



ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

MÁSTER UNIVERSITARIO EN VISIÓN ARTIFICIAL

**TRABAJO FIN DE MÁSTER**

**Predicción de Fotogramas  
con Redes Neuronales Profundas**

Autor: Nuria Oyaga de Frutos

Tutor: José María Cañas Plaza

Cotutor: Inmaculada Mora Jiménez

Curso académico 2019/2020



©2020 Nuria Oyaga de Frutos

Esta obra está distribuida bajo la licencia de  
“Reconocimiento-CompartirIgual 4.0 Internacional (CC BY-SA 4.0)”

de Creative Commons.

Para ver una copia de esta licencia, visite  
<http://creativecommons.org/licenses/by-sa/4.0/> o envíe  
una carta a Creative Commons, 171 Second Street, Suite 300,  
San Francisco, California 94105, USA.

*“It’s kind of fun to do the impossible.”*

-Walt Disney-

*A mi rosa mas bonita,  
la más bella del rosal,  
gracias por enseñarme,  
lo que es vivir y luchar.*

# Agradecimientos

En primer lugar, quiero dar las gracias a mis tutores, Inmaculada y José María, por confiar en mí para el desarrollo de este trabajo y por su infinita paciencia. Además tengo que agradecerles su guía, apoyo y motivación durante todos estos meses. También me gustaría dar las gracias a Pablo, mi tutor durante mis primeras prácticas, por enseñarme a valorar todo el potencial que hay en mí y ser un gran consejero todo este tiempo.

Por otro lado, quiero agradecer a mis compañeros y amigos Vanessa, Mireya y David, por todos estos años de apoyo conjunto, desde el comienzo del grado hasta la finalización de este máster, y por todos los buenos y malos momentos que eso conlleva.

Gracias también a mis compañeros de máster que han compartido conmigo esta breve pero intensa etapa. En especial a Adrián, Borja, Leyre, y Clara, por seguir presentes en mi vida y hacer más llevadero todo el proceso. Además, gracias a todos esos buenos amigos que ya estaban de antes: Nazaret, Sara, Iván, Sarai, Miguel Ángel y Julia, entre otros, por su confianza, sus ratitos de desconexión y su incansable ánimo.

Gracias a toda mi familia porque a pesar de las dificultades siempre confiaron en que lo lograría y por el tremendo orgullo que sienten de todos y cada uno de mis logros. Por encima de todo quiero agradecer a mis padres, Iñigo y Pilar, y a mis hermanas, Elena y Arancha, que con sus ánimos, sus piques y su empuje han logrado que termine este trabajo con la mejor de mis actitudes.

Por último quiero dar las gracias a mi pareja, Gonzalo, por hacerme sentir una persona grande con mis metas logradas, por soportar el estrés y los malos ratos, especialmente estos últimos meses, por sacar mi mejor versión, por quererme como lo hace y por estar ahí siempre.

*Muchísimas gracias a todos!*

# Resumen

En los últimos años, la investigación en Visión Artificial para que las máquinas puedan percibir el mundo físico que les rodea, al igual que hace el ser humano mediante la vista, ha experimentado un gran desarrollo. En este aspecto, el uso de arquitecturas neuronales profundas cuyo aprendizaje está dirigido por conjuntos de datos representativos de la tarea a abordar, ha permitido mejorar las prestaciones de los algoritmos tradicionales. De las tres tareas que pueden realizar las estructuras neuronales, detección, clasificación y predicción, la predicción visual es la menos común. Además, dicha tarea tiene un largo recorrido en su investigación y una gran utilidad. Es por ello que este trabajo fin de máster aborda la tarea de predicción haciendo uso de secuencias de imágenes con un elemento móvil.

Se ha realizado un estudio sobre cómo distintas estructuras neuronales, de naturaleza recurrente y no recurrente, pueden utilizarse para realizar la predicción de fotogramas en una secuencia de vídeo, donde las correlaciones espacio-temporales entre imágenes son importantes. Para ello, se ha creado una serie de secuencias sintéticas formadas por fotogramas en los que un único píxel activo se desplaza siguiendo una determinada dinámica temporal: lineal, parabólica o sinusoidal. Así mismo, se han considerado los fotogramas en dos formatos distintos, uno modelado, que reduce toda la imagen a las posiciones ( $x$ ,  $y$ ) del píxel activo, y otro crudo, que representa la imagen como una matriz 2D de píxeles. Para obtener las secuencias sintéticas, se ha desarrollado un generador que permite crear ejemplos adaptados a un tipo de estudio concreto.

Con la realización de esta investigación se ha comprobado que, bajo determinadas hipótesis, es posible predecir satisfactoriamente la posición del objeto móvil en secuencias de imágenes. Además, la recurrencia en las estructuras neuronales aporta un gran valor para abordar este tipo de tarea, pues permite capturar la correlación temporal existentes en la secuencia.

# Índice general

Índice de figuras	VIII
Índice de tablas	XIII
Acrónimos	XIV
<b>1. Introducción</b>	<b>1</b>
1.1. Visión artificial . . . . .	1
1.2. Redes neuronales artificiales . . . . .	5
1.2.1. No recurrentes . . . . .	9
1.2.1.1. Perceptrón multicapa . . . . .	9
1.2.1.2. Redes convolucionales . . . . .	10
1.2.2. Recurrentes . . . . .	11
1.2.2.1. Redes LSTM . . . . .	12
1.2.2.2. Redes (ConvLSTM) . . . . .	13
1.3. Predicción con Redes Neuronales Artificiales . . . . .	14
1.4. Objetivos . . . . .	17
1.5. Metodología . . . . .	18
1.6. Estructura de la memoria . . . . .	19
<b>2. Estado del arte</b>	<b>21</b>
2.1. Estructuras neuronales para la predicción . . . . .	21
2.2. Predicción de valores . . . . .	23
2.3. Predicción en imágenes . . . . .	28
2.4. Infraestructura utilizada . . . . .	33
2.4.1. Lenguaje Python . . . . .	33
2.4.2. Biblioteca OpenCV . . . . .	33
2.4.3. Biblioteca Matplotlib . . . . .	35
2.4.4. Middleware neuronal Keras . . . . .	35
2.4.4.1. Modelos en Keras . . . . .	36

2.4.4.2. Capas en Keras . . . . .	37
2.4.5. Servidor Tamino . . . . .	39
<b>3. Generación de secuencias de vídeo sintéticas</b>	<b>40</b>
3.1. Propiedades del <i>dataset</i> . . . . .	41
3.1.1. Estructura del <i>dataset</i> . . . . .	42
3.1.2. Tipos de imágenes . . . . .	43
3.1.2.1. Imágenes modeladas . . . . .	43
3.1.2.2. Imágenes crudas . . . . .	44
3.1.3. Tipos de dinámicas . . . . .	45
3.1.3.1. Dinámica lineal . . . . .	45
3.1.3.2. Dinámica parabólica . . . . .	48
3.1.3.3. Dinámica sinusoidal . . . . .	50
3.2. Herramienta de generación . . . . .	52
3.2.1. Fichero de configuración . . . . .	52
3.2.2. Generación del <i>dataset</i> . . . . .	54
<b>4. Figuras de mérito y evaluación</b>	<b>59</b>
4.1. Figuras de mérito . . . . .	59
4.2. Metodología de evaluación . . . . .	63
<b>5. Predicción con imágenes modeladas</b>	<b>66</b>
5.1. Arquitectura no recurrente: Perceptrón Multicapa . . . . .	67
5.1.1. Influencia del número muestras . . . . .	68
5.1.2. Predicción con dinámicas lineales . . . . .	70
5.1.3. Predicción con dinámicas parabólicas . . . . .	72
5.1.4. Predicción con dinámicas sinusoidales . . . . .	75
5.1.5. Resumen de resultados . . . . .	77
5.2. Arquitectura recurrente: LSTM-1 . . . . .	78
5.2.1. Predicción con dinámicas lineales . . . . .	79
5.2.2. Predicción con dinámicas parabólicas . . . . .	80
5.2.3. Predicción con dinámicas sinusoidales . . . . .	81
5.2.4. Resumen de resultados . . . . .	84
5.3. Arquitectura recurrente: LSTM-4 . . . . .	85

5.3.1.	Aumento de neuronas . . . . .	86
5.3.2.	Aumento de capas . . . . .	87
5.3.3.	Predicción con dinámicas lineales . . . . .	88
5.3.4.	Predicción con dinámicas parabólicas . . . . .	88
5.3.5.	Predicción con dinámicas sinusoidales . . . . .	89
5.3.6.	Predicción con dinámica combinada . . . . .	90
5.3.7.	Predicción a largo plazo . . . . .	91
5.3.8.	Resumen de resultados . . . . .	92
5.4.	Comparativa global . . . . .	93
<b>6.</b>	<b>Predicción con imágenes crudas</b>	<b>94</b>
6.1.	Arquitectura no recurrente: Red convolucional . . . . .	95
6.1.1.	Influencia del número de muestras . . . . .	96
6.1.2.	Predicción con dinámicas lineales . . . . .	97
6.1.3.	Predicción con dinámicas parabólicas . . . . .	99
6.1.4.	Predicción con dinámicas sinusoidales . . . . .	102
6.1.5.	Resumen de resultados . . . . .	103
6.2.	Arquitectura recurrente: Convolucional + LSTM . . . . .	104
6.2.1.	Predicción con dinámicas lineales . . . . .	105
6.2.2.	Extensión gradual del punto activo . . . . .	106
6.2.3.	Resumen de resultados . . . . .	108
6.3.	Arquitectura recurrente: ConvLSTM-1 . . . . .	109
6.3.1.	Predicción con dinámicas lineales . . . . .	110
6.3.2.	Predicción con dinámicas parabólicas . . . . .	112
6.3.3.	Predicción con dinámicas sinusoidales . . . . .	112
6.3.4.	Resumen de resultados . . . . .	113
6.4.	Arquitectura recurrente: ConvLSTM-4 . . . . .	114
6.4.1.	Aumento de capas . . . . .	115
6.4.2.	Predicción con dinámicas lineales . . . . .	116
6.4.3.	Predicción con dinámicas parabólicas . . . . .	116
6.4.4.	Predicción con dinámicas sinusoidales . . . . .	117
6.4.5.	Predicción con dinámica combinada . . . . .	120
6.4.6.	Resumen de resultados . . . . .	120
6.5.	Comparativa global . . . . .	121

<b>7. Conclusiones</b>	<b>124</b>
7.1. Conclusiones . . . . .	124
7.2. Líneas futuras . . . . .	127
<b>Bibliografía</b>	<b>129</b>

# Índice de figuras

1.1.	Esquema de las relaciones entre la VA y otras áreas. Gráfico obtenido de [1].	2
1.2.	Detección de tumores cerebrales mediante VA. Imagen obtenida de [2]. . . . .	3
1.3.	Flujo de una aplicación <i>onboarding</i> . . . . .	4
1.4.	Control con drones del aforo en las playas mediante VA. Imagen obtenida de [3]. . . . .	4
1.5.	Seguimiento de personas con VA. Imagen obtenida de [4]. . . . .	5
1.6.	Ejemplo de aplicación de <i>AlexNet</i> . Imagen obtenida de [5]. . . . .	7
1.7.	Ejemplo de funcionamiento de <i>YOLO</i> . Imagen obtenida de [6]. . . . .	8
1.8.	Estructura de perceptrón multicapa con 4 entradas, 1 salida y 2 capas ocultas. Imagen obtenida de [7]. . . . .	9
1.9.	Estructura de CNN. Imagen obtenida de [8]. . . . .	10
1.10.	Estructura de RNN. Imagen obtenida de [9]. . . . .	11
1.11.	Estructura de LSTM. Imagen obtenida de [9]. . . . .	12
1.12.	Estructura de <i>ConvLSTM</i> . Imagen obtenida de [10]. . . . .	13
1.13.	Tareas de: (a) Clasificación, (b) Detección y (c) Predicción. . . . .	14
1.14.	Ejemplo de predicción en <i>trading</i> . Imagen obtenida de [11]. . . . .	15
1.15.	Predicción de acciones en secuencia de vídeo. Imagen obtenida de [12]. . . . .	16
1.16.	Predicción del fotograma en un videojuego. Imagen obtenida de [13]. . . . .	16
1.17.	Predicción en asistencia a la conducción. Imagen obtenida de [14]. . . . .	17
1.18.	Enfoque en espiral. . . . .	19
2.1.	Red LSTM. Imagen obtenida de [15]. . . . .	23
2.2.	Resultados comparativos de ARIMA, RNA y ARIMA+RNA. Imagen obtenida de [16]. . . . .	24
2.3.	Diagrama de flujo del estudio [17]. . . . .	25
2.4.	Estructura de RMNM-ANN [18]. . . . .	25
2.5.	Estructura propuesta por [19] con LSTM. . . . .	26
2.6.	Estructura de SocialLSTM [20]. . . . .	27
2.7.	Estructura propuesta por [21]. . . . .	28
2.8.	Estructuras propuestas en [13]. . . . .	29

2.9.	Estructura propuesta en [12]. . . . .	29
2.10.	Estructura de red MCnet [22]. . . . .	30
2.11.	Funcionamiento de PHD [23]. . . . .	30
2.12.	Estructura de red PHD [23]. . . . .	31
2.13.	Estructura propuesta en [24]. . . . .	32
2.14.	Resultados de estructura propuesta en [24]. . . . .	32
2.15.	Funciones de activación: (a) Lineal, (b) ReLu, (c) Tanh y (d) Softmax. . .	39
3.1.	Diagrama “caja negra” del generador. . . . .	40
3.2.	Vista general de un <i>dataset</i> . . . . .	41
3.3.	Estructura creada para el almacenamiento del <i>dataset</i> . . . . .	42
3.4.	Ejemplo de muestra cruda. . . . .	44
3.5.	Ejemplo de dinámica lineal. . . . .	46
3.6.	Ejemplo del caso con pendiente nula. . . . .	47
3.7.	Ejemplo de dinámica parabólica. . . . .	48
3.8.	Ejemplo de dinámica sinusoidal. . . . .	50
3.9.	Diagrama de flujo del generador. . . . .	54
4.1.	Ejemplo de gráfica con la distribución de distintas figuras de mérito y estadísticos asociados. . . . .	62
4.2.	Diagrama de flujo del evaluador. . . . .	63
5.1.	Estructura de MLP con 1 capa oculta y 10 neuronas para imágenes modeladas. . . . .	67
5.2.	Resultados de MLP con dinámica lineal, pendiente nula y altura fija en imágenes modeladas (100 muestras de <i>test</i> ). . . . .	68
5.3.	Resultados de MLP con dinámica lineal, pendiente nula y altura aleatoria en imágenes modeladas (100 muestras de <i>test</i> ). . . . .	69
5.4.	Resultados de MLP con dinámica lineal, pendiente nula y altura fija en imágenes modeladas (500 muestras de <i>test</i> ). . . . .	70
5.5.	Resultados de MLP con dinámica lineal de 1 DOF en imágenes modeladas (1000 muestras de <i>test</i> ). . . . .	71
5.6.	Resultados de MLP con dinámica lineal de 2 DOF en imágenes modeladas (1000 muestras de <i>test</i> ). . . . .	72

5.7. Resultados de MLP con dinámica parabólica de 1 DOF en imágenes modeladas (1000 muestras de <i>test</i> ) . . . . .	73
5.8. Resultados de MLP con dinámica parabólica de 2 DOF en imágenes (1000 muestras de <i>test</i> ) . . . . .	74
5.9. Resultados de MLP con dinámica parabólica de 3 DOF (1000 muestras de <i>test</i> ) . . . . .	74
5.10. Resultados de MLP con dinámica sinusoidal de 1 DOF (1000 muestras de <i>test</i> ) . . . . .	75
5.11. Resultados de MLP con dinámica sinusoidal de 1 DOF (10000 muestras de <i>test</i> ) . . . . .	76
5.12. Resultados de MLP con dinámica sinusoidal de 2 DOF (10000 muestras de <i>test</i> ) . . . . .	77
5.13. Estructura de LSTM-1 para imágenes modeladas. . . . .	78
5.14. Resultados de LSTM-1 con dinámica lineal de 1 DOF (1000 muestras de <i>test</i> ) . . . . .	79
5.15. Resultados de LSTM-1 con dinámica lineal de 2 DOF (1000 muestras de <i>test</i> ) . . . . .	80
5.16. Resultados de LSTM-1 con dinámica sinusoidal de 1 DOF (10000 muestras de <i>test</i> ) . . . . .	81
5.17. Resultados de LSTM-1 con dinámica sinusoidal de 2 DOF (10000 muestras de <i>test</i> ) . . . . .	82
5.18. Resultados de LSTM-1 con dinámica sinusoidal de 3 DOF (10000 muestras de <i>test</i> ) . . . . .	83
5.19. Resultados de LSTM-1 con dinámica sinusoidal de 4 DOF (10000 muestras de <i>test</i> ) . . . . .	84
5.20. Estructura de LSTM-4 para imágenes modeladas. . . . .	85
5.21. Resultados de LSTM de 50 neuronas con dinámica sinusoidal de 4 DOF (10000 muestras de <i>test</i> ) . . . . .	86
5.22. Comparación del error relativo al aumentar el número de capas LSTM (Sinusoidal, 4 DOF, 10000 muestras de <i>test</i> ) . . . . .	87
5.23. Resultados de LSTM-4 con dinámica lineal de 2 DOF (10000 muestras de <i>test</i> ) . . . . .	88

5.24. Resultados de LSTM-4 con dinámica parabólica de 3 DOF (10000 muestras de <i>test</i> ) . . . . .	89
5.25. Resultados de LSTM-4 con dinámica sinusoidal de 4 DOF (10000 muestras de <i>test</i> ) . . . . .	90
5.26. Resultados de LSTM-4 con dinámica combinada (10000 muestras de <i>test</i> ) . . . . .	91
5.27. Comparación del error relativo al aumentar el <i>gap</i> (Combinada, 10000 muestras de <i>test</i> ) . . . . .	92
6.1. Estructura de CNN para imágenes crudas. . . . .	95
6.2. Comparación del error relativo al aumentar el número de muestras de entrenamiento con CNN (Lineal, 1 DOF, 1000 muestras de <i>test</i> ) . . . . .	96
6.3. Resultados de CNN con dinámica lineal de 1 DOF (1000 muestras de <i>test</i> ) . . . . .	97
6.4. Resultados de CNN con dinámica lineal de 2 DOF (1000 muestras de <i>test</i> ) . . . . .	98
6.5. Resultados de CNN con dinámica lineal de 2 DOF (10000 muestras de <i>test</i> ) . . . . .	99
6.6. Resultados de CNN con dinámica parabólica de 1 DOF (10000 muestras de <i>test</i> ) . . . . .	100
6.7. Resultados de CNN con dinámica parabólica de 2 DOF (10000 muestras de <i>test</i> ) . . . . .	101
6.8. Resultados de CNN con dinámica parabólica de 3 DOF (10000 muestras de <i>test</i> ) . . . . .	101
6.9. Resultados de CNN con dinámica sinusoidal de 1 DOF (10000 muestras de <i>test</i> ) . . . . .	102
6.10. Resultados de CNN con dinámica sinusoidal de 2 DOF (10000 muestras de <i>test</i> ) . . . . .	103
6.11. Estructura de CNN+LSTM para imágenes crudas. . . . .	104
6.12. Resultados de CNN con dinámica lineal, pendiente nula y altura fija (100 muestras de <i>test</i> ) . . . . .	105
6.13. Resultados de CNN+LSTM con dinámica lineal, pendiente nula y altura fija (100 muestras de <i>test</i> ) . . . . .	106
6.14. Ejemplos de píxel: (a) Discreto y (b) Expandido. . . . .	107
6.15. Resultados de CNN+LSTM con píxel extendido en dinámica lineal, pendiente nula y altura fija (100 muestras de <i>test</i> ) . . . . .	108
6.16. Estructura de ConvLSTM-1 para imágenes crudas. . . . .	109

6.17. Resultados de ConvLSTM-1 con dinámica lineal de 1 DOF (1000 muestras de <i>test</i> ) . . . . .	110
6.18. Resultados de ConvLSTM-1 con dinámica lineal de 2 DOF (10000 muestras de <i>test</i> ) . . . . .	111
6.19. Estructura de ConvLSTM-4 para imágenes crudas. . . . .	114
6.20. Comparación del error relativo al aumentar el número de capas ConvLSTM (Sínusoidal, 4 DOF, 10000 muestras de <i>test</i> ) . . . . .	115
6.21. Resultados de ConvLSTM-4 con dinámica lineal de 1 DOF (10000 muestras de <i>test</i> ) . . . . .	116
6.22. Resultados de ConvLSTM-4 con dinámica parabólica de 2 DOF (10000 muestras de <i>test</i> ) . . . . .	117
6.23. Resultados de ConvLSTM-4 con dinámica sinusoidal de 2 DOF (10000 muestras de <i>test</i> ) . . . . .	118
6.24. Resultados de ConvLSTM-4 con dinámica sinusoidal de 3 DOF (10000 muestras de <i>test</i> ) . . . . .	119
6.25. Resultados de ConvLSTM-4 con dinámica sinusoidal de 4 DOF (10000 muestras de <i>test</i> ) . . . . .	119
6.26. Resultados de ConvLSTM-4 con dinámica combinada (10000 muestras de <i>test</i> ) . . . . .	120

# Índice de tablas

3.1. Ejemplo de imagen modelada. . . . .	44
5.1. Promedio del error relativo en <i>test</i> al evaluar el MLP con imágenes modeladas y distintas dinámicas (10000 muestras de <i>test</i> ). . . . .	77
5.2. Error relativo en la dinámica parabólica con MLP y LSTM-1 (1000 muestras de <i>test</i> ). . . . .	81
5.3. Promedio del error relativo en <i>test</i> al evaluar la red LSTM-1 con imágenes modeladas y distintas dinámicas (10000 muestras de <i>test</i> ). . . . .	84
5.4. Promedio del error relativo en <i>test</i> al evaluar la red LSTM-4 con imágenes modeladas y distintas dinámicas (10000 muestras de <i>test</i> ). . . . .	92
5.5. Comparativa del promedio de error relativo en las distintas redes para imágenes modeladas con las distintas dinámicas (10000 muestras de <i>test</i> ). .	93
6.1. Promedio del error relativo en <i>test</i> al evaluar la CNN con imágenes modeladas y distintas dinámicas (10000 muestras de <i>test</i> ). . . . .	104
6.2. Promedio del error relativo en <i>test</i> al evaluar la red CNN+LSTM con imágenes modeladas y distintas dinámicas (100 muestras de <i>test</i> ). . . . .	109
6.3. Error relativo en la dinámica parabólica con CNN y ConvLSTM-1 (10000 muestras de <i>test</i> ). . . . .	112
6.4. Error relativo en la dinámica sinusoidal con ConvLSTM-1 (10000 muestras de <i>test</i> ). . . . .	113
6.5. Promedio del error relativo en <i>test</i> al evaluar la red ConvLSTM-1 con imágenes modeladas y distintas dinámicas (10000 muestras de <i>test</i> ). . . . .	113
6.6. Promedio del error relativo en <i>test</i> al evaluar la red ConvLSTM-4 con imágenes modeladas y distintas dinámicas (10000 muestras de <i>test</i> ). . . . .	121
6.7. Comparativa del promedio de error relativo en las distintas redes para imágenes crudas con las distintas dinámicas (10000 muestras de <i>test</i> ). . . .	122



# **Capítulo 1**

## **Introducción**

En este capítulo se situará el Trabajo Fin de Máster en el marco existente en la actualidad, explicando el campo que comprende la Visión Artificial (VA), las Redes Neuronales Artificiales (RNA) y la tarea de predicción. Además se exponen los objetivos a alcanzar en el desarrollo del trabajo y la metodología empleada para ello.

### **1.1. Visión artificial**

Desde que apareció el primer ordenador en el planeta, el ser humano ha soñado con que estas máquinas puedan realizar tareas humanas, incluyendo su capacidad de pensar y tomar decisiones. Algunos de los ejemplos de estas tareas, propiamente humanas, se encuentran en el reconocimiento del habla, el entendimiento de las imágenes o la automatización de procesos. El campo de estudio que trata estos aspectos y se focaliza en el aprendizaje de las máquinas es la denominada Inteligencia Artificial (IA)[25], cuya presencia es cada vez mayor en la vida cotidiana.

Hoy en día el campo de la IA abarca muchas de las tareas que antes requerían de la acción humana; sin embargo, los resultados que pueden alcanzar los algoritmos en las máquinas no es siempre suficiente. A pesar de que la capacidad de cómputo que tienen algunas máquinas es mucho mayor que la del ser humano, aún no se les ha logrado dotar de otros aspectos importantes como la empatía o las emociones. Este hecho hace que la actividad del cerebro humano siga siendo necesaria en muchas de las situaciones a las que se pueden enfrentar las personas.

## CAPÍTULO 1. INTRODUCCIÓN

---

La IA se aplica en la solución de problemas como la identificación biométrica, el diagnóstico de enfermedades, los estudios de mercado o la robótica. Este campo hace, hoy en día, uso de desarrollos en el ámbito del *Machine Learning*, las RNA, el procesamiento del lenguaje natural, el procesamiento del habla o la VA, donde se pone el foco de este trabajo. Todas estas alternativas de desarrollo no se presentan de forma única, sino que las soluciones suelen aplicar varias de ellas, complementadas entre sí, para abordar distintos problemas.

La Visión Artificial (VA) es la disciplina que permite a las máquinas entender el mundo físico a través del procesamiento de las imágenes. Su objetivo es emular la actividad de los ojos y el cerebro del ser humano para explorar el mundo real, actuando en consecuencia con lo procesado. Este campo presenta distintas alternativas y se ve influido por distintas áreas afines que aportan distintos conceptos a la disciplina, representadas en la Figura 1.1. Las áreas representadas en color azul aportan distintos valores a la disciplina, y son necesarias para su buen funcionamiento. Áreas como las matemáticas o la física ayudan a la comprensión del mundo real, mientras que otras, como la técnica de imagen, ayudan a la adquisición de datos.

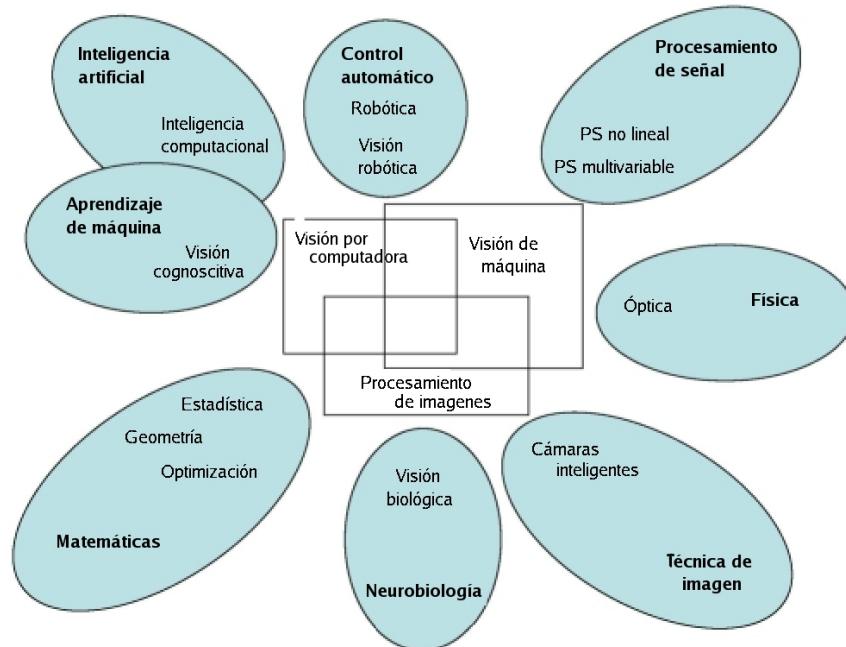


Figura 1.1: Esquema de las relaciones entre la VA y otras áreas. Gráfico obtenido de [1].

## CAPÍTULO 1. INTRODUCCIÓN

---

La VA tiene aplicaciones muy diversas en campos tan distintos como la medicina y la fabricación industrial. Además, en los últimos meses ha jugado un papel muy importante durante la pandemia del COVID-19 con soluciones que han permitido realizar controles de aforo, identificación a distancia o toma de temperatura sin contacto.

Uno de los ejemplos de aplicación más extendidos es el uso de algoritmos sobre imágenes de naturaleza médica (TACs, radiografías, ecografías...) como sistemas de ayuda al diagnóstico de enfermedades como el cáncer, mostrado en la Figura 1.2. La empresa Microsoft está desarrollando su propio proyecto de investigación, *InnerEye* [26], cuyo objetivo radica, precisamente, en facilitar el diagnóstico proporcionando mayor precisión en el mismo y la posibilidad de combinar la imagen con otro tipo de datos que complementen la información extraída a través de imagen médica.

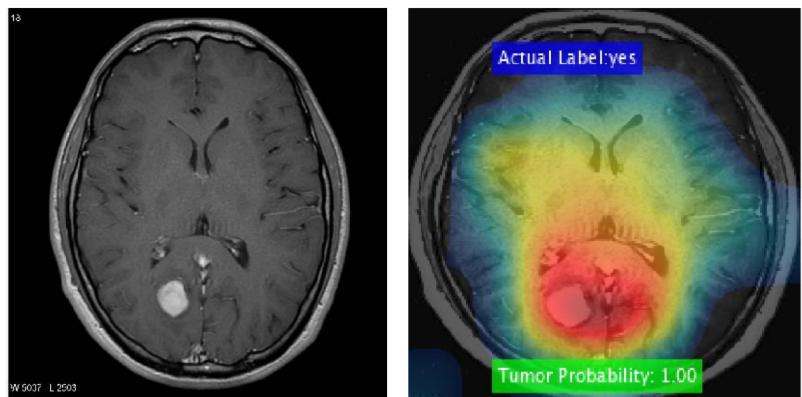


Figura 1.2: Detección de tumores cerebrales mediante VA. Imagen obtenida de [2].

Otro ejemplo de aplicación de la VA, esta vez en el mundo de la biometría, consiste en facilitar la identificación de las personas de forma remota. Mediante la extracción de la información de un documento de identidad y la petición de fotografías o vídeos al usuario, se desarrollan los llamados sistemas de *onboarding* (véase la Figura 1.3). Estos sistemas facilitan el proceso de alta en determinados productos, como una cuenta bancaria, o plataformas, como una casa de apuestas, que por su naturaleza necesitan de una identificación veraz. Durante los últimos meses, este sistema ha experimentado un gran impulso en su uso ya que facilita no tener contacto interpersonal, un factor determinante para frenar la expansión del COVID-19.

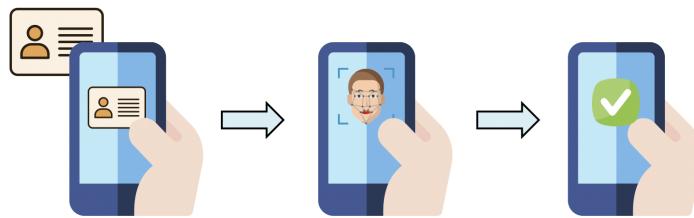


Figura 1.3: Flujo de una aplicación *onboarding*.

Un último ejemplo de aplicación de la VA se encuentra en la detección de personas u objetos. Esta tarea no se suele realizar de forma independiente, sino que se aúna con otras como el conteo o el seguimiento. En relación a la primera, el conteo de personas, si bien es una tarea bastante sencilla tras la detección, ha tomado un alto grado de importancia en los últimos meses. La necesidad de control de aforo en lugares abiertos, como las playas (véase Figura 1.4), y cerrados, como los aeropuertos, ha impulsado la presencia de estas tecnologías de VA en el día a día.



Figura 1.4: Control con drones del aforo en las playas mediante VA. Imagen obtenida de [3].

En cuanto al seguimiento, tras detectar a la persona en imágenes sucesivas es posible establecer la trayectoria que ésta ha seguido, según se muestra en la Figura 1.5. En este caso es posible introducir la predicción en una secuencia de imágenes, tema sobre el que versa este trabajo, como una solución a las pérdidas del objeto, es decir, los momentos en los que el objeto que estaba siendo seguido por una máquina no logra ser detectado

por la misma. Cuando el sistema no detecta el objeto, por ejemplo por una oclusión, el sistema puede realizar una estimación de la posición en la que el objeto puede aparecer en instantes sucesivos. Ello permite reducir la zona de una nueva búsqueda para volver a detectar el objeto y dar continuidad al seguimiento.

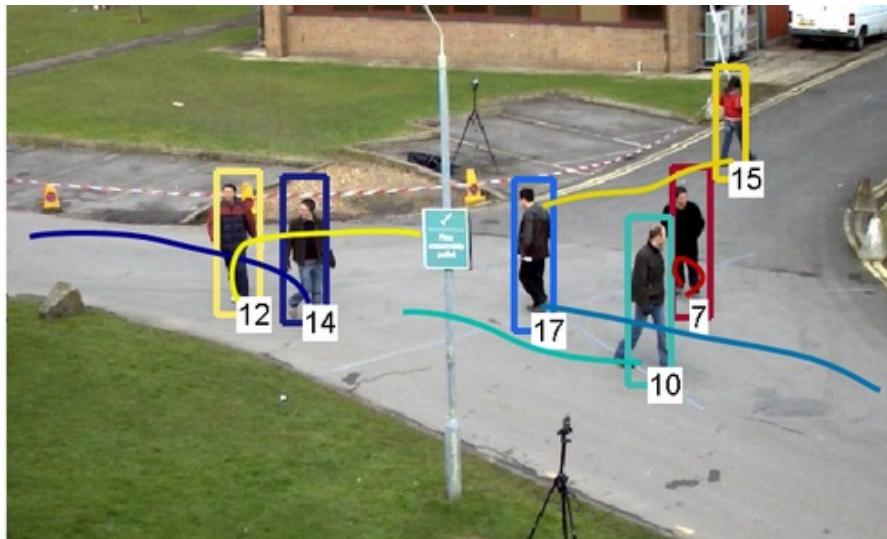


Figura 1.5: Seguimiento de personas con VA. Imagen obtenida de [4].

## 1.2. Redes neuronales artificiales

El aprendizaje máquina, *Machibe Learning*, se utiliza en muchos campos, incluyendo la VA. El proceso de aprendizaje de las máquinas puede realizarse de forma supervisada o no supervisada. Si cuando se proporcionan las muestras al algoritmo para que aprenda, además de las características, se le proporciona una etiqueta que indica la salida deseada del algoritmo, se está ante un proceso supervisado. Si, por el contrario, la única entrada que alimenta el algoritmo son las características crudas, el algoritmo de aprendizaje se considera no supervisado. Los algoritmos de aprendizaje supervisado abordan tareas de clasificación y estimación, cuyo principal objetivo es encontrar la función que mapea entradas con salidas. Por otro lado, los algoritmos no supervisados abordan tareas de agrupamiento, creando grupos de ejemplos con características similares entre sí y distintas a las del resto.

