un día).

Tabla 2.24: Descripción del riesgo “Tarjeta gráfica no suficientemente potente”.

6 - INCUMPLIMIENTO DE LA PLANIFICACIÓN
Descripción Al tratarse del primer proyecto de investigación que se realiza, y, debido a contratiempos en la ejecución del proyecto o a fallos en la planificación temporal, es posible que algunas tareas no se puedan completar a tiempo, retrasando las sucesivas. **Prevención** Elaborar la planificación temporal con la suficiente holgura para que los imprevistos no supongan un retraso demasiado grande. **Plan de contingencia** Dedicar más horas de trabajo diarias para contrarrestar el atraso. **Probabilidad** Alta. **Impacto** Muy alto (más de dos semanas).

Tabla 2.25: Descripción del riesgo “Incumplimiento de la planificación”.

7 - BAJA PERSONAL
Descripción El alumno puede verse incapacitado para continuar con el desarrollo del proyecto, ya sea por enfermedad, accidente u otras razones. **Prevención** Llevar hábitos de vida saludables. **Plan de contingencia** Al ser este riesgo posible causa del anterior (6 - “Incumplimiento de la planificación”), el plan de contingencia es el mismo. **Probabilidad** Baja. **Impacto** Moderado (menos de una semana).

Tabla 2.26: Descripción del riesgo “Baja personal”.

Como plan de seguimiento de los riesgos, se realizarán seguimientos bisemanales de los riesgos moderados, altos y muy altos: 1- “Pérdida de la documentación”, 2- “Pérdida del código”, 3- “Daños en el ordenador personal”, 6- “Incumplimiento de la planificación” y 7- “Baja personal”. Asimismo, si durante el desarrollo del proyecto el plan de prevención o de contingencia de algún riesgo se modifica, o un nuevo riesgo se añade o elimina, se reflejará en el plan de riesgos.

2.6 Evaluación económica

En esta sección se expondrá la evaluación económica del proyecto, compuesta por los costes directos e indirectos.

2.6.1 Costes directos

Los costes directos son costes que se pueden asignar al proyecto de forma directa y clara.

Horas trabajadas

Para calcular los costes por horas trabajadas, se ha tomado el número total de horas estimado para la realización del proyecto, calculado en la sección 2.2 (495 horas) y el coste por cada una: 12€/hora. Este número ha sido calculado mediante 3 datos: el salario mínimo bruto anual de un investigador predoctoral, 16 422€ [37]; la cotización a la seguridad social de la empresa por cada trabajador, alrededor de un 30 % del salario; y la jornada máxima anual en ingeniería para 2019, 1780 horas [33]. Por lo tanto, el coste por horas trabajadas es 5 940€.

Viajes

Las reuniones con el tutor del proyecto tienen lugar en la Escuela de Ingeniería de Bilbao, a donde se llega en tren. El coste el billete de ida y vuelta (con descuento de familia numerosa) es 2.26€. Según la EDT del proyecto (Figura 2.1), este tendrá una duración de 8 meses, y las reuniones serán bisemanales. Por lo tanto, el coste por viajes es 36€.

2.6.2 Costes indirectos

Los costes indirectos que se tienen en cuenta para este proyecto son el equipamiento informático, las licencias *software* y los servicios del puesto de trabajo.

Respecto al equipamiento informático, el proyecto se realiza en el ordenador personal, que tuvo un coste aproximado de 1 000€. Si su vida útil estimada es de 6 años, se obtiene un gasto de 10€ imputable al proyecto.

En cuanto a las licencias *software*, la mayoría de los programas utilizados son gratuitos, exceptuando *Visual Paradigm*, cuya licencia mensual (en su versión estándar) cuesta 17€. Por lo tanto, para este proyecto tiene un coste de 136€ (17×8).

Para calcular el coste de los servicios del puesto de trabajo, tales como la electricidad, el agua o la conexión a internet, se ha tomado un 5 % de los costes directos, resultando en 299€.

2.6.3 Coste total

Finalmente, el coste total es la suma de los gastos directos e indirectos, como se muestra en la tabla 2.27.

Concepto	Coste
Horas trabajadas	5 940€
Viajes	36€
Equipamiento informático	10€
Licencias *software*	136€
Servicios del puesto de trabajo	299€
Total	6 421€

Tabla 2.27: Coste total del proyecto.

3. Antecedentes

3.1 Marco teórico

Dado que este proyecto hace uso del *deep learning* mediante redes neuronales, en esta sección se explicará qué es el aprendizaje automático y una red neuronal.

3.1.1 ¿Qué es el aprendizaje automático?

Para dar respuesta a esta pregunta se expondrá un ejemplo: *«Para resolver un problema en un ordenador se necesita un algoritmo. Un algoritmo es una secuencia de instrucciones que se deben llevar a cabo para transformar una entrada en una salida. Por ejemplo, se puede crear un algoritmo de ordenamiento, donde la entrada es un conjunto de números y la salida es la lista ordenada.* [...] *Sin embargo, para algunas tareas no se dispone de un algoritmo; por ejemplo, para distinguir un correo no deseado de uno legítimo. Se sabe que la entrada es una cadena de caracteres y la salida un sí/no, indicando si es un correo basura. En este caso, es difícil saber cómo transformar la entrada en una salida.* [...] *Pero lo que falta en conocimiento se compensa con datos, ya que se pueden recopilar fácilmente miles de mensajes, algunos de los cuales se sabe que son no deseados. Lo que se pretende es "aprender" lo que hace que un correo sea no deseado. En otras palabras, se quiere que el ordenador extraiga automáticamente el algoritmo para esta labor.»* [1, página 1]

En resumidas cuentas, *«el aprendizaje automático es programar un ordenador para que optimice un criterio de rendimiento, usando datos de entrenamiento o la propia experiencia. Teniendo un modelo definido por unos parámetros, este aprende mediante la ejecución de un programa que los optimice.»* [1, página 3]

Existen varios tipos de aprendizaje automático, como el *supervisado*, *no supervisado* y *reforzado*, siendo el primero el tipo de aprendizaje del problema de este proyecto. El aprendizaje supervisado comprende problemas en los que los datos de entrenamiento están formados por un vector de entrada y su correspondiente vector objetivo. Por ejemplo, en el reconocimiento de dígitos, la entrada es un vector de píxeles y la salida un número del 0 al 9. Si, como en este caso, la salida es una categoría discreta, estos problemas se llaman de *clasificación*. Por el contrario, si la salida es una variable continua, son problemas de *regresión*. [2, página 3] La tarea de este proyecto es un

problema de regresión, puesto que la salida debe ser una imagen; es decir, un vector de píxeles en el intervalo [0, 255]. A pesar de que los píxeles son valores discretos, esta tarea no se considera de clasificación, ya que el sistema produce valores continuos que posteriormente se redondean para formar la imagen.

En aprendizaje automático hay varios algoritmos de regresión, como la regresión polinomial [1, página 75], los árboles de regresión [1, página 192] o las redes neuronales (o perceptrones multicapa) [1, página 233]. Se ha escogido este último para resolver el problema de interpolación de fotogramas, por su capacidad y facilidad para modelar problemas muy complejos [1, capítulo 9].

3.1.2 ¿Qué es una red neuronal?

El objetivo de este apartado es que el lector adquiera el conocimiento suficiente para entender la segunda parte del proyecto; esto es, el entrenamiento de la red neuronal. Por esta razón, se explicará, de manera superficial, qué es una red neuronal y cómo entrenarla.

Las redes neuronales, como se puede suponer por su nombre, toman al cerebro como fuente de inspiración. Están formadas por conjuntos de unidades llamadas neuronas, organizándose estas en capas. Una neurona es una función matemática que recibe una o varias entradas y produce una salida, casi siempre mediante dos fases: primero, calcula la suma ponderada de las entradas, ya que cada una tiene un peso asociado a cada neurona con la que está conectada. Y después, este valor pasa por una función no lineal, llamada "de activación". Además, cada neurona tiene conectada una entrada igual a 1, con su correspondiente peso, llamado sesgo o *bias*. Varias neuronas apiladas forman una red neuronal, que tiene un mínimo de dos capas de profundidad: la "de entrada" y la "de salida". Si existen capas intermedias, estas se denominan "ocultas". Una red neuronal puede aproximar funciones matemáticas más complejas que una sola neurona, y, si es lo suficientemente profunda, casi cualquier función. [1, capítulo 11] [4, capítulo 6]

Una red neuronal aprende a través del ajuste de los pesos de sus capas; generalmente, utilizando los algoritmos del descenso estocástico del gradiente y retropropagación del error. En el entrenamiento, cuando la red produce una salida, esta se compara con la salida real y se obtiene el error cometido. Este error se utiliza para ajustar los pesos de la red, propagando el gradiente de actualización, desde las últimas capas hacia las primeras. [4, capítulos 4.3, 6.5]

Hay muchos tipos de redes neuronales, como las *convolucionales*, recursivas y recurrentes. La que se va a utilizar en este proyecto es una red neuronal convolucional [12]. Este tipo de redes están diseñadas para procesar datos que tienen una topología de cuadrícula, como, por ejemplo, imágenes. Su funcionamiento se basa, a grandes rasgos, en que en cada capa se va recorriendo los datos, desplazando por toda la imagen un núcleo de convolución o *kernel* (una pequeña matriz que se aprende durante el entrenamiento), con el que se realiza la operación de convolución. Al utilizar la misma matriz de valores para toda la imagen, se consigue reducir enormemente el número de parámetros, haciendo a estas redes muy eficientes. [4, capítulo 9] Pese a que se crearon hace 30 años, es en la última década cuando han adquirido más relevancia, debido a la gran cantidad de datos y poder computacional disponible.