Una de las ramas de aprendizaje máquina más exitosas en los últimos años es la del

## CAPÍTULO 1. INTRODUCCIÓN

---

aprendizaje profundo, que maneja Redes Neuronales Artificiales (RNA). Las RNA[27] son modelos matemáticos que tratan de emular el funcionamiento del cerebro humano mediante la interconexión de neuronas distribuidas en varias capas. El objetivo principal de estos modelos es conseguir que las máquinas sean capaces de aprender a través de su entrenamiento con un conjunto de datos. Gracias al entrenamiento se consigue que estas redes puedan ofrecer resultados satisfactorios cuando se les presenta un ejemplo que no han visto antes, lo que indica que la red ofrece buena capacidad de generalización.

El proceso de entrenamiento o aprendizaje de las RNA, proceso en el que se fijan los pesos de las distintas conexiones entre las neuronas de cada capa, se afronta como un problema de minimización de una función de coste o función de *loss*. Para ajustar los pesos de la red se emplea la propagación del error “hacia atrás” haciendo uso de un método de cálculo del gradiente una vez la red se ha inicializado con una serie de pesos. Al introducir un ejemplo en la estructura, éste se propaga desde la primera capa a las sucesivas hasta generar una salida. Esta salida se compara con la salida deseada y se obtiene un error, que es trasladado desde la capa de salida hasta la de entrada, permitiendo el ajuste de los pesos en consecuencia. Es decir, se puede ver el entrenamiento como una aplicación directa de la teoría de optimización.

Existen redes monocapa, compuestas por una única capa en la que las neuronas crean conexiones laterales para conectarse entre ellas en la propia capa, como la red de Hopfield. Sin embargo, en este trabajo se utilizan las redes multicapa, compuestas por múltiples capas conectadas entre sí. Cuando se trabaja con varias capas, éstas se clasificadas en tres tipos:

- **Capa de entrada:** Formada por las neuronas que introducen los datos a la red.
- **Capas ocultas:** Las capas intermedias, que realizan diferentes transformaciones. Las transformaciones que realizan estas capas, no tienen por qué coincidir dentro de una misma red.
- **Capa de salida:** Constituida por las neuronas que se corresponden con la salida.

La introducción del aprendizaje profundo, con las RNA, en el mundo de la VA ha supuesto una gran revolución, comenzada en el año 2012. Las RNA han permitido descubrir relaciones no lineales que no era posible captar con los algoritmos tradicionales. Las

## CAPÍTULO 1. INTRODUCCIÓN

---

RNA han introducido un alto grado de mejora en las aplicaciones que utilizan las técnicas de VA tradicional, como los filtros de *Kalman*. Se consigue un incremento en la calidad de las aplicaciones sustituyendo, además, un gran esfuerzo humano para la obtención de descriptores por un procesamiento de máquina mucho más rápido y eficaz. Otro factor que hace de las RNA una buena alternativa a las técnicas tradicionales es que el ajuste de una aplicación en un entorno nuevo no suele conllevar un desarrollo muy complejo, se puede conseguir una buena adaptación con un nuevo reentrenamiento específico.

El aprendizaje profundo ha sido aplicado con éxito en una gran variedad de problemáticas, de distinta naturaleza, favoreciendo su alto grado de implementación en la actualidad. Un ejemplo de aplicación de RNA, en un contexto distinto a la VA, es reconocimiento del lenguaje. Aplicaciones como *Siri* o *Alexa* hacen uso de esta tecnología para “escuchar” al ser humano y realizar acciones como poner una canción o llamar a una persona.

Dentro de la VA un ejemplo de la implementación de las RNA en la tarea de clasificación es la conocida como *AlexNet* [5], cuyo ejemplo de aplicación se muestra en la Figura 1.6. Esta red fue desarrollada con el objetivo de clasificar las imágenes existentes en el conjunto *ImageNet*, con 15 millones de imágenes pertenecientes a 22000 categorías.



Figura 1.6: Ejemplo de aplicación de *AlexNet*. Imagen obtenida de [5].

Para la detección, una de las estructuras con mejores prestaciones y más utilizada, es la denominada *YOLO* [6], cuyo ejemplo de funcionamiento se muestra en la Figura 1.7. Esta estructura destaca en la tarea de detección visual por rapidez y buen funcionamiento en tiempo real, gracias a la filosofía de buscar una única vez en la imagen. Su gran éxito

ha hecho que la estructura haya ido evolucionando con el tiempo, dando lugar a distintas versiones que proporcionan diferentes prestaciones al usuario.

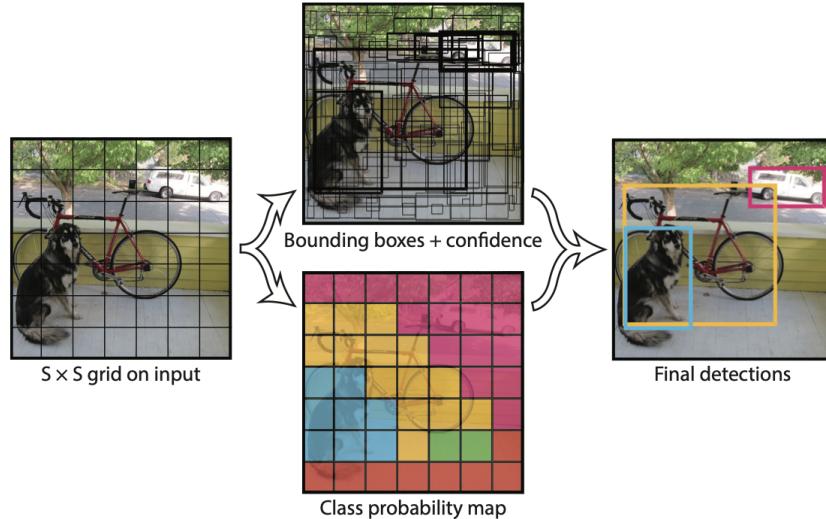


Figura 1.7: Ejemplo de funcionamiento de *YOLO*. Imagen obtenida de [6].

Por último, existen distintos *middleware* neuronales que proporcionan al desarrollador distintas herramientas para el entrenamiento y ejecución de las redes. Algunos de estos *middleware* son:

- *Keras*
- *Tensorflow*, desarrollado por *Google*
- *Darknet*
- *Pytorch*, desarrollado por *Facebook*
- *Caffe*

Para la investigación de este proyecto se han considerado dos tipos de redes, recurrentes y no recurrentes, cuya principal divergencia es la persistencia del conocimiento a lo largo del tiempo. Entre ellas hay diferencias en la arquitectura de la red, y por tanto el proceso de aprendizaje. Ambos tipos se detallan a continuación.

### 1.2.1. No recurrentes

Las redes neuronales no recurrentes no utilizan la persistencia a lo largo del tiempo, ya que las salidas o procesamientos intermedios únicamente alimentan a la siguiente capa y no vuelven a las entradas de la propia red. Si se asemeja el funcionamiento de estas redes con la forma de pensar del ser humano, es como si cada vez que éste leyera un texto, procesase cada palabra por separado, ignorando el contexto en la que se encuentra [9].

Este tipo de redes son las más utilizadas para abordar tareas de clasificación o detección en las que sólo es necesario procesar una imagen puntual para extraer información de la misma. En este trabajo se plantea el uso de dos estructuras no recurrentes: el perceptrón multicapa, para las posiciones, y las redes convolucionales, para las imágenes.

#### 1.2.1.1. Perceptrón multicapa

Un *MultiLayer Perceptron* (MLP)[28] es un conjunto de varios perceptrones que se apilan en varias capas para resolver problemas complejos. El perceptrón, o neurona, es una estructura simple de clasificación binaria que fue propuesto por Cornell Frank Rosenblat en el año 1958. Se trata de una máquina de aprendizaje muy simple que toma una serie de entradas ponderadas por un peso y genera una salida de binaria: “0” ó “1”. Sin embargo, el poder de estas estructuras se acentúa cuando se combinan varias de ellas, organizadas en distintas capas. En la Figura 1.8 se muestra un ejemplo de una estructura MLP.

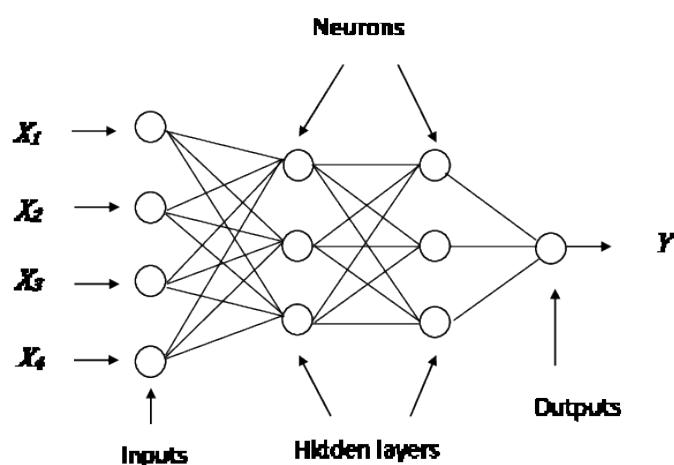


Figura 1.8: Estructura de perceptrón multicapa con 4 entradas, 1 salida y 2 capas ocultas. Imagen obtenida de [7].

Al tratarse de una red multicapa, ésta tiene una capa de entrada, un conjunto de capas ocultas y una capa de salida. En el MLP, cada una de las neuronas de una capa envía información a las neuronas de la capa siguiente, lo que hace que la complejidad se eleve exponencialmente con el aumento de capas y de neuronas por capa oculta. Con esta estructura se pueden aproximar relaciones no lineales entre los datos de entrada y salida.

### 1.2.1.2. Redes convolucionales

Al trabajar con imágenes éstas se deben de tratar como un conjunto de píxeles en entre los que existe una determinada correlación espacial. Las *Convolutional Neural Networks* (CNN) [8] son un tipo especial de RNA que pretende obtener correlaciones espaciales invariantes a ciertas transformaciones de la entrada. Existen CNN que aplican la convolución sobre una dimensión, en un vector, sobre dos dimensiones, en una imagen, y sobre tres dimensiones, en un volumen. Para este trabajo se utilizan redes 2D, cuya estructura se representa en la Figura 1.9. El origen de este tipo de redes se sitúa en la solución a un problema común en los MLP. Este problema tiene su origen en que, en las estructuras de tipo MLP, todas las capas están conectadas entre sí, obviando posibles relaciones espaciales en la imagen.

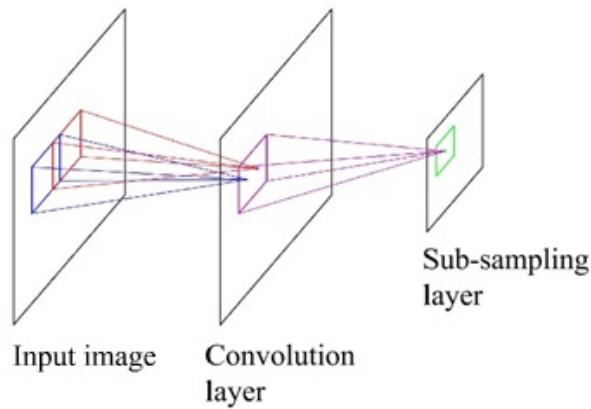


Figura 1.9: Estructura de CNN. Imagen obtenida de [8].

Para formar una CNN se utilizan dos tipos de capas distintas conectadas entre sí. En primer lugar se emplea una capa convolucional que realiza una operación de convolución espacial sobre la imagen. Esta convolución se realiza, en cada neurona, con un filtro local cuyos pesos se modifican en durante el aprendizaje. A esta capa le sigue siempre una de submuestreo o agrupación, encargada de generar las características invariantes mediante

el cálculo de estadísticas de las activaciones obtenidas a partir de un pequeño campo receptivo. Al contrario que en las MLP, cada neurona en una capa oculta se conecta a un pequeño campo de la capa anterior, campo receptivo local, permitiendo mantener esa información espacial. Por otro lado, al igual que en el MLP, las neuronas se organizan en varias capas consecutivas que dan lugar a distintos mapas de características. En resumen, cada mapa de características se conecta a un campo receptivo local. Además, en un mismo mapa de características se comparte el mismo parámetro de pesos, filtro o *kernel*.

En definitiva, las CNN realizan un procesamiento que trata de imitar al cortex visual del ojo humano, con el objetivo de identificar características en la entrada que permitan “ver” a la máquina.

Por último, en este trabajo no se emplean imágenes aisladas, se utilizan secuencias de vídeo que centran el elemento a procesar en un fotograma, una imagen que es tratada como un conjunto de píxeles en el que las relaciones espacio-temporales de los mismos importan.

### 1.2.2. Recurrentes

Las *Recurrent Neural Networks* (RNN) [9], al contrario que las anteriores, sí que introducen el concepto de la persistencia en su forma de aprendizaje. Para lograr esto se introduce un bucle en la estructura que permite a los pasos siguientes de la red utilizar cierta información de los anteriores, de forma que ahora la información sí puede progresar de las siguientes capas a las anteriores, las precedentes en el flujo de datos. Para comprender mejor su funcionamiento, en la Figura 1.10 se muestra un ejemplo de esta estructura, tanto de forma compacta como desenrollada.

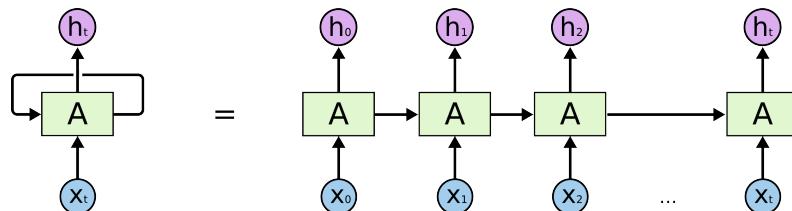


Figura 1.10: Estructura de RNN. Imagen obtenida de [9].

Una RNN puede ser vista como una misma red estática reproducida varias veces, de forma que cada copia le pasa un mensaje a la siguiente. La naturaleza de estructura en cadena hace que este tipo de redes esté íntimamente relacionado con listas y datos secuenciales, tema sobre el que versa este trabajo.

### 1.2.2.1. Redes LSTM

Las redes *Long Short-Term Memory* (LSTM) [9] son un tipo concreto de RNN que trata de solucionar el problema de las dependencias a largo plazo. Dependiendo de la naturaleza del problema a tratar, es posible que se necesite de un contexto de mayor o menor alcance. Cuando la cantidad de características previas utilizadas no es muy grande, las RNN tradicionales ofrecen buenas prestaciones. No obstante, según se amplía esta cantidad de características utilizadas, las prestaciones decrecen en consecuencia, algo que no ocurre con las LSTM. La estructura de este tipo de redes, mostrada en la Figura 1.11, se caracteriza por tener cuatro capas en una única celda de memoria, equivalente a las neuronas en las redes no recurrentes.

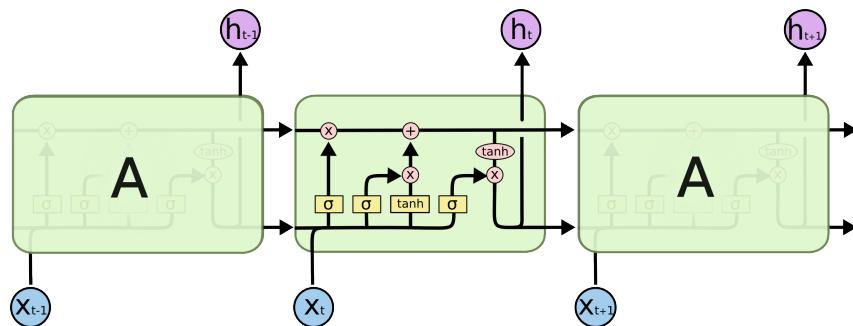


Figura 1.11: Estructura de LSTM. Imagen obtenida de [9].

La clave de las LSTM se encuentra en la parte superior de la celda, su estado. Esta parte es la encargada de transportar las características de una celda a otra, con las modificaciones reguladas por tres estructuras llamadas puertas. Las puertas están compuestas por una capa sigmoidea, que genera valores entre cero y uno describiendo cuánto de cada componente de un vector de características se debe dejar pasar, seguida de una operación de multiplicación puntual. Para la modificación de los datos corren de una celda a otra se realizan cuatro pasos:

1. Decidir las características antiguas a olvidar mediante la “puerta del olvido”.

2. Decidir las características nuevas a almacenar mediante la combinación de la “puerta de entrada”, responsable qué valores se van a actualizar, con una capa sigmoidea que crea los nuevos valores a tomar.
3. Actualizar las características y combinarlas en un nuevo estado que se transmitirá a la siguiente celda.
4. Obtener la salida de la celda ( $h$ ) como una versión filtrada del estado de la celda. Para ello se combina la “puerta de salida”, que decide qué parte del estado de celda se considera como salida, con el propio estado de celda ponderado con una  $tanh$  para que los valores se encuentren entre -1 y 1.

Con este funcionamiento se consigue mantener la memoria tanto a corto plazo como a largo plazo, proporcionando buenos resultados en tareas de predicción haciendo uso de secuencias.

### 1.2.2.2. Redes (ConvLSTM)

Las redes (ConvLSTM) [29] son un caso especial de las redes LSTM que ponen su foco en las secuencias de imágenes. La Figura 1.12 muestra la estructura de estas redes.

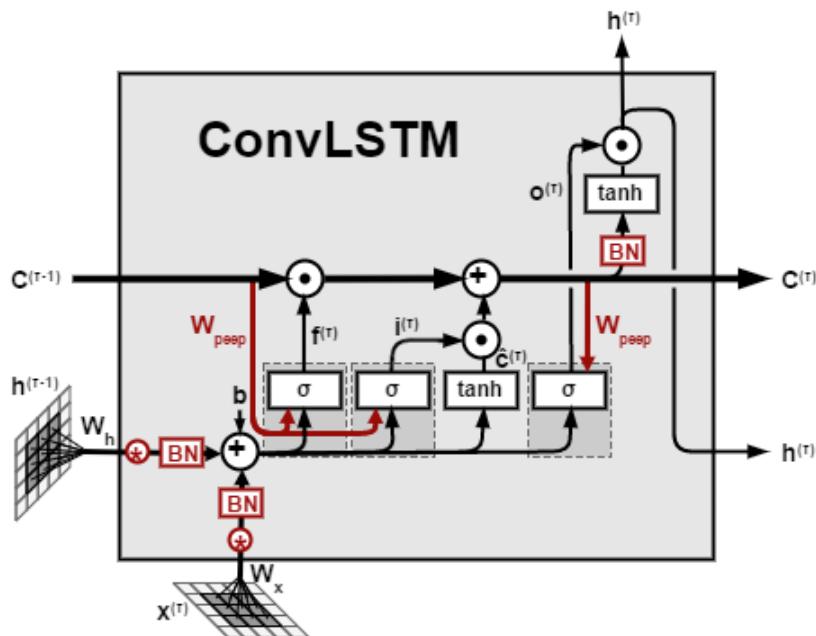


Figura 1.12: Estructura de *ConvLSTM*. Imagen obtenida de [10].

La característica principal de este tipo de redes es la capacidad de mantener las relaciones temporales, al igual que las LSTM, añadiendo las relaciones espaciales, propio de las CNN. Para ello, se sustituyen todas las operaciones de multiplicación existentes en una celda LSTM, representada en la Figura 1.11, por operaciones de convolución, manteniendo las dimensiones de entrada. Este hecho consigue que esta estructura sea considerada la más idónea en tareas de predicción haciendo uso de secuencias de vídeo.

### 1.3. Predicción con Redes Neuronales Artificiales

En este trabajo se plantea un proceso de aprendizaje supervisado con secuencias de imágenes, es decir, de VA. Los algoritmos supervisados utilizados en este campo típicamente pueden abordar tareas de tres tipologías diferentes: clasificación, detección y predicción. Los dos primeros tipos abarcan la gran mayoría de soluciones actuales en el mercado. En cuanto a la predicción, actualmente se está investigando mucho y hay numerosas soluciones satisfactorias en distintos ámbitos, debido a su gran potencial.

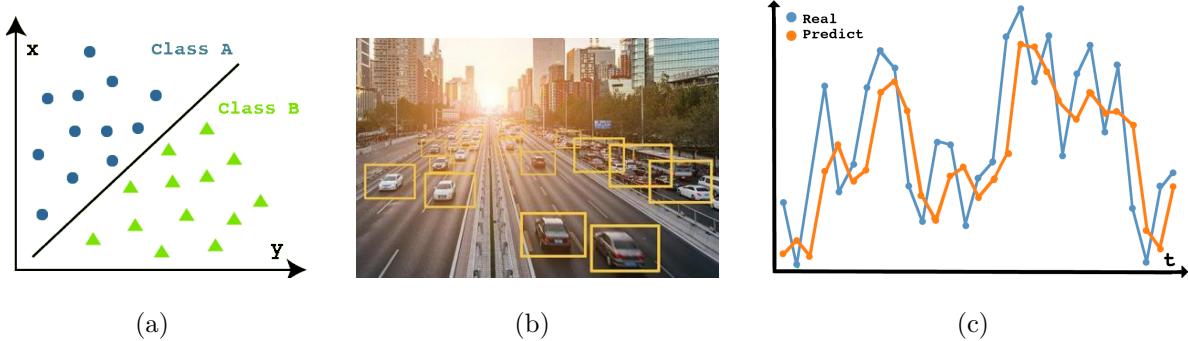


Figura 1.13: Tareas de: (a) Clasificación, (b) Detección y (c) Predicción.

En las tareas de clasificación, el objetivo principal de los algoritmos es establecer la pertenencia de un ejemplo a una clase de un conjunto previamente definido. En la detección, el algoritmo busca un patrón en la imagen y establece la posición del objeto dentro de la misma. Por último, la predicción se suele afrontar como una tarea de regresión, en el que el valor de aproximación que se obtiene a la salida no tiene ningún límite.

Muchos de los ejemplos de predicción actuales están enfocados a valores económicos, como la acción el Bolsa, la evolución de los precios de recursos naturales o las ventas

## CAPÍTULO 1. INTRODUCCIÓN

---

futuras de una compañía. Todos ellos toman como entrada un conjunto de datos numéricos y su salida tiene la misma naturaleza. En la Figura 1.14 se presenta una de estas aplicaciones para la predicción del precio de las acciones en actividades de *trading*.

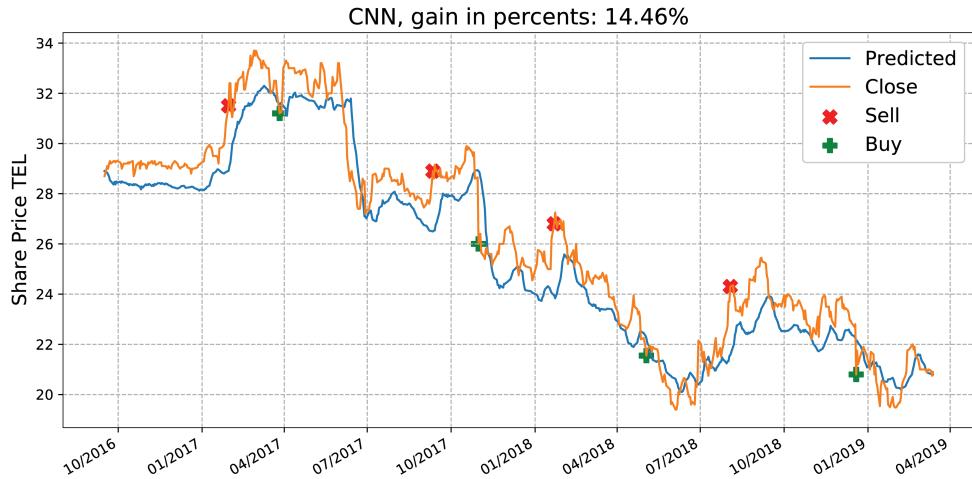


Figura 1.14: Ejemplo de predicción en *trading*. Imagen obtenida de [11].

La predicción de fotogramas por una máquina es algo que puede resultar muy útil en contextos como la videovigilancia o la programación de autómatas, y que cada vez tiene mayor interés en el mundo tecnológico y, en concreto, en el campo de la Visión Artificial. En los últimos años se ha producido un gran progreso de la predicción en el ámbito de secuencias numéricas. En contraste, la tarea de predicción sobre las imágenes de un vídeo tiene aún un amplio campo de exploración.

Respecto al uso de secuencias de imágenes, un ejemplo de aplicación es la predicción de la acción futura en una escena, cuyo ejemplo se muestra en la Figura 1.15. Este tipo de aplicación puede resultar útil, por ejemplo en la mejora de la interacción entre un robot y una persona. Al analizar el movimiento de la persona frente a la cámara del robot, éste puede predecir el próximo movimiento del humano y anticiparse para ofrecer una mejor experiencia al usuario.



Figura 1.15: Predicción de acciones en secuencia de vídeo. Imagen obtenida de [12].

Otro ejemplo de aplicación de la predicción con secuencias de vídeo se focaliza en el ámbito de los videojuegos. Gracias a la tecnología de predicción se mejora la experiencia del jugador, mostrando una situación del juego acorde a los movimientos que éste realiza con los elementos de control (véase la Figura 1.16). De esta forma se consigue que el juego siga una dinámica más fluida y se adapte al nivel del propio jugador, evitando niveles imposibles y el consecuente abandono de la partida.

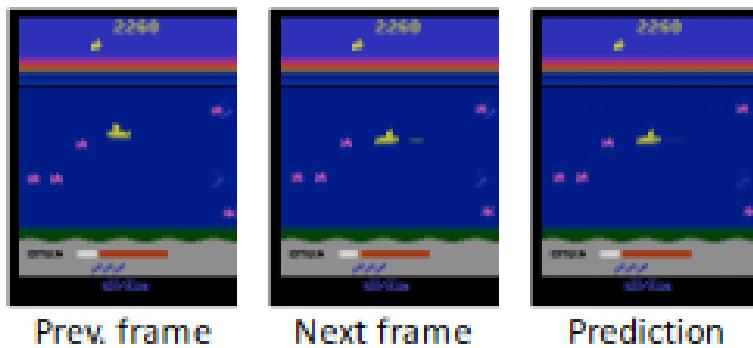


Figura 1.16: Predicción del fotograma en un videojuego. Imagen obtenida de [13].

Un último ejemplo de aplicación a la orden del día, mostrado en la Figura 1.17 es la asistencia a la conducción. Son muchas las situaciones que hacen que un conductor tenga que planificar sus decisiones en un corto espacio de tiempo. El procesamiento de las imágenes tomadas por el vehículo para predecir lo que puede pasar en el entorno que rodea al mismo, puede ayudar en la toma de decisiones por parte del conductor. Esta mejora en la conducción acerca, a su vez, la posibilidad de que en un futuro los vehículos no necesiten de la acción humana.



Figura 1.17: Predicción en asistencia a la conducción. Imagen obtenida de [14].

## 1.4. Objetivos

El objetivo principal de este trabajo es diseño y el análisis de distintas redes neuronales como predictores visuales mediante el procesamiento de secuencias de vídeo. Se propone investigar las prestaciones de varias estructuras de redes neuronales con secuencias de fotogramas que muestran por diferentes dinámicas de movimiento más o menos complejas de predecir.

Para cumplir con el objetivo se han considerado dos tipos de imágenes: las crudas, imágenes normales que representan cada fotograma como una matriz 2D donde cada píxel tiene un valor de luminancia, y las modeladas, donde se asume que hay un único píxel blanco sobre fondo negro que se va moviendo a lo largo de la secuencia. Cada imagen modelada es un resumen de la posición ( $x, y$ ) de ese píxel activo, de modo que la secuencia de fotogramas se resume en un vector de posiciones a lo largo del tiempo.

El anterior objetivo general ha sido articulado en los siguientes 4 subobjetivos:

1. **Desarrollo *software*** del código necesario para ejecución de redes neuronales, conexión con imágenes de entrada, muestra de sus inferencias y grabación de sus estimaciones. También se programará el *software* para la evaluación automáticas de varias figuras de mérito.
2. **Creación de las bases de datos** con las que entrenar las distintas redes propuestas y evaluar sus prestaciones. Los conjuntos se crearán bajo la premisa

de un píxel activo que se mueve en la imagen siguiendo una dinámica determinada. Se programará un generador de fotogramas sintéticos.

3. **Estudio y evaluación** de las prestaciones de varias redes no recurrentes y de varias redes recurrentes para predecir en secuencias de **imágenes modeladas**.
4. Estudio y evaluación de las prestaciones de varias redes no recurrentes y de varias redes recurrentes para predecir en secuencias de **imágenes crudas**.

### 1.5. Metodología

Con el propósito de alcanzar los objetivos establecidos se han utilizado una serie de herramientas y métodos que han favorecido un seguimiento fluido y dinámico de los avances en el proyecto.

La herramienta principal para mantener una interacción continua con los tutores ha sido el establecimiento de reuniones, generalmente semanales, en las que actualizar el trabajo realizado durante la semana, poner en común y discutir los resultados obtenidos, consultar algunas dudas y establecer los pasos a seguir durante la siguiente semana. De forma adicional, para permitir un seguimiento durante la semana a todos los miembros del grupo y dejar patentes los avances obtenidos, se ha empleado un repositorio de GitHub<sup>1</sup> donde se dispone del código y una bitácora<sup>2</sup> en la que se presentan esos avances.

En cuanto a la metodología de la realización del trabajo, se ha utilizado un enfoque en espiral, como el representado en la Figura 1.18. Se definen cuatro fases principales, que son recorridas en forma de caracola:

1. Fase de planificación en la que se marcan los objetivos a alcanzar.
2. Fase de análisis de los posibles problemas a afrontar durante el desarrollo.
3. Fase de desarrollo, que generalmente coincide con el entrenamiento y la obtención de resultados de las distintas redes.

---

<sup>1</sup><https://github.com/RoboticsLabURJC/2017-tfm-nuria-oyaga>

<sup>2</sup><https://roboticslaburjc.github.io/2017-tfm-nuria-oyaga/>

4. Fase de evaluación, en la que se analizan los resultados y se toman decisiones sobre los pasos a seguir según las conclusiones.

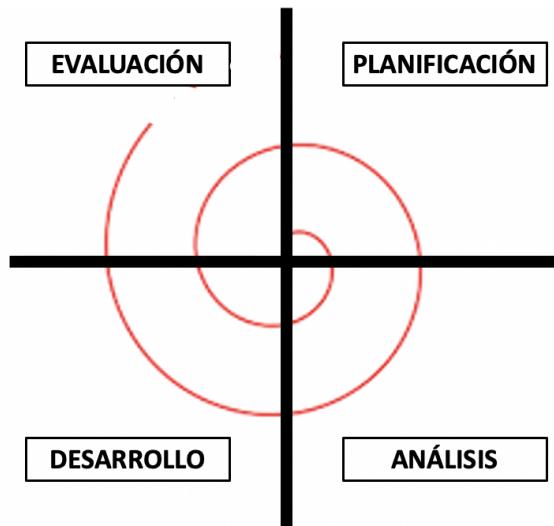


Figura 1.18: Enfoque en espiral.

Aunque no se establece un número fijo de iteraciones sobre este esquema, generalmente cada iteración coincide con el tiempo transcurrido entre reunión y reunión, aumentando el alcance del trabajo.

## 1.6. Estructura de la memoria

En el **Capítulo 1, Introducción**, se sitúa el trabajo en el marco actual de la tecnología, la IA y la sociedad en general. Se establece el contexto de la VA y de la tarea de predicción, sobre la que gira este trabajo. Además, se explican las RNA y se describen las principales estructuras consideradas en el desarrollo. También se establecen las metas que se pretenden alcanzar con el mismo.

En el **Capítulo 2, Estado del arte**, se exponen algunas líneas de investigación referentes a la predicción, tanto en datos numéricos como en imágenes, que proporcionaron avances en dicho campo. Posteriormente se explica la infraestructura, tanto *software* como *hardware*, empleada durante el desarrollo del trabajo.

## CAPÍTULO 1. INTRODUCCIÓN

---

En el **Capítulo 3, Generación de secuencias de vídeo sintéticas**, se explican los tipos de imágenes utilizados, modeladas y crudas, las dinámicas que rigen el movimiento del píxel en la imagen durante una secuencia, y la herramienta desarrollada para la generación de los conjuntos de datos.

En el **Capítulo 4, Figuras de mérito y evaluación**, se presentan las métricas calculadas para la evaluación de las distintas estructuras neuronales y la comparación entre ellas. Además, se describe la metodología de evaluación desarrollada y la forma de presentar los resultados obtenidos.

En el **Capítulo 5, Predicción con imágenes modeladas**, se exponen todos los experimentos realizados con imágenes modeladas y las diferentes dinámicas, mostrando sus resultados y extrayendo conclusiones sobre los mismos.

En el **Capítulo 6, Predicción con imágenes crudas**, se presentan todos los experimentos realizados con imágenes crudas y las distintas dinámicas, mostrando sus resultados y extrayendo conclusiones sobre los mismos.

Por último, el **Capítulo 7, Conclusiones**, resume las conclusiones obtenidas en el trabajo y establece un posible plan de actuación futuro para continuar con la investigación en el tema abordado.

# Capítulo 2

## Estado del arte

En este capítulo se presenta el estado del arte sobre la predicción mediante redes neuronales de aprendizaje profundo. Se exponen distintos trabajos enfocados en las estructuras neuronales para la predicción y la realización de esta tarea con secuencias de valores, así como otros que ponen el foco en la predicción con secuencias de imágenes en un vídeo. También se presenta la infraestructura *software* y *hardware* utilizada en el desarrollo del trabajo.

### 2.1. Estructuras neuronales para la predicción

Antes de llegar al conocimiento y el uso de las RNN y las LSTM, estructuras dinámicas utilizadas en este trabajo, se desarrollaron varias investigaciones que permitían, en cierta forma, predecir un elemento en una secuencia a partir de elementos previos de la misma secuencia haciendo uso de estructuras de naturaleza estática, es decir, no recurrentes.

En 1990, Elman realizó un estudio sobre la estructura temporal con el que logró dotar de cierta memoria a las estructuras de aprendizaje [30]. En este artículo, se llega a importantes conclusiones que permitieron avanzar en el desarrollo de estructuras dinámicas que faciliten la tarea de predicción:

- Algunos problemas pueden cambiar el enfoque por el hecho de incluir en ellos la variable temporal.

- La señal de error variable en el tiempo puede ser de ayuda para establecer la estructura temporal.
- Aumentar las dependencias secuenciales en una tarea no tiene por qué implicar un peor rendimiento.
- La representación tiempo-memoria depende de la tarea que se quiera llevar a cabo.
- Las representaciones de los datos no tienen por qué tener una estructura concreta.

El trabajo de Elman fue muy utilizado en estudios posteriores que se centran en la predicción con secuencias temporales mediante estructuras dinámicas. Por ejemplo, en el año 1996, Koskela *et al.* [31] desarrollaron un trabajo en el que se comparaban tres tipos de redes para la tarea de predicción, obteniendo resultados muy similares entre las redes Elman y los MLP. Para el uso de los MLP se modifica su estructura introduciendo las entradas retardadas de manera sucesiva, logrando así obtener un comportamiento dinámico. Los resultados obtenidos en el estudio muestran que la el algoritmo de aprendizaje es un factor más importante que el modelo de red neuronal utilizado. Además, a pesar de ser mucho más sencillo, el modelo MLP obtiene un rendimiento muy bueno en varias de las series, reduciendo sustancialmente el tiempo de entrenamiento.