3.2 Interpolación de fotogramas

A pesar de que existen técnicas de interpolación de fotogramas sin utilizar aprendizaje automático (como las que ofrece el programa *After Effects* de *Adobe* [22]), en esta sección se limitará a explicar las que sí hacen uso de él.

El éxito que ha tenido el *deep learning* en el campo de la visión artificial [20] ha provocado que haya mayor interés en aplicarlo a esta disciplina. Por ejemplo, Long *et al.* [14] utilizaron una red neuronal convolucional capaz de interpolar fotogramas para resolver un problema de emparejamiento de imágenes.

En 2009 Mahajan *et al.* [16] crearon un método para la interpolación de imágenes. Este método se basa en la idea de que cada píxel traza un camino desde un fotograma hasta el siguiente. Su sistema mueve y copia los gradientes de estos píxeles a lo largo de este camino. Además, implementan una mejora significativa respecto a métodos anteriores, que consiste en utilizar puntos de transición arbitrarios (asimétricos), donde el camino cambia de una imagen a otra, con lo que se reduce el *ghosting* (o imagen fantasma)[1] y la borrosidad y se mantiene la coherencia temporal.

Años más tarde, en 2017, Niklaus *et al.* [18] desarrollaron un método para la síntesis de los píxeles del fotograma interpolado, basado en una operación de convolución sobre dos fotogramas de entrada. Por cada píxel del fotograma a interpolar, una red neuronal convolucional genera una imagen (núcleo de convolución) a partir de segmentos de las dos imágenes de origen. Esta se convoluciona con otros dos segmentos de cada imagen de entrada y se produce un píxel. A pesar de que se obtienen resultados de alta calidad, tanto el coste computacional como el espacial son elevados, debido al gran número de cálculos que se deben hacer por cada píxel.

Ese mismo año, Jiang *et al.* [10] crearon un algoritmo capaz de estimar múltiples fotogramas a partir de dos fotogramas consecutivos. La importancia de este algoritmo en la alta calidad de los resultados y la capacidad de producir múltiples fotogramas intermedios, mientras que la mayoría de métodos existentes solo producen un único fotograma. Como se explica en su artículo "*Super SloMo: High Quality Estimation of Multiple Intermediate Frames for Video Interpolation*", su sistema está compuesto por dos subsistemas: el cálculo del flujo óptico[2] y la interpolación de flujo. Primero, se usa una red neuronal convolucional con arquitectura U-Net [19] para aproximar el flujo óptico bidireccional entre las dos imágenes. Después, y, por cada fotograma intermedio, otra red similar refina los resultados y predice mapas de visibilidad (imágenes con información sobre el flujo óptico). Por último, los mapas son aplicados al fotograma y este se combina con las dos imágenes de entrada, produciendo un fotograma interpolado.

[1] El efecto *ghosting* ocurre cuando un objeto se mueve a través de varias imágenes y estas son combinadas en una. Para más información, véase [29].

[2] El flujo óptico es el patrón de movimiento aparente de los objetos de una imagen entre dos fotogramas consecutivos, causado por el movimiento del objeto o de la cámara. [3]

4. Captura de requisitos

En este capítulo se explicarán los requisitos que se han identificado para el *software* a desarrollar, mediante el diagrama de casos de uso y el modelo de dominio. El *software* está pensado para funcionar sin interfaz de usuario; por ello, los requisitos aquí presentados responden a la "lógica de negocio" (o *backend*) de una aplicación convencional.

4.1 Casos de uso

El diagrama de casos de uso muestra las relaciones entre los actores y el sistema. Se denomina actor al rol bajo el que una persona puede utilizar el sistema.

En este proyecto se han determinado dos actores: desarrollador y usuario. El desarrollador es un caso particular de usuario, y, como tal, hereda su único caso de uso ("Aplicar el modelo a vídeos"). Por su parte, el desarrollador cuenta con 3 casos de uso propios. Todos estos se muestran en la figura 4.1 y se explican a continuación:

- **Generar datos de entrenamiento:** consiste en producir el conjunto de datos con el que se entrenará la red. Consta de dos pasos: obtener vídeos y extraer sus fotogramas.
- **Entrenar el modelo:** utilizando el conjunto de entrenamiento generado previamente, el desarrollador crea el entrenamiento y lo ejecuta, obteniéndose un modelo.
- **Evaluar el modelo:** consiste en evaluar la calidad del modelo, comparando los resultados que ofrece en un conjunto de datos externo.
- **Aplicar el modelo a vídeos:** proceso por el que se producen fotogramas interpolados en un vídeo concreto, usando el modelo obtenido en el entrenamiento. Este caso de uso lo pueden realizar tanto el usuario como el desarrollador, ya que el sistema está preparado para que un usuario externo lo pueda usar sin ningún conocimiento sobre su desarrollo.

4.2 Modelo de dominio

El modelo de dominio recoge los objetos del sistema y sus relaciones. El sistema a desarrollar está compuesto a base de *scripts* de código, representados como cuadrados en la figura 4.2. Esto significa que son archivos ejecutables, donde la mayoría del código es considerado un guión. Cada

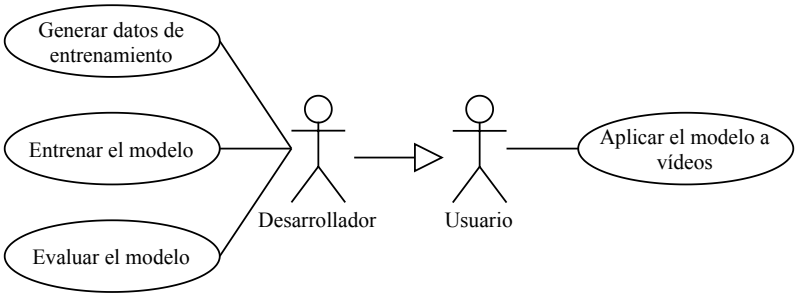


Figura 4.1: Diagrama de casos de uso del proyecto. Cada actor está unido a sus casos de uso, y la flecha entre actores indica herencia de casos de uso.

script tiene atributos, que toman diferentes valores según la funcionalidad deseada. Conjuntamente, el modelo de dominio refleja datos, expresados en el diagrama mediante rectángulos redondeados.

Las relaciones entre estos objetos se simbolizan con una línea que los une. Por cada una, dos números indican las cardinalidades de los objetos en la relación. Por ejemplo, la relación entre "Aplicar" y "Vídeo" tiene las cardinalidades 1 - 2, que significa que el *script* "Aplicar" se relaciona con dos vídeos (uno como entrada y otro como salida). Asimismo, las cardinalidades 1 - 1..* de la relación entre "Descargar" y "Vídeo" simbolizan que el *script* "Descargar" se relaciona con uno o más vídeos. A continuación se explica el diagrama detalladamente:

- El sistema lo componen 5 *scripts*: "Entrenar", "Evaluar", "Aplicar", "Descargar" y "GenerarDatos". Cada uno con sus propios atributos.
- El *script* "Entrenar" toma un conjunto de entrenamiento y otro de validación para producir un modelo.
- El *script* "Evaluar" utiliza un conjunto de evaluación y un modelo para generar unas métricas.
- El *script* "Aplicar" produce un vídeo en cámara lenta a partir de un modelo y un vídeo de origen.
- El *script* "Descargar" genera uno o más vídeos, especificados en una lista de reproducción que posee como atributo.
- El *script* "GenerarDatos" toma uno o más vídeos y crea dos conjuntos de datos (que posteriormente serán el de entrenamiento y validación).
- El objeto "Conjunto de datos" es una generalización de los 3 objetos "Conjunto de entrenamiento", "Conjunto de validación" y "Conjunto de evaluación".
- Un conjunto de datos se compone de una o más instancias.
- Una instancia está formada por exactamente 3 fotogramas.
- Un vídeo se constituye por uno o más fotogramas.

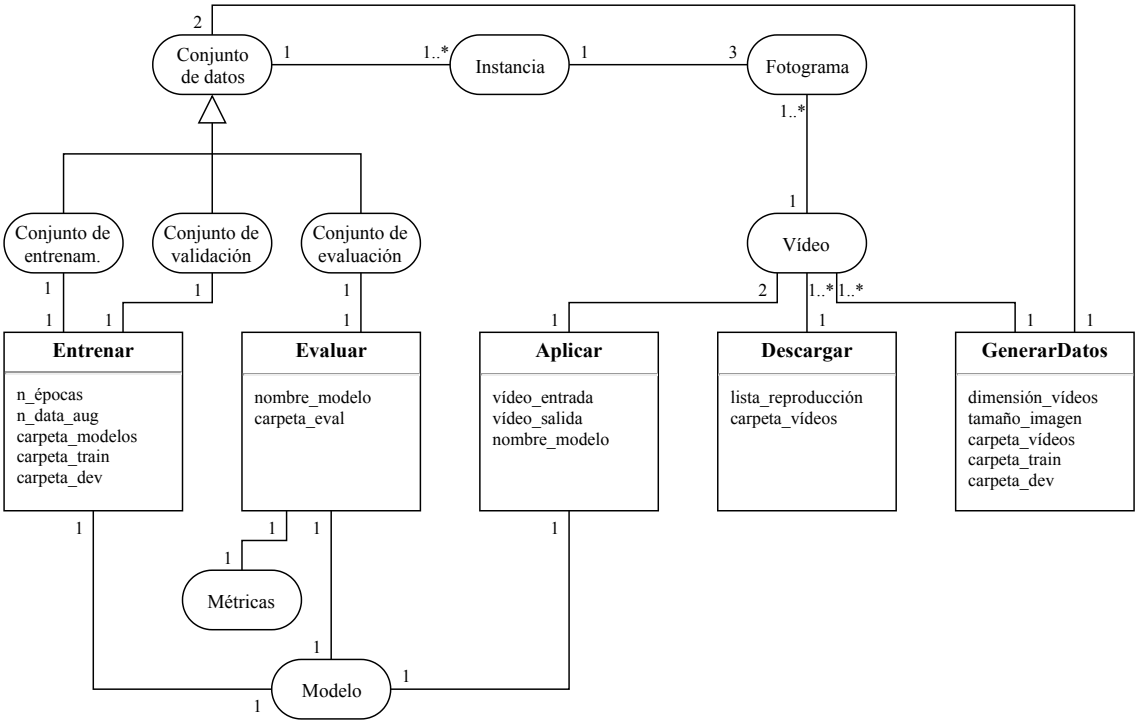


Figura 4.2: Modelo de dominio del proyecto. Los cuadrados representan *scripts*, mientras que los rectángulos redondeados indican datos.