También en los años 90, Kevin J.Lang *et al.* [32] introdujeron una nueva forma para dotar de memoria a las estructuras: las Redes Neuronales de Retardo Temporal (TDNN). Se trata de redes multicapa realimentadas cuyas neuronas ocultas y de salida se replican en el tiempo, de forma que las salidas de una capa se almacenan durante varios instantes de tiempo y, posteriormente, alimentan a la siguiente capa. La topología de este tipo de redes se incluye dentro de un MLP, donde cada conexión sináptica está representada por un filtro de respuesta finita.

Finalmente, se llega a las RNN. Estas redes fueron inventadas en la década de los 80, con el desarrollo por John Hopfield de una red que lleva su mismo nombre. Este tipo de redes se usan como sistemas de memoria asociativa con unidades binarias. Están diseñadas para converger a un mínimo local, pero dicha no está garantizada [33]. Tras esta primera RNN se fueron desarrollando muchas otras, como las LSTM, cuyo concepto se introdujo en el año 1997 por Sepp Hochreiter y Jürgen Schmidhuber [15]. En este estudio se analiza

la transmisión del error hacia atrás y se desarrolla una nueva arquitectura formada por una capa de entrada, una oculta y la capa de salida. La capa oculta está completamente conectada y la forman un conjunto de celdas de memoria y sus correspondientes puertas, que se encargan de evitar un conflicto con los pesos de entrada. La celda de memoria es el elemento principal de estas redes y pueden ser agrupadas en bloques que compartan las puertas de entrada y de salida, facilitando el almacenamiento de la información. En la Figura 2.1 se muestra una red de este tipo que consta de 8 neuronas en la capa de entrada, cuatro en la salida, y dos celdas de memoria en la oculta.

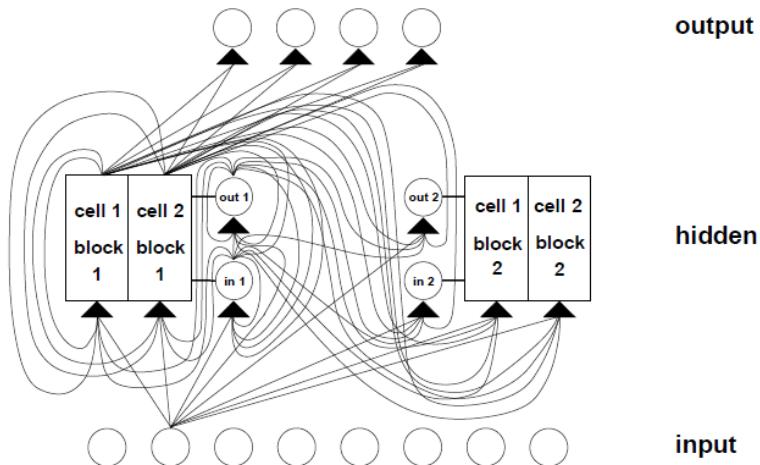


Figura 2.1: Red LSTM. Imagen obtenida de [15].

Los trabajos expuestos anteriormente dan lugar a la definición de una serie de estructuras dinámicas que permiten abordar con aprendizaje profundo la tarea de la predicción.

## 2.2. Predicción de valores

El uso de secuencias de valores para la predicción mediante el uso de RNA marca la primera vía de investigación en esta tarea. Así mismo, actualmente es la vía más desarrollada y que presenta muchos de los ejemplos existentes.

En el año 2002, miembros del *Institute of Electrical and Electronics Engineers* llevaron a cabo un estudio [34] en el se desarrolla un método para predecir los precios de la

electricidad al día siguiente según los modelos *Autoregressive Integrated Moving Average* (ARIMA). Este tipo de modelos se basan en el uso de una serie de procesos estocásticos para analizar series de tiempo, y fue introducida por Box y Jenkins [35]. El estudio propone dos modelos de este tipo para predecir los precios en cada hora para los mercados eléctricos de España y California, que alcanzan resultados satisfactorios. De entre las conclusiones a las que llega esta investigación cabe destacar que el histórico temporal necesario para realizar la predicción de una forma satisfactoria varía según el mercado, siendo mayor en el español. Además, se plantea el uso de variables explicativas que permiten mejorar los resultados en aquellos meses con una alta correlación entre la producción hidroeléctrica disponible y el precio de la electricidad.

En relación con el trabajo anterior, otro estudio que hace uso de los modelos ARIMA es el desarrollado por G. Peter Zhang [16], en el que se propone combinar dichos modelos con las RNA para distintas secuencias de valores. El desarrollo de un modelo híbrido viene motivado por la naturaleza combinada de las series temporales: es difícil determinar si una serie de tiempo se genera a partir de un proceso subyacente lineal o no lineal. Con un modelo híbrido se tiene la flexibilidad para modelar una variedad de problemas aprovechando la fuerza en la correlación lineal de ARIMA y en la no lineal de las RNA. El estudio concluye que, efectivamente, un modelo híbrido ofrece mejores prestaciones que cada modelo por separado, sin embargo la diferencia no es demasiado significativa, según se muestra en la Figura 2.2.

Table 4  
Exchange rate forecasting results<sup>a</sup>

	1 month		6 months		12 months	
	MSE	MAD	MSE	MAD	MSE	MAD
ARIMA	3.68493	0.005016	5.65747	0.0060447	4.52977	0.0053597
ANN	2.76375	0.004218	5.71096	0.0059458	4.52657	0.0052513
Hybrid	2.67259	0.004146	5.65507	0.0058823	4.35907	0.0051212

<sup>a</sup>Note: All MSE values should be multiplied by  $10^{-5}$ .

Figura 2.2: Resultados comparativos de ARIMA, RNA y ARIMA+RNA. Imagen obtenida de [16].

Un estudio similar es el desarrollado por un grupo de investigadores japoneses en el año

2015 [17]. En dicho trabajo se combina los modelos ARIMA con un tipo de red estocástica llamada *Restricted Boltzmann Machine* (RBM) seguida de un MLP. En la Figura 2.3 se muestra el flujo del modelo propuesto, en el que se integran los dos modelos. Se sitúa el foco de la investigación en cómo afecta el orden de ambos modelos en la predicción de distintos valores económicos. El estudio concluye que el orden adecuado depende del tipo de datos con el que se esté trabajando.

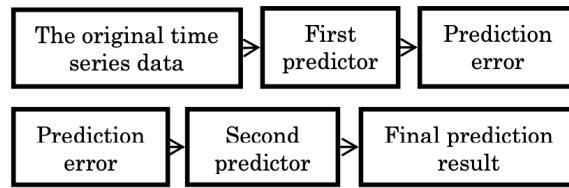


Figura 2.3: Diagrama de flujo del estudio [17].

Respecto al uso de redes recurrentes, Erol Egrioglu, Ufuk Yolcu, Cagdas Hakan Aladag y Eren Basen desarrollan un estudio en el que se propone una arquitectura combinada de RNN y RNA (RMNM-ANN) [18]. La RNN propuesta se basa en el modelo de neuronas multiplicativas que se caracterizan por el uso de funciones de agregación multiplicativas en lugar de aditivas. El nuevo modelo, representado en la Figura 2.4, puede producir variables de error rezagadas y utilizarlas como entradas gracias a su estructura de retroalimentación recurrente, y hace uso de una única neurona en la capa oculta de la RNA. El modelo propuesto se aplicó para estimar la cantidad de dióxido de carbono medida mensualmente en Ankara entre marzo de 1995 y abril de 2006, proporcionando mejores resultados que otros modelos de la literatura.

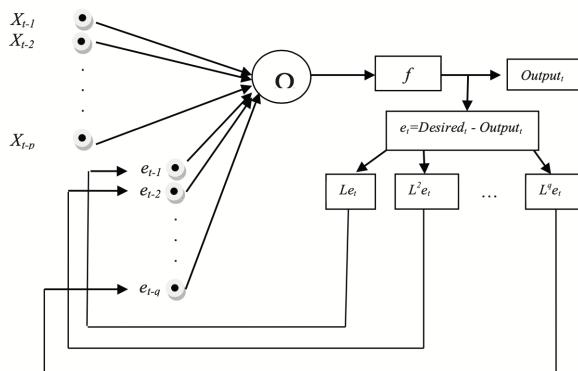


Figura 2.4: Estructura de RMNM-ANN [18].

Algunos estudios han utilizado una combinación de datos de distinta naturaleza para lograr una predicción más precisa. A modo de ejemplo, un artículo de 2018 combina datos temporales con información en forma de texto no estructurado para estimar la demanda de taxi en la ciudad de Nueva York [19]. Para ello se proponen dos estructuras, utilizando una red para el texto y otra para la información temporal y que conjuntamente logran superar de modo significativo a otros métodos populares. La principal diferencia entre ambas estructuras se encuentra en la red utilizada para el procesamiento de la información temporal, que puede ser LSTM (véase la Figura 2.5) o *fully connected*.

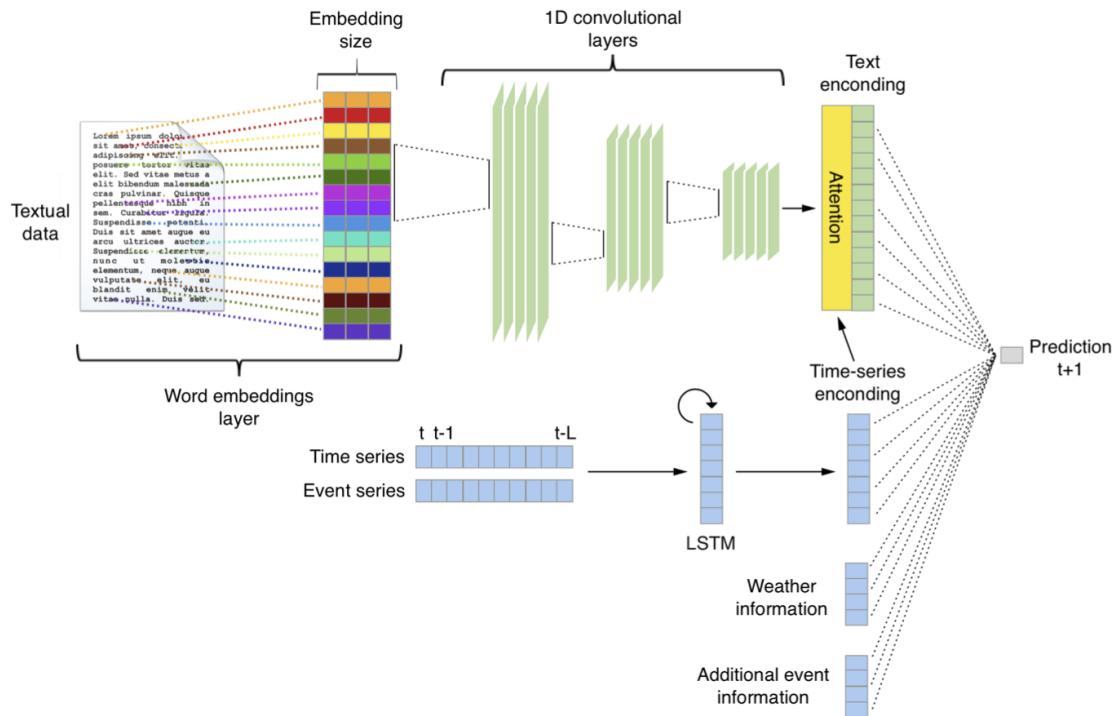


Figura 2.5: Estructura propuesta por [19] con LSTM.

Finalmente, en cuanto al seguimiento de trayectorias, se han llevado a cabo varios estudios. En el año 2016, se presentó en un artículo la red *SocialLSTM*, cuyo objetivo es lograr predecir la dinámica del movimiento de peatones en escenas abarrotadas. Esta red es capaz de predecir de forma conjunta las trayectorias de todas las personas que aparecen en una escena teniendo en cuenta las reglas de sentido común y las convenciones sociales que los humanos utilizan mientras se desplazan. Se utiliza una LSTM independiente para cada trayectoria y, posteriormente, se conectan todas ellas a través de una capa

de agrupación social (*S-pooling*), compartiendo información entre sí. En la Figura 2.6 se muestra la estructura de la red propuesta. El método propuesto logra superar a los métodos de vanguardia en dos conjuntos de datos públicos (ETH y UCY). Además, es capaz de predecir con éxito varios comportamientos no lineales que surgen de interacciones sociales.

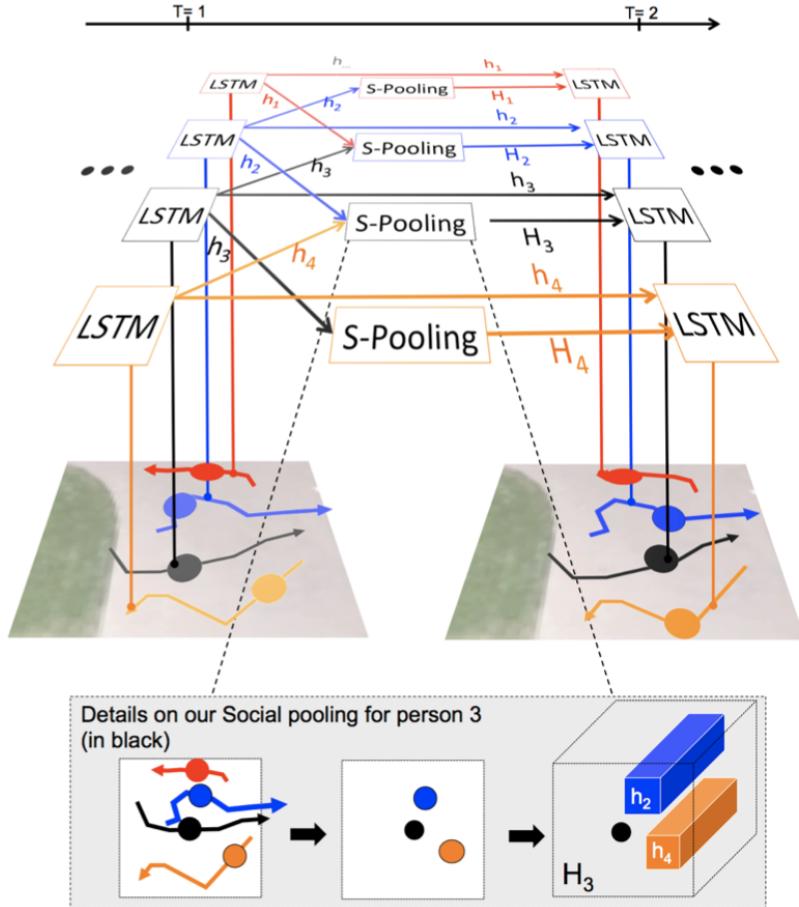


Figura 2.6: Estructura de SocialLSTM [20].

Otro estudio en relación con trayectorias, en esta ocasión de vehículos [21], se publica en el 2018 y se basa en la arquitectura de red neuronal del codificador-decodificador *lstm*<sup>1</sup>. El sistema propuesto, mostrado en la Figura 2.7, se basa en el análisis de varias observaciones pasadas del vehículo por el codificador para que el decodificador genere un número  $K$  de posibles trayectorias a seguir. Este método es capaz de lograr una mejora

<sup>1</sup><https://machinelearningmastery.com/encoder-decoder-long-short-term-memory-networks/>

significativa con respecto a los métodos existentes en términos de precisión de predicción, al mismo tiempo que puede generar la secuencia completa de la trayectoria predicha.

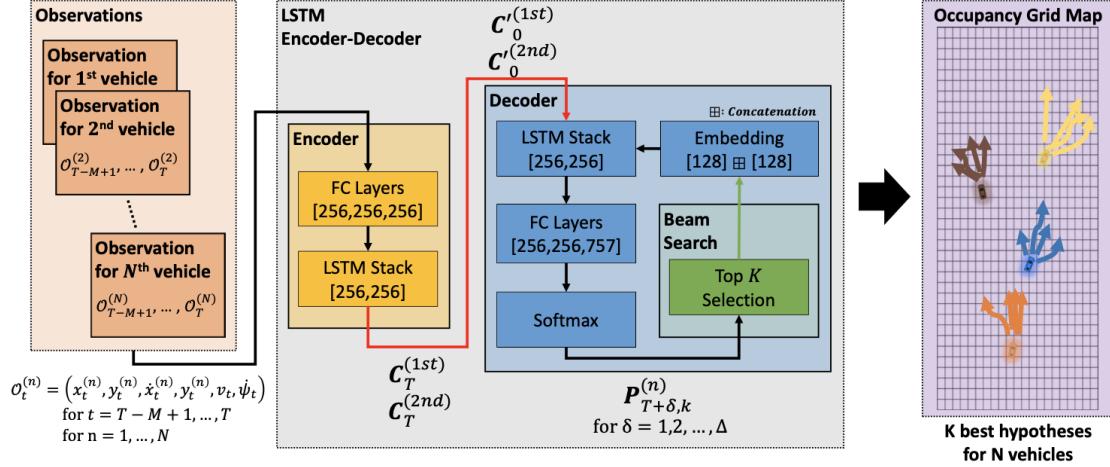


Figura 2.7: Estructura propuesta por [21].

## 2.3. Predicción en imágenes

La investigación en la tarea de predicción con secuencias de imágenes o de vídeo está menos desarrollada, puesto que surge más tarde que para las secuencias numéricas.

Uno de los campos en los que se han realizado estudios de predicción con imágenes es el de los videojuegos. En el año 2015 se publica un artículo [13] que propone dos estructuras, mostradas en la Figura 2.8, novedosas hasta el momento para la predicción de fotogramas que dependen de las acciones del jugador. Los resultados, tanto cualitativos como cuantitativos, muestran que ambas redes son capaces de predecir fotogramas visualmente realistas y útiles para el control del juego, con una separación temporal (*gap*) de 100 pasos en varios juegos de Atari de distinto dominio. Se trata de uno de los primeros artículos que muestra buenas predicciones en los juegos de Atari.

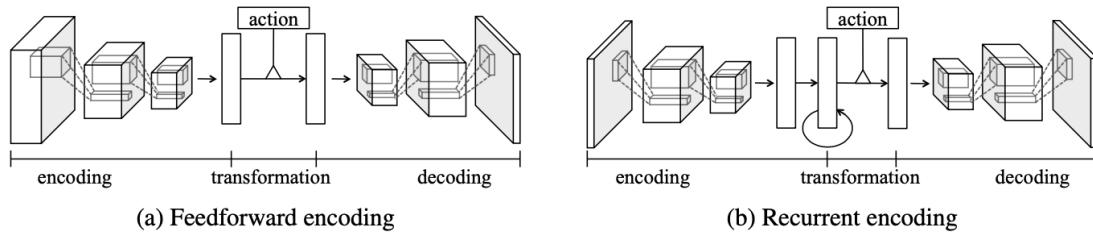


Figura 2.8: Estructuras propuestas en [13].

Otro campo de estudio es la predicción de secuencias de vídeo, anticipándose a las acciones que sucederán en la secuencia. Un ejemplo de ello es el artículo propuesto por Carl Vondrick, Hamed Pirsiavash y Antonio Torralba, que se centra en dicha predicción mediante aprendizaje no supervisado [12]. Se propone la estructura definida en la Figura 2.9, entrenada con más de 600 horas de vídeo de *YouTube* con diferentes temáticas. Con las predicciones obtenidas realizan una clasificación de la acción que se está desarrollando en la escena, obteniendo resultados bastante satisfactorios aunque con margen de mejora.

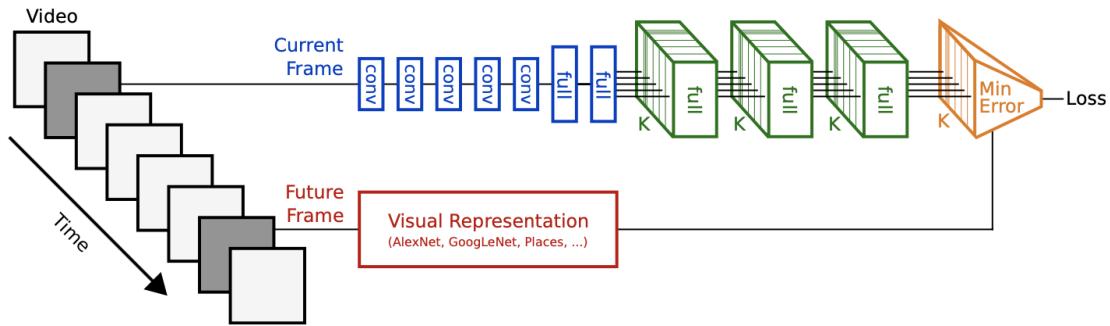


Figura 2.9: Estructura propuesta en [12].

En en año 2018 se publica un estudio que propone la descomposición del movimiento y el contenido para manejar de forma efectiva la evolución los píxeles en los vídeos [22]. El modelo propuesto (MCnet), mostrado en la Figura 2.10, se basa en un codificador-decodificador con CNN y una LSTM convolucional que obtiene la predicción a nivel de píxel. Debido a la separación entre movimiento y contenido, la predicción se obtiene convirtiendo el contenido extraído en el contenido del siguiente fotograma mediante las características de movimiento identificadas, simplificando la tarea. Para evaluar la red y

compararla con otras estructuras han empleado los conjuntos de datos KTH, Weizmann action y UCF-101, que están formados por vídeos de actividad de personas. La estructura propuesta es considerada una de las primeras arquitecturas que realizan la separación de movimiento y contenido en la predicción a nivel de píxeles en vídeos naturales.

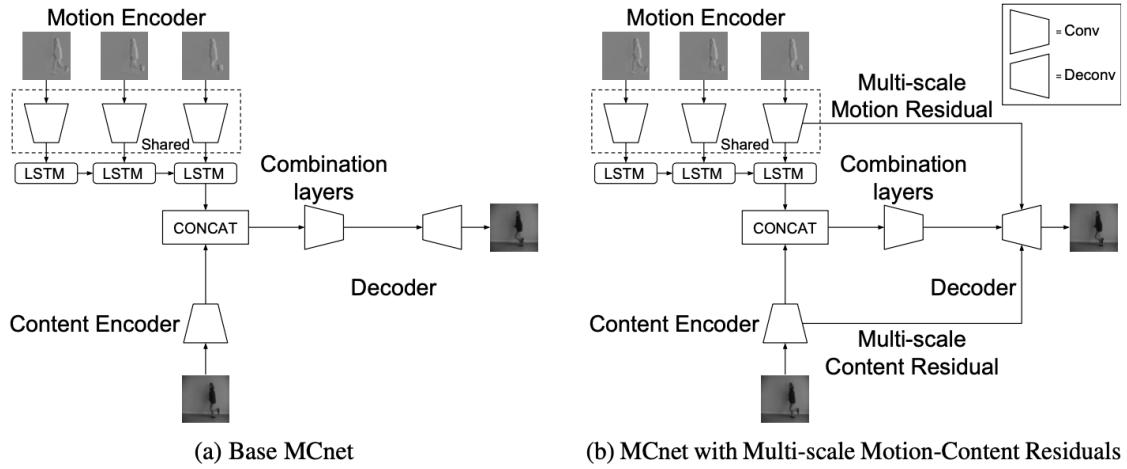


Figura 2.10: Estructura de red MCnet [22].

En el año 2019, Jason Y. Zhang, Panna Felsen, Angjoo Kanazawa y Jitendra Malik llevaron a cabo un estudio para predecir el movimiento 3D de una persona a partir de una secuencia de vídeo [23]. Es uno de los primeros enfoques que tratan de predecir una malla 3D a partir de una serie de fotogramas, por lo que el margen de mejora es todavía bastante amplio. Este modelo (PHD) centra su atención en acciones periódicas como el caminar, lanzar a los bolos o realizar un ejercicio de gimnasio. En la Figura 2.11 se muestra un ejemplo de funcionamiento.

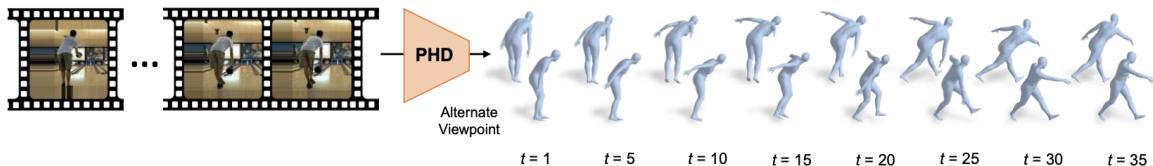


Figura 2.11: Funcionamiento de PHD [23].

La arquitectura definida se basa en modelos autorregresivos y no necesita de vídeos etiquetados en 3D, se puede entrenar con vídeos que sólo dispongan de anotaciones 2D. El

modelo desarrollado en este estudio, mostrado en la Figura 2.12, está formado por cuatro partes:

- Una *Resnet* que se encarga de extraer las características de cada fotograma.
- Un codificador temporal causal que aprende una “tira de película”, es decir, una representación latente de la dinámica humana en 3D.
- Un regresor 3D que puede leer la malla 3D de la tira de película.
- Un modelo autorregresivo en el espacio latente que toma las tiras de películas pasadas para predecir las películas futuras, capturando así la dinámica humana.

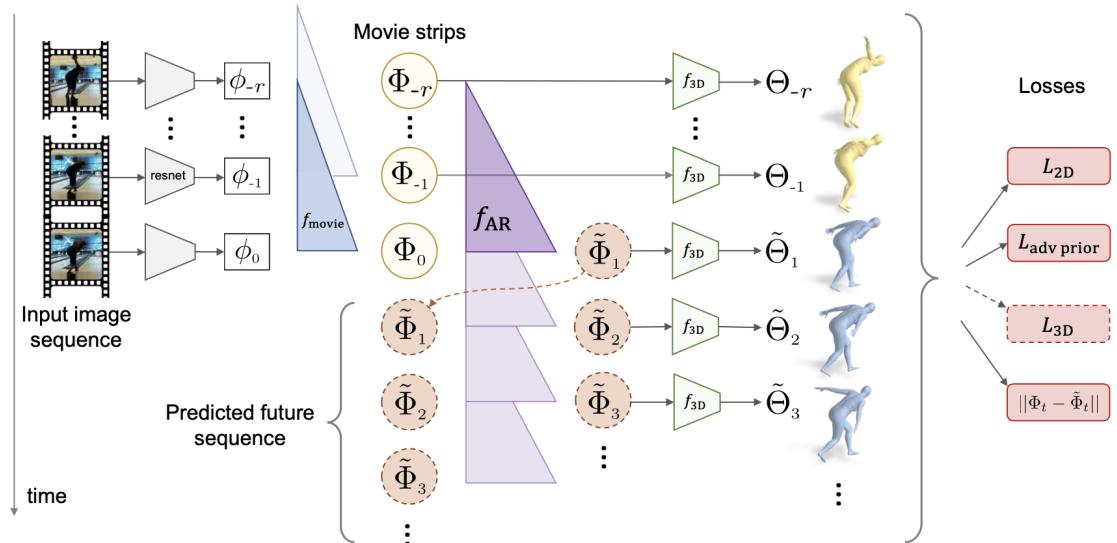


Figura 2.12: Estructura de red PHD [23].

Finalmente, uno de los estudios más recientes [24] trata de obtener la predicción del fotograma lo más nítida y realista posible, evitando las distorsiones que generan otros métodos de predicción. Para ello se hace uso de la estructura definida en la Figura 2.13, una red residual profunda con arquitectura jerárquica donde cada capa hace una predicción del estado futuro a diferentes resoluciones espaciales para después fusionarlas a través de conexiones de arriba hacia abajo.

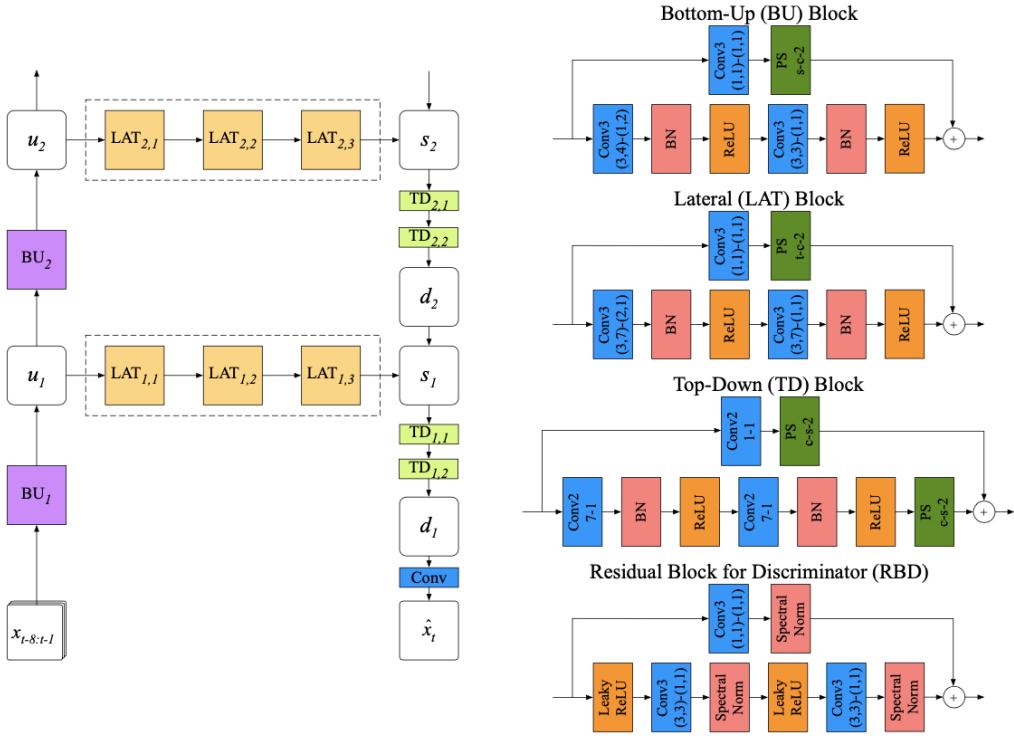


Figura 2.13: Estructura propuesta en [24].

El modelo propuesto supera cuantitativamente las líneas del estado del arte en la predicción de fotogramas futuros y es capaz de generar fotogramas con detalles más finos y texturas que son perceptualmente más realistas, según se muestra en la Figura 2.14.

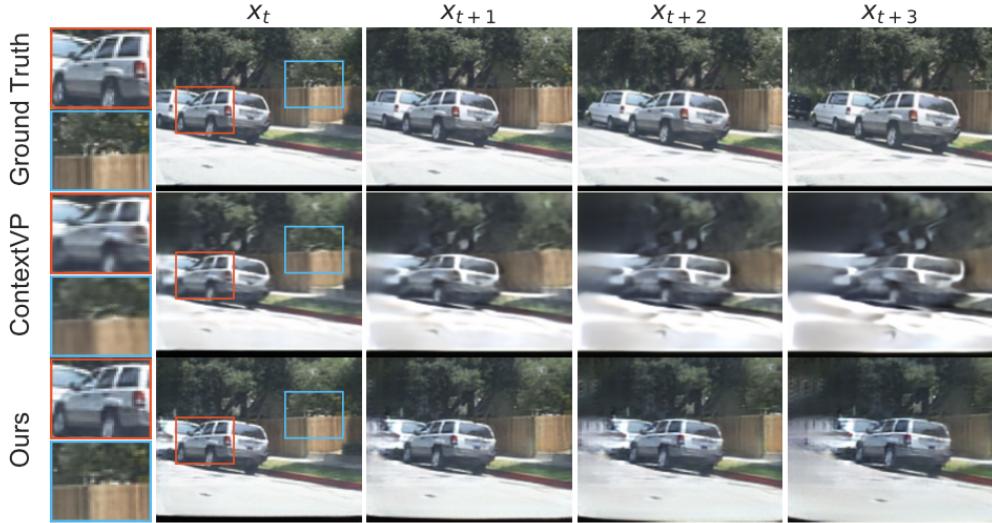


Figura 2.14: Resultados de estructura propuesta en [24].

## 2.4. Infraestructura utilizada

Esta sección describe los elementos *software* y *hardware* utilizados en el desarrollo de este trabajo. Se explica el lenguaje de programación empleado, las herramientas *software* necesarias, y el servidor en el que se han ejecutado todos los experimentos.

### 2.4.1. Lenguaje Python

Python<sup>2</sup> es un lenguaje de programación orientado a objetos, interpretado e interactivo que combina una alta potencia con una sintaxis muy clara. Esta sencillez en la sintaxis hace que sea fácil de aprender. Además la comunidad de desarrolladores proporciona conferencias, documentación y listas de correo que facilitan su aprendizaje.

La potencia de este lenguaje viene dada por la fácil combinación de la biblioteca estándar con miles de módulos de terceros que permite aumentar casi infinitamente el rango de posibilidades de sus aplicaciones. Además, puede ser utilizado como lenguaje de extensión para aplicaciones escritas en otros lenguajes que necesitan interfaces de automatización o *scripts* fáciles de usar. Todo esto hace que su uso se haya extendido en las distintas aplicaciones de IA. Es un lenguaje de código abierto que permite su libre uso y distribución, cuya licencia está administrada por la “*Python Software Foundation*”.

Actualmente, la versión de Python más reciente es la 3.8.6, lanzada el 24 de septiembre de 2020. Sin embargo, en la realización de este trabajo se ha utilizado la versión 3.6.9, pues era la última en el momento de comenzar el desarrollo.

### 2.4.2. Biblioteca OpenCV

OpenCV (*Opne Source Computer Vision Library*)<sup>3</sup> es una librería de código abierto para realizar tareas de VA y aprendizaje automático. Fue creado con el objetivo de facilitar una infraestructura común para diversas aplicaciones de VA y acelerar su uso en productos comerciales.

---

<sup>2</sup><https://www.python.org>

<sup>3</sup><https://opencv.org>

Su uso está muy extendido, pues cuenta con más de 47 mil usuarios en la comunidad y un número aproximado de descargas superior a 18 millones. Su expansión abarca desde grupos de investigación hasta empresas tan potentes como *Google*, y cuenta con algunos usos desplegados como la unión de imágenes en *streetview* o la comprobación de pistas en busca de escombros en Turquía.

Presenta interfaces para los lenguajes C++, Java, MATLAB y Python, esta última utilizada en este trabajo. En cuanto a sistemas operativos, es compatible con Windows, Linux, Android y Mac OS. Dispone de más de 2500 algoritmos, clásicos y de última generación, que han sido optimizados y que permiten realizar tareas como la detección de rostros y su identificación, el seguimiento de objetos, la extracción de modelos 3D de los objetos o la búsqueda de patrones.

El principal uso de OpenCV en el trabajo se ha centrado en el procesamiento de las imágenes utilizadas para el entrenamiento y la evaluación de las redes mediante tres métodos:

- ***cv2.imwrite(filename, image):*** Almacena la imagen (*image*) en el fichero (*filename*). Se utiliza en la generación de las imágenes para almacenar las secuencias de imágenes en la base de datos.
- ***cv2.imread(path, flag=0):*** Lee la imagen almacenada en *path* en escala de grises (*flag=0*). Se utiliza en la lectura las secuencias de imágenes de la base de datos para entrenar o evaluar las distintas redes.
- ***cv2.GaussianBlur(image, (5, 5), 0):*** Aplica un filtro gaussiano a la imagen para expandir el píxel en una ventana de 5x5. Se utiliza como preprocesamiento de las imágenes en algunos estudios.

La versión más reciente de la biblioteca OpenCV en la actualidad es la 4.4.0, mientras que la empleada en este trabajo es la 4.2.0 con su interfaz de Python.

### 2.4.3. Biblioteca Matplotlib

La librería Matplotlib<sup>4</sup> está enfocada a la creación de gráficos 2D de matrices en Python. Su origen se sitúa en la emulación de comandos gráficos de MATLAB pero ha desembocado en una biblioteca completamente independiente de este lenguaje que puede ser usada en Python y de forma orientada a objetos. Está escrito principalmente en Python puro, aunque utiliza de forma intensiva la librería Numpy para el tratamiento de matrices, y otros códigos de extensión para un buen rendimiento incluso con matrices grandes. La filosofía de esta biblioteca radica en la capacidad de crear gráficos simples con unos pocos comandos, o tan solo uno.

Los usos de Matplotlib son muy variados. En este trabajo se emplean para la representación gráfica de las distintas métricas calculadas para comparar las prestaciones de las diferentes redes. En concreto, se hace uso de dos gráficos:

- ***pyplot.bar***: Dibuja un diagrama de barras dados un par de vectores  $x$  e  $y$  con el mismo número de valores. Permite la modificación de parámetros como el ancho de la barra, su alineación respecto el valor de  $x$  o el color de la misma.
- ***pyplot.boxplot***: Dibuja un diagrama de caja y bigotes sobre un conjunto de datos. En este gráfico se representa la caja, con los valores del cuartil inferior al superior de los datos con una línea en la mediana; y los bigotes, que se extienden desde la caja para mostrar el rango de los datos. Además aparecen representados los distintos *outliers* como puntos más allá del final de los bigotes.

La versión más reciente de Matplotlib es la 3.3.2, pero en este trabajo se hace uso de la versión 3.1.3.

### 2.4.4. Middleware neuronal Keras

Keras<sup>5</sup> es un *middleware* de aprendizaje profundo escrito en Python que se ejecuta sobre la plataforma Tensorflow. Su enfoque de desarrollo se sitúa en lograr una experimentación rápida, pasando de la idea al resultado de una forma fluida. Keras es la API de alto nivel de TensorFlow 2.0 que proporciona abstracciones y bloques de construcción

---

<sup>4</sup><https://matplotlib.org>

<sup>5</sup><https://keras.io>

esenciales para desarrollar diversas soluciones de aprendizaje automático con una alta velocidad de iteración.

Las estructuras de datos centrales de este *middleware*, sobre las que se entrará en detalle a continuación, son los llamados modelos y capas.

#### 2.4.4.1. Modelos en Keras

Un modelo es una estructura de datos que engloba a otras, las capas, para la creación de redes neuronales de aprendizaje profundo. El tipo de modelo más sencillo definido en Keras, y el utilizado en este trabajo, es el *Sequential()* que apila todas las capas de forma lineal. Para trabajar con estos modelos se utilizan las funciones que se exponen a continuación:

- ***model.add(layer)*:** Se utiliza para añadir una nueva capa (*layer*) al modelo (*model*).
- ***model.compile(...)*:** Sirve para, una vez definida la estructura, configurar la red para el entrenamiento. A esta función se le pueden pasar varios argumentos, siendo empleados en este trabajo dos de ellos:
  - *loss*: Define la función de pérdida a optimizar y varía según el experimento a realizar.
  - *optimizer*: Define el tipo de optimizador a utilizar.
- ***model.fit(...)*:** Entrena el modelo bajo una serie de condiciones. Los parámetros utilizados son:
  - *x*: Datos de entrenamiento.
  - *y*: Etiquetas de entrenamiento.
  - *batch\_size*: Tamaño del lote, número de muestras utilizadas en cada actualización de gradiente.
  - *epochs*: Número de épocas a entrenar.
  - *validation\_data*: Datos de validación.
  - *callbacks*: Lista con los objetos para realizar acciones en varias etapas del entrenamiento. En concreto se hace uso de dos de ellos.

- *early\_stopping*: Define un criterio de parada en el entrenamiento cuando el valor de la función de *loss* no mejora un número de épocas consecutivas.
- *checkpoint*: Guarda el modelo en un punto de control determinado. En este trabajo, se guarda el modelo si éste es mejor que el anterior en términos de *loss* con el conjunto de validación.
- *verbose*: Establece la forma de obtener información sobre el entrenamiento del modelo durante el avance del mismo.
- ***model.fit\_generator(...)***: Realiza la misma operación que el *fit* pero con una división por lotes del conjunto de entrenamiento. Se utilizan los mismos parámetros que los definidos en la función anterior salvo tres de variaciones:
  - *generator*: Especifica la forma en la que se obtienen los distintos lotes para el entrenamiento. Para este trabajo se trata de una función que lee un bloque de secuencias del tamaño indicado. Esta función sustituye a los valores de *x* e *y*.
  - *steps\_per\_epoch*: Establece el número de lotes que se procesarán en cada época. Este valor se obtiene como *Tamaño conjunto entrenamiento / Tamaño lote*.
  - *validation\_steps*: Establece el número de lotes que se validarán en cada época. Este valor se obtiene como *Tamaño conjunto validación / Tamaño lote*.
- ***model.predict(dataX)***: Genera las predicciones sobre los datos de entrada que se proporcionen (*dataX*).

Por último, para poder obtener una representación esquemática de la estructura de la red diseñada se hace uso del método *utils.vis\_utils.plot\_model(model, path)* que almacena la estructura del modelo (*model*) en la ruta definida (*path*). Por otro lado, la función *models.load\_model(path)* carga en memoria un modelo de Keras previamente guardado.

#### 2.4.4.2. Capas en Keras

Como se mencionó anteriormente, los modelos en Keras están compuestos por una combinación de varias de sus capas. A continuación se exponen las capas utilizadas en este trabajo, un subconjunto de todas las posibilidades que proporciona el API.

- ***Dense***: Define una capa completamente conectada (*fully connected*) con un número de neuronas determinado.

- ***Conv2D***: Se trata de una capa de convolución en la que un núcleo (*kernel*) convoluciona con la entrada de la capa para producir un tensor de salidas.
- ***MaxPooling2D***: Es una capa que realiza una operación de agrupación máxima para datos espaciales 2D. Reduce la representación de entrada utilizando el valor máximo sobre una ventana de tamaño definido en el parámetro *pool\_size*.
- ***LSTM***: Crea una capa recurrente de tipo LSTM con un número de celdas de memoria determinado y la posibilidad de devolver las secuencias para apilar varias capas de este tipo.
- ***ConvLSTM2D***: Establece una capa de tipo ConvLSTM en dos dimensiones, en la que las multiplicaciones en una celda de memoria se sustituyen por operadores de convolución.
- ***Flatten***: Se trata de una capa cuya función es reestructurar la entrada que recibe como un vector unidimensional.
- ***TimeDistributed***: No es una capa como tal. Se trata de un contenedor que permite aplicar una capa a cada segmento temporal de una entrada. La entrada debe ser al menos 3D, y la primera dimensión será considerada la dimensión temporal.

En todas estas capas se definen el número de neuronas que tiene cada capa y la función de activación a utilizar, encargada de normalizar el valor de la salida en un rango determinado. Para este trabajo han sido consideradas cuatro funciones de activación, representadas en la Figura 2.15.

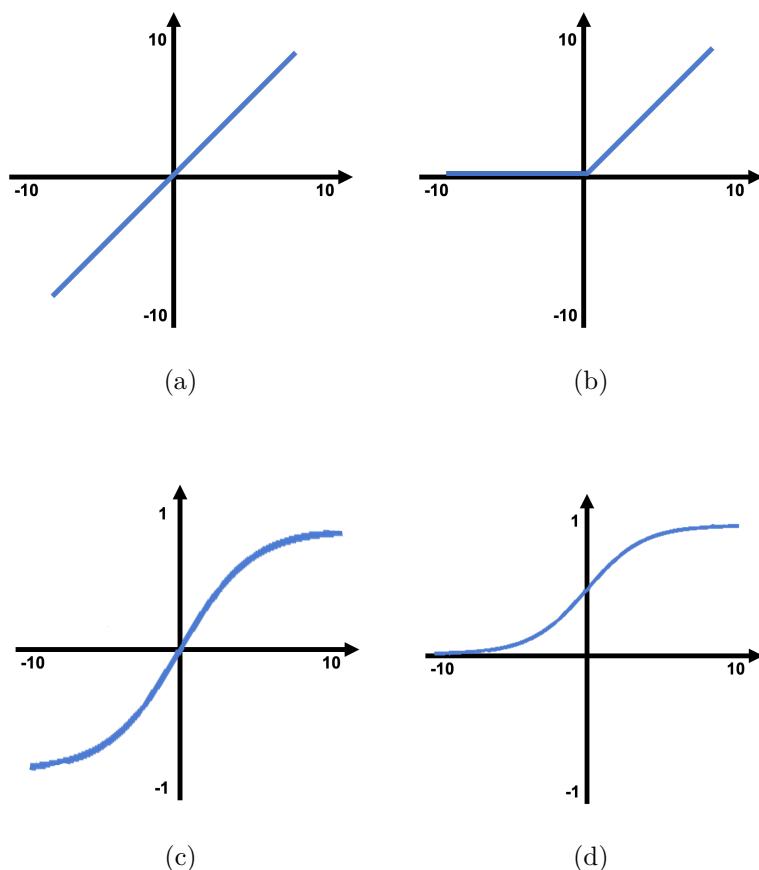


Figura 2.15: Funciones de activación: (a) Lineal, (b) ReLu, (c) Tanh y (d) Softmax.

#### 2.4.5. Servidor Tamino

Para la ejecución de los distintos procesos de este trabajo se ha utilizado un servidor Linux, Tamino, que está situado en el CPD de la ETSIT-URJC y pertenece a *RoboticsLabURJC*. Las principales características de este servidor son:

- **Procesador:** Intel(R) Xeon(R) CPU E5-2609 v4 @ 1.70GHz.
- **Número de núcleos:** 8 cores.
- **Memoria RAM:** 64GB.
- **GPU:** GeForce GTX 1080.

El uso de esta máquina ha permitido realizar entrenamientos más potentes así como el uso de conjuntos de datos más grandes.

# Capítulo 3

## Generación de secuencias de vídeo sintéticas

En este trabajo se va a evaluar el poder de predicción de distintas redes neuronales profundas. La exploración en este ámbito se realizará de forma progresiva: primero con píxeles que se mueven linealmente, posteriormente con otros que se mueven parabólicamente y finalmente otros que se mueven siguiendo una función sinusoidal. Es decir, se modifican las dinámicas de movimiento con una complejidad creciente. Para entrenar estas redes neuronales es necesario tener un conjunto grande de imágenes .etiquetas con las que aprender a predecir. Se ha programado una aplicación que genera automáticamente esos conjuntos voluminosos siguiendo esas dinámicas con sus parámetros correspondientes, a la vez que divide el conjunto en los subconjuntos de entrenamiento, validación y *test*. En la Figura 3.1 se presenta un diagrama “caja negra” del funcionamiento de dicha aplicación, en la que se parte de un fichero de configuración y se obtiene una base de datos con las muestras deseadas.

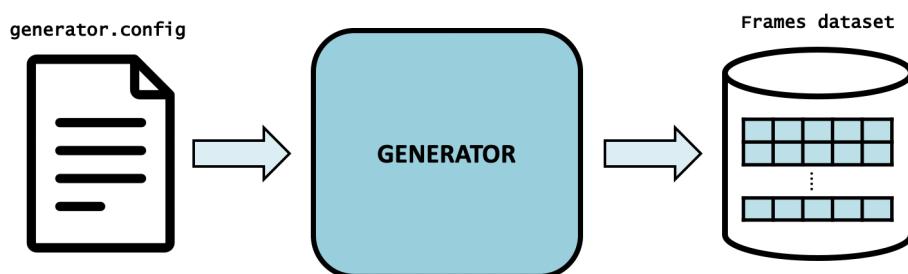


Figura 3.1: Diagrama “caja negra” del generador.

En este capítulo se explican los dos tipos de imágenes que se han considerado, modeladas y crudas, y se exponen las distintas dinámicas de movimiento entre fotogramas que se aplican dentro de un mismo subconjunto. Así mismo, se expondrá el código desarrollado para la generación de los conjuntos de datos que se utilizarán para entrenar las redes neuronales profundas de predicción y para validar experimentalmente la calidad sus predicciones, el nivel de acierto.

### 3.1. Propiedades del *dataset*

Esta sección hace hincapié en las características de los *datasets* generados con secuencias de vídeo. Se explica en profundidad la estructura que tiene el directorio que almacena los distintos conjuntos utilizados, los dos tipos de imágenes considerados, modeladas y crudas, y las tres dinámicas que han sido estudiadas, lineal, parabólica y sinusoidal.

Un *dataset* generado se corresponde con una base de datos en la que se almacenan  $n\_samples$  muestras, cada una de las cuales es una colección de  $n\_points+1$  fotogramas. En la Figura 3.2 se muestra un esquema general del aspecto que presenta un conjunto de datos utilizado en este trabajo.

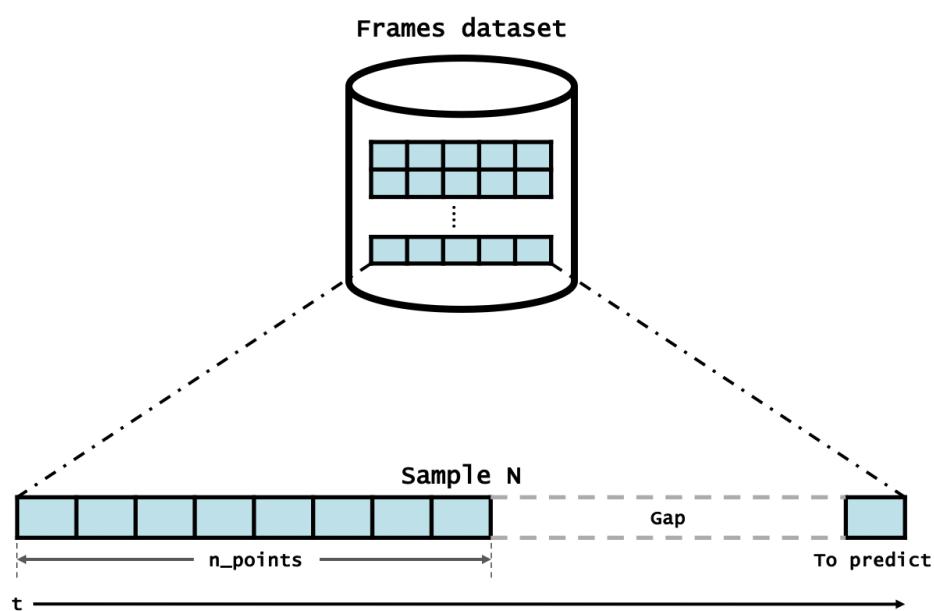


Figura 3.2: Vista general de un *dataset*.

### 3.1.1. Estructura del *dataset*

En la Figura 3.3 se presenta un esquema de la estructura resultante con la ejecución del código explicado en la Sección 3.2 para diferentes parámetros (de 1 a n).

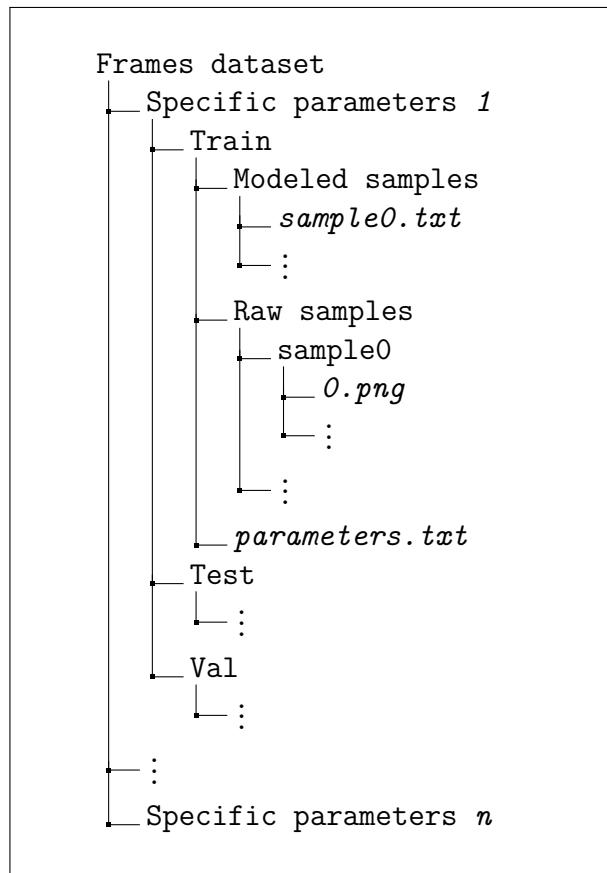


Figura 3.3: Estructura creada para el almacenamiento del *dataset*.

En el primer nivel de esta estructura, *Frames dataset*, se distingue entre los distintos tipos de datos que se pueden generar. A pesar de que en este trabajo se ha centrado en las secuencias de imágenes, el generador fue diseñado para poder obtener otros dos tipos de datos: funciones y vectores, que tendrían su directorio homólogo en este mismo nivel.

En el segundo nivel se hace una distinción en función de los parámetros concretos que se hayan establecido en el fichero de configuración, explicado en el Apartado 3.2.1. Estos valores son:

- Tipo de dinámica.
- Forma y color del objeto.

- Grados de libertad.
- Número de muestras.
- Tamaño de la imagen.

Dentro de cada uno de estos directorios se presenta siempre la misma estructura. La primera división existente es la de los subconjuntos de entrenamiento, *test* y validación. Si no se hubiese indicado la realización de esta división, este nivel sería obviado y se pasaría directamente al siguiente. Todas estas carpetas presentan a su vez la misma estructura, sobre la que la herramienta de generación escribe directamente las distintas muestras: el fichero de parámetros, la carpeta con las muestras modeladas y la correspondiente a las muestras en crudo.

### 3.1.2. Tipos de imágenes

Se han considerado dos tipos de imágenes para los distintos estudios sobre predicción visual que se han desarrollado en el trabajo. Ambos tipos de imágenes tienen una correspondencia entre sí, es decir, para una misma muestra se presenta la misma información en dos formatos distintos: valores numéricos e imágenes puras de píxeles. En cuanto al contenido, todas las imágenes constan de un fondo negro y un único punto de luz móvil blanco, un píxel iluminado, que se desplaza dentro del fotograma siguiendo cierta dinámica.

#### 3.1.2.1. Imágenes modeladas

Las datos modelados son una forma simplificada de mostrar las distintas posiciones que adquiere el píxel móvil en cada instante de tiempo. Con este tipo de imágenes se consigue representar la misma información de una forma compacta, se utiliza un total de  $2 * n\_points$  valores de entrada a la red, y 2 valores de salida de la misma, reduciendo considerablemente la complejidad de las distintas redes que se estudien. En la Tabla 3.1 se presenta un ejemplo concreto de cómo se representan estas muestras.

<i>t</i>	<i>x</i>	<i>y</i>
<b>0</b>	0	40
<b>1</b>	3	38
<b>2</b>	6	37
<b>3</b>	9	36
$\vdots$	$\vdots$	$\vdots$
<b>29</b>	87	4

Tabla 3.1: Ejemplo de imagen modelada.

Estos datos se almacenan en la carpeta *modeled\_samples* mediante un fichero *"sample[n].txt*, donde *n* hace referencia al número del ejemplo durante su creación. Este fichero presenta una cabecera y un número de filas que se corresponde al número de instantes conocidos más el instante que se quiere predecir. La información viene distribuida en dos columnas, correspondientes con los valores *x* e *y* del píxel en cada instante de tiempo, que no se representa de forma explícita sino que se asocia con el orden de las filas.

### 3.1.2.2. Imágenes crudas

Las imágenes crudas son las imágenes propiamente dichas. En la Figura 3.4 se muestra una composición donde se superponen todas las imágenes que forman la secuencia agrupadas en una única imagen.

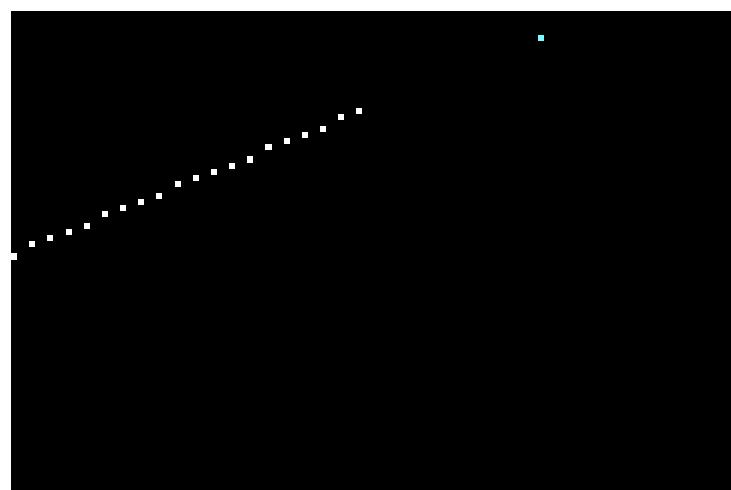


Figura 3.4: Ejemplo de muestra cruda.

Cada uno de los fotogramas presentaría un único píxel activo que se corresponde con el objeto del que se simula el movimiento y se almacena con el nombre “[n].png” en la carpeta “sample[m]”, donde “n” sería el orden del instante de tiempo y “m” el de la muestra generada. Cada una de estas carpetas que identifica una muestra es almacenada en el directorio *raw\_samples* que confluye todos los ejemplos del mismo tipo.

En este caso, a pesar de que la información es la misma, la cantidad de valores que se utilizan para representarla es mucho mayor. Se utiliza un total de  $h * w * n\_points$  valores de entrada a la red, y  $h * w$  valores de salida de la misma, complicando las redes a estudiar.

### 3.1.3. Tipos de dinámicas

El movimiento del píxel en el vídeo viene determinado por tres dinámicas distintas, que van aumentando el grado de complejidad del mismo y ponen a prueba la capacidad de predicción de las redes. Además de las tres dinámicas que se han considerado: lineal, parabólica y sinusoidal, dentro de cada una de ellas se aumenta de forma escalonada la dificultad incrementando el número de variables que adquieren un valor aleatorio, es decir, los *Degrees Of Freedom* (DOF) de las mismas.

Estas dinámicas afectan únicamente a la posición  $y$  del píxel, y se utilizan los valores de  $x$  obtenidos mediante la aplicación de un Movimiento Rectilíneo Uniforme (MRU) al instante de tiempo con una velocidad aleatoria.

#### 3.1.3.1. Dinámica lineal

La primera dinámica, y la más sencilla de todas, simula el movimiento del objeto en una línea recta a distintas velocidades y con distintas pendientes. Este movimiento se puede asimilar con el movimiento de un vehículo con una velocidad constante, sin tener en cuenta los efectos del rozamiento por la superficie ni la perspectiva. Para ello, utiliza la ecuación de la recta:

$$y = m * x + y_0$$

que se materializa en el código de la siguiente forma:

```
self.g = lambda x, y0, m: (m * x) + y0
```

En la Figura 3.5 se presenta un ejemplo de esta dinámica con todas las imágenes que lo conforman superpuestas en una sola.

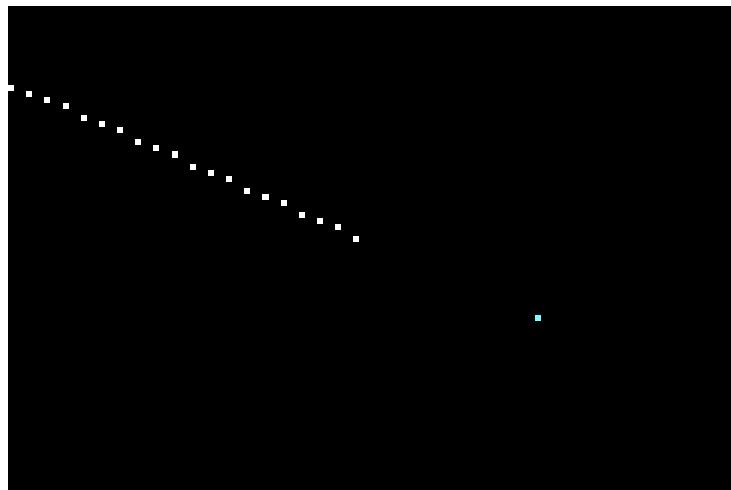


Figura 3.5: Ejemplo de dinámica lineal.

Por la propia naturaleza de la ecuación, es posible ir modificando la complejidad del movimiento a medida que incrementamos la cantidad de parámetros que se asignan de forma aleatoria, es decir, los DOF. A continuación, se exponen los distintos casos que se dan en la dinámica actual.

### Pendiente nula

El movimiento más sencillo que se puede encontrar es un caso especial de la dinámica lineal. Ocurre cuando se fija la pendiente de la recta ( $m$ ) en un valor nulo, igualando el valor de la posición a una altura inicial. De esta forma se obtiene la ecuación simplificada:

$$y = y_0$$

que se presenta en el código de la siguiente manera:

```
self.g = lambda x, y0: y0
```

La altura inicial se puede asignar de dos maneras distintas::

```
y0 = int(self.h / 2) #1  
y0 = random.randint(1, self.h - 1) #2
```

La primera línea asigna la altura a la mitad de la imagen ( $h/2$ ), mientras que con la segunda se establece un valor aleatorio entre los límites de la imagen, que introduce un nuevo DOF.

En la Figura 3.6 se muestra un ejemplo de este caso especial de la dinámica lineal.

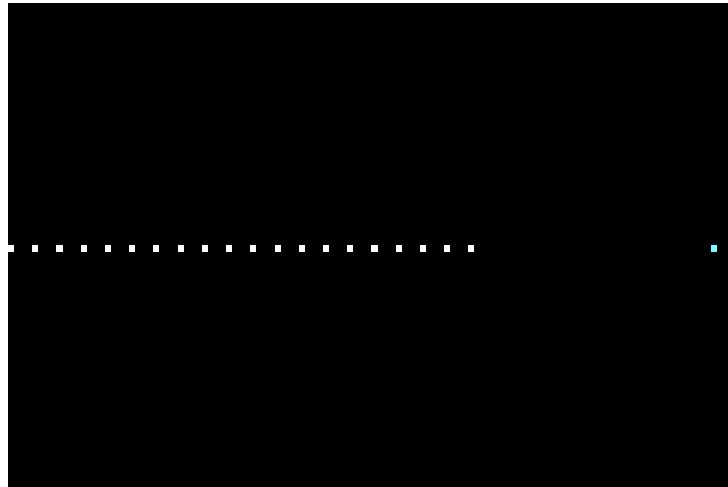


Figura 3.6: Ejemplo del caso con pendiente nula.

## 1 DOF

El primer parámetro de la ecuación que toma un valor aleatorio en cada muestra es la pendiente de la recta, dejando fija la altura inicial del punto. El código que se utiliza en cada iteración para la definición de los parámetros es el siguiente:

```
m = np.round(random.uniform(-self.h/10, self.h/10), 2)  
y0 = int(self.h / 2)
```

Los límites que se establecen para obtener el valor aleatorio son orientativos. Pretenden reducir al máximo el intervalo que asegure la presencia del punto en todas las imágenes pero esta presencia no llega a asegurarse en su totalidad. Es por esto que, posteriormente, se realiza una comprobación de las posiciones obtenidas.

## 2 DOF

Se deja que todos los parámetros de la ecuación tomen un valor aleatorio distinto en cada muestra, tanto la pendiente como la altura inicial. Para esta asignación se implementa el siguiente código:

```
m = np.round(random.uniform(-self.h/10, self.h/10), 2)
y0 = random.randint(1, self.h - 1)
```

Al igual que en el caso anterior, los límites establecidos son orientativos para cumplir con las limitaciones del *dataset*.

### 3.1.3.2. Dinámica parabólica

La dinámica parabólica, que aumenta el grado de complejidad mediante el incremento del número de parámetros necesarios para su definición, utiliza la ecuación de la parábola para obtener las posiciones  $y$ :

$$y = a * t^2 + b * t + c$$

que se implementa en el código de mediante las siguiente línea:

```
self.g = lambda x, a, b, c: a * (x ** 2) + b * x + c
```

En la Figura 3.7 se puede ver un ejemplo, con el mismo formato que el de la dinámica anterior, de este tipo de movimiento.

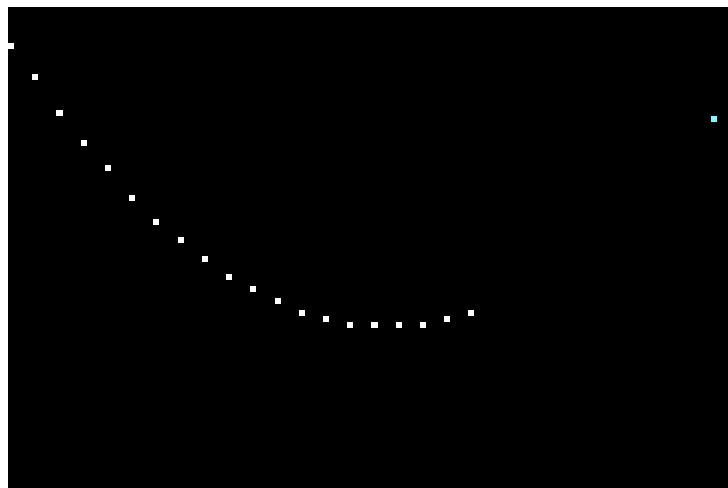


Figura 3.7: Ejemplo de dinámica parabólica.

Al igual que en el caso lineal, se va incrementando la complejidad de la dinámica aumentando el número de DOF, explicados a continuación.

## 1 DOF

El parámetro que se modifica en este primer caso es el que acompaña al término cuadrático ( $a$ ), de tal forma que la definición de los parámetros queda de la siguiente manera:

```
a = round(random.uniform(-0.75, 0.75), 3)  
b = 1.5  
c = int(self.h / 2)
```

El valor otorgado a  $b$  deriva de una serie de pruebas para obtener aquel que mejor se ajustase a lo que se buscaba mientras que la altura inicial ( $c$ ) se establece en la mitad de la imagen. Por otro lado, al igual que ocurre en la dinámica lineal, los límites escogidos de la muestra son orientativos para cumplir con la presencia del píxel en todas las imágenes, pero en ningún caso la asegura.

## 2 DOF

El siguiente parámetro a modificar es la altura inicial, que en este caso se establece con el parámetro  $c$ . Siguiendo la definición del caso lineal, los parámetros de este caso quedan definidos de la siguiente manera:

```
a = round(random.uniform(-0.75, 0.75), 3)  
b = 1.5  
c = random.randint(1, self.h - 1)
```

## 3 DOF

En el último caso se definen todos los parámetros de la ecuación mediante un valor aleatorio. Con esta premisa, la definición de los parámetros presenta la siguiente forma:

```
a = round(random.uniform(-0.75, 0.75), 3)
b = round(random.uniform(-2, 2), 3)
c = random.randint(1, self.h - 1)
```

### 3.1.3.3. Dinámica sinusoidal

La dinámica más compleja considerada en este trabajo, la sinusoidal, hace uso de la función trigonométrica del seno para el cálculo de posiciones en el eje vertical. Su expresión es la siguiente:

$$y = a * \sin(2 * \pi * f * x + b) + c$$

y se traduce en el código en la línea:

```
self.g = lambda x, a, b, c, f:
    a * math.sin(2 * math.pi * f * math.radians(x) + b) + c
```

En la Figura 3.8 se muestra un ejemplo de la dinámica mediante la superposición de las imágenes en una sola.

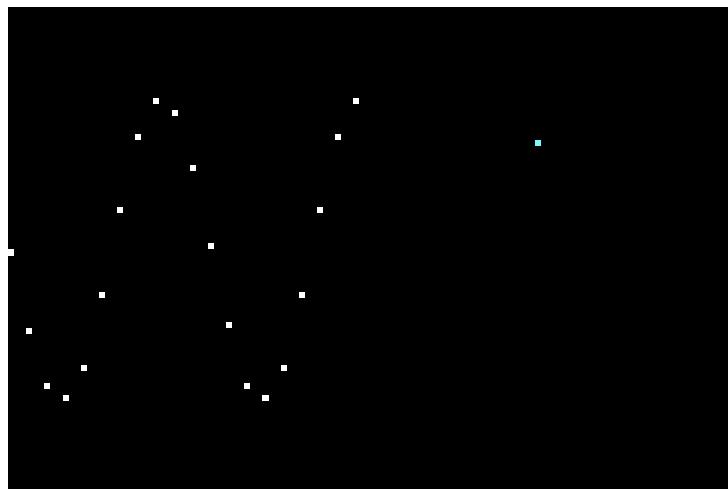


Figura 3.8: Ejemplo de dinámica sinusoidal.

De la misma forma que sucede con las otras dos dinámicas, se aumenta la dificultad del movimiento mediante el aumento de los DOF, que se exponen a continuación.

## 1 DOF

El primer parámetro que se establece de forma aleatoria es la frecuencia ( $f$ ) del seno. La definición de los parámetros para este caso queda de la siguiente manera:

```
a = 25  
f = round(random.uniform(0.5, 5), 2)  
b = 0  
c = int(self.h / 2)
```

Como ocurre en el caso parabólico, los valores de los parámetros fijados se han obtenido mediante distintas pruebas con el objetivo de que sean lo más adecuados para los estudios realizados. De igual forma, los límites establecidos son orientativos para cumplir con las limitaciones del *dataset*.

## 2 DOF

El segundo parámetro que se deja libre es, de nuevo, la altura inicial de la imagen en la que comienza el movimiento, dando lugar a la siguiente definición de parámetros:

```
a = 25  
f = round(random.uniform(0.5, 5), 2)  
b = 0  
c = random.randint(1, self.h - 1)
```

## 3 DOF

El tercer parámetro que obtiene valores aleatorios en esta dinámica es la amplitud del seno, y se definen los parámetros de este caso con el siguiente código:

```
a = random.randint(15, 40)  
f = round(random.uniform(0.5, 5), 2)  
b = 0  
c = random.randint(1, self.h - 1)
```

## 4 DOF

Por último se permite que todos los parámetros tomen valores aleatorios, dando lugar al caso más complejo de este movimiento. Para ello se usa el siguiente código:

```
a = random.randint(15, 40)
f = round(random.uniform(0.5, 5), 2)
b = round(random.uniform(0, 2 * math.pi), 3)
c = random.randint(1, self.h - 1)
```

## 3.2. Herramienta de generación

Para la elaboración propia de los distintos conjuntos de datos, se ha desarrollado un código<sup>1</sup> en Python que, a través de la fijación de varios parámetros en un fichero de configuración, elabora un conjunto de muestras que se adapta al estudio concreto en el que se vaya a aplicar. En esta sección se explican las dos partes fundamentales de este código: la especificación de los parámetros y la propia generación y almacenamiento de las muestras.

### 3.2.1. Fichero de configuración

A través de este fichero se establecen los parámetros principales que definen las características del *dataset* a generar. A continuación se explican los distintos valores a definir y lo que supone modificar cada uno de ellos.