5. Análisis y diseño

En este capítulo se explicará qué solución se ha dado para el proyecto, así como las partes en las que se divide. Para ello, se usarán diagramas de secuencia, que indican el flujo de eventos de cada *script*.

5.1 Diagrama de clases

Debido a que los *scripts* no tienen relaciones entre ellos (véase figura 4.2), se ha decidido no diseñar un diagrama de clases, ya que no se ha usado de programación orientada a objetos, y, en consecuencia, no hay clases.

5.2 Diagramas de secuencia

Se ha diseñado un diagrama de secuencia por cada *script*. En estos se especifica el flujo de eventos que siguen los *scripts* “Descargar”, “GenerarDatos”, “Entrenar”, “Evaluar” y “Aplicar”.

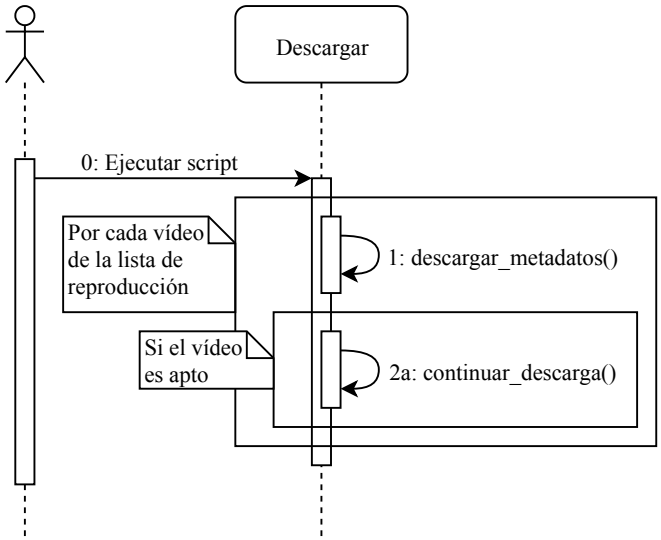


Figura 5.1: Diagrama de secuencia del *script* "Descargar". Por cada vídeo de la lista de reproducción (definida como atributo) se descargan sus metadatos y, si es apto (tiene el formato correcto y una duración pequeña), se continúa la descarga.

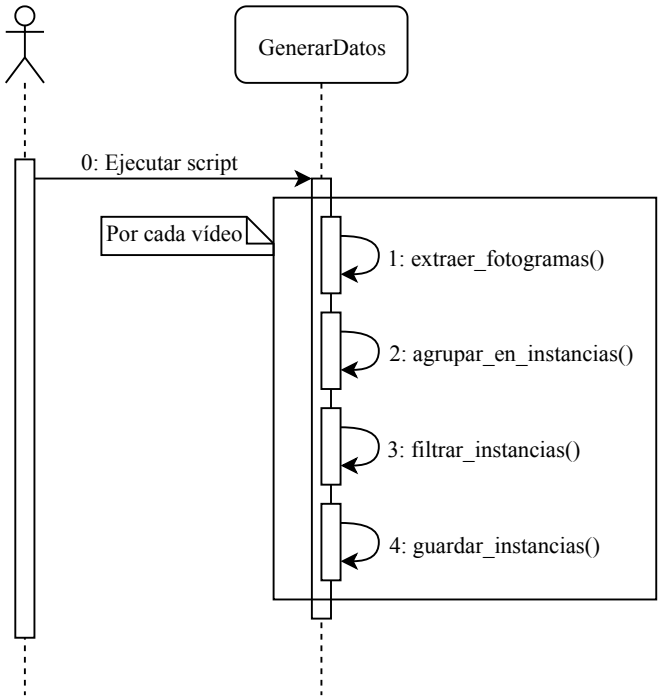


Figura 5.2: Diagrama de secuencia del *script* "GenerarDatos". Por cada vídeo descargado, se extraen sus fotogramas y se agrupan en instancias, formadas por 3 de ellos. Después de filtrar las instancias y descartar las que no sirven, se guardan en el disco, formando un conjunto de datos.

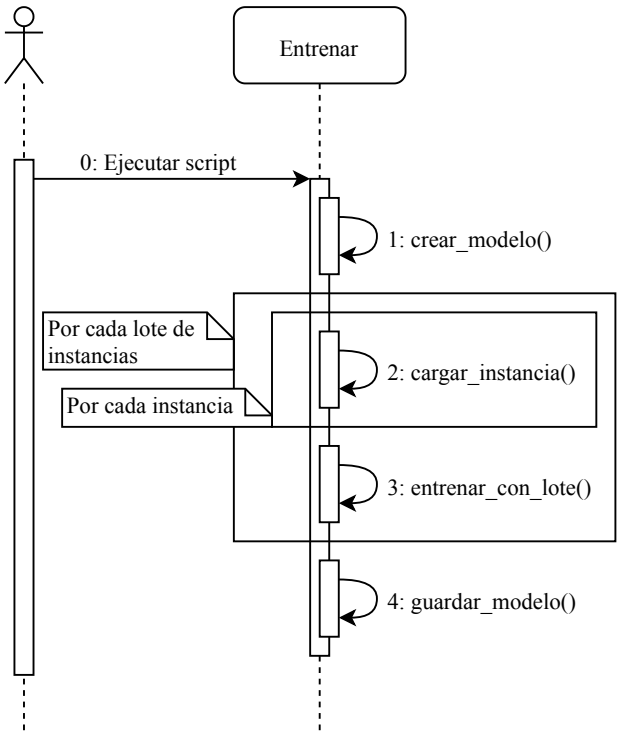


Figura 5.3: Diagrama de secuencia del *script* "Entrenar". Primero, se crea el modelo, que se compone de una red neuronal. Después, y, por cada lote, se cargan las instancias que lo forman y se entrena el modelo con él. Por último, una vez acabado el entrenamiento, se guarda el modelo.

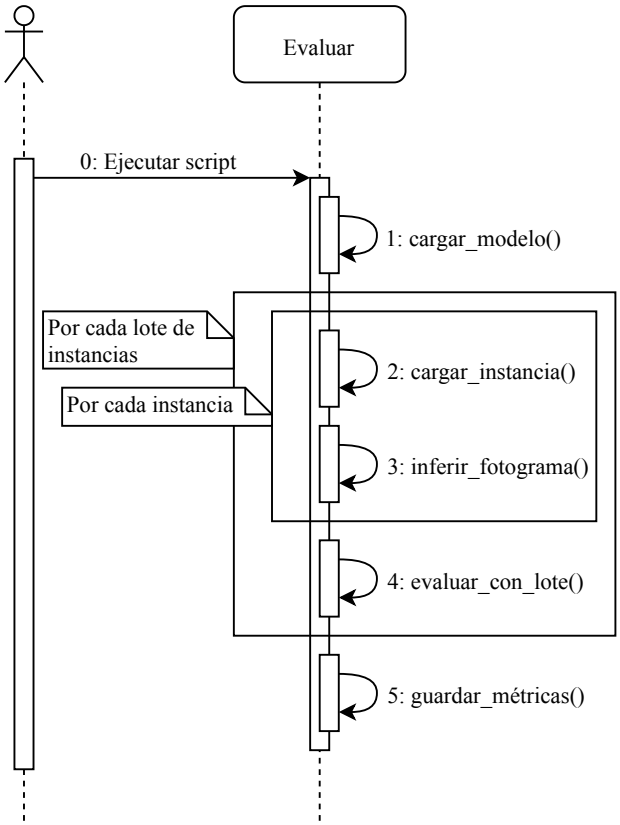


Figura 5.4: Diagrama de secuencia del *script* “Evaluar”. Como en el *script* anterior, “Entrenar”, se carga el modelo y, después de formar cada lote de instancias, el modelo se evalúa con él. Finalmente, se guardan las métricas obtenidas en la evaluación.

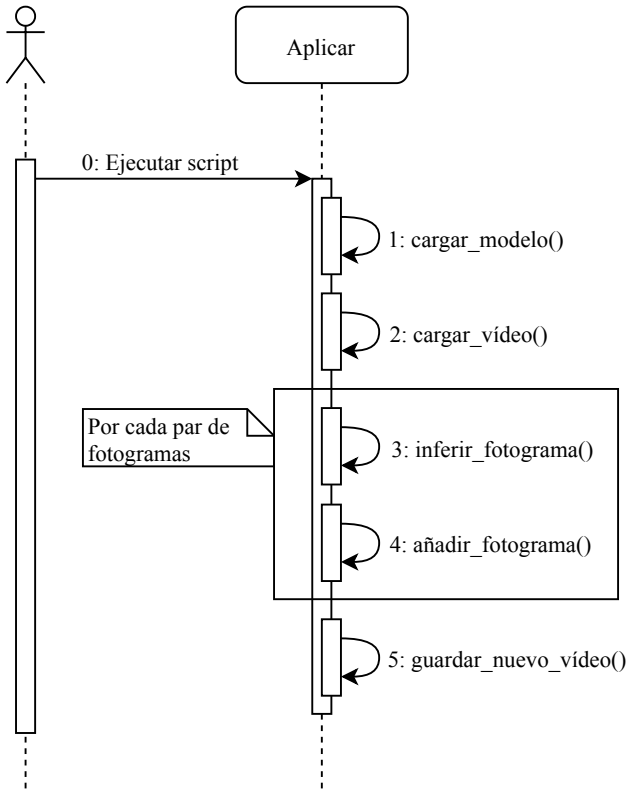


Figura 5.5: Diagrama de secuencia del *script* "Aplicar". Inicialmente, se cargan el modelo y el vídeo de origen. A continuación y, por cada par de fotogramas del vídeo, se infiere uno intermedio y se añade al vídeo resultante. Al acabar, este vídeo se guarda.

6. Desarrollo

6.1 Conjunto de entrenamiento

Al tratarse de un algoritmo basado en una red neuronal, se necesita un conjunto de datos con el que entrenarla. Esta red recibe como instancias de entrenamiento únicamente fotogramas; por tanto, para conseguir el conjunto de entrenamiento basta con reunir vídeos y posteriormente extraer sus fotogramas. En esta sección se explicará cómo se ha creado el conjunto de entrenamiento.

6.1.1 Reunir vídeos

Se han descargado de YouTube cerca de 300 vídeos con licencia *Creative Commons*, de distintas temáticas, como paisajes, animales y deportes. Se ha prescindido de los que tenían una duración demasiado grande (mayor que 6 minutos), para que no ocuparan demasiado espacio en disco. Además, y, por la misma razón, se ha requerido que tuvieran una resolución de 1080p, adecuada para poder apreciar el movimiento de los píxeles.

6.1.2 Obtener fotogramas

Una vez descargados los vídeos de entrenamiento, se deben extraer y tratar los fotogramas. Para lo primero, se han ido leyendo y agrupando en conjuntos de tres fotogramas, donde el primero y el tercero son la entrada para la red y el segundo es la clase a predecir. Estas tres imágenes forman una instancia.

En la mayoría de casos, un vídeo se compone de varios planos y escenas, por lo que si se tomaran dos fotogramas consecutivos pertenecientes a planos diferentes (donde el fotograma 1 es el último fotograma de un plano y el fotograma 2 es el primero del siguiente) como atributos para la red neuronal, se estarían creando datos erróneos (véase Figura 6.1). Por tanto, se han de ignorar las instancias cuyos fotogramas pertenecen a diferentes planos.

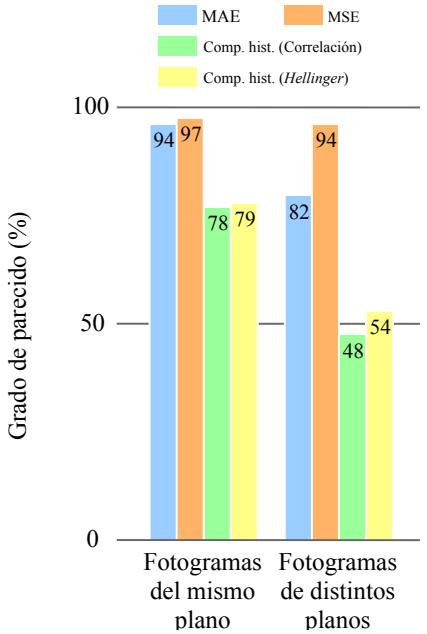
Para medir el parecido de dos fotogramas, se ha experimentado con varios métodos: diferencia absoluta (*MAE*, por sus siglas en inglés), error cuadrático medio (o *MSE*) y comparación de histogramas de color con dos métricas diferentes (véase Figura 6.2). Después de realizar diversas pruebas, se ha escogido el error cuadrático medio, ya que es más fiable que los demás métodos probados en la tarea de encontrar el cambio de plano. Por un lado, los que comparan histogramas

(a) Trío de imágenes válido porque pertenecen al mismo plano.

(b) Trío de imágenes no válido porque coinciden en un cambio de plano.

Figura 6.1: Dos ejemplos de agrupaciones de imágenes.

(a) Dos casos de fotogramas consecutivos. A la izquierda, pertenecientes al mismo plano. A la derecha, último fotograma de un plano y primero del siguiente.



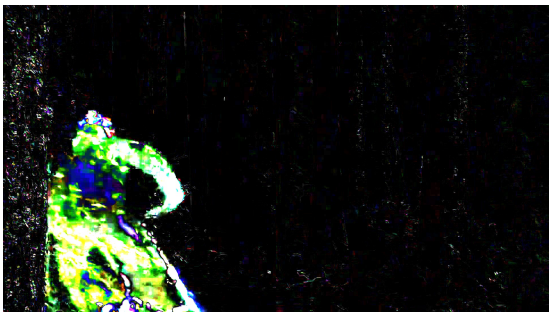
(b) Comparativa de cuatro métodos diferentes de comparación de imágenes, en los dos casos de la izquierda.

Figura 6.2: Comparación de fotogramas consecutivos. El eje Y del gráfico indica, entre 0 % y 100 %, el grado de semejanza de una imagen respecto de la anterior, establecido por 4 métodos diferentes: En azul, la diferencia absoluta, donde 0 % es 255 y 100 % es 0. En naranja, el error cuadrático medio, donde 0 % es 255^2 y 100 % es 0. En verde, el resultado de la comparación de sus histogramas de color (canales rojo, verde y azul) con la métrica "Correlación". 0 % es 0 y 100 % es 1. En amarillo, el resultado de la comparación de histogramas con la métrica "Hellinger", donde 0 % es 1 y 100 % es 0.

de color, a pesar de que son lo suficientemente sensibles comparando imágenes de la misma escena, no predicen tan bien si se ha cambiado de una a otra, porque un histograma solo contiene frecuencias de valores de un canal de color, y no la distribución de estos en píxeles de la imagen.

Dos fotogramas con un cambio significativo en un sector de la imagen.

(a) Diferencia en valor absoluto de las imágenes. $\bar{x} = 3.1$.



(b) Error cuadrático medio de las imágenes. $\bar{x} = 81.5$. Los valores superiores a 255 han sido rebajados a 255.

Figura 6.3: Ejemplo de la comparación de dos fotogramas con la diferencia en valor absoluto y el error cuadrático medio. Se puede apreciar que la media del error de la primera imagen es muy inferior a la de la segunda, ya que el cambio importante en la parte izquierda se contrarresta con el resto de la imagen, que no varía.

Así, comparar histogramas de una imagen y de la misma volteada, daría como resultado que son idénticas. Por lo tanto, si, por ejemplo, a un plano de un bosque le siguiera otro del mismo bosque pero desde otro ángulo, un método de comparación de histogramas no sería fiable para distinguir dicho cambio. Por otro lado, con el error cuadrático medio se obtienen menos falsos positivos que con la resta entre imágenes, porque con esta última, diferencias notables en ciertas partes de la imagen son compensadas con la similitud de la mayoría de la imagen. Con el error cuadrático medio, al penalizar en mayor medida los errores más grandes, se reduce esa compensación. En la figura 6.3 se puede ver un ejemplo de lo anterior.

Una vez se tiene la métrica para comparar fotogramas, lo siguiente es establecer un umbral de aceptación. En un primer momento se estableció un intervalo de MSE, fuera del cual los fotogramas eran descartados. Estos dos valores eran obtenidos empíricamente; o lo que es lo mismo, ajustados mediante prueba y error. Pero, dado que cada vídeo tiene una naturaleza diferente, la distribución de los valores del MSE es diferente, y un intervalo que servía para un vídeo concreto, podía no servir para otro. Por esta razón y, para no tener que probar con la mayoría de los vídeos, se pensó en establecer este intervalo según la media de dichos valores de cada vídeo. De esta manera, el intervalo iba ajustándose según este parámetro. Sin embargo, este método descarta los valores altos del MSE y no los cambios abruptos o repentinos, que es lo que se pretendía (encontrar el cambio de plano). Por esto, se decidió utilizar como métrica la diferencia entre los valores de MSE. Por cada trío de fotogramas se calcula primero los MSE de los dos pares consecutivos de fotogramas y, después, la diferencia absoluta de estos dos valores (véase Figura 6.4). Al usar las

diferencias entre los valores, se consigue pasar por alto su valor (si son altos o bajos) y solo se tiene en cuenta la diferencia con el anterior. El umbral máximo de aceptación actualmente empleado, ya que ha dado buenos resultados es $\bar{x} + \frac{S}{4}$, donde $\bar{x}$ y S son la media y desviación típica muestrales, respectivamente.

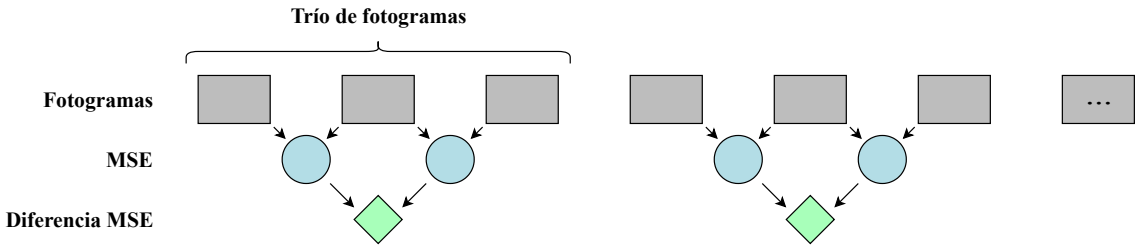


Figura 6.4: Las diferencias del MSE son computadas por cada trío de fotogramas sin solapamiento. La razón por la que no se toman todos los tríos posibles es porque habitualmente un vídeo posee suficientes fotogramas para lo que requiere este proyecto. Además, disponer de cada trío triplicaría el coste computacional.