- |                    |   |
|--------------------|---|
| <b>root</b>        | Define la ruta en la que se almacenará la estructura de carpetas y archivos que se genera en la ejecución del código.   |
| <b>to_generate</b> | Indica el tipo de dato que va a ser generado. A pesar de que este proyecto se centra en el estudio con imágenes, el generador se desarrolló con la posibilidad de generar datos de tres tipos: funciones, vectores unidimensionales e imágenes. |

---

<sup>1</sup><https://github.com/RoboticsLabURJC/2017-tfm-nuria-oyaga/tree/master/Generator>

<b><i>motion_type</i></b>	Especifica el tipo de movimiento, la dinámica que sigue el objeto entre cada <i>frame</i> , explicadas en el apartado 3.1.3.
<b><i>height</i></b>	Número entero que indica la altura de la imagen, su dimensión <i>y</i> . En este trabajo se utiliza una altura de 80 píxeles.
<b><i>width</i></b>	Número entero que indica el ancho de la imagen, su dimensión <i>x</i> . En este trabajo se utiliza un ancho de 120 píxeles.
<b><i>object</i></b>	Define el tipo de objeto que se mueve en la imagen. En este trabajo únicamente se utiliza el objeto píxel, de tal forma que toda la imagen será negra a excepción de un punto activo que será considerado el objeto.
<b><i>obj_color</i></b>	Permite definir el color del objeto. Se utiliza un único nivel de intensidad en el caso del píxel y una terna RGB para otros objetos.
<b><i>dof</i></b>	Establece los grados de libertad de la dinámica de movimiento, es decir, el número de variables que se generan de forma aleatoria en la ecuación que la define.
<b><i>n_samples</i></b>	Número entero que indica el número de muestras totales que se generarán en el conjunto.
<b><i>n_points</i></b>	Número entero que indica la cantidad de instantes de tiempo que serán utilizados para predecir.
<b><i>gap</i></b>	Número entero que define la separación temporal, instantes de tiempo, entre la última muestra conocida y la muestra a predecir.
<b><i>noise</i></b>	Permite añadir ruido de un determinado tipo a las muestras para un estudio más avanzado.
<b><i>split</i></b>	Indica si se dividirá el conjunto en los tres subconjuntos ( <i>train</i> , <i>validation</i> , <i>test</i> ) y la proporción en la que lo hará si así de define.

De esta manera, se permite un ajuste más sencillo del *dataset* a generar al tipo de problema que se quiera estudiar, dejando al gusto del usuario las características del mismo.

### 3.2.2. Generación del *dataset*

En la Figura 3.9 se expone el diagrama del flujo que sigue el programa encargado de la generación del *dataset*, centrándose en el tipo de dato que es de interés en este trabajo: los fotogramas.

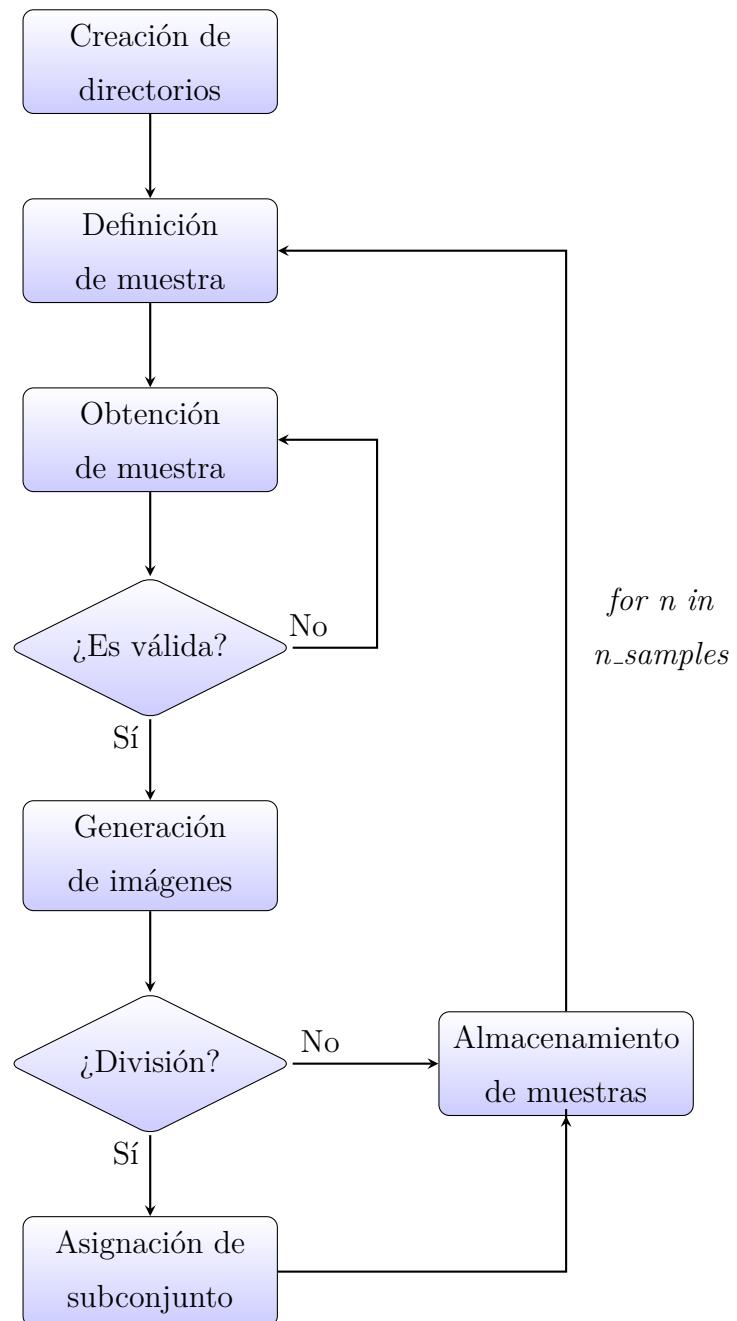


Figura 3.9: Diagrama de flujo del generador.

En primer lugar se comprueban y se crean, en caso de ser necesario, los directorios que son necesarios para el almacenamiento del conjunto según la ruta que ha sido indicada en el fichero de configuración. La estructura de ficheros que se crea con el fin de almacenar todas las muestras del conjunto es explicada en el apartado 3.1.1.

Tras la creación de la estructura necesaria se entra en un bucle, que se itera el número de muestras que haya sido definido, en el que se define, obtiene, valida, genera y almacena cada uno de los ejemplos que conforman el *dataset*.

Para la definición de la muestra se utiliza la clase correspondiente al tipo de movimiento que se vaya a implementar, la cual tiene definida de forma interna la ecuación que se corresponde con la dinámica. Este tipo de dinámica a implementar viene indicada en el fichero de configuración y todas las clases y ecuaciones que son utilizadas se explican en profundidad en el Apartado 3.1.3.

Tras definir la muestra se llama a la función encargada de generarla. A grandes rasgos, esta función obtiene las posiciones  $(x,y)$  del píxel en cada uno de los fotogramas y comprueba que esta posición obtenida no se salga de los límites de la imagen, evitando así occlusiones y desapariciones. Para obtener esas posiciones se realizan varios pasos que, aunque pueden variar en número en función de la dinámica y de los grados de libertad, siguen siempre la misma estructura. Esta estructura queda encapsulada en un bucle *while*, que será detenido una vez se obtenga un conjunto de posiciones válidas, con las siguientes operaciones:

1. Se define el movimiento que sigue el píxel en el eje  $x$ , identificado con el instante temporal  $t$  y, por tanto, con la velocidad a la que se mueve el píxel. Éste es el mismo independientemente de la dinámica con la que se esté trabajando, se corresponde con un MRU y viene definido de la siguiente forma:

```
self.f = lambda t, x0, u_x: x0 + u_x * t
```

Para poder definir el movimiento MRU en los conjuntos desarrollados se fija el punto de inicio ( $x0$ ) en el instante 0 y se asigna una velocidad aleatoria para cada una de las muestras. Esta asignación se realiza mediante el siguiente código:

```
limit = int(self.w / (self.n_points + self.gap))
u_x = random.randint(1, limit)
```

Así se establece un límite en la velocidad, en función del número de instantes de tiempo conocidos y la separación entre el instante a predecir y el último conocido, que asegura la aparición del píxel en todas las imágenes de la muestra.

2. Se obtienen los valores concretos de la posición  $x$  en cada instante de tiempo de la siguiente forma:

```
numbers_x = [self.f(x, x0, u_x) for x in range(self.n_points)]
numbers_x.append(self.f(self.n_points + self.gap - 1, x0, u_x))
```

Primero se calculan todos los valores de los instantes de tiempo conocidos ( $n\_points$ ) y, posteriormente, se añade el valor correspondiente con el instante a predecir ( $n\_points+gap$ ).

3. Se define el valor de todas las variables que son necesarias para aplicar la ecuación de la dinámica y obtener las posiciones en el eje  $y$ . La forma de definir estas variables, explicadas en el Apartado 3.1.3, dependen del grado de libertad que se haya indicado en el fichero de configuración. A medida que aumenta la complejidad aumenta el número de variables que se generan de forma aleatoria.
4. Se calculan las posiciones en el eje  $y$ , en función de la dinámica que se haya establecido, mediante la siguiente línea:

```
numbers_y = [self.g(n_x, *args) for n_x in numbers_x]
```

Los argumentos que se pasan a la función de la dinámica serán las variables que previamente han sido definidas.

5. Se comprueba que todas las posiciones se encuentren dentro de la imagen y, en caso positivo, se detiene el bucle y se continúa con el flujo del programa de generación. Si es negativo se repetirá el proceso hasta alcanzar un conjunto de posiciones válido.

Una vez se han obtenido las posiciones del píxel válidas se procede a la generación de la propia secuencia de imágenes. Para ello se recorren todas las posiciones que han sido

generadas ( $n\_points + 1$ ) y se crea una imagen por cada una de ellas. La imagen se inicia con todos los píxeles nulos, completamente negra, y posteriormente se activa el píxel que se corresponde con la posición  $(y,x)$  con el valor indicado en el fichero de configuración, para este proyecto fijado en 255. Se debe trasponer el vector de posición por la forma en la que Numpy establece el acceso a las posiciones en las imágenes<sup>2</sup>, primero las filas ( $y$ ) y luego a las columnas ( $x$ ).

Antes de almacenar la muestra, si se ha establecido la división del *dataset* en los subconjuntos de entrenamiento, validación y *test*, se asignará la misma al conjunto al que pertenece. Para esta asignación se utiliza un criterio de orden de la siguiente forma:

- $n\_sample < n\_train \Rightarrow$  Conjunto de entrenamiento
- $n\_train \leq n\_sample < n\_train + n\_test \Rightarrow$  Conjunto de *test*
- $n\_train + n\_test \leq n\_sample < n\_train + n\_test + n\_val \Rightarrow$  Conjunto de validación

Los valores que definen el número de muestras que tiene cada conjunto vienen dados por los distintos parámetros de división establecidos en el fichero de configuración. El código empleado para calcular estos valores es el siguiente:

```
n_test = int(n_samples * float(conf['split']['fraction_test']))
n_val = int(n_samples * float(conf['split']['fraction_validation']))
n_train = n_samples - n_val - n_test
```

En los conjuntos desarrollados para este trabajo, el *flag* que indica la división estará siempre activado, pues se hace uso de los tres subconjuntos diferenciados en las distintas fases de la investigación. Por lo tanto, el flujo del generador siempre será el mismo independientemente del tipo de dato que se esté generando, pasando la muestra por el proceso de asignación correspondiente.

Para el almacenamiento de la muestra en la carpeta correspondiente, la del subconjunto si se ha hecho división o la general en caso contrario, se realizan tres operaciones que permiten obtener información de la misma en distintos formatos.

---

<sup>2</sup>[https://scikit-image.org/docs/dev/user\\_guide/numpy\\_images.html](https://scikit-image.org/docs/dev/user_guide/numpy_images.html)

- ***parameters.txt***: Se escriben las distintas variables utilizadas para la generación de cada muestra. Cada fila de este fichero se corresponde con un ejemplo generado.
- ***modeled\_samples***: Se guarda un fichero para cada muestra con las posiciones de la misma.
- ***raw\_samples***: Se almacena una carpeta por muestra con las imágenes que la conforman.

Con todo esto se obtiene un directorio, cuya estructura se ha explicado más a fondo en el apartado 3.1.1, que contiene todo lo necesario para entrenar y evaluar las distintas redes que se estudian en el desarrollo de este proyecto.

# Capítulo 4

## Figuras de mérito y evaluación

Para evaluar de modo fiable la calidad de las redes neuronales profundas como predictores visuales se ha desarrollado una herramienta *software* que obtienen unas figuras de mérito objetivas. El proceso de evaluación aplica las redes sobre los conjuntos de datos de test supervisados que se han descrito en el Capítulo 3, computa las figuras de mérito elegidas y las representa gráficamente para un mejor análisis. La comparación de estas gráficas de las distintas redes estudiadas es lo que permitirá extraer conclusiones sobre distintos parámetros.

En este capítulo se explican dichas figuras de mérito y se realiza un análisis en profundidad del código Python desarrollado para la evaluación, realizando también una interpretación sobre los gráficos y resultados que arroja dicho código al terminar la ejecución.

### 4.1. Figuras de mérito

A continuación se describen las figuras de mérito que se han utilizado para evaluar las prestaciones de las redes, que se aplican tanto a imágenes modeladas como crudas. Para obtener conclusiones sobre la capacidad predictiva de las redes estudiadas se realiza una comparación entre dichas medidas. Estas comparaciones permiten obtener conclusiones sobre parámetros como el número de muestras utilizadas para entrenar, la complejidad de la dinámica o el horizonte temporal de predicción. Dichas conclusiones serán las que ayuden a tomar decisiones en la mejora del entrenamiento y la estructura de las redes para obtener la mejor red posible para la predicción.

### Distancia entre píxeles

La medida que se utiliza para cuantificar el error cometido en cada una de las predicciones es la distancia Euclídea<sup>1</sup> entre el píxel predicho ( $p$ ) y el real ( $q$ ). En dos dimensiones, su expresión viene dada por:

$$d_E(p, q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2}$$

Además, para un análisis más profundo, esta distancia se desglosa en cada una de las dimensiones de la imagen,  $x$  e  $y$ , simplemente calculando la diferencia en términos absolutos del valor en dichas coordenadas:

$$d_{E,dim}(p_{dim}, q_{dim}) = |p_{dim} - q_{dim}|$$

Las distancias anteriores cuantifican el error absoluto cometido en el plano de la imagen y sus dimensiones  $x$  e  $y$ . Para analizar de una forma más completa la bondad de la red, se considera también el error relativo a la a la máxima distancia entre dos píxeles de la imagen, así como distintos estadísticos sobre las medidas de distancia.

### Distancia relativa

La distancia absoluta puede dar una visión incompleta sobre el fallo que se ha cometido. Puesto que no es lo mismo desviarse una distancia de 5 píxeles en una imagen de 5x5 que en una de 1920x1080, se hace uso del error relativo<sup>2</sup>. La medida relativa ( $\eta$ ) normaliza la distancia absoluta ( $\epsilon$ ) respecto a la máxima distancia en la imagen ( $v$ ), haciendo uso de la siguiente fórmula:

$$\eta = \frac{\epsilon}{v}$$

En el caso de las imágenes, el máximo error a cometer es la mayor distancia entre dos píxeles de la imagen. Para una imagen de altura  $h$  y anchura  $w$ , la máxima distancia se corresponde con su diagonal, cuyo valor se obtiene de la siguiente manera:

$$v = \sqrt{h^2 + w^2}$$

---

<sup>1</sup>[https://en.wikipedia.org/wiki/Euclidean\\_distance](https://en.wikipedia.org/wiki/Euclidean_distance)

<sup>2</sup>[https://en.wikipedia.org/wiki/Approximation\\_error](https://en.wikipedia.org/wiki/Approximation_error)

La distancia relativa se suele indicar de forma porcentual, por lo que el valor calculado se multiplica por 100 para su expresión como porcentaje:

$$\delta = \eta * 100 \%$$

La distancia relativa consigue normalizar la medida para una comparación más justa entre imágenes de distinto tamaño, permitiendo tener una idea más realista del fallo que se está cometiendo.

### Estadísticos

Para que la medición de la calidad de las predicciones sea fiable típicamente se aplica la red neuronal sobre un conjunto de secuencias, no sobre una única, es decir sobre varias secuencias supervisadas de fotogramas, que incluyen el fotograma verdadero en el tiempo de predicción. A partir de las medidas anteriores en dicho conjunto se obtienen una serie de estadísticos que permiten comparar las prestaciones de distintas redes y extraer conclusiones. A continuación se describen los estadísticos considerados.

- **Máximo:** Se trata del error máximo cometido sobre todas las secuencias analizadas. Este valor identifica la imagen para la que más distancia existe entre la posición real y la predicha, que será uno de los elementos a tener en cuenta para examinar la bondad de la red.
- **Media:** Se trata de la media de los vectores de error resultantes, absoluto y relativo, para obtener un único valor que permita hacer comparaciones rápidas.

### Formas de representación

Para un mejor análisis de las figuras de mérito y los estadísticos presentados, se crean una serie de representaciones gráficas que contienen la información relevante para la comparación. Estas representaciones tienen su base en dos tipos:

- **Histograma**<sup>3</sup>: Se trata de un gráfico de barras que enfrenta la frecuencia con la que se comete un error con el valor del mismo, agrupado en intervalos para una representación más compacta.

---

<sup>3</sup><https://en.wikipedia.org/wiki/Histogram>

- **Boxplot**<sup>4</sup>: Se trata de un diagrama de caja y bigotes en el que la caja representa los cuartiles y las líneas que se extienden desde ella indican variabilidad fuera de los cuartiles superior e inferior. Por otro lado, los valores atípicos u *outliers* se representan como puntos individuales dispersos por el eje vertical de la caja.

Todas las representaciones se combinan en una única figura, similar a la representada en la Figura 4.1, que se divide en cuatro regiones: una para el histograma del error absoluto, otra para el del relativo, una tercera para el *boxplot* y la última con información de la media y el máximo.

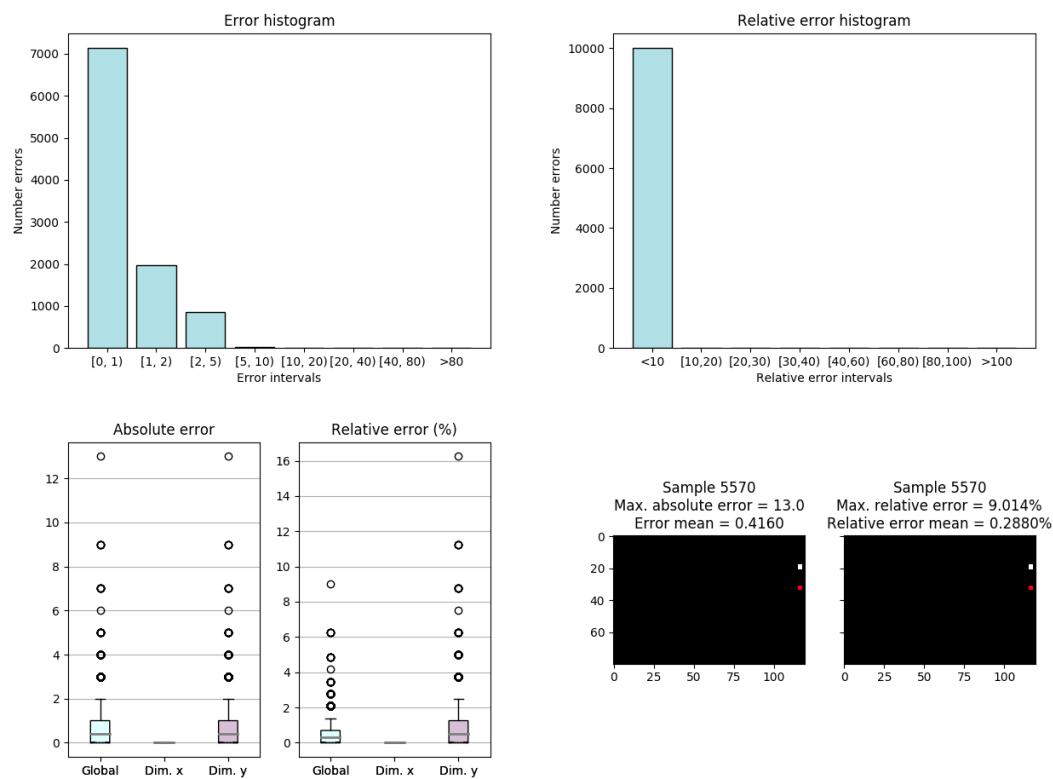


Figura 4.1: Ejemplo de gráfica con la distribución de distintas figuras de mérito y estadísticos asociados.

Con esta figura se representa de una forma resumida y clara toda la información referente las prestaciones de la red, a su calidad como predictor visual, facilitando la posterior tarea de análisis.

---

<sup>4</sup>[https://matplotlib.org/3.3.1/api/\\_as\\_gen/matplotlib.pyplot.boxplot.html](https://matplotlib.org/3.3.1/api/_as_gen/matplotlib.pyplot.boxplot.html)

## 4.2. Metodología de evaluación

Para el cálculo de las medidas y la obtención de la Figura 4.1 se ha desarrollado un código en Python sobre el que se profundiza en la siguiente sección.

En la Figura 4.2 se muestra el flujo que sigue el código para evaluar una red concreta con un conjunto de *test* determinado. El programa sigue un flujo muy sencillo y lineal cuyo resultado son la Figura 4.1 y los valores asociados a las figuras de mérito que permiten la comparación de las distintas redes.

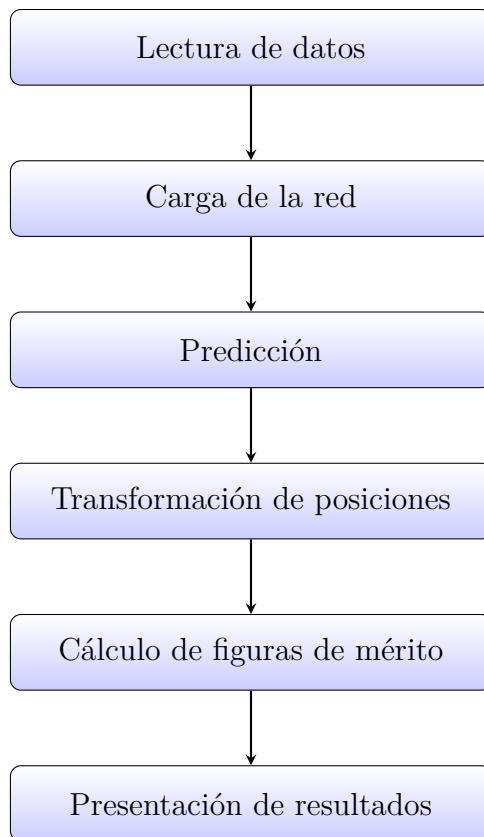


Figura 4.2: Diagrama de flujo del evaluador.

En primer lugar se realizan las operaciones de lectura del conjunto de *test* y carga de la red en memoria, cuyas rutas se obtienen del fichero de configuración correspondiente, para posteriormente pasar por la red los datos y obtener sus predicciones.

Puesto que las posiciones que se obtienen a la salida de la red tienen un formato diferente al de las posiciones reales, para obtener las figuras de mérito es necesario realizar

una transformación previa. Esta transformación dependerá del tipo de datos, modelados o crudos, que se estén tratando, ya que el problema se afronta de manera distinta en función de ello.

Cuando se trata con imágenes modeladas, se realiza una tarea de regresión. Es por ello que la salida de la red es un par de valores que corresponden a las coordenadas de la posición estimada. En este caso, la única acción a realizar es el redondeo de la estimación, para obtener un número entero. El procesamiento en el código queda de la siguiente manera:

```
p = np.round(p).astype(np.float64)  
predict_pos.append(p)
```

En el caso de trabajar con imágenes en crudo, el problema de predicción se aborda como un problema de clasificación binaria (“0” ó “1”) donde la salida es un vector de longitud  $h \times w$ , con la codificación de la clase en cada píxel. En este caso se redimensiona el vector de salida al del tamaño de la imagen, obteniendo la posición del píxel activo de la siguiente forma:

```
p = p.reshape(dim)  
predict_pos.append(np.unravel_index(p.argmax(), p.shape))
```

Una vez obtenidas las posiciones predichas y reales en el mismo formato, se obtienen las figuras de mérito mediante el siguiente proceso:

1. Se calcula, para cada imagen a predecir, la distancia entre la posición real del píxel activo y la estimada, es decir, el error absoluto.
2. Se calcula el error relativo cometido, sobre el plano de imagen y las dimensiones  $x$  e  $y$ , en cada una de las imágenes a predecir.
3. Se calculan los estadísticos para cada uno de los tipos de error obtenidos.
4. Se crean los histogramas asociados al error absoluto y relativo en el plano de imagen.
5. Se obtiene la imagen de máximo error, tanto absoluto como relativo, en el plano de imagen. Tras su identificación, se utilizan distintos colores para representar ambas posiciones (predicha y real) en una única figura.

6. Se crea el *boxplot* para los errores sobre el plano de la imagen, la dimensión *x* y la dimensión *y*.

Finalmente, tras el cálculo de las figuras de mérito y la creación de los gráficos que facilitarán la tarea de análisis, se almacenan los resultados en dos formatos distintos:

- El fichero *error\_result.txt* almacena, para cada una de las imágenes predichas, el número que la identifica, la posición real, la posición predicha y los errores absoluto y relativo.
- Una figura que aúna cuatro gráficos representativos: los histogramas de los errores absolutos y relativos de cada imagen a predecir, el *boxplot* de ambos tipos de error y la representación de la imagen de máximo error.

Con esta metodología para evaluar las prestaciones de una red se dispone de una forma objetiva de comparación entre redes, permitiendo extraer conclusiones sobre distintos aspectos de las mismas.

# Capítulo 5

## Predicción con imágenes modeladas

En este capítulo se presentan todos los estudios realizados en cuanto a la predicción en imágenes modeladas, con el objetivo de explorar distintas estructuras para atacar el problema. En particular, se quiere analizar el aporte de las redes recurrentes, que incorporan cierta memoria, a la solución.

En el mundo de las imágenes modeladas se ha afrontado la predicción como un problema de regresión, en el que la entrada es el conjunto de instantes de tiempo que se consideran conocidos (*n-points*) con sus pares de posiciones ( $x, y$ ) y la salida par de coordenadas en el instante de tiempo futuro. Cada una de estas coordenadas pueden tomar cualquier valor numérico decimal por lo que, como se explicó en la Sección 4.2, es necesario realizar un redondeo para obtener la posición final, los píxeles están siempre representados por números enteros.

Para abordar la capacidad de predicción de las distintas redes propuestas se han utilizado los conjuntos generados, cuyas muestras siguen la estructura definida en la Figura 3.2, manteniendo fijos algunos parámetros:

- El número de instantes temporales conocidos (*n-points*) es en todos los casos 20.
- La separación (*gap*) entre la última muestra conocida y la que se quiere predecir se establece en 10 instantes temporales.
- La división del conjunto se realiza con el 80 % de las muestras para el entrenamiento, el 10 % para validación, utilizado para el *early stopping*, y el 10 % restante para el *test*.

A continuación se realiza el recorrido por las distintas estructuras neuronales entrenadas, presentando los resultados obtenidos y las conclusiones que dan lugar a la exploración de distintas vías para la mejora de los mismos.

## 5.1. Arquitectura no recurrente: Perceptrón Multicapa

La primera aproximación para abordar la predicción de las imágenes modeladas considera una arquitectura de MLP con una única capa oculta, explicado en detalle en el Apartado 1.2.1.1. En la Figura 5.1 se muestra un esquema que ilustra la arquitectura del MLP utilizado para predecir la posición del píxel móvil en este formato de imagen. Esta estructura tiene como entrada 20 pares de valores ( $x, y$ ) que alimentan una única capa oculta con 10 neuronas, proporcionando a la salida un par de valores que se corresponde con la posición estimada.

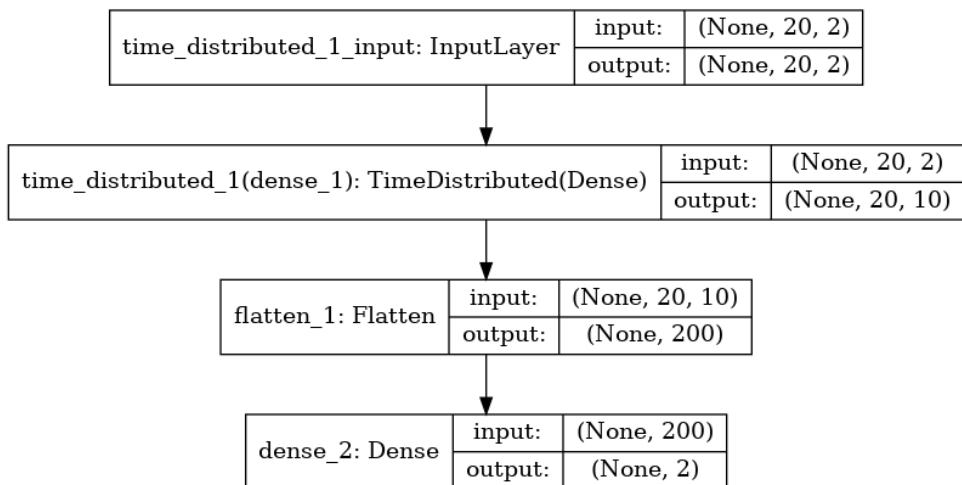


Figura 5.1: Estructura de MLP con 1 capa oculta y 10 neuronas para imágenes modeladas.

Tras la definición de la red, se realizan diversos experimentos con el objetivo de lograr la predicción en el escenario más complejo considerado. Se aumenta de forma progresiva la complejidad en el movimiento, considerando en primera instancia más grados de libertad, y posteriormente otras dinámicas.

### 5.1.1. Influencia del número muestras

Antes de comenzar con el estudio de las distintas redes con las dinámicas consideradas se realiza un breve estudio sobre el número de muestras utilizando el caso especial de la dinámica lineal, con pendiente nula, por su sencillez. Bajo esta premisa, se emplea en primer lugar un conjunto de 1000 muestras, de las cuales 800 se emplean para entrenamiento, 100 para validación y las 100 restantes para *test*.

En la Figura 5.2 se muestran los resultados obtenidos al evaluar la red entrenada con el conjunto cuya altura inicial del píxel en la imagen permanece fija.

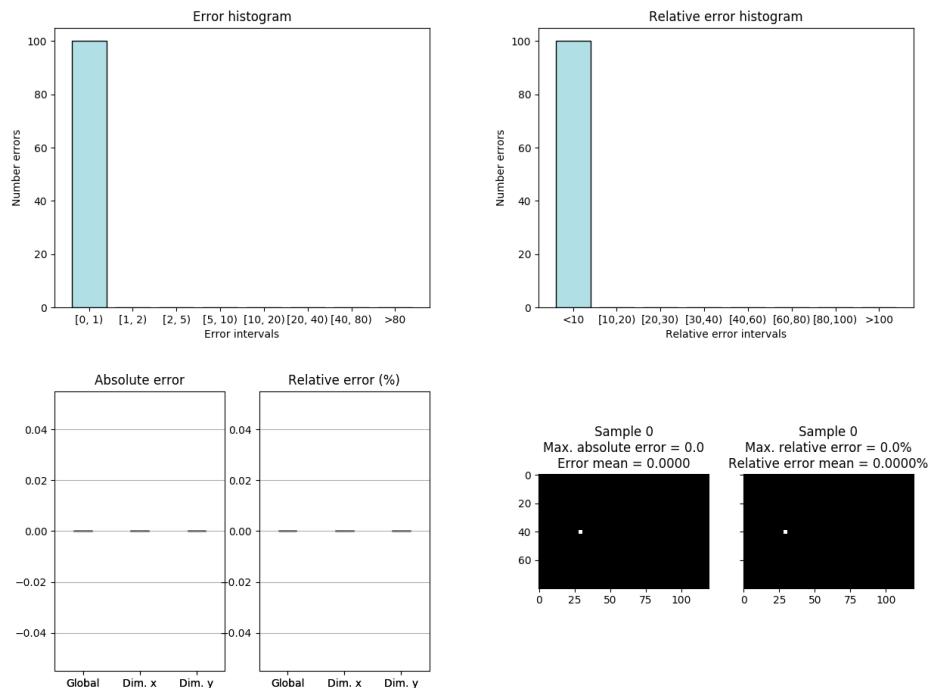


Figura 5.2: Resultados de MLP con dinámica lineal, pendiente nula y altura fija en imágenes modeladas (100 muestras de *test*)

El error obtenido en este caso es nulo ya que, al mantener fija la altura y aplicar las restricciones de aparición del píxel en todas la imágenes, el número de muestras distintas es mucho menor a las 1000 creadas. Este hecho hace que la red aprenda todos los ejemplos posibles en el entrenamiento, evitando que se pueda equivocar en cualquiera de los ejemplos de *test*.

Con el objetivo de incrementar la complejidad de la dinámica, se entrena la misma estructura de red con el mismo número de muestras pero dejando la altura inicial como parámetro libre en cada predicción. Los resultados de la evaluación de esta nueva red, que se muestran en la Figura 5.3, ya no son tan buenos como en el caso anterior. La razón es que, al dejar que la altura inicial tome un valor aleatorio ha aumentado la variabilidad de las muestras, impidiendo que la red vea la totalidad de las variantes en la fase de entrenamiento, presentando casos desconocidos en la evaluación.

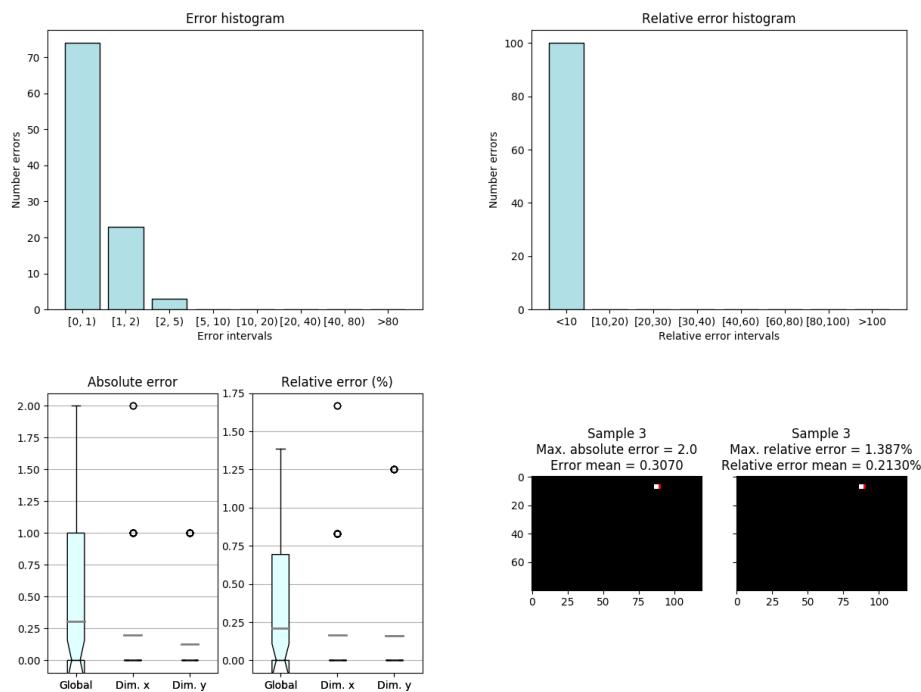


Figura 5.3: Resultados de MLP con dinámica lineal, pendiente nula y altura aleatoria en imágenes modeladas (100 muestras de *test*).