Después de filtrar y agrupar los fotogramas, se han excluido los que menos información proporcionan; esto es, las instancias con menor diferencia de MSE, puesto que sus fotogramas cambian poco. Por cada vídeo se han tomado las 300 instancias con mayor diferencia de MSE. Este número se ha elegido influenciado por la cantidad de imágenes del conjunto de entrenamiento de trabajos similares, citados en el capítulo 3 [10, 18]: 300 000 y 250 000, respectivamente. Como se tiene un total de 300 vídeos y de cada uno de ellos se extraen 300 instancias (estando cada una formada por tres imágenes), el conjunto de entrenamiento tiene un volumen de alrededor de 270 000 imágenes. Si se almacenaran todas en su tamaño original, estos datos requerirían un almacenamiento del orden de *terabytes*, lo que provocaría, por falta de espacio en disco, que fuera necesario extraer los fotogramas y eliminarlos iterativamente durante el proceso de entrenamiento. De realizar el entrenamiento de esta manera, la red no aprendería de manera imparcial respecto de los datos, ya que estos necesitan ser seleccionados aleatoriamente para reducir el sesgo [4, página 277]. Por este motivo, se ha cortado cada imagen a un segmento aleatorio de 144×144 píxeles (el mismo para cada instancia); número que es también cercano al utilizado en los trabajos mencionados, además de ser compatible con las operaciones que realiza la red neuronal, pues es potencia de 2. Para introducir variabilidad en los datos y no saturar el conjunto de entrenamiento con pocas gamas de colores por tomar siempre la misma parte de la imagen, este segmento se ha seleccionado según la siguiente función de masa de probabilidad:

$$P(s_i) = \frac{MMSE(s_i)}{\sum_{j=1}^{n} MMSE(s_j)} \quad \forall i \in \{1, ..., n\}, \tag{6.1}$$

donde $\sum_{j=1}^{n} MMSE(s_j) \neq 0$, n es el número de segmentos posibles y s es un segmento de la imagen. $MMSE(s_i)$ es la media entre el error cuadrático medio de la primera imagen de la instancia con la segunda, y el de la segunda con la tercera.

Para formar el conjunto de validación, se ha tomado un 10 % de los datos del conjunto de entrenamiento.

6.2 Entrenar la red neuronal

Se ha usado una única red neuronal que, dadas dos imágenes, produce una imagen intermedia, del mismo tamaño. Se ha entrenado durante 7 épocas[1], pero en las 2 últimas no se redujo el error

[1] Las épocas son el número de iteraciones de entrenamiento sobre el conjunto de datos [4, página 243].

de validación. En total, el entrenamiento tardó 6 horas.

La función de error[2] utilizada está compuesta por dos partes: el error cuadrático medio (MSE) y la diferencia de gradientes [13]. Esta última es la diferencia entre los gradientes de píxeles de la imagen real y de la predicha, que se computan restando a cada píxel por su colindante horizontal y vertical. Introduciendo este término, se busca que la red sea capaz de identificar los cambios bruscos (bordes de objetos), y evite el efecto borroso y el *ghosting*.

El optimizador elegido ha sido *ADAM* [11], con una tasa de aprendizaje (o *learning rate*) de 10^{-3}, ya que con esta configuración se obtuvo un buen rendimiento. A continuación, se explicará con más detalle qué arquitectura se ha adoptado, además de los regularizadores utilizados.

6.2.1 Arquitectura de la red

La red recibe como entrada las dos imágenes de origen, apiladas sobre su tercer eje; es decir, una encima de la otra. De esta manera, se obtiene una matriz de 6 píxeles de profundidad, pues cada imagen tiene tres canales de color. Se ha usado la arquitectura U-Net [19], una red neuronal que consiste en un codificador y un decodificador, con conexiones salto (o *skip connections*) entre ellos. Al ser completamente convolucional, admite un tamaño de entrada variable. La versión implementada en este proyecto (véase Figura 6.5) tiene 5 jerarquías, cada una compuesta por un bloque en el codificador y otro en el decodificador (excepto en la quinta jerarquía). Cada bloque está compuesto por dos operaciones idénticas, ambas constituidas por una capa de convolución con un núcleo de 3×3 píxeles, la función de activación Leaky ReLU [15] y una capa de *Batch Normalization* [9] (o normalización de lotes). Estas operaciones están representadas en la figura 6.5 por las flechas azules. En el primer y segundo bloque del codificador, el núcleo de convolución utilizado ha sido de 7×7 y 5×5 píxeles, respectivamente. Según Jiang *et al.* [10], esto es importante para capturar el movimiento de largo recorrido. Asimismo, el núcleo de convolución de la última capa de la red es de 1 píxel, ya que solo se requiere asignar el vector de atributos de 32 píxeles de profundidad a uno de 3. El número de filtros usados en cada capa de convolución corresponde con la profundidad del mapa de atributos que generan, indicado encima de cada uno de estos. Entre cada bloque del codificador y, representado con una flecha roja, se realiza una operación de *pooling* (o remuestreo hacia abajo) promedio para reducir la dimensionalidad de los atributos, con el objetivo de disminuir el coste computacional y espacial. De manera análoga, entre los bloques del decodificador se llevan a cabo las operaciones de *upsampling* (o remuestreo hacia arriba) y convolución, junto con las capas de Leaky ReLU y *Batch Normalization*. Esta serie de operaciones se indica con las flechas verdes. Por cada par de bloques del codificador y decodificador de la misma jerarquía, se copia el último mapa de atributos del codificador (flecha gris) y se concatena con el primero del decodificador (signo más). Este procedimiento, llamado conexión salto, ayuda a solucionar el problema de la desaparición de gradientes [6] [7], por el que los gradientes de los pesos de las redes neuronales, a medida que estas se hacen más profundas, decrecen tanto que, en algunos casos, desaparecen.

6.2.2 Regularizadores utilizados

«La regularización es cualquier modificación hecha a un algoritmo de aprendizaje automático que esté destinada a reducir su error de generalización, y no el de entrenamiento.»

Los modelos tienden a ajustarse demasiado a los datos, esto es, a aprender su ruido. Al aplicar un regularizador se consigue que el modelo no capte las características particulares de los datos, reduciéndose el *overfitting* (o sobreajuste). De igual modo, una regularización excesiva provocaría *underfitting* (o subajuste). Un ejemplo de regularizador es un parámetro en la función de error de una red neuronal, que la penaliza dependiendo del valor que tomen los pesos de la red en ese

[2] La función de error (también llamada función objetivo o de coste) es la función que el algoritmo de optimización busca minimizar [4, página 80].

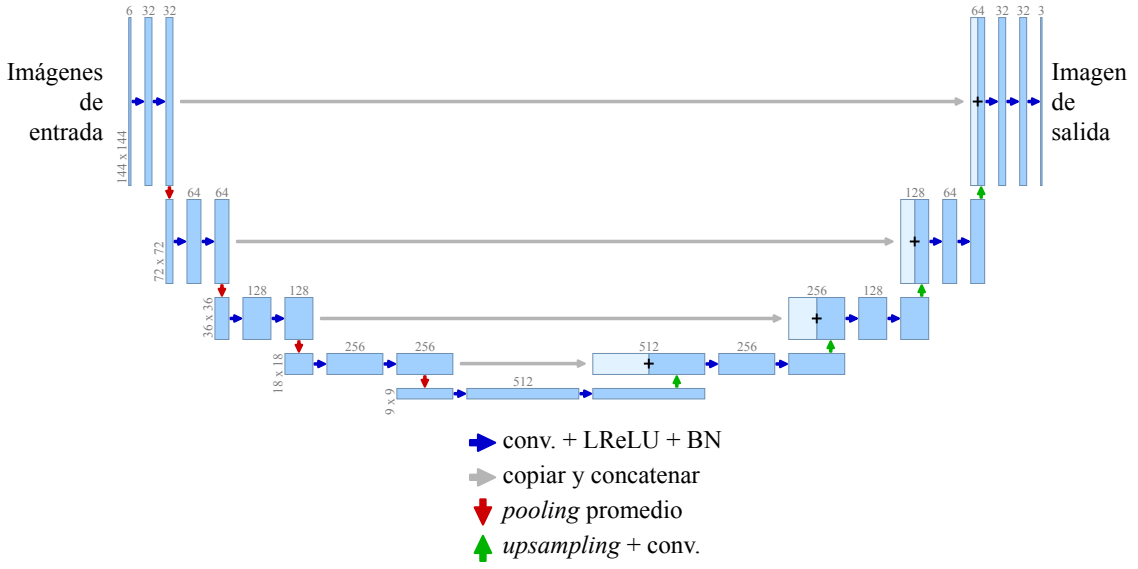


Figura 6.5: Arquitectura de la red neuronal. Los rectángulos azules representan los mapas de atributos (o *feature maps*) tridimensionales. El largo y ancho está indicado a la izquierda de cada uno y, la profundidad (el número de canales), encima. Los rectángulos blancos denotan los mapas copiados, que se concatenan con el de su derecha. Las flechas indican las diferentes operaciones que se realizan.

momento. [4, página 117] En este apartado se explicarán los regularizadores utilizados en este proyecto.

Existen muchas maneras de aplicar regularización a un algoritmo, siendo el aumento de datos una de las más comunes [4, página 236]. Consiste en añadir al conjunto de entrenamiento datos artificiales, creados mediante la modificación de instancias existentes. Para este proyecto, el aumento de datos se ha realizado sobre la marcha durante el entrenamiento, de 4 maneras diferentes: invirtiendo el orden temporal de las imágenes (intercambiando la primera y la tercera), volteándolas horizontalmente, verticalmente y en ambos ejes. Para saber cuántas instancias se deben añadir por cada instancia real, se ha entrenado la red con 5 configuraciones distintas, variando el número de instancias agregadas. Las modificaciones y su orden se eligen aleatoriamente. Después de observar las pruebas, reflejadas en a tabla 6.1, se ha decidido añadir 2 instancias, ya que, como se muestra en la tabla, a medida que se incrementa el número de instancias agregadas, el error de entrenamiento aumenta. No así el de validación, que desciende al agregar una instancia, pero prácticamente se detiene al añadir más. Probablemente, esto ocurre porque, al tener un tamaño de lote constante, la proporción de instancias reales se hace cada vez menor al añadir más instancias artificiales, lo que no ayuda a la generalización. El tamaño de lote se ha mantenido constante en vez de irse aumentando por limitaciones en cuanto a memoria de la tarjeta gráfica. Este ha sido 16 (el máximo número potencia de 2 que la tarjeta gráfica puede soportar), cuando lo habitual suele ser de 32 a 256 [4, página 276]. No obstante, utilizar tamaños de lote pequeños no es tan ineficaz como puede parecer intuitivamente. De hecho, según un artículo de Masters y Luschi [17], los mejores resultados en cuanto estabilidad de entrenamiento y error de generalización se consiguen con lotes de entre 2 y 32 instancias.

Otra forma de regularización muy habitual en *deep learning* es el *early stopping* (o parada temprana), por su simplicidad y efectividad. Consiste en parar el entrenamiento antes de cumplirse el número establecido de épocas, cuando el error de validación esté en su mínimo. Y es que, al entrenar modelos con suficiente capacidad para sobreajustar la tarea, estos tienden a aumentar el error del conjunto de validación a partir de un cierto punto. Utilizando esta técnica, se consigue

Instancias agregadas	Error de entrenamiento	Error de validación
0	**500**	323
1	508	318
2	511	**315**
3	514	316
4	525	318

Tabla 6.1: Resultados del entrenamiento con diferente cantidad de instancias agregadas por cada una real. Los errores corresponden al instante del entrenamiento en el que el modelo era óptimo.

que el modelo generalice mejor, obteniendo un menor error con datos nuevos. En la figura 6.6 se muestra un ejemplo del entrenamiento de un modelo, donde el error de entrenamiento disminuye continuamente, mientras que el de validación asciende pasado un tiempo. [4, páginas 241, 243]

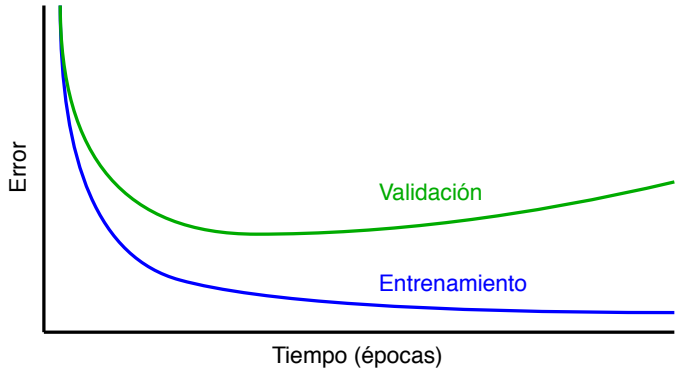


Figura 6.6: Curvas que indican el error de entrenamiento y validación durante el entrenamiento de un modelo.

Otro regularizador empleado es la normalización de lotes [9]. Este algoritmo es un método de reparametrización[3], que funciona normalizando el lote en la capa correspondiente; es decir, restando a cada neurona su media en el lote y dividiendo el resultado por su desviación típica. Esto reduce significativamente el problema de coordinar actualizaciones a través de muchas capas, por el que una red puede producir resultados inesperados. Porque todas sus capas se actualizan simultáneamente, pero estas actualizaciones se calculan asumiendo que las demás capas permanecen constantes. Como consecuencia el tiempo de entrenamiento se reduce notablemente. Tras el entrenamiento y, durante la inferencia, la media y la desviación que se utilizan son el promedio de todas las que se calcularon en el entrenamiento. Esto permite poder inferir una sola instancia, sin necesidad de un lote con el que computar dichos estadísticos. Pero también produce que el error en los conjuntos de validación y evaluación sea menor que en el de entrenamiento, ya que la media y desviación típica utilizadas en estos conjuntos son más representativas de los datos que las calculadas durante el entrenamiento. [4, páginas 313-315] Aunque este método no fue creado para ser un regularizador, se ha comprobado que en algunas tareas reduce el error de generalización [4, página 420].

[3]La reparametrización es un concepto matemático que sirve para realizar un cambio de variables que definen una función. Para más detalles, véase [39].

7. Evaluación

En este capítulo se explicarán las pruebas realizadas y las métricas obtenidas, y se compararán con otras soluciones.

7.1 Modelo base

Se ha implementado un modelo base, que aunque no hace uso de aprendizaje automático, es útil para comparar el grado de mejora del modelo principal, ya que se supone que este último no producirá peores resultados que el modelo base. Consiste en la superposición de los dos fotogramas de entrada, calculando la media píxel a píxel de ambos fotogramas. En la figura 7.1 se muestra el resultado con dos imágenes de ejemplo.

(a) Fotograma 1. (b) Media de los fotogramas. (c) Fotograma 2.

Figura 7.1: Ejemplo de un resultado del modelo base.

7.2 Pruebas realizadas

Se han realizado evaluaciones de los modelos base y principal respecto al conjuntos de datos *UCF101* [21], pues es bastante empleado en problemas de visión por computador, como [8, 10, 13]; y además, se trata de uno de los conjuntos de vídeos más grande y con mayor diversidad, ya que lo conforman 13 320 vídeos pertenecientes a 101 categorías. Los resultados también se comparan con los obtenidos en otros proyectos similares: *FlowNet2* [8] y *Super SloMo* [10]. La métrica utilizada es el error de interpolación (EI) [5], definido por la raíz cuadrada de la media cuadrática de la diferencia entre la imagen real y la predicha:

$$EI = \sqrt{\frac{1}{N}\sum_{x,y}(I(x,y) - \hat{I}(x,y))^2}\,, \tag{7.1}$$

donde N es el número de canales de color y x, y son las coordenadas de la imagen. Se ha elegido esta métrica porque se utiliza en la gran mayoría de proyectos de este ámbito.

Como se puede observar en la tabla 7.1, el error del modelo principal es 4 puntos superior que el de *FlowNet2* [8], pero está a la misma distancia por debajo del modelo base. Se podría asegurar, por tanto, que la calidad del modelo principal está entre medias del estado del arte y el modelo base.

Modelo	EI
Modelo base	16.4
Modelo principal	12.5
FlowNet2 [8]	8.4
Super SloMo [10]	**7.8**

Tabla 7.1: Resultados en el conjunto de datos *UCF101*.

7.3 Ejemplos

Los modelos base y principal han sido aplicados a vídeos de prueba, produciendo los resultados mostrados en las figuras 7.2 y 7.3. Como se puede apreciar, el modelo principal tiene un buen rendimiento en cuanto a síntesis de formas en regiones con poco movimiento. En cambio, en áreas donde los fotogramas trazan un largo recorrido, el error es superior y la imagen intermedia es más borrosa, ya que es más difícil aproximarla. Aún así, en estas situaciones, el modelo principal supera al base la gran mayoría de las veces, porque, si bien su resultado se asemeja a una superposición de fotogramas, las partes donde las formas no se superponen están más atenuadas.

Respecto a los colores, el modelo los produce fielmente, utilizando una gama muy similar. Sin embargo, en algunas ocasiones tiende a acentuar los colores más claros, produciendo una sensación de parpadeo en la visualización del vídeo.

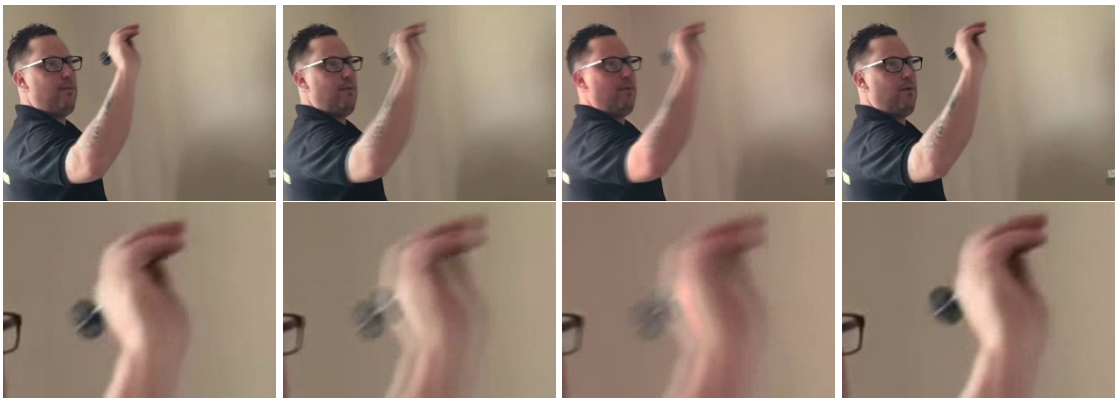
Fotograma 1. Modelo base. Modelo principal. Fotograma 2.