La primera acción para tratar de mejorar los resultados es aumentar el número de muestras para incrementar el número de ejemplos de los que la red aprende. En esta línea, se genera un nuevo conjunto de 5000 muestras con la misma proporción en la generación de las tres particiones: 4000 destinadas al entrenamiento, 500 a la validación y otras 500 al *test*. Los resultados de la red entrenada con este nuevo conjunto se muestran en la Figura 5.4, en la que se puede comprobar que el error vuelve a ser nulo.

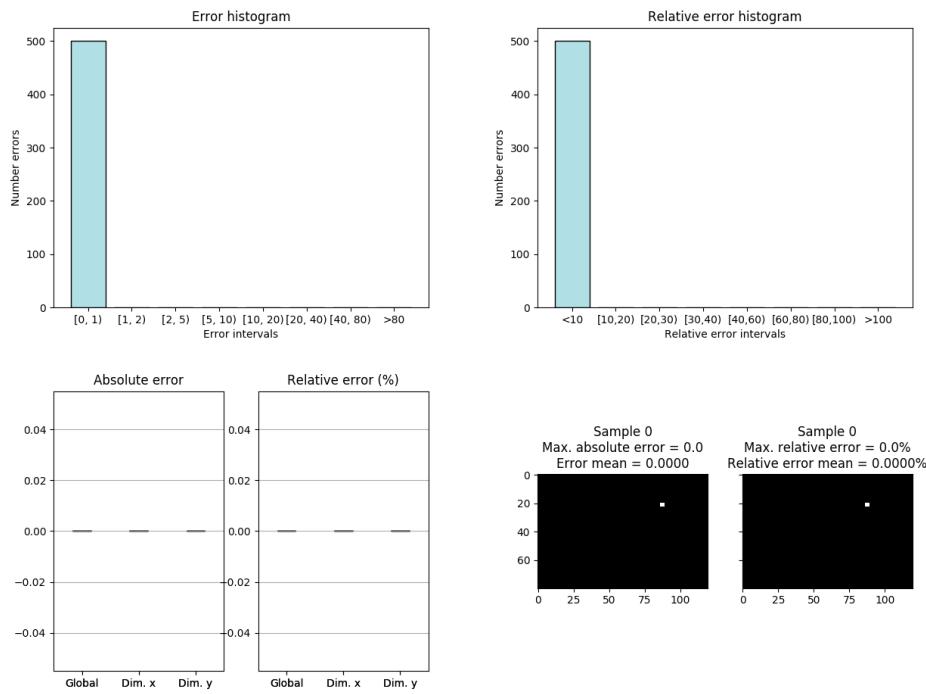


Figura 5.4: Resultados de MLP con dinámica lineal, pendiente nula y altura fija en imágenes modeladas (500 muestras de *test*).

Con los resultados obtenidos se concluye que utilizar un mayor número de muestras es un factor que hace mejorar los resultados, pues la red aprende de una mayor variedad de ejemplos. Sin embargo, cuando se obtiene un acierto del 100 %, como en el primer caso, no hay margen de mejora y únicamente se consigue aumentar la complejidad computacional. Es decir, el aumento de muestras es un factor de mejora siempre que la naturaleza de las mismas sea lo suficientemente compleja como para necesitar un mayor número de observaciones para modelar la estadística subyacente en los datos.

### 5.1.2. Predicción con dinámicas lineales

Para abordar la predicción de fotogramas que representan objetos siguen una dinámica lineal se emplea un conjunto con un total de 10000 muestras dividido en 8000 muestras para entrenamiento, 1000 para validación y 1000 para evaluación.

En primer lugar se realiza el estudio de esta dinámica con un único DOF, la pendiente de la trayectoria del píxel activo, cuyos resultados de evaluación de la red entrenada queda reflejado en la Figura 5.5.

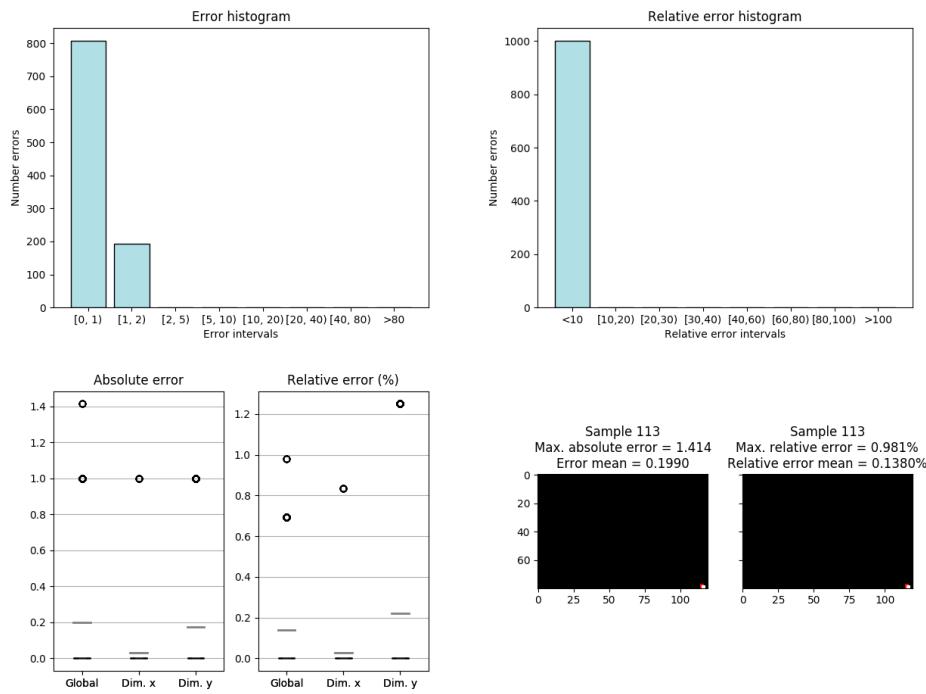


Figura 5.5: Resultados de MLP con dinámica lineal de 1 DOF en imágenes modeladas (1000 muestras de *test*).

Se puede comprobar que la capacidad de predicción de esta red es buena, ofreciendo un error relativo muy reducido tanto en términos de media como de máximo.

Se aumenta el grado de complejidad de la dinámica con un nuevo DOF, la altura inicial, y se repite el experimento con la misma estructura de red, cuyos resultados se reflejan en la Figura 5.6. Como era de esperar, dejar mayor libertad de movimiento al píxel activo, aumentando la complejidad, repercute de forma directa en la capacidad de predicción de la red. Sin embargo, los resultados obtenidos indican que la red continúa siendo capaz de predecir bien bajo estas condiciones.

Dados los resultados para el caso más complejo de la dinámica lineal, 2 DOF, se concluye que, con el número de muestras modeladas establecidas y la estructura de red propuesta, es posible predecir satisfactoriamente la posición de un píxel móvil que sigue una dinámica lineal, independientemente de sus DOF.

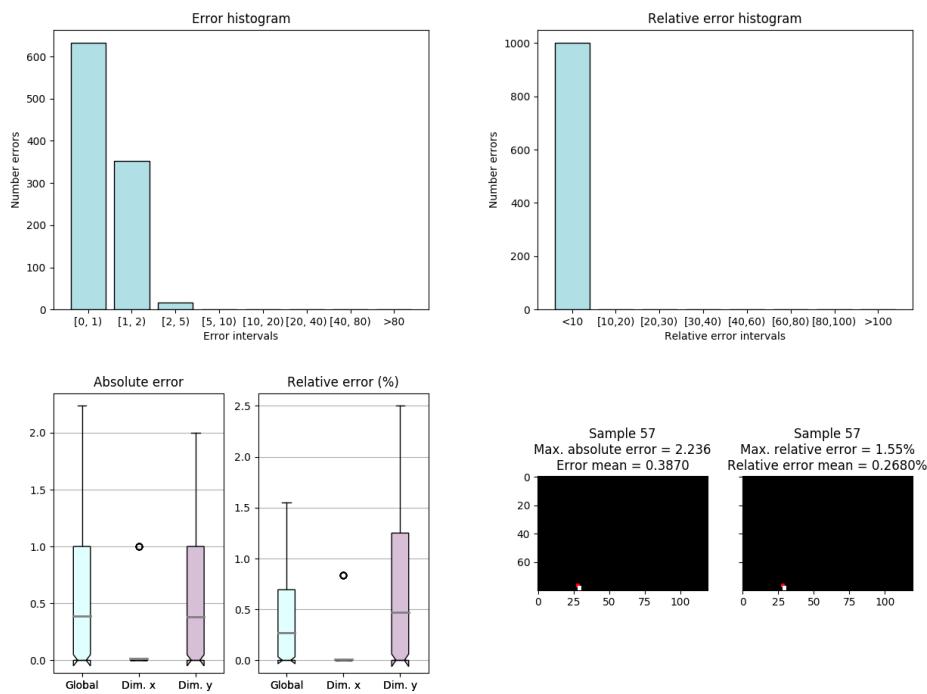


Figura 5.6: Resultados de MLP con dinámica lineal de 2 DOF en imágenes modeladas (1000 muestras de *test*).

### 5.1.3. Predicción con dinámicas parabólicas

Para abordar la predicción de fotogramas que representan objetos siguen una dinámica parabólica se utiliza un conjunto con 10000 ejemplos, divididos de la misma manera que en la dinámica anterior.

El análisis de la capacidad predictiva de la red comienza con el caso más sencillo, un único DOF, el valor de  $a$  en esta dinámica. La Figura 5.7 muestra los resultados obtenidos al evaluar la red entrenada bajo estas condiciones. Estos resultados demuestran que la red propuesta es capaz de predecir bien la posición del píxel móvil en este primer caso.

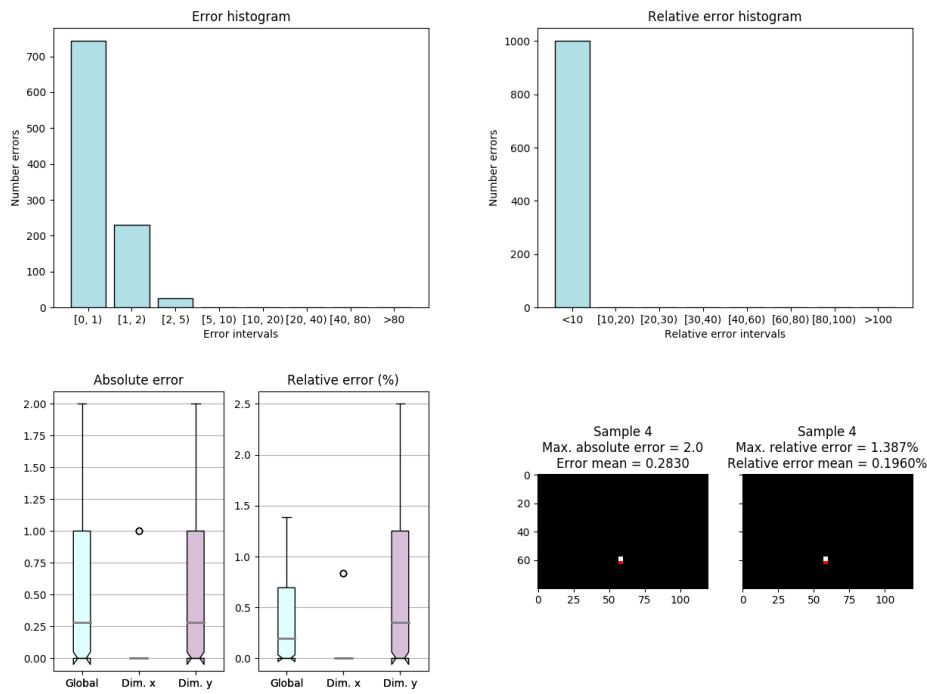


Figura 5.7: Resultados de MLP con dinámica parabólica de 1 DOF en imágenes modeladas (1000 muestras de *test*).

Tras los buenos resultados obtenidos con el primer caso de la dinámica, 1 DOF, se prosigue con el aumento del los DOF para poner a prueba la estructura.

En las Figuras 5.8 y 5.9 se muestran los resultados obtenidos para las predicción de la dinámica con 2 y 3 DOF, la altura inicial y  $c$ , respectivamente. A la vista de estos resultados se comprueba que, a pesar de que los resultados empeoran ligeramente al introducir más grados de libertad, éste se mantiene en unos valores razonables. Se concluye que, para imágenes modeladas, el MLP es capaz de predecir el desplazamiento del píxel según la dinámica parabólica, independientemente del nivel de complejidad que se establezca.

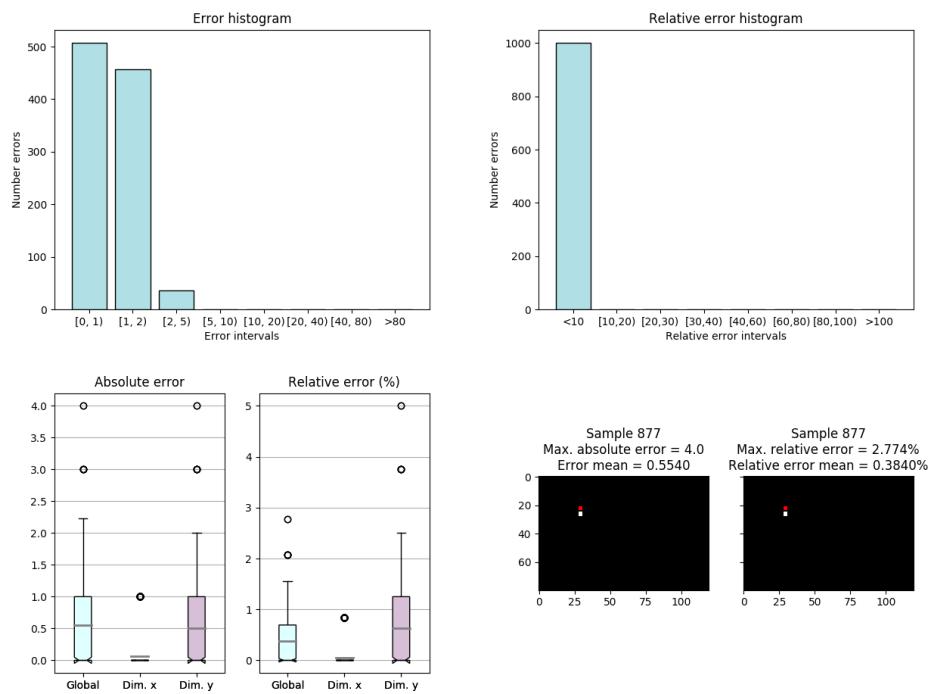


Figura 5.8: Resultados de MLP con dinámica parabólica de 2 DOF en imágenes (1000 muestras de *test*).

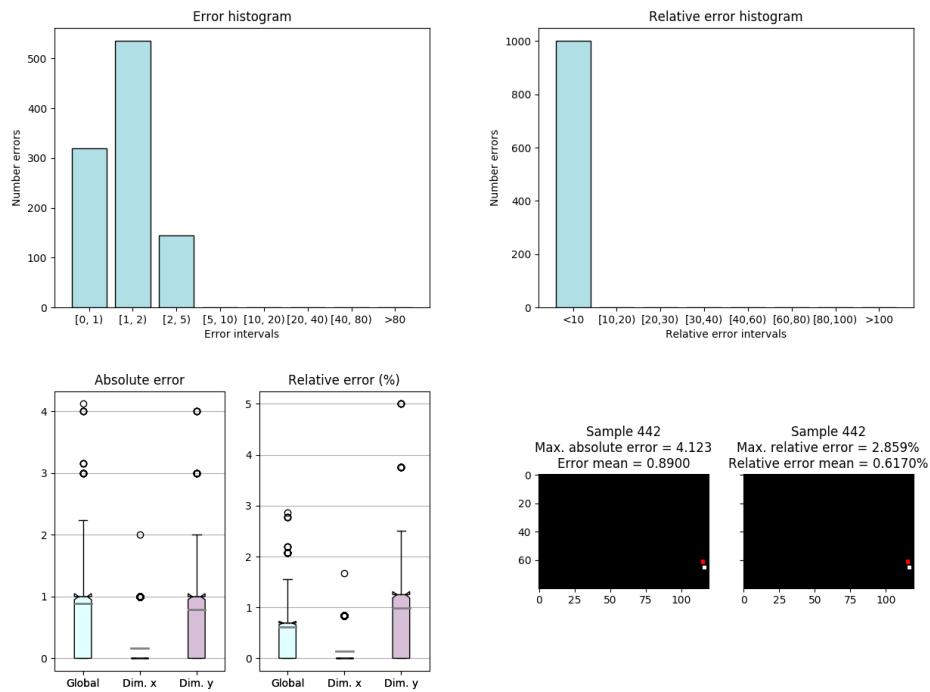


Figura 5.9: Resultados de MLP con dinámica parabólica de 3 DOF (1000 muestras de *test*).

### 5.1.4. Predicción con dinámicas sinusoidales

Para abordar la predicción de fotogramas que representan objetos siguen una dinámica sinusoidal se utiliza un *dataset* compuesto por 10000 muestras, dividido de la misma forma que en dinámicas anteriores. En la Figura 5.10 se muestra la evaluación de la estructura de red propuesta cuando el movimiento del píxel sigue una función sinusoidal con un solo DOF, la frecuencia.

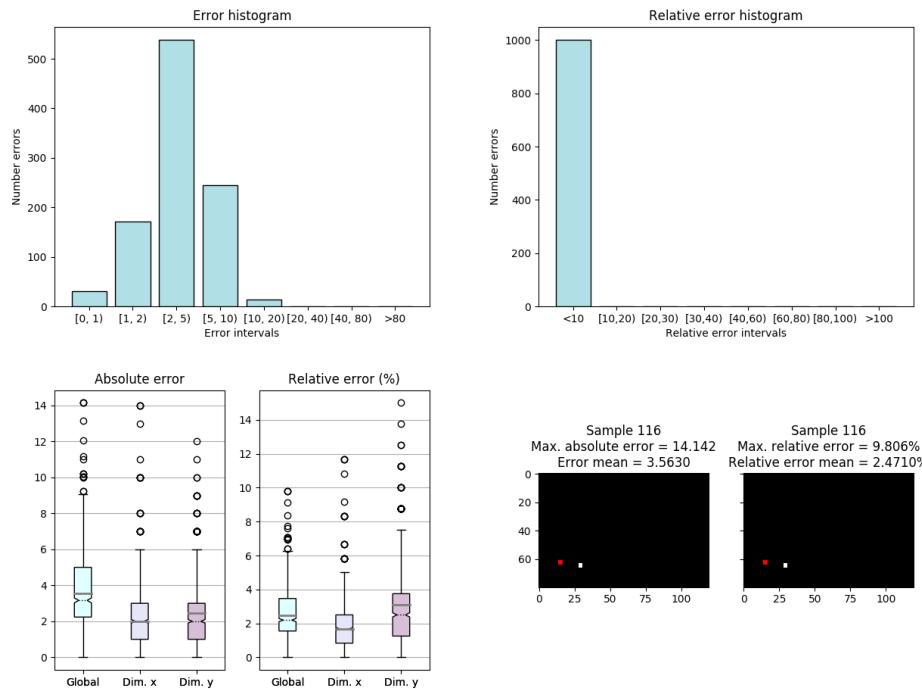


Figura 5.10: Resultados de MLP con dinámica sinusoidal de 1 DOF (1000 muestras de *test*)).

En este caso, los resultados no son tan buenos como los obtenidos para las dinámicas anteriores, que eran más sencillas. Siguiendo la conclusión extraída del Apartado 5.1.1, la solución más inmediata a este deterioro es el aumento de muestras, pasando a un conjunto con un total del 100000 ejemplos: 80000 de entrenamiento, 10000 de validación y 10000 de *test*. Los resultados obtenidos con este nuevo conjunto queda reflejado en los gráficos de la Figura 5.11.

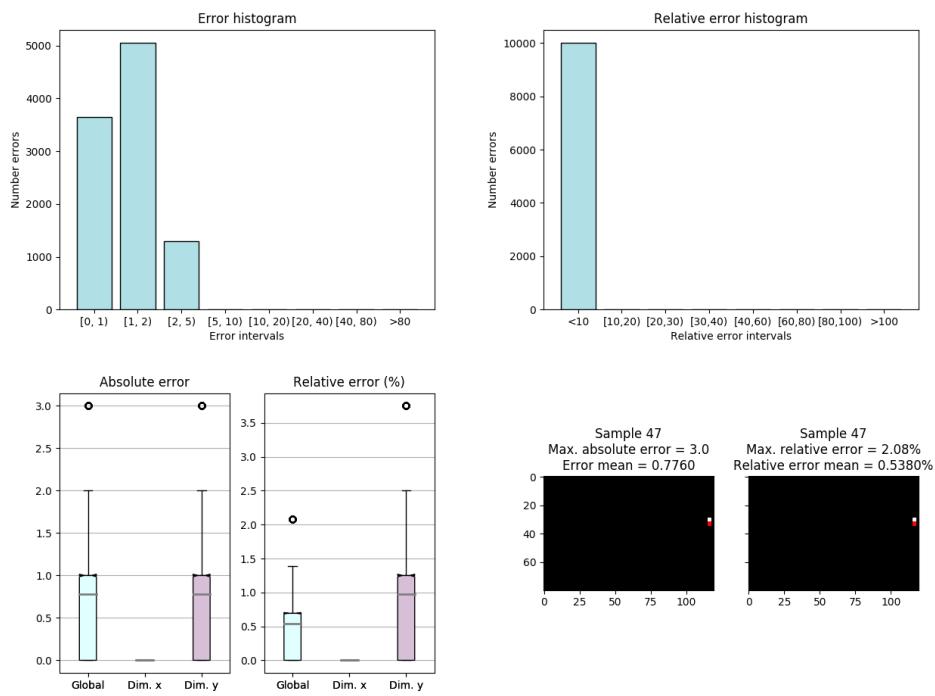


Figura 5.11: Resultados de MLP con dinámica sinusoidal de 1 DOF (10000 muestras de *test*).

Con el incremento en el número de muestras se obtienen mejores resultados en la evaluación, alcanzando la capacidad predictiva conseguida en las dinámicas anteriores.

Manteniendo en 100000 el número de muestras, se aumenta la libertad de movimiento del píxel a 2 DOF, con la altura inicial. La red se entrena y evalúa bajo estas premisas dando lugar a los resultados mostrados en la Figura 5.12. Estos datos hacen ver que el aumento de un grado de libertad en esta dinámica incrementa de forma significativa su complejidad, perdiendo la red la capacidad predictiva alcanzada.

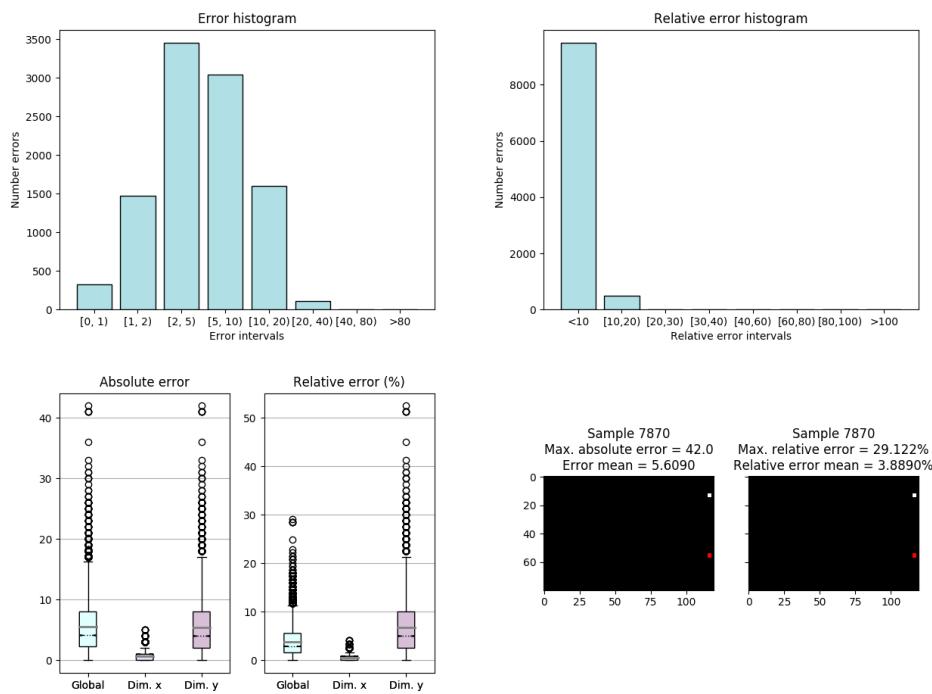


Figura 5.12: Resultados de MLP con dinámica sinusoidal de 2 DOF (10000 muestras de *test*).

Con estos resultados se concluye que la estructura del MLP propuesto no es capaz de abarcar la dinámica sinusoidal en su totalidad, poniendo su límite en un único DOF.

### 5.1.5. Resumen de resultados

En la Tabla 5.1 se muestra un resumen de los resultados alcanzados en cada dinámica.

Dinámica		MLP
Lineal	1 DOF	0.21 %
	2 DOF	0.31 %
Parabólica	1 DOF	0.28 %
	2 DOF	0.42 %
	3 DOF	0.65 %
Sinusoidal	1 DOF	0.54 %
	2 DOF	3.89 %

Tabla 5.1: Promedio del error relativo en *test* al evaluar el MLP con imágenes modeladas y distintas dinámicas (10000 muestras de *test*).

Para la interpretación de esta tabla se ha establecido un código de cuatro colores que indiquen a simple vista los mejores resultados, en términos de promedio y máximo de error relativo, para cada uno de los casos:

- **Verde oscuro:** Indica muy buenos resultados.
- **Verde claro:** Indica resultados satisfactorios con cierto grado de mejoría.
- **Naranja:** Indica resultados intermedio, generalmente con un buen valor promedio pero un máximo elevado.
- **Rojo:** Indica malos resultados.

Además, en el interior de cada celda se refleja el promedio del error relativo de cada uno de los casos para una ilustración cuantitativa de los resultados. Con el objetivo de obtener una evaluación equitativa se han evaluado todas las redes con un conjunto de 10000 muestras, repitiendo la evaluación en aquellas que se utilizaron 1000.

## 5.2. Arquitectura recurrente: LSTM-1

Otro punto de estudio en el trabajo es el análisis de la capacidad de predicción con redes recurrentes, capaces de captar la relación temporal. Más concretamente se ha optado por el uso de las LSTM, explicadas en el Apartado 1.2.2.1, por su uso extendido en problemas de esta tipología. En la Figura 5.13 se muestra la estructura de red creada para el caso más sencillo, en el que se emplea una única capa LSTM de 25 neuronas.

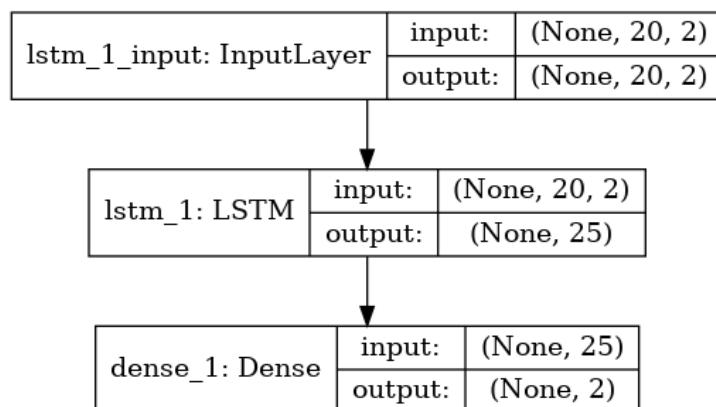


Figura 5.13: Estructura de LSTM-1 para imágenes modeladas.

Las entradas y la salida de esta nueva estructura de red coinciden con las establecidas para el MLP: 20 posiciones como entrada y 1 como salida.

### 5.2.1. Predicción con dinámicas lineales

Al aplicar la red recurrente sobre los *dataset* cuyo movimiento del píxel en la imagen se rige por la función lineal se obtienen unos resultados muy similares que en el caso no recurrente. En las Figuras 5.14 y 5.15 se puede comprobar cómo se obtienen resultados muy similares a los del MLP, mejorando ligeramente los resultados en términos de media.

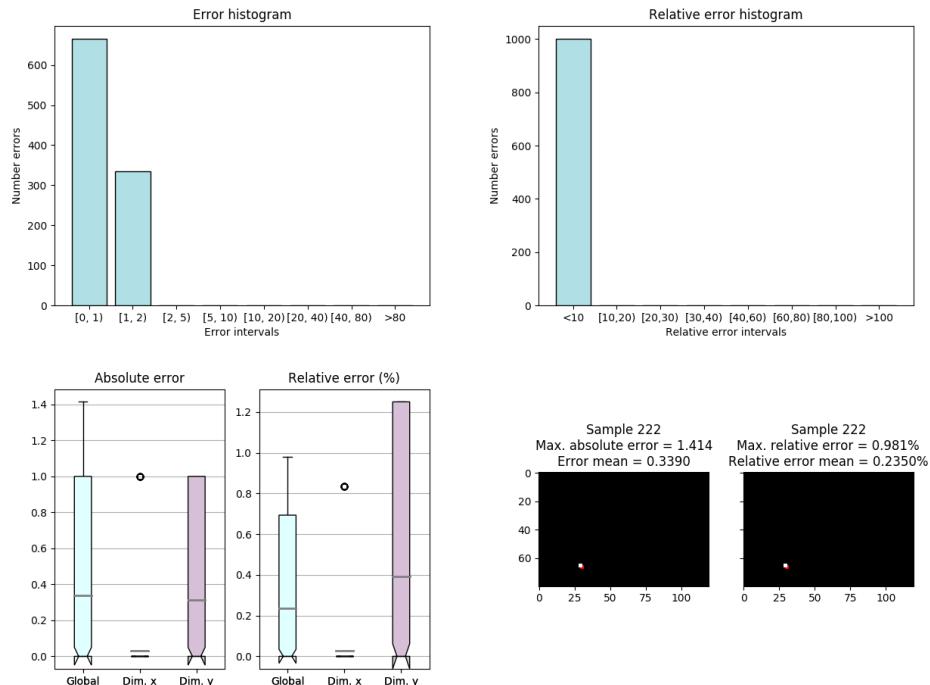


Figura 5.14: Resultados de LSTM-1 con dinámica lineal de 1 DOF (1000 muestras de *test*).

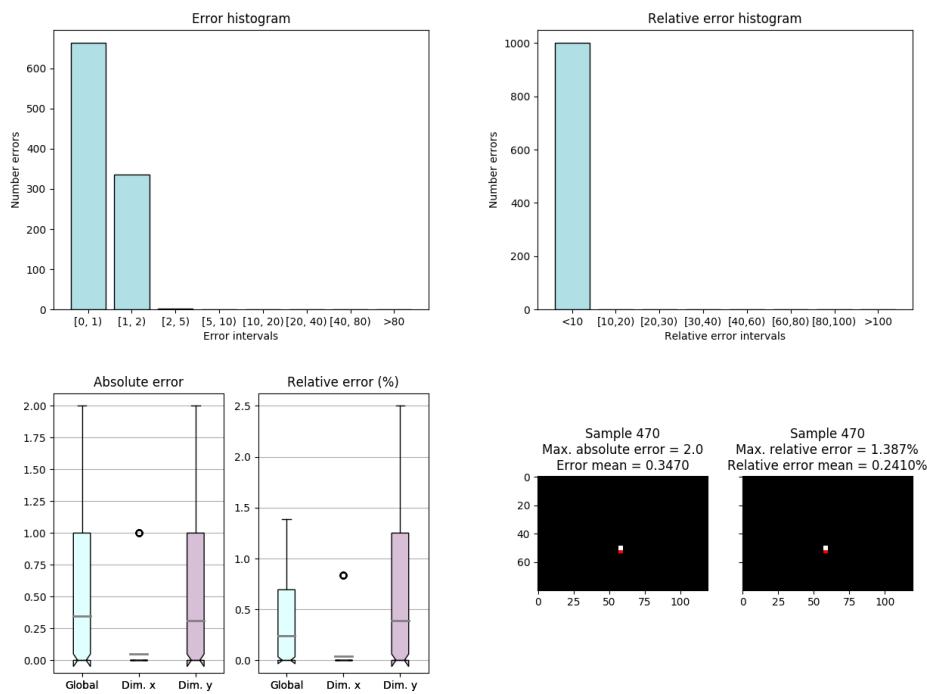


Figura 5.15: Resultados de LSTM-1 con dinámica lineal de 2 DOF (1000 muestras de *test*).

Estos resultados permiten concluir que en las dinámicas cuyos resultados en el MLP son buenos, al utilizar redes LSTM la capacidad predictiva se mantiene. Es decir, las redes recurrentes no empeoran significativamente la predicción cuando ésta funciona sin recurrencia.

### 5.2.2. Predicción con dinámicas parabólicas

El caso parabólico es muy similar al lineal. Con la red no recurrente ya se consigue que la capacidad predictiva sea buena, por lo que introducir la recurrencia no aportará gran diferencia aunque puede mejorar los resultados. En la Tabla 5.2 se muestra una comparación de los resultados obtenidos para esta dinámica con sus 3 DOF y 10000 muestras, tanto con la estructura MLP como con la LSTM.

Estos resultados corroboran lo concluido en el caso lineal: introducir la recurrencia en los casos donde se obtenían buenos resultados, no modifica significativamente los mismos.

DOF	MLP		LSTM-1	
	Max.	Mean	Max.	Mean
1 (a)	1.4 %	0.2 %	1.4 %	0.1 %
2 (c)	2.8 %	0.4 %	2.2 %	0.3 %
3 (b)	2.9 %	0.6 %	2.3 %	0.6 %

Tabla 5.2: Error relativo en la dinámica parabólica con MLP y LSTM-1 (1000 muestras de *test*)).

### 5.2.3. Predicción con dinámicas sinusoidales

En la dinámica sinusoidal, con el MLP, únicamente se conseguía dominar el caso más sencillo de 1 DOF (frecuencia). En la Figura 5.16 se muestran los resultados obtenidos para este caso utilizando la estructura recurrente propuesta. Se mantienen los buenos resultados arrojados por el MLP, reforzando las conclusiones extraídas.

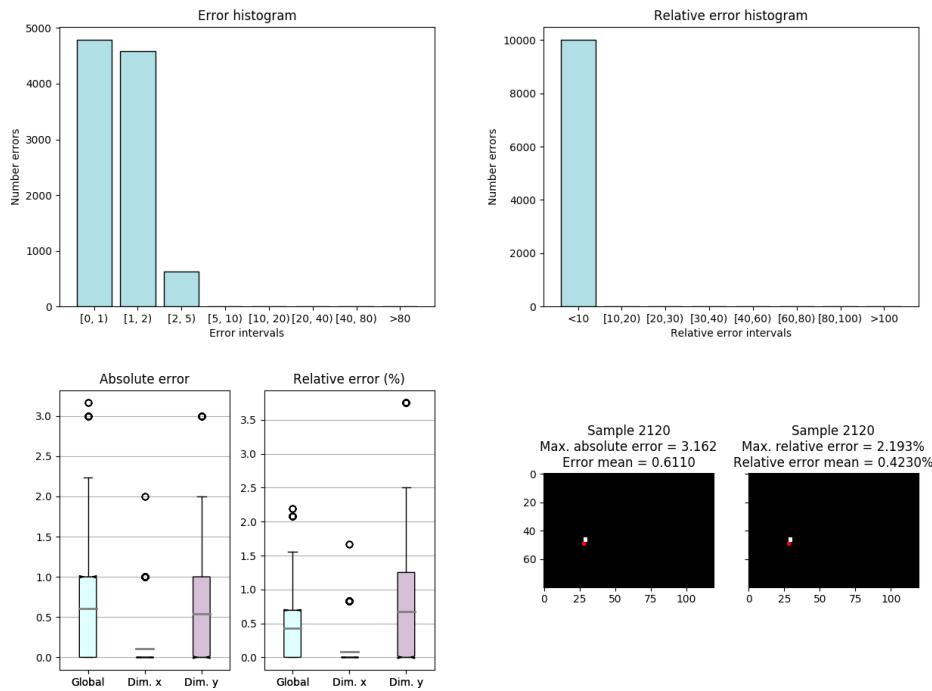


Figura 5.16: Resultados de LSTM-1 con dinámica sinusoidal de 1 DOF (10000 muestras de *test*).

Al complicar la dinámica con el aumento de la libertad de movimiento del píxel, la red no recurrente estudiada no es capaz de predecir correctamente. Se espera que al introducir la recurrencia, no solo se mantengan los resultados como anteriormente, sino que la capacidad predictiva en estos casos mejore considerablemente. En la Figura 5.17 se muestran los resultados del primer experimento en este sentido. Se entrena y evalúa la red recurrente propuesta con un conjunto de 100000 muestras y permitiendo 2 DOF, frecuencia y altura inicial.

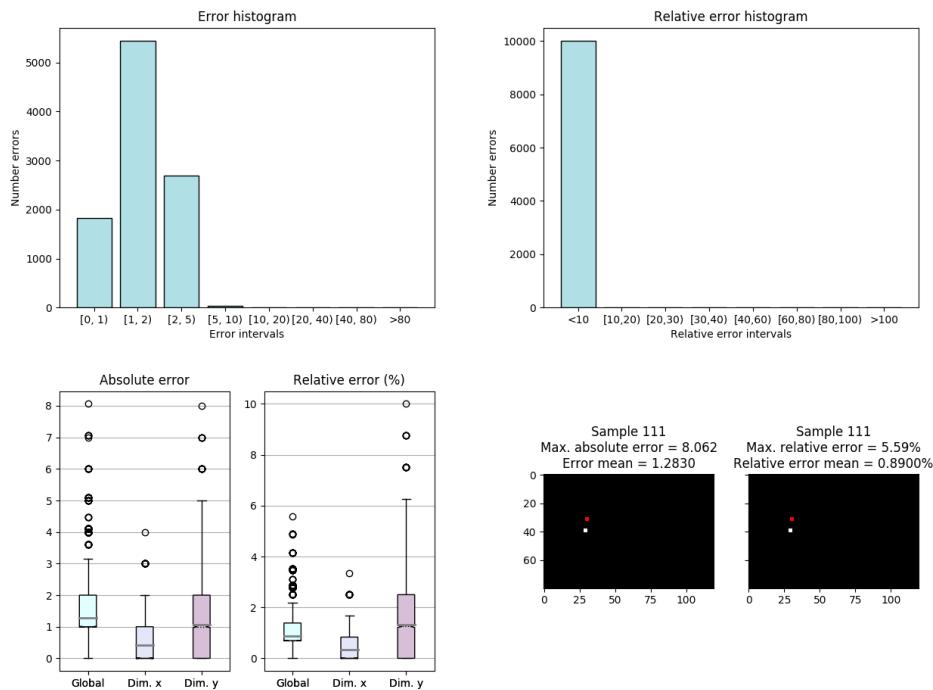


Figura 5.17: Resultados de LSTM-1 con dinámica sinusoidal de 2 DOF (10000 muestras de *test*).

Se puede observar una mejora considerable respecto a los resultados de la Figura 5.12, que evalúa una red no recurrente bajo las mismas circunstancias. Esta comparación corrobora la idea mencionada anteriormente de que la recurrencia mejora la capacidad predictiva de una red cuando esta no es del todo buena.

Se añade un nuevo DOF a la dinámica, la amplitud y se realiza el entrenamiento y evaluación con un conjunto del mismo número de muestras (100000) que en los casos anteriores. En la Figura 5.18 se muestran los resultados para este caso.

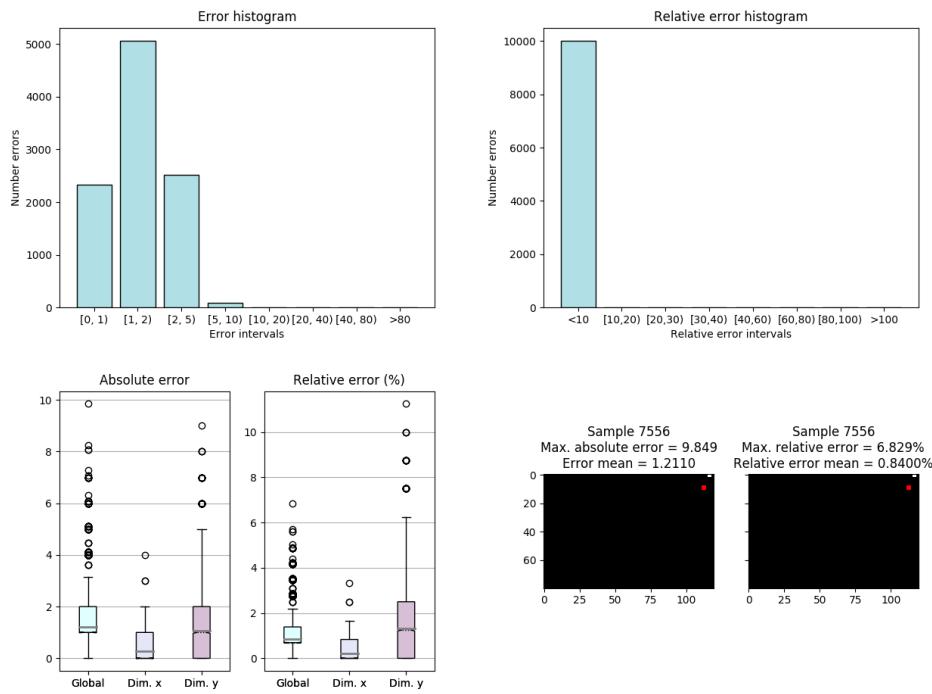


Figura 5.18: Resultados de LSTM-1 con dinámica sinusoidal de 3 DOF (10000 muestras de *test*).

Los resultados obtenidos indican que la red tiene una buena capacidad predictiva, lo que refuerza la idea de que introducir la recurrencia en problemas de esta naturaleza facilita la tarea de predicción.

Se entrena la red con el siguiente grado de complejidad, 4 DOF, cuyos resultados se reflejan en la Figura 5.19. Para este último caso, el más complejo de la dinámica, la red no tiene una buena capacidad predictiva a pesar de introducir la recurrencia. Este hecho puede derivarse de la sencillez de la propia red propuesta, que dispone de una única capa con 25 neuronas, algo que se tratará posteriormente.

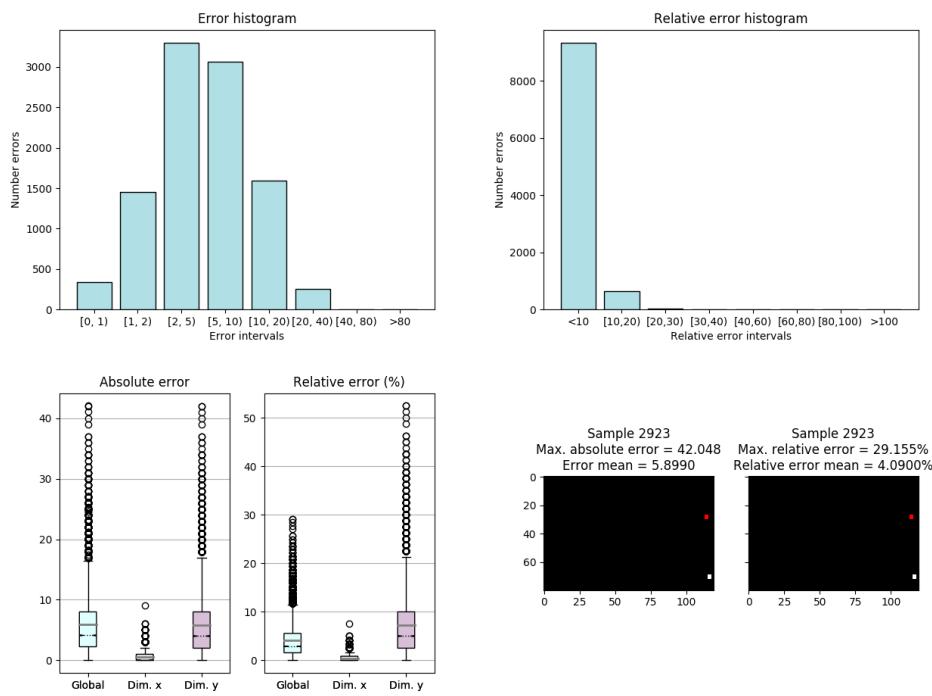


Figura 5.19: Resultados de LSTM-1 con dinámica sinusoidal de 4 DOF (10000 muestras de *test*).

#### 5.2.4. Resumen de resultados

La Tabla 5.3 resume los resultados obtenidos con la red *LSTM-1* para cada dinámica.

Dinámica		LSTM-1
Lineal	1 DOF	0.16 %
	2 DOF	0.25 %
Parabólica	1 DOF	0.12 %
	2 DOF	0.35 %
	3 DOF	0.58 %
Sinusoidal	1 DOF	0.42 %
	2 DOF	0.89 %
	3 DOF	0.84 %
	4 DOF	4.1 %

Tabla 5.3: Promedio del error relativo en *test* al evaluar la red LSTM-1 con imágenes modeladas y distintas dinámicas (10000 muestras de *test*).

Al igual que en el caso anterior se han evaluado todas las redes con un conjunto de 10000 muestras para una comparación equitativa, repitiendo la evaluación en aquellas que se utilizaron 1000.

### 5.3. Arquitectura recurrente: LSTM-4

Para tratar de obtener una arquitectura capaz de predecir en todas las dinámicas en su caso más complejo se realizan una serie de experimentos que dan lugar a la estructura recurrente definida en la Figura 5.20.

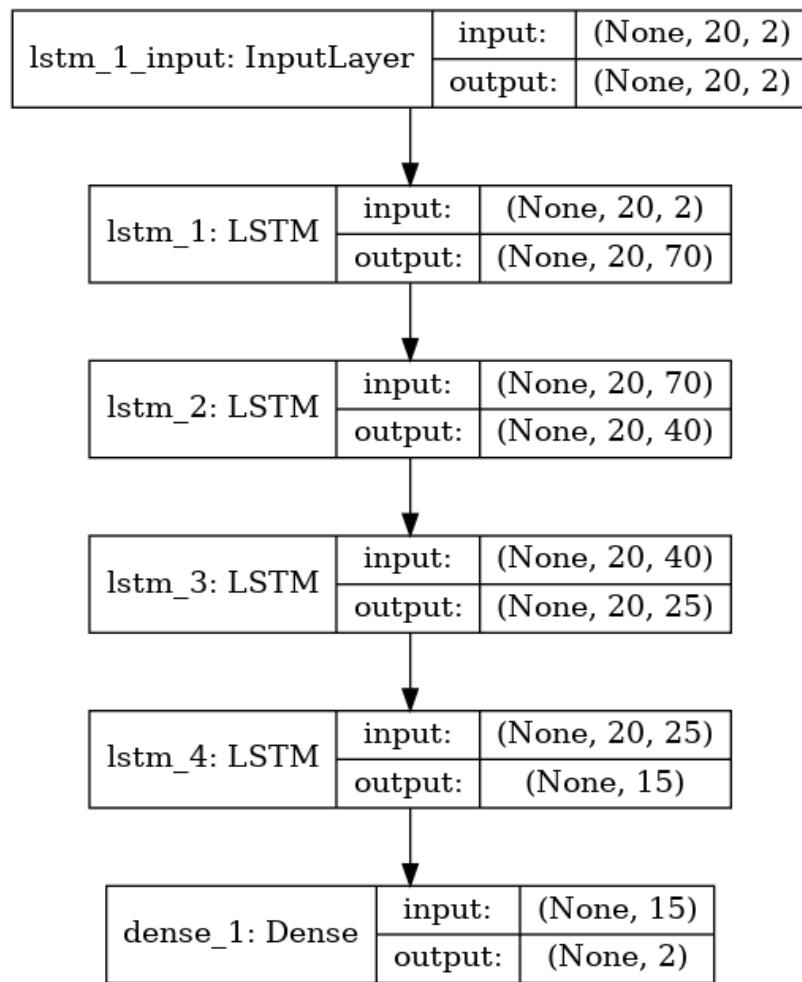


Figura 5.20: Estructura de LSTM-4 para imágenes modeladas.

Esta nueva red está compuesta por 4 capas LSTM de 70, 40, 25 y 15 neuronas respectivamente, cuya entrada son las 20 posiciones conocidas y la salida la posición futura estimada.

Para llegar a definir la nueva estructura de red se han explorado dos vías: el aumento de neuronas y el de capas, que serán explicados a continuación. Además, se presentan los resultados obtenidos para cada una de las dinámicas consideradas.

### 5.3.1. Aumento de neuronas

Una vía para la mejora de la red recurrente en las imágenes modeladas es el aumento de neuronas manteniendo una única capa. En este sentido, se duplica el número de neuronas, pasando de las 25 provistas para la red *LSTM-1* a un total de 50. Con esta nueva red, utilizando el conjunto de 100000 muestras que sigue la dinámica sinusoidal con 4 DOF, se obtienen los resultados de la Figura 5.21.

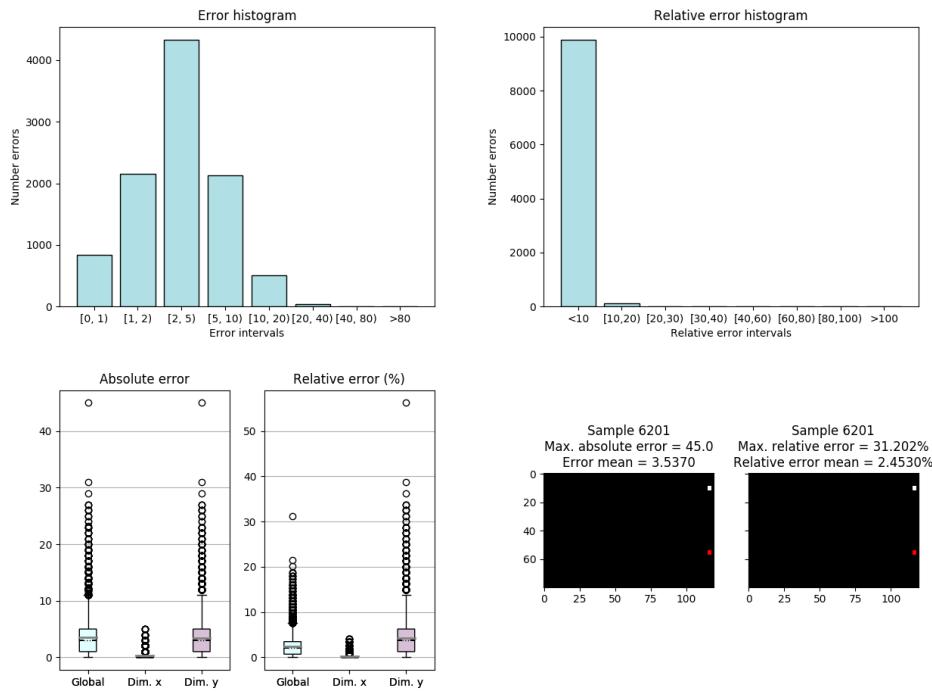


Figura 5.21: Resultados de LSTM de 50 neuronas con dinámica sinusoidal de 4 DOF (10000 muestras de *test*).

Al comparar los resultados con los equivalentes para la red con menor número de neuronas, en la Figura 5.19, se observa una mejora en promedio del error relativo. Este hecho parece corroborar la idea de que la poca capacidad predictiva de la red *LSTM-1*, bajo estas circunstancias, es consecuencia de su simplicidad. Sin embargo, esta mejora no es suficiente para lograr una red que sea capaz de predecir en el caso más complejo.

### 5.3.2. Aumento de capas

La segunda vía a explorar en la mejora de la red *LSTM-1* es el aumento de del número de capas. En este sentido, se comienza con el aumento de una única capa, que ya proporciona unos datos mucho mejores que los expuestos anteriormente. Este hecho hace ver que esta mecánica es más efectiva que el aumento de neuronas, y reafirma la idea de que una red más compleja captará mejor las distintas relaciones.

Con la idea de que el aumento del número de capas es la estrategia más efectiva, se debe establecer el límite de capas a añadir. Para ello se realiza un estudio en el que se aumenta de forma gradual el número de capas, disminuyendo de la misma forma el número de neuronas en cada una de ellas. En la Figura 5.22 se muestra un gráfico en el que se representa la evolución del valor del error relativo, en términos de media y máximo, a medida que se aumentan las capas. Para la elaboración de esta tabla se ha utilizado el mismo conjunto que en el aumento de neuronas, 100000 muestras de sinusoidal con todos sus DOF.

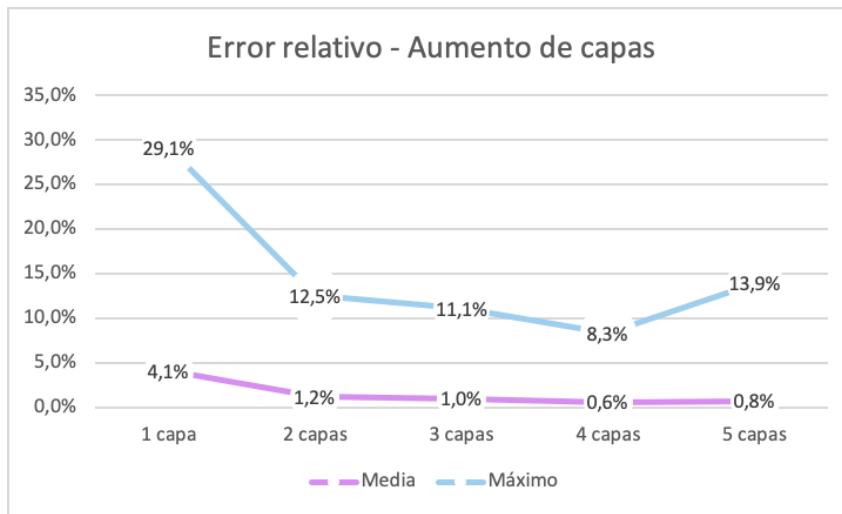


Figura 5.22: Comparación del error relativo al aumentar el número de capas LSTM (Síntesis sinusoidal, 4 DOF, 10000 muestras de *test*).

En esta gráfica se puede comprobar que los resultados mejoran a medida que se añaden más capas, hasta llegar a 4 capas. Cuando se emplean 5 capas el error vuelve a subir, empeorando los resultados, lo que hace que se opte por definir la estructura con 4 capas LSTM.

### 5.3.3. Predicción con dinámicas lineales

En la Figura 5.23 se muestran los resultados obtenidos para la dinámica lineal de 2 DOF con esta nueva estructura de red.

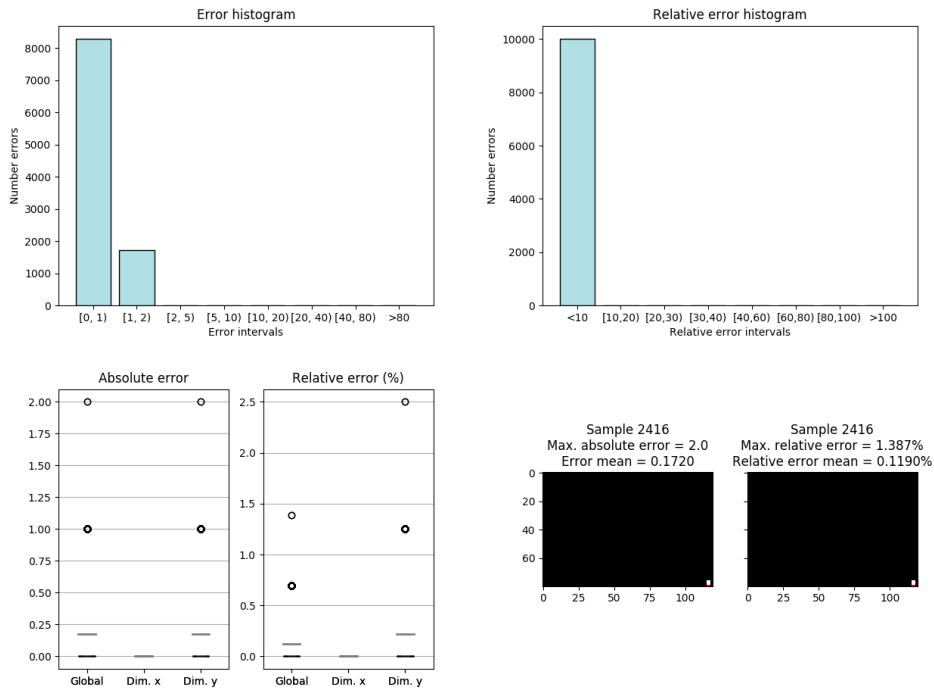


Figura 5.23: Resultados de LSTM-4 con dinámica lineal de 2 DOF (10000 muestras de *test*).

Los resultados obtenidos son muy similares a los de la estructura LSTM con una única capa y 10000 muestras. Este hecho hace concluir que, aunque los resultados son muy buenos, no compensa el aumento del coste computacional pues no se obtiene una mejora significativa.

### 5.3.4. Predicción con dinámicas parabólicas

Para esta dinámica, cuyo máximo grado de complejidad se alcanza al tener 3 DOF, los resultados obtenidos al evaluar la estructura de red propuesta se muestran en la Figura 5.24.

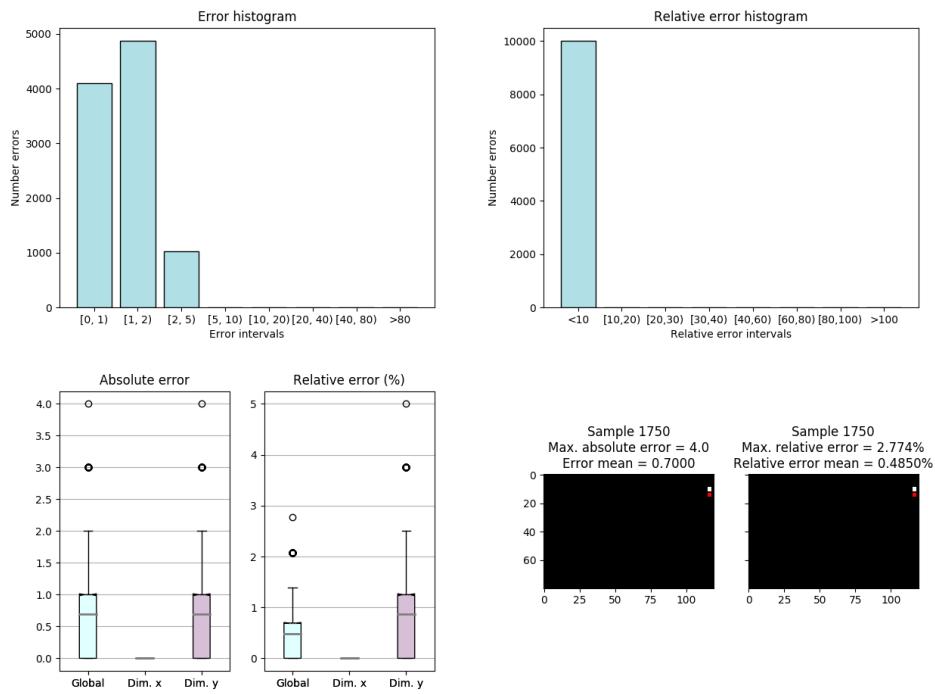


Figura 5.24: Resultados de LSTM-4 con dinámica parabólica de 3 DOF (10000 muestras de *test*).

Al igual que ocurre en el caso anterior, la capacidad de predicción de la red *LSTM-1* ya es buena para esta dinámica con un conjunto de 10000 muestras. Los resultados arrojados al utilizar un conjunto más grande con una estructura más compleja reafirman lo concluido en el caso lineal: no compensa el coste computacional por la mejora.

### 5.3.5. Predicción con dinámicas sinusoidales

La última dinámica considerada hasta el momento, regida por la función sinusoidal, establece su máxima complejidad en 4 DOF. Los resultados de entrenar y evaluar la nueva red con el *dataset* de 100000 muestras y dichos grados de libertad se muestran en la Figura 5.25.

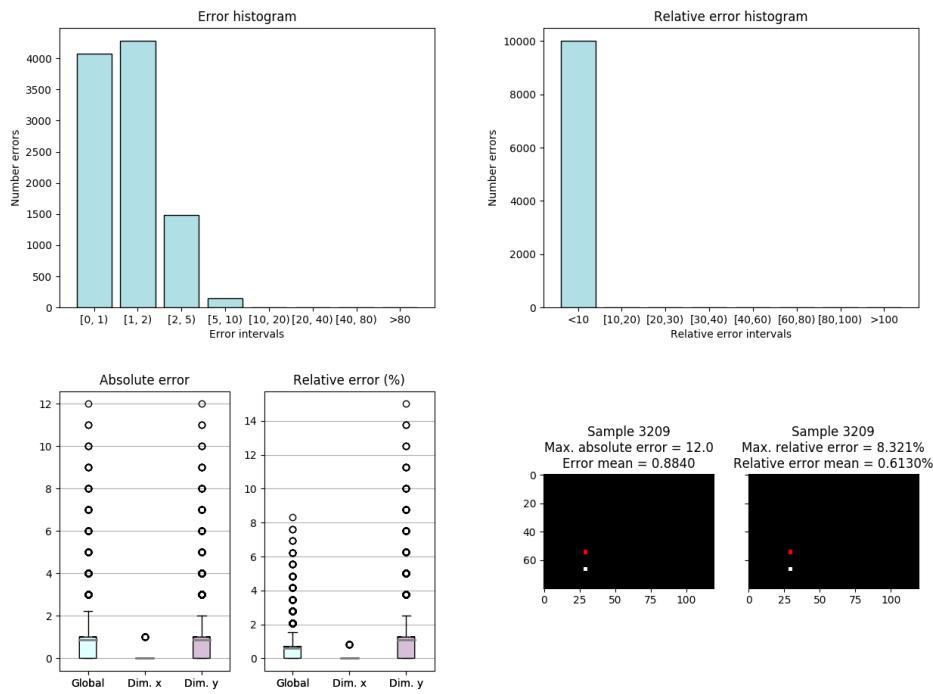


Figura 5.25: Resultados de LSTM-4 con dinámica sinusoidal de 4 DOF (10000 muestras de *test*).

En esta última dinámica no se habían conseguido unos resultados demasiado buenos con la estructura *LSTM-1*, había un alto grado de incertidumbre. Con la nueva estructura de red los resultados de la misma mejoran considerablemente, lo que hace que se pueda predecir con mayor certeza. Por tanto, sí que resulta efectivo aumentar la complejidad y, con ello el coste computacional, por una mejora en la capacidad predictiva para la dinámica.

### 5.3.6. Predicción con dinámica combinada

Tras haber dominado por separado las dinámicas en su grado más complejo, se ha considerado un nuevo conjunto que contiene 33000 ejemplos lineales y parabólicos, y 34000 sinusoidales, con el objetivo de explorar una dinámica combinada. La red se entrena y evalúa con este nuevo conjunto y se obtienen los resultados mostrados en la Figura 5.26.

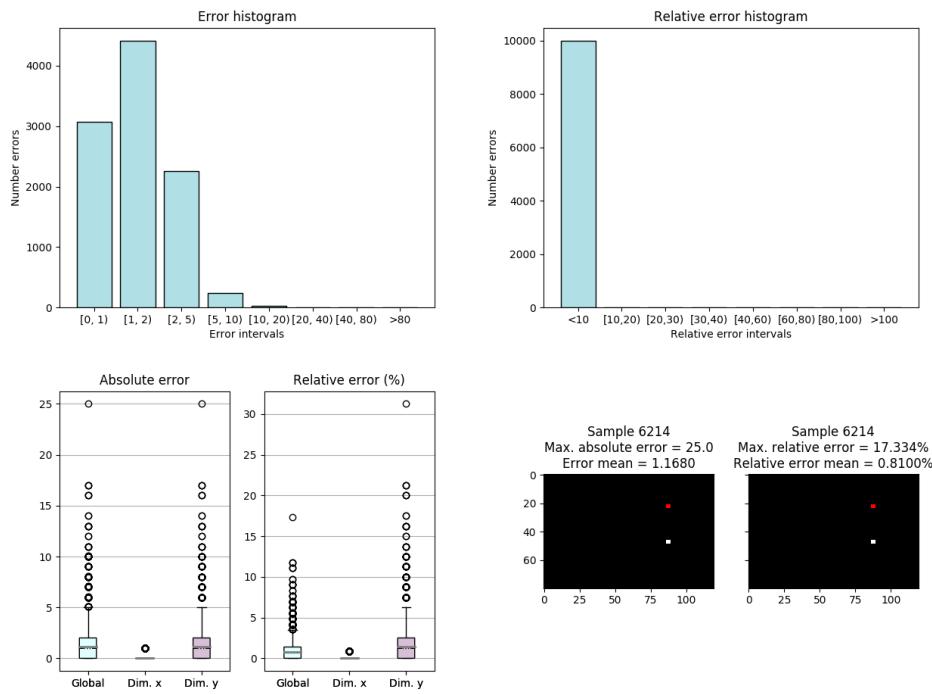


Figura 5.26: Resultados de LSTM-4 con dinámica combinada (10000 muestras de *test*).

Los resultados están limitados por los obtenidos para cada dinámica por separado. Al mezclar las muestras en la misma proporción, con algún ejemplo más de la sinusoidal, no se espera mejorar los resultados de la que peor predice. De esta forma, aunque se comete mayor error respecto al caso sinusoidal, se consigue mantener la capacidad de predicción en términos generales.

### 5.3.7. Predicción a largo plazo

El último experimento propuesto en cuanto a la predicción de las imágenes modeladas consiste en el análisis de la repercusión del horizonte temporal en la calidad de la predicción. Para ello se ha utilizado el conjunto de la dinámica combinada con 100000 muestras incrementando el valor del *gap* gradualmente. En la Figura 5.27 se muestra la evolución del error relativo obtenido, en términos de media y máximo, a medida que se modifica este parámetro.

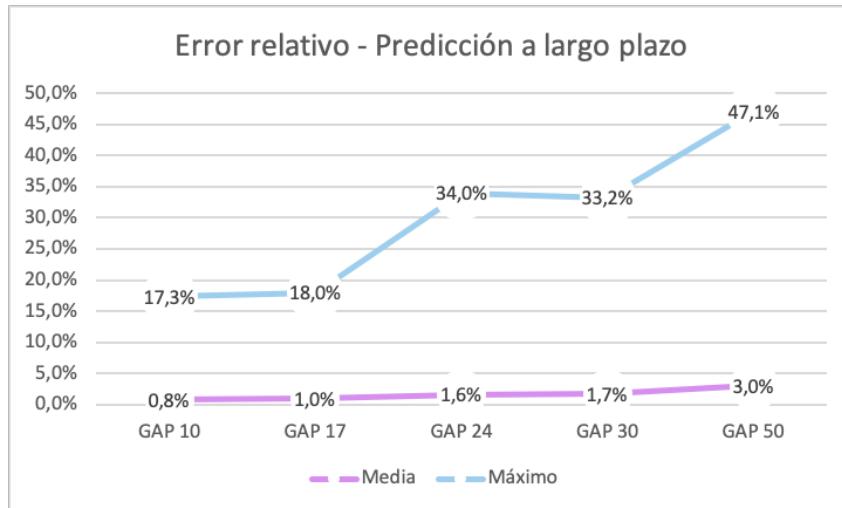


Figura 5.27: Comparación del error relativo al aumentar el *gap* (Combinada, 10000 muestras de *test*)

A la vista de estos resultados queda patente el hecho de que poner la vista en una posición más lejana en el tiempo complica la tarea de predicción de la red. A medida que se aumenta la separación temporal entre el último elemento conocido y el que se quiere estimar la relación entre ambas se va difuminando, perdiendo la capacidad de predicción progresivamente. Sin embargo, esta pérdida se realiza siempre en umbrales admisibles: en una imagen de 640x480, por ejemplo, se obtienen 14 píxeles de media de error a 30 fotogramas (1.7%) y 24 a 50 fotogramas (3%).

### 5.3.8. Resumen de resultados

En la Tabla 5.4 se muestra el resumen de los resultados obtenidos en las distintas dinámicas consideradas para la red *LSTM-4*.

Dinámica		LSTM-4
Lineal	2 DOF	0.12 %
Parabólica	3 DOF	0.5 %
Sinusoidal	4 DOF	0.61 %
Combinada		0.81 %

Tabla 5.4: Promedio del error relativo en *test* al evaluar la red LSTM-4 con imágenes modeladas y distintas dinámicas (10000 muestras de *test*).

Como anteriormente, se han evaluado todas las redes con un conjunto de 10000 muestras para que la comparación entre ellas sea equitativa.

## 5.4. Comparativa global

Tras realizar los experimentos con imágenes modeladas se obtiene un conjunto de redes entrenadas para las distintas dinámicas. Con el objetivo de compararlas de una forma rápida, se ha elaborado la Tabla 5.5. Esta tabla sigue el mismo código de colores especificado en el Apartado 5.1.5, añadiendo el gris para aquellos experimentos que no han sido considerados. Además, el valor numérico asociado, promedio del error relativo, ilustra de una forma cuantitativa las prestaciones obtenidas, aunque no es la única figura de mérito utilizada para la asignación del color en la celda. También se han evaluado todas las redes con un conjunto de 10000 muestras para una comparación equitativa.

Dinámica		MLP	LSTM-1	LSTM-4
Lineal	1 DOF	0.21 %	0.16 %	
	2 DOF	0.31 %	0.25 %	0.12 %
Parabólica	1 DOF	0.28 %	0.12 %	
	2 DOF	0.42 %	0.35 %	
	3 DOF	0.65 %	0.58 %	0.6 %
Sinusoidal	1 DOF	0.54 %	0.42 %	
	2 DOF	3.89 %	0.89 %	
	3 DOF		0.84 %	
	4 DOF		4.09 %	0.61 %
Combinada				0.81 %

Tabla 5.5: Comparativa del promedio de error relativo en las distintas redes para imágenes modeladas con las distintas dinámicas (10000 muestras de *test*).

Se puede observar que la red que mejores resultados aporta es la *LSTM-4*, que es capaz de realizar una predicción razonable en todas las dinámicas consideradas. Además, se refleja que el uso de la recurrencia aporta una mejora en la calidad de la predicción y que el aumento de capas recurrentes hace que se lleguen a dominar todas las dinámicas en su grado más complejo.