Fotograma 1. Modelo base. Modelo principal. Fotograma 2.

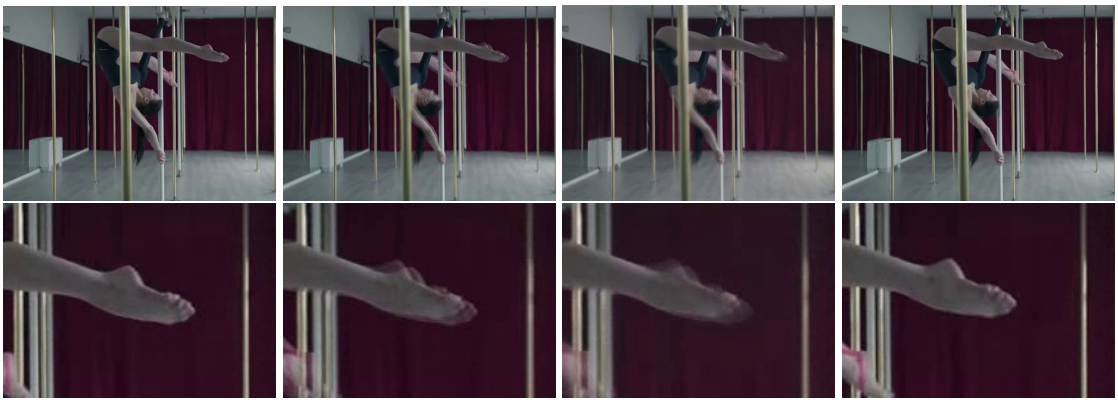
Fotograma 1. Modelo base. Modelo principal. Fotograma 2.

Figura 7.2: Resultados de los modelos base y principal.

Fotograma 1. Modelo base. Modelo principal. Fotograma 2.



Fotograma 1. Modelo base. Modelo principal. Fotograma 2.



Fotograma 1. Modelo base. Modelo principal. Fotograma 2.

Figura 7.3: Resultados de los modelos base y principal.

8. Conclusiones

8.1 Conclusiones de gestión

En esta sección se explicarán las conclusiones respecto al planteamiento inicial (capítulo 2), referente a los objetivos, la planificación temporal y, por consiguiente, la evaluación económica.

8.1.1 Objetivos

En este apartado se analizará el cumplimiento de los objetivos expuestos en la sección 2.1. Respecto a los objetivos personales, al no existir un método para verificar su comprobación, se asumirá que se han cumplido.

De *software*

- **Conseguir un modelo con una calidad aceptable:** se ha cumplido este objetivo, ya que, el modelo, aunque posee gran margen de mejora, logra producir vídeos donde se aprecia el efecto de cámara lenta. Además, los resultados mostrados en el capítulo 7, Evaluación corroboran su calidad.
- **Escribir buen código:** se ha implementado un *software* con buenas características, ya que posee comentarios explicativos, tiene en cuenta posibles fallos y actúa en consecuencia, y es modular, pues está organizado en funciones.

Académicos

- **Entregar el proyecto en las convocatorias de junio o julio:** se ha cumplido este objetivo, porque a día de hoy (18 de julio de 2019) se se han iniciado los trámites de entrega.
- **Obtener una nota mínima de "Notable":** a pesar de que este objetivo aún no se puede verificar, ya que la calificación se recibirá después de entregar el proyecto, se da por cumplido, puesto que el alumno está satisfecho con el trabajo realizado.

8.1.2 Planificación temporal

Una vez que se ha terminado el proyecto y se han recogido las duraciones reales de las tareas, se comparan con las estimadas en el Alcance, sección 2.2. En la tabla 8.1 se muestra esta comparativa, y de forma más visual, en la figura 8.1.

Tarea	Duración estimada	Duración real
Gestión	130	145
Estudio	110	100
Captura de requisitos	4	3
Análisis y diseño	7	3
Desarrollo	225	270
Evaluación	19	12
TOTAL	495	533

Tabla 8.1: Comparativa de la estimación temporal de las tareas y su duración real, expresado en horas.

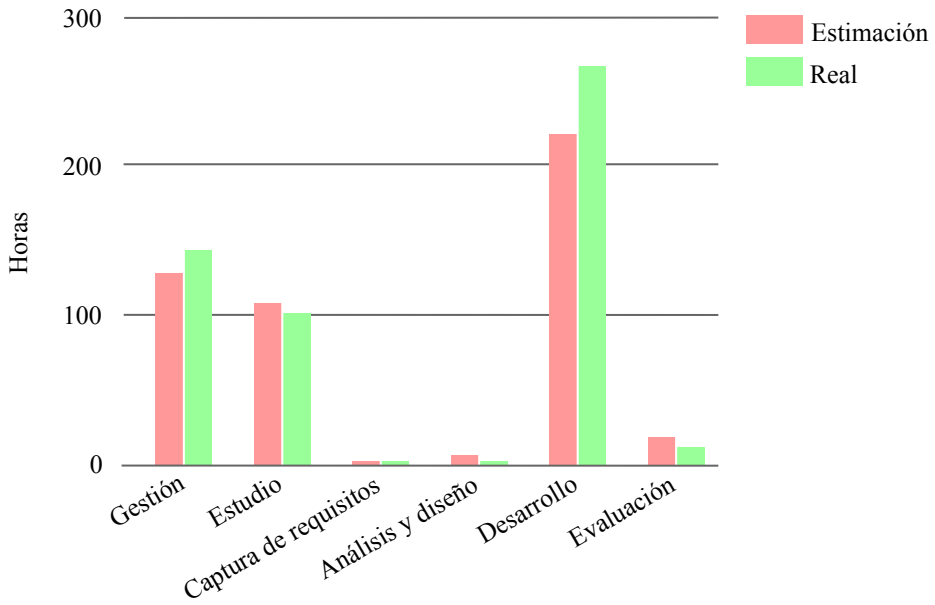


Figura 8.1: Comparativa de la estimación temporal de las tareas y su duración real.

Como se puede apreciar en la figura anterior, se ha invertido menos tiempo del previsto en todas las tareas menos Gestión y Desarrollo, donde el incremento de la duración ha sido de un 11 % y 20 %, respectivamente. El incremento en la tarea de Desarrollo responde a que se subestimó la complejidad del proyecto, por tratarse del primero de estas características que se realizaba. Estas variaciones provocan un aumento del 8 % en la duración total, resultando en 533 horas.

8.1.3 Evaluación económica

El incremento en la duración de las tareas se refleja también en la evaluación económica. Así, los costes por horas trabajadas pasan a ser 6 396€; y, sumando el resto de costes, el total asciende a 6 877€, 456€ más que lo estimado en la sección 2.6.

8.1.4 Riesgos

A lo largo del proyecto, solo se ha cumplido un riesgo: Incumplimiento de la planificación (véase tabla 2.25), debido principalmente a que se establecía una hora de trabajo al día durante todos los días, mientras que ha habido días en los que no ha sido posible dedicar tiempo al proyecto. Además, la tarea de Desarrollo ha tenido más duración de la prevista, todo esto resultando en un retraso general.

Sin embargo, gracias a la prevención, por una parte, ya que se estableció la suficiente holgura para minimizar el efecto del riesgo; y al plan de contingencia, por otra, puesto que se dedicaron más horas de trabajo diarias; este riesgo no ha imposibilitado la consecución de los objetivos del proyecto (véase sección 2.1).

8.2 Conclusiones del proyecto

Se ha desarrollado un sistema de interpolación de fotogramas con inteligencia artificial, que, aunque cuyo desempeño dista de ser excelente, logra resultados aceptables en la mayoría de vídeos probados. A la hora de visualizar el vídeo resultante, salvando el efecto de parpadeo descrito en el capítulo 7, Evaluación, se percibe claramente el efecto de cámara lenta, cumpliéndose el propósito del proyecto. A continuación, se explicarán los principales hitos del proyecto y los problemas surgidos durante su realización.

Por una parte, se ha generado un conjunto de entrenamiento que combina cantidad (270 000 imágenes), variedad (de 300 vídeos con diferentes planos), aporte de información (por el proceso de selección descrito en el Desarrollo) y poco espacio (1.5 GB). Por otra, se ha construido el modelo de *deep learning* entrenado con el conjunto de entrenamiento. Este genera un fotograma artificial a partir de dos originales, produciendo un vídeo con efecto de cámara lenta. Como se ha mostrado en el capítulo 7, Evaluación, la calidad del modelo es buena comparada con el modelo base. No obstante, cuando el movimiento de los píxeles entre los dos fotogramas de entrada es grande, el resultado deja bastante que desear, ya que la calidad del fotograma generado se resiente notablemente.

Durante la creación del conjunto de entrenamiento se han topado varios problemas. Primero, se necesitaba distinguir y separar los diferentes planos del vídeo, para no crear datos erróneos (véase figura 6.1). Para ello, es necesaria una métrica con la que comparar los fotogramas y un método de comparación que haga uso de ella. La dificultad de este proceso reside en poder diferenciar todos los cambios de plano, descartando el mínimo posible de falsos positivos; esto es, fotogramas que pertenecen al mismo plano pero son identificados como lo contrario. Esto se ha conseguido gracias a numerosas pruebas, mediante las cuales se determinó un umbral de aceptación adaptado a cada vídeo.

Después de seleccionar los fotogramas, se recortaban en pequeños segmentos para almacenarlos como instancias. En un principio, este procedimiento se hacía aleatoriamente, pudiéndose escoger cualquier parte de la imagen con la misma probabilidad. Esto causaba que hubiera muchas instancias con poco o ningún cambio, ya que, generalmente, el mayor movimiento se concentra en el centro de la imagen. Para solucionarlo, se decidió implementar una función de probabilidad, según la cual los segmentos tendrían una probabilidad mayor de ser escogidos cuanto mayor movimiento tuvieran, calculado como MSE.

En cuanto al entrenamiento de la red neuronal, han surgido diferentes complicaciones. En primer lugar, la arquitectura de la red fue difícil de construir, ya que, al tratarse de la primera vez que se realizaba un proyecto de este tipo, las modificaciones y su elección se hacían mediante prueba y error. En segundo lugar, y, relacionado con lo anterior, la duración de los entrenamientos de la red neuronal ha supuesto una dificultad añadida, porque para elegir si se consolidaba una modificación se necesitaba esperar la gran mayoría del tiempo del entrenamiento.

8.3 Trabajo futuro

A juicio personal, los dos principales problemas del modelo (borrosidad en el movimiento de largo recorrido y efecto de parpadeo con algunos colores) podrían verse resueltos o minimizados por un modelo mayor. Y es que, cuantos más parámetros posee un modelo, más capacidad para representar la solución adquiere (generalmente y dentro de unos límites) [4, página 21]. Para entrenar este modelo mayor sería necesario mucho más tiempo de ejecución, recurso muy limitado durante la realización del proyecto.

Además, hay configuraciones de la red neuronal (llamadas hiperparámetros) que, a pesar de ser buenas, actualmente no son óptimas, por el tiempo que supondría hacer una búsqueda exhaustiva de estas. Ejemplos de hiperparámetros son la tasa de aprendizaje, el decaimiento de pesos[1] (o *weight decay*), el tamaño del lote, la función de error y las funciones de activación de las capas [35].

8.4 Agradecimientos

Las imágenes de cabecera de los capítulos han sido diseñadas por *rawpixel.com*, *Freepik*.

[1] El decaimiento de pesos es un parámetro de la función de error que provoca que los pesos de la red neuronal sean más pequeños, ayudando a la generalización [4, página 117].

Referencias

Libros

[1] Ethem Alpaydin. *Introduction to Machine Learning*. 2nd. The MIT Press, 2010. ISBN: 9780262012430 (véanse páginas 33, 34).

[2] Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006 (véase página 33).

[3] A. Burton y J. Radford. *Thinking in Perspective: Critical Essays in the Study of Thought Processes*. Psychology in progress. Methuen, 1978. ISBN: 9780416858402 (véase página 35).

[4] Ian Goodfellow, Yoshua Bengio y Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016 (véanse páginas 11, 34, 50-53, 62).

Artículos

[5] Simon Baker y col. «A database and evaluation methodology for optical flow». En: *International Journal of Computer Vision* 92.1 (2011), páginas 1-31 (véase página 56).

[6] Xavier Glorot y Yoshua Bengio. «Understanding the difficulty of training deep feedforward neural networks». En: Proceedings of Machine Learning Research 9 (mayo de 2010), páginas 249-256. URL: http://proceedings.mlr.press/v9/glorot10a.html (véase página 51).

[7] Kaiming He y col. «Deep Residual Learning for Image Recognition». En: *CoRR* abs/1512.03385 (2015). arXiv: 1512.03385. URL: http://arxiv.org/abs/1512.03385 (véase página 51).

[8] Eddy Ilg y col. «Flownet 2.0: Evolution of optical flow estimation with deep networks». En: *Proceedings of the IEEE conference on computer vision and pattern recognition* (2017), páginas 2462-2470 (véase página 56).

[9] Sergey Ioffe y Christian Szegedy. «Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift». En: *CoRR* abs/1502.03167 (2015). arXiv: 1502.03167. URL: http://arxiv.org/abs/1502.03167 (véanse páginas 51, 53).

[10] Huaizu Jiang y col. «Super SloMo: High Quality Estimation of Multiple Intermediate Frames for Video Interpolation». En: *CoRR* abs/1712.00080 (2017). arXiv: 1712.00080. URL: http://arxiv.org/abs/1712.00080 (véanse páginas 12, 35, 50, 51, 56).

[11] Diederik P Kingma y Jimmy Ba. «Adam: A method for stochastic optimization». En: *arXiv preprint arXiv:1412.6980* (2014) (véase página 51).

[12] Yann LeCun y col. «Generalization and network design strategies». En: *Connectionism in perspective* 19 (1989) (véase página 34).

[13] Ziwei Liu y col. «Video Frame Synthesis using Deep Voxel Flow». En: *CoRR* abs/1702.02463 (2017). arXiv: 1702.02463. URL: http://arxiv.org/abs/1702.02463 (véanse páginas 51, 56).

[14] Gucan Long y col. «Learning Image Matching by Simply Watching Video». En: *CoRR* abs/1603.06041 (2016). arXiv: 1603.06041. URL: http://arxiv.org/abs/1603.06041 (véase página 34).

[15] Andrew L. Maas, Awni Y. Hannun y Andrew Y. Ng. «Rectifier nonlinearities improve neural network acoustic models». En: *in ICML Workshop on Deep Learning for Audio, Speech and Language Processing* (2013) (véase página 51).

[16] Dhruv Mahajan y col. «Moving Gradients: A Path-based Method for Plausible Image Interpolation». En: *ACM Trans. Graph.* 28.3 (jul. de 2009), 42:1-42:11. ISSN: 0730-0301. DOI: 10.1145/1531326.1531348. URL: http://doi.acm.org/10.1145/1531326.1531348 (véase página 35).

[17] Dominic Masters y Carlo Luschi. «Revisiting Small Batch Training for Deep Neural Networks». En: *CoRR* abs/1804.07612 (2018). arXiv: 1804.07612. URL: http://arxiv.org/abs/1804.07612 (véase página 52).

[18] Simon Niklaus, Long Mai y Feng Liu. «Video Frame Interpolation via Adaptive Convolution». En: *CoRR* abs/1703.07514 (2017). arXiv: 1703.07514. URL: http://arxiv.org/abs/1703.07514 (véanse páginas 35, 50).

[19] Olaf Ronneberger, Philipp Fischer y Thomas Brox. «U-Net: Convolutional Networks for Biomedical Image Segmentation». En: *CoRR* abs/1505.04597 (2015). arXiv: 1505.04597. URL: http://arxiv.org/abs/1505.04597 (véanse páginas 35, 51).

[20] Jürgen Schmidhuber. «Deep learning in neural networks: An overview». En: *Neural networks* 61 (2015), páginas 85-117. URL: https://doi.org/10.1016/j.neunet.2014.09.003 (véanse páginas 12, 34).

[21] Khurram Soomro, Amir Roshan Zamir y Mubarak Shah. «UCF101: A Dataset of 101 Human Actions Classes From Videos in The Wild». En: *CoRR* abs/1212.0402 (2012). arXiv: 1212.0402. URL: http://arxiv.org/abs/1212.0402 (véase página 56).

Recursos web (accedidos el 19/7/2019)

[22] *Adobe After Effects*. URL: https://helpx.adobe.com/es/after-effects/using/keyframe-interpolation.html (véase página 34).

[23] *Amazon Web Services*. URL: https://aws.amazon.com/es/machine-learning/amis (véase página 29).

[24] François Chollet y col. *Keras*. https://keras.io. 2015 (véanse páginas 19, 26).

[25] *Cisco AnyConnect VPN*. URL: https://www.cisco.com/c/en/us/products/security/anyconnect-secure-mobility-client/index.html (véase página 26).

[26] *Draw.io*. URL: https://www.draw.io (véase página 26).

[27] *Dropbox*. URL: https://www.dropbox.com (véase página 26).

[28] Python Software Foundation. *Python Language Reference, versión 3.6.7*. URL: http://www.python.org (véase página 26).

[29] Pye Jirsa. *Motion Blur Vs. Ghosting: The Difference Between These 2 Artifacts*. URL: https://www.slrlounge.com/motion-blur-vs-ghosting-the-difference-between-these-2-artifacts (véase página 35).

[30] *Microsoft Azure*. URL: https://azure.microsoft.com/es-es (véase página 29).

[31] BBC Mundo. *Qué es la cuarta revolución industrial (y por qué debería preocuparnos)*. URL: https://www.bbc.com/mundo/noticias-37631834 (véase página 12).

[32] *Online Charts*. URL: https://www.onlinecharttool.com (véase página 26).

[33] CC. OO. *XIX Convenio Estatal de Ingenierías y Oficinas de Estudios Técnicos*. URL: https://www.ccoo-servicios.es/html/41549.html (véase página 31).

[34] *Overleaf*. URL: https://es.overleaf.com (véase página 26).

[35] Prabhu. *Understanding Hyperparameters and its Optimisation techniques*. URL: https://towardsdatascience.com/understanding-hyperparameters-and-its-optimisation-techniques-f0debba07568 (véase página 62).

[36] *PyCharm*. URL: https://www.jetbrains.com/pycharm (véase página 26).

[37] Ministerio de ciencia, innovación y universidades. *Estatuto del personal investigador predoctoral en formación*. URL: https://www.boe.es/eli/es/rd/2019/03/01/103/dof/spa/pdf (véase página 31).

[38] *Visual Paradigm*. URL: https://www.visual-paradigm.com (véase página 26).

[39] Stuart Wilson. *Reparameterization*. MathWorld–A Wolfram Web Resource, creado por Eric W. Weisstein. URL: http://mathworld.wolfram.com/Reparameterization.html (véase página 53).

Índice alfabético

N

O

P

R

U

V