# Capítulo 6

## Predicción con imágenes crudas

En este capítulo se presentan los estudios realizados para la predicción en imágenes crudas, con el objetivo de explorar distintas estructuras neuronales para abordar el problema. Al igual que en las imágenes modeladas, se pone el foco en el análisis de las redes recurrentes y, más concretamente, de aquéllas que mantienen la relación espacio-temporal.

Para las imágenes crudas, la tarea no se aborda como un problema clásico de regresión. Se quiere predecir una imagen en la que todos los píxeles toman un valor nulo salvo uno de ellos, considerado como activo. Esta naturaleza de las imágenes a predecir hace que se pueda afrontar el problema como una tarea de clasificación binaria donde la salida se corresponde con un vector de longitud  $h \times w$ , con la codificación de la clase en cada píxel. Para volver a obtener una imagen, según se explicó en la Sección 4.2, se redimensiona el vector de salida a una matriz  $h \times w$  que se pasa a una imagen en escala de grises con 8 bits por píxel.

La características que siguen todos los conjuntos de datos utilizados para este estudio, en lo que a *n\_points*, *gap* y división en subconjuntos se refiere, son las mismas que las establecidas en el Capítulo 5 para imágenes modeladas. Además, cada una de las imágenes que forma una muestra tienen un tamaño de  $80 \times 120$  píxeles.

A continuación se realiza el recorrido por las estructuras neuronales entrenadas con imágenes crudas, mostrando los resultados obtenidos y las conclusiones que originan nuevas vías de investigación para la mejora de los mismos.

## 6.1. Arquitectura no recurrente: Red convolucional

Para abordar el estudio de la predicción en imágenes crudas con una arquitectura sin recurrencia se utiliza una CNN, introducida en el Apartado 1.2.1.2. En la Figura 6.1 se presenta la estructura escogida para abordar la tarea de predicción con imágenes crudas. En este caso, la entrada a la red es una secuencia con 20 imágenes de tamaño  $h \times w$ , que alimenta dos capas consecutivas de convolución seguidas de una de reducción o *pooling*. A la salida se obtiene un vector de longitud  $h \times w$  que será redimensionado posteriormente para obtener una imagen.

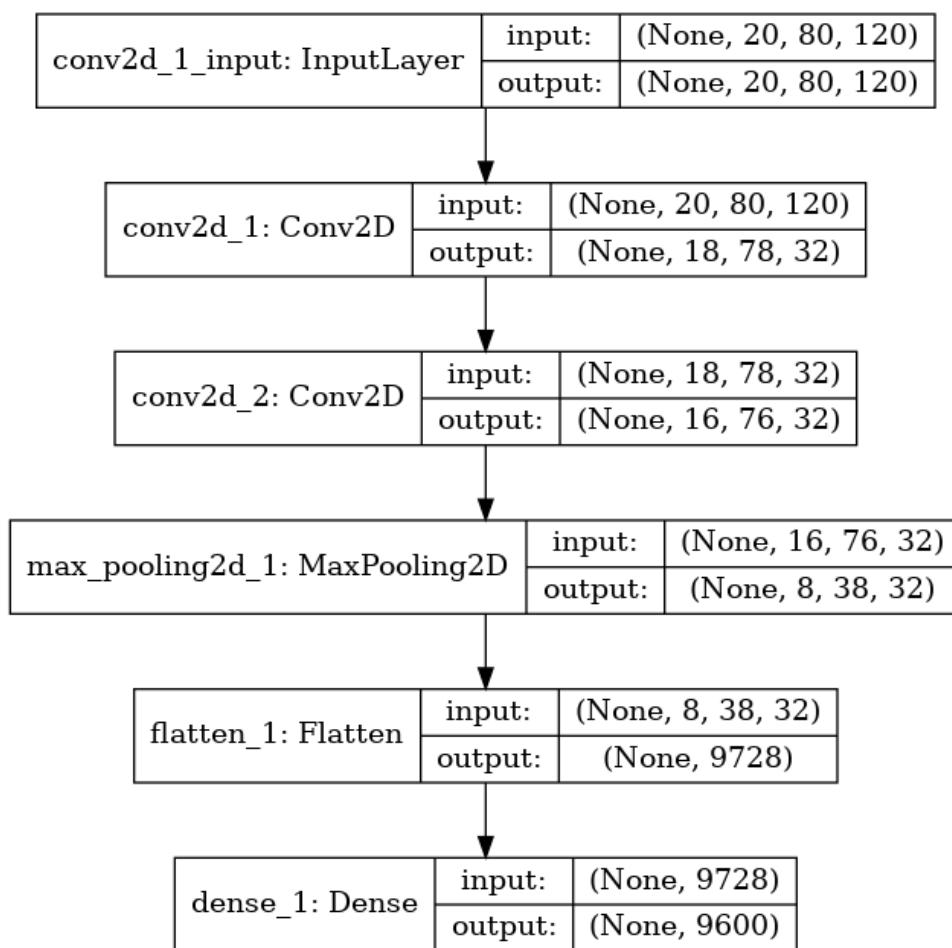


Figura 6.1: Estructura de CNN para imágenes crudas.

Una vez definida la red, se realizan experimentos con conjuntos de secuencias de imágenes crudas donde la complejidad de movimiento del píxel activo va aumentando de forma progresiva, tanto con el tipo de dinámica como en el número de grados de libertad para la misma dinámica.

### 6.1.1. Influencia del número de muestras

Con el objetivo de analizar la influencia que el número de muestras tiene en el aprendizaje de la red, se ha desarrollado un experimento en el que dicho número aumenta de forma progresiva, entrenando la estructura descrita anteriormente. Se han empleado muestras cuyos píxeles siguen una dinámica lineal de un único DOF, evaluando todas las redes con un mismo conjunto de *test* de 1000 muestras. Los resultados se muestran en la gráfica de la Figura 6.2.

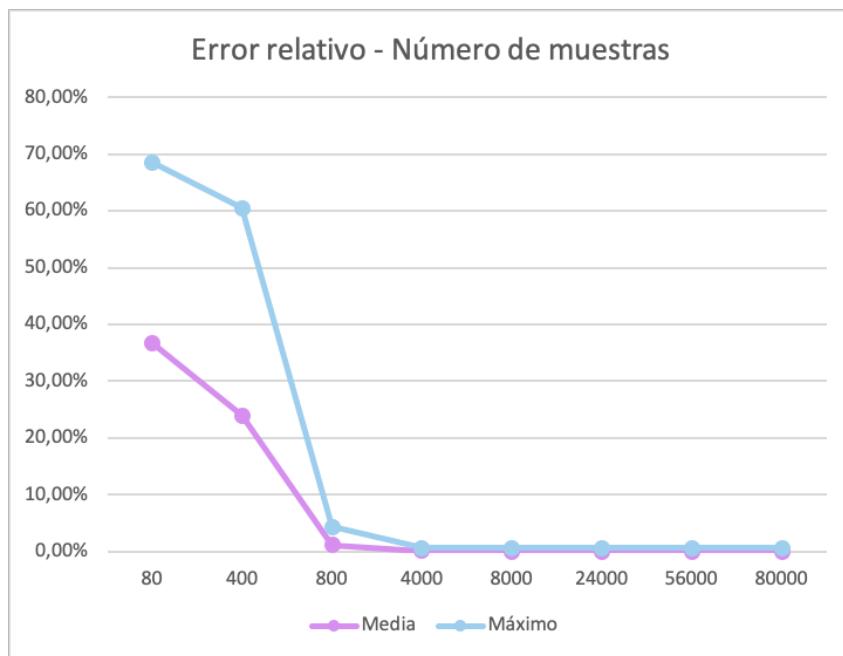


Figura 6.2: Comparación del error relativo al aumentar el número de muestras de entrenamiento con CNN (Lineal, 1 DOF, 1000 muestras de *test*).

En el gráfico se puede comprobar cómo aumentar el número de muestras de entrenamiento produce un efecto positivo en las prestaciones de la red hasta un límite, a partir del cuál las prestaciones se mantienen. Este valor límite en el número de muestras dependerá de la complejidad de la dinámica dinámica que rige el movimiento. Para las dinámicas más sencillas se necesitará un número menor de muestras de entrenamiento que para las más complejas. Mediante el uso de un número adecuado de muestras en el entrenamiento se evita que no haya suficientes secuencias para que la red sea capaz de aprender y, por otro lado, que la red vea todos los ejemplos de la dinámica y memorice las relaciones entre la entrada y la salida, evitando la generalización.

### 6.1.2. Predicción con dinámicas lineales

Para abordar la predicción de fotogramas donde se representan objetos que siguen una dinámica lineal, se utiliza un *dataset* compuesto por 10000 muestras. Este conjunto de datos se divide en tres, de tal forma que 8000 muestras se utilizan para el entrenamiento, 1000 para la validación y 1000 para el *test*.

En la Figura 6.3 se muestran los resultados obtenidos al evaluar la CNN propuesta entrenada con dicho conjunto cuando se considera un único DOF, la pendiente de la trayectoria del píxel activo. Se observa que las prestaciones de la red son buenas, pues se consigue un error relativo muy reducido en términos de media y máximo.

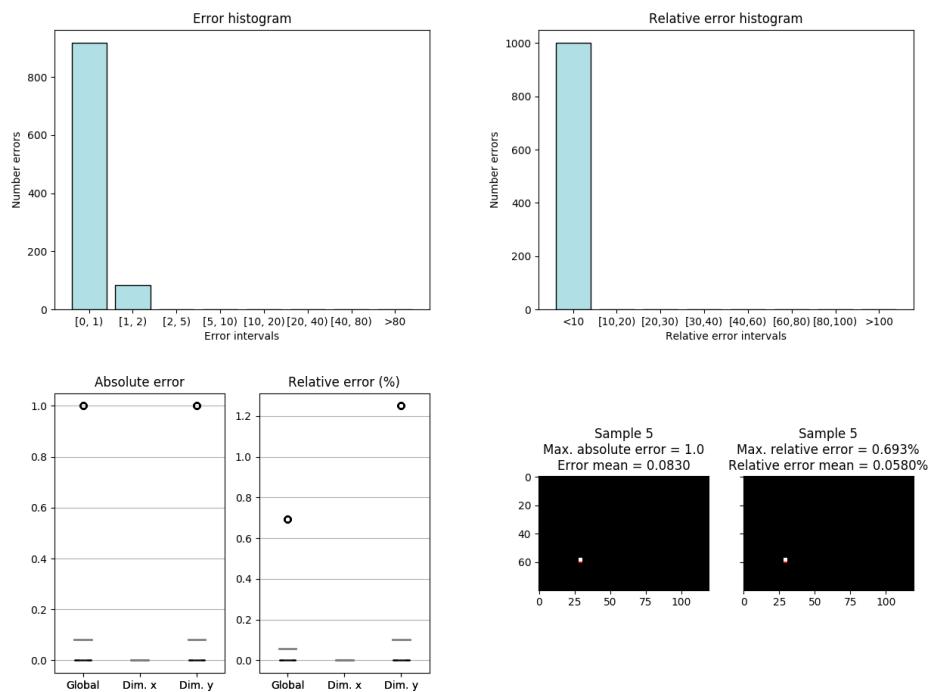


Figura 6.3: Resultados de CNN con dinámica lineal de 1 DOF (1000 muestras de *test*).

Al aumentar el grado de complejidad de la dinámica añadiendo un nuevo DOF, la posición vertical en la imagen desde la que el píxel activo comienza su movimiento recto. Los resultados obtenidos bajo esta nueva premisa, utilizando el mismo número de muestras y distribución de los subconjuntos que en el caso anterior, se muestran en la Figura 6.4.

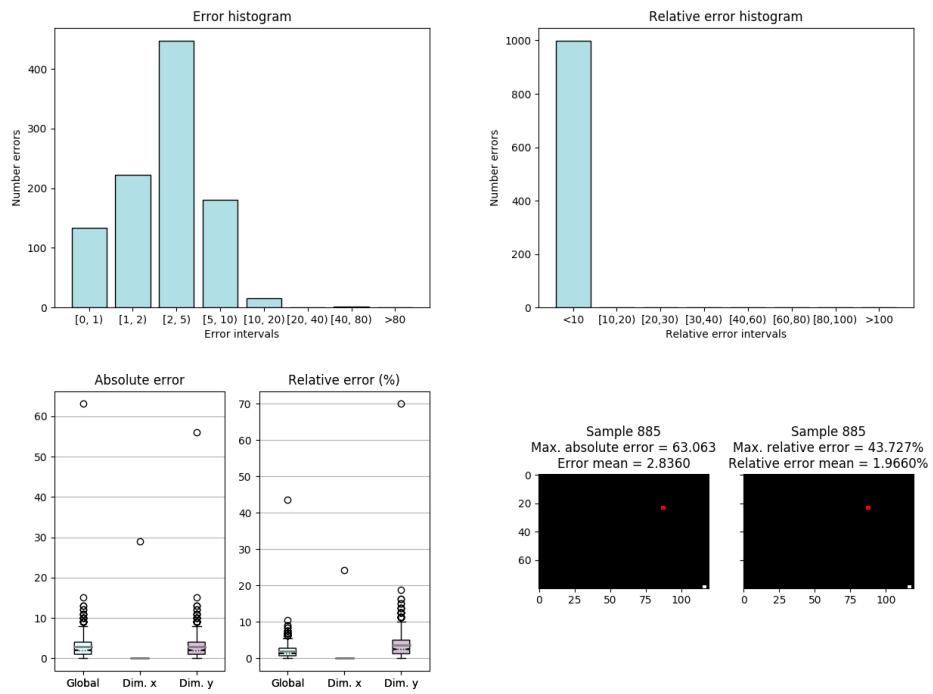


Figura 6.4: Resultados de CNN con dinámica lineal de 2 DOF (1000 muestras de *test*).

En esta ocasión, el resultado obtenido muestra un deterioro de las prestaciones de la red. Al tratar directamente con todos los píxeles de la imagen y no reducir la información a un par de valores ( $x, y$ ), como en el caso de las modeladas, el número de parámetros que utiliza la red se ha elevado enormemente. Este hecho hace que, para el caso que se está tratando (dinámica lineal con 2DOF), la cantidad de ejemplos que se introducen a la red durante el entrenamiento pueda no ser suficiente. Para solucionar este problema se decide aumentar el número de muestras del conjunto en un factor de 10, pasando de los 10000 ejemplos a 100000. La división en los tres subconjuntos se realiza con una asignación de 80000 para entrenamiento, 10000 para validación y 10000 para *test*. Con este incremento en el número de muestras se vuelve a entrenar y a evaluar la misma estructura CNN, obteniendo los resultados de la Figura 6.5.

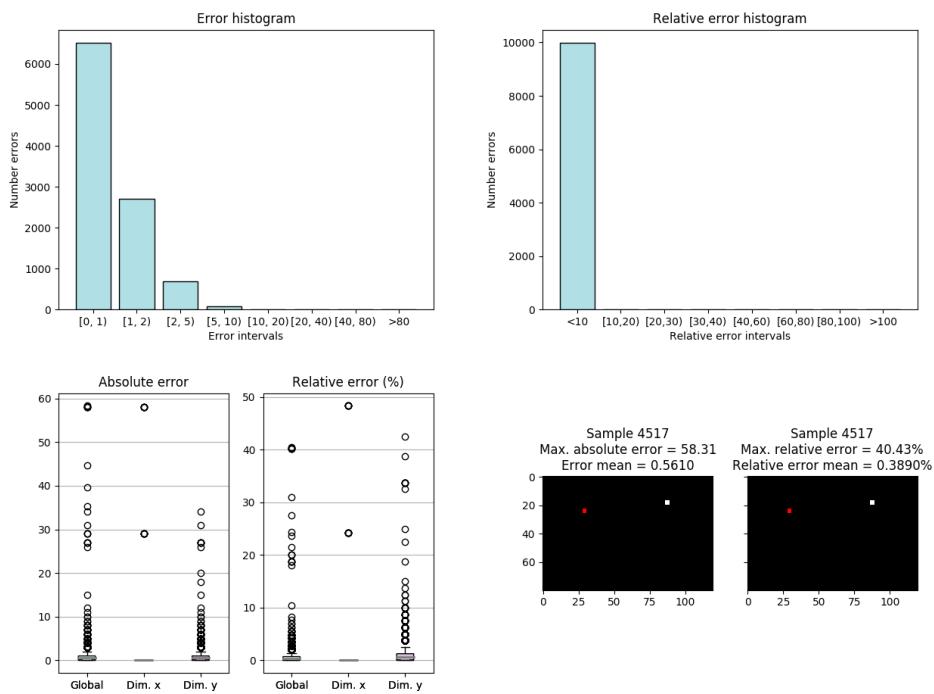


Figura 6.5: Resultados de CNN con dinámica lineal de 2 DOF (10000 muestras de *test*).

Se observa que con el aumento en el número de muestras de entrenamiento se mejoran las prestaciones de la red en términos de promedio del error relativo, pero siguen existiendo algunos *outliers*, valores atípicos que perjudican a la predicción y no permiten obtener resultados óptimos.

### 6.1.3. Predicción con dinámicas parabólicas

Para abordar la predicción de fotogramas donde se representan objetos que siguen una dinámica parabólica se utiliza un *dataset* compuesto por 100000 muestras. Este conjunto de datos queda dividido en tres subconjuntos, utilizando 80000 muestras para el entrenamiento, 10000 para la validación y 10000 para el *test*.

Se comienza el estudio con el caso más sencillo de la dinámica, un único DOF (valor de  $a$ ). Los resultados obtenidos se reflejan en la Figura 6.6. Las prestaciones de la red son muy buenas, se obtienen valores muy bajos de error y el número de *outliers* es también muy reducido.

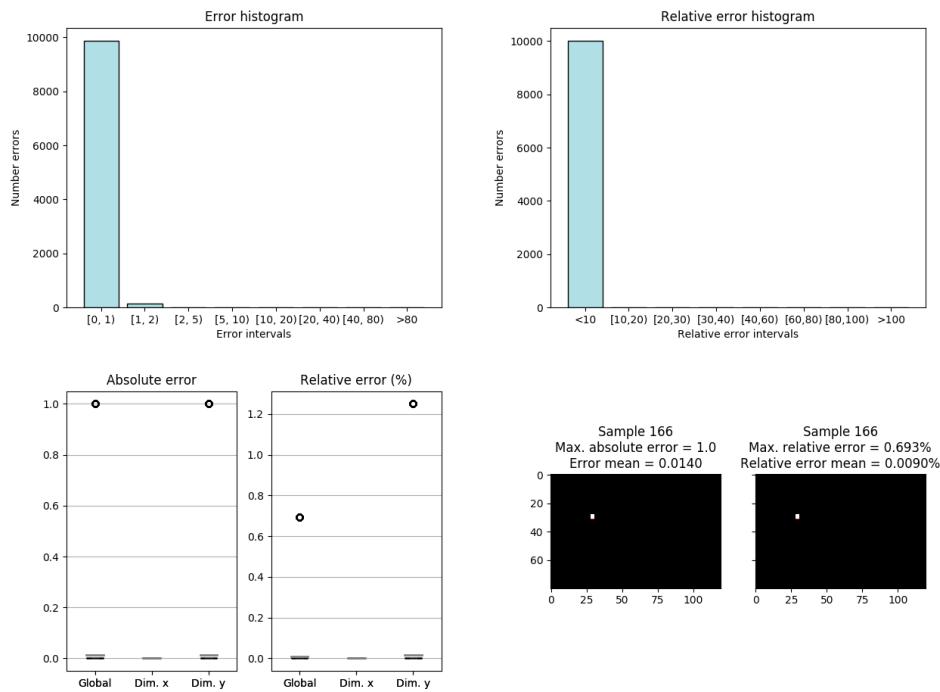


Figura 6.6: Resultados de CNN con dinámica parabólica de 1 DOF (10000 muestras de *test*).

Se aumenta la complejidad de la dinámica con un nuevo DOF, la posición vertical inicial del píxel en la imagen (valor de  $c$ ), manteniendo el número de muestras y la distribución de los subconjuntos del caso anterior. Los resultados obtenidos sobre la red entrenada con este conjunto se muestran en la Figura 6.7. En esta ocasión, al contrario que ocurría en la dinámica lineal, el aumento de la complejidad, aunque empeora los resultados, no provoca que la red pierda su capacidad predictiva.

Finalmente, dentro de esta dinámica se establecen 3 DOF. Se utiliza el mismo número de muestras por subconjunto que en los casos anteriores, obteniendo los resultados de la Figura 6.8. En ella se refleja que los valores de error son elevados, se produce un gran número de *outliers* y la caja de los *boxplot* se separa ligeramente del valor nulo. Con todo esto, las prestaciones de la red se reducen respecto a los casos anteriores, empeorando la capacidad predictiva de la red.

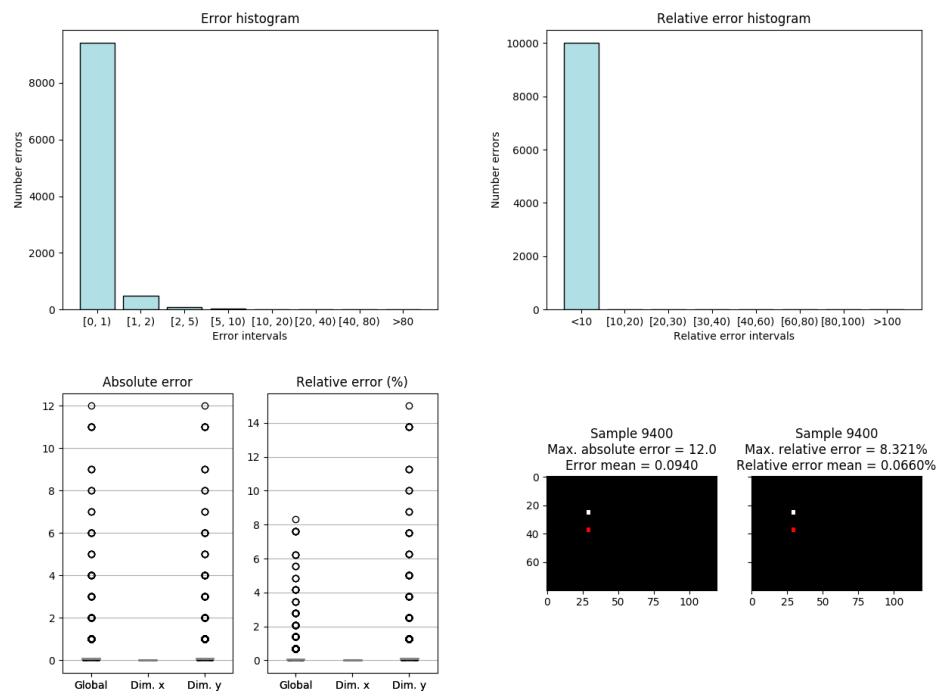


Figura 6.7: Resultados de CNN con dinámica parabólica de 2 DOF (10000 muestras de *test*).

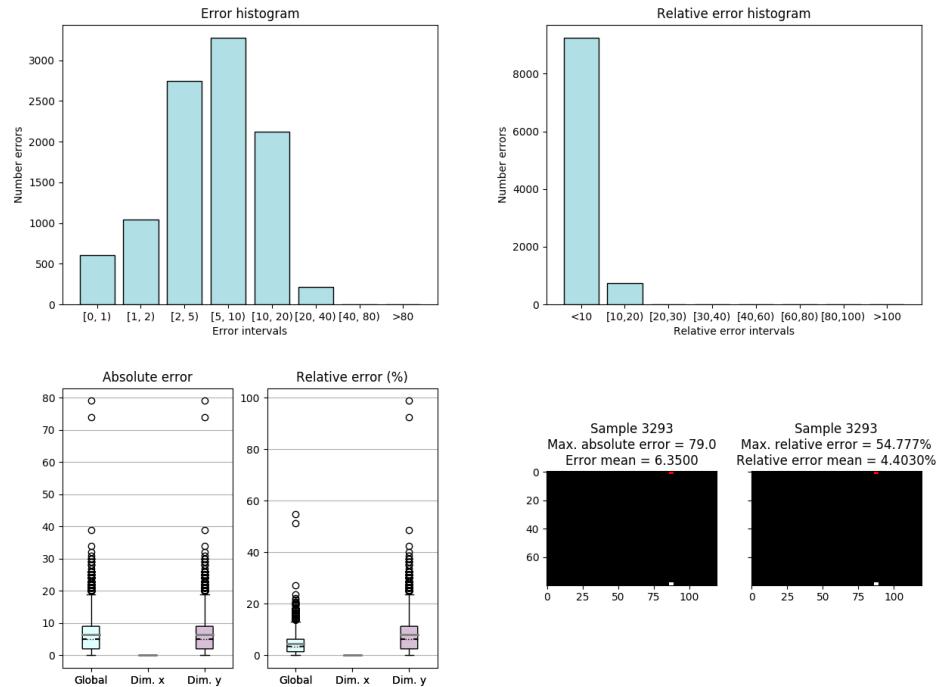


Figura 6.8: Resultados de CNN con dinámica parabólica de 3 DOF (10000 muestras de *test*).

### 6.1.4. Predicción con dinámicas sinusoidales

Para abordar la predicción de fotogramas que representan objetos que siguen una dinámica sinusoidal, se utiliza un *dataset* compuesto por 100000 muestras. Sobre el total de muestras establecido se realiza una división de tal forma que 80000 muestras se utilizan para el entrenamiento, 10000 para la validación y 10000 para el *test*.

Se comienza con la variante más sencilla de esta dinámica, que deja como parámetro aleatorio la frecuencia de la sinusoide, 1 DOF. En la Figura 6.9 se muestran los resultados obtenidos, donde se puede comprobar que la red tiene muy buenas prestaciones para este caso.

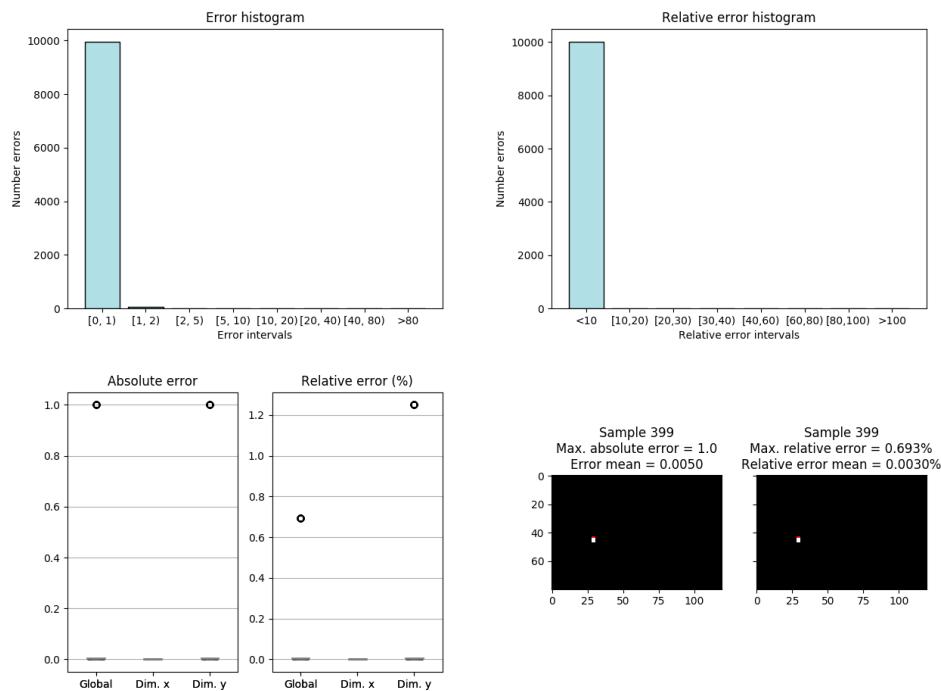


Figura 6.9: Resultados de CNN con dinámica sinusoidal de 1 DOF (10000 muestras de *test*).

A continuación, se permite que la posición vertical inicial del píxel en la imagen tome también un valor aleatorio, estableciendo la dinámica con 2 DOF. Tras entrenar y evaluar la red propuesta con los subconjuntos correspondientes, cuyas características son las mismas que en casos anteriores, se obtienen los resultados de la Figura 6.10. Estos valores reflejan un deterioro de las prestaciones de la red por la complejidad de la dinámica. Los valores de error han aumentado considerablemente, especialmente el

máximo, y se obtiene un número de *outliers* mucho mayor que en el caso más sencillo. Con estos resultados, se establece el límite de predicción con la CNN propuesta para dinámicas sinusoidales en un único DOF.

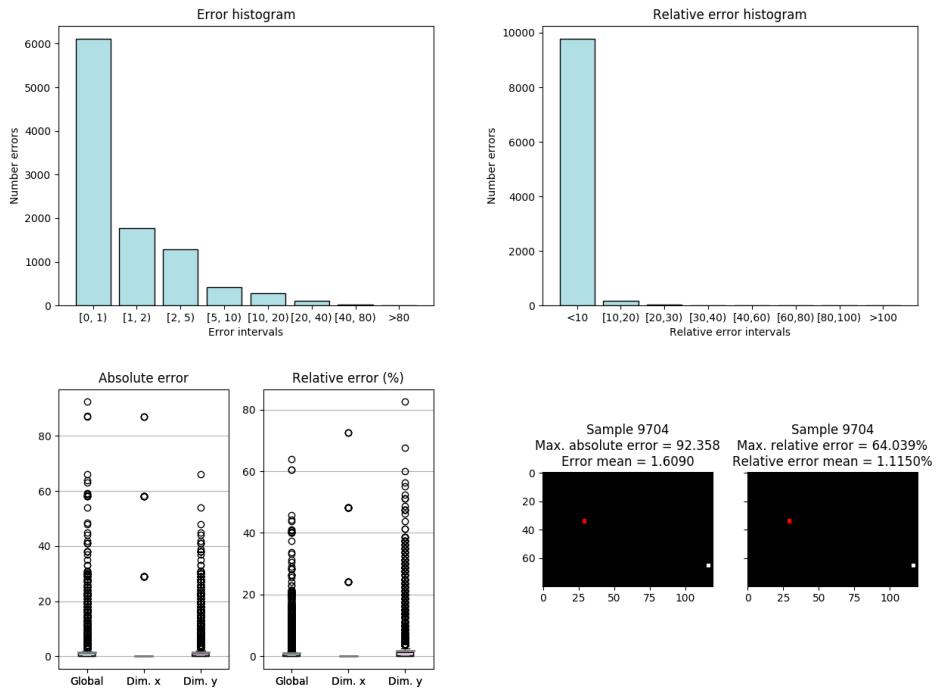


Figura 6.10: Resultados de CNN con dinámica sinusoidal de 2 DOF (10000 muestras de *test*).

### 6.1.5. Resumen de resultados

En la Tabla 6.1 se muestra un resumen de los resultados alcanzados en cada dinámica con imágenes de píxeles 2D. Para la interpretación de esta tabla se ha establecido un código de cuatro colores que reflejan, de mejor a peor, los resultados obtenidos en cada experimento. La secuencia de colores utilizada comienza con el verde oscuro para resultados muy buenos, pasando por el verde claro y el naranja, para terminar con el rojo, que indica malos resultados. Para realizar una comparación equitativa, se ha vuelto a evaluar la dinámica lineal de 1 DOF con 10000, proporcionando un promedio de error relativo de 0.07 %.

Dinámica		CNN
Lineal	1 DOF	0.07 %
	2 DOF	0.39 %
Parabólica	1 DOF	0.01 %
	2 DOF	0.07 %
	3 DOF	4.40 %
Sinusoidal	1 DOF	0.003 %
	2 DOF	1.12 %

Tabla 6.1: Promedio del error relativo en *test* al evaluar la CNN con imágenes modeladas y distintas dinámicas (10000 muestras de *test*).

## 6.2. Arquitectura recurrente: Convolucional + LSTM

Una vía de investigación en cuanto al efecto de introducir la recurrencia en la predicción de imágenes es combinar una capa convolucional, que capte las relaciones espaciales, con una capa LSTM posterior, para las relaciones temporales. La estructura propuesta se muestra en la Figura 6.11.

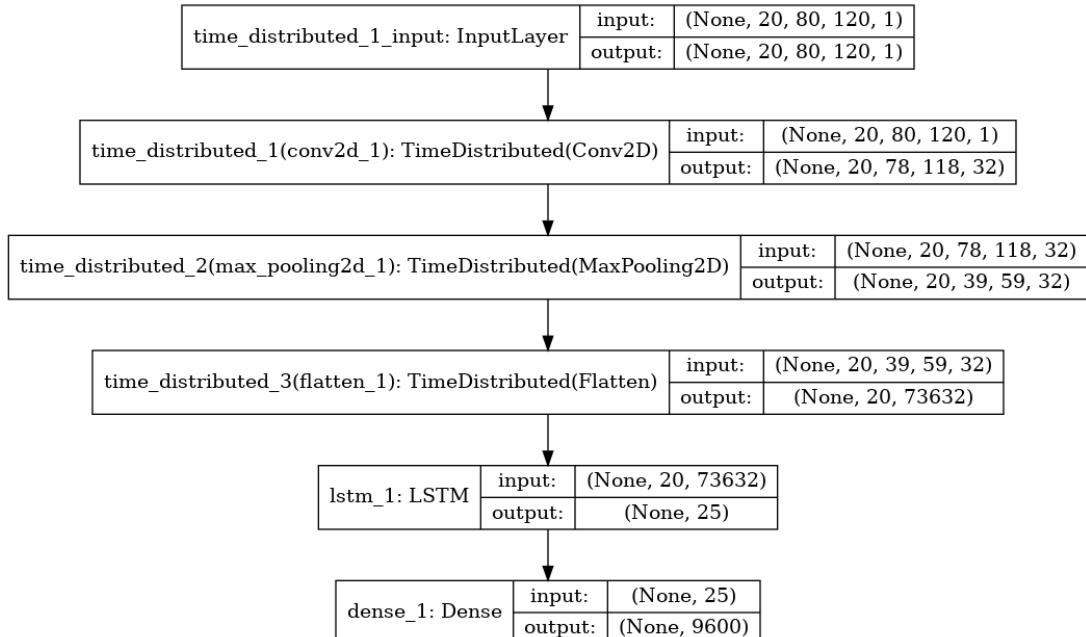


Figura 6.11: Estructura de CNN+LSTM para imágenes crudas.

La red consta de una capa convolucional de 32 neuronas seguida de su correspondiente capa de reducción. Tras la estructura propia de las CNN se añade una capa para *flatten* para obtener un vector de una dimensión que será la entrada a la única capa LSTM con 25 neuronas.

### 6.2.1. Predicción con dinámicas lineales

Para estudiar el efecto de la recurrencia con la arquitectura de red propuesta, se comienza con el caso más sencillo de todos: dinámica lineal de pendiente nula y con posición vertical inicial del píxel fija. El conjunto consta de 1000 muestras, de las cuales 800 son para entrenamiento, 100 para validación y otras 100 para *test*. Al evaluar la red CNN propuesta en la Sección 6.1 con el mismo conjunto se obtiene un error completamente nulo, según se muestra en la Figura 6.12; sin embargo, al introducir la capa LSTM, la red pierde toda su capacidad predictiva pasando de un 100 % de acierto a un 70 %, según se muestra en la Figura 6.13.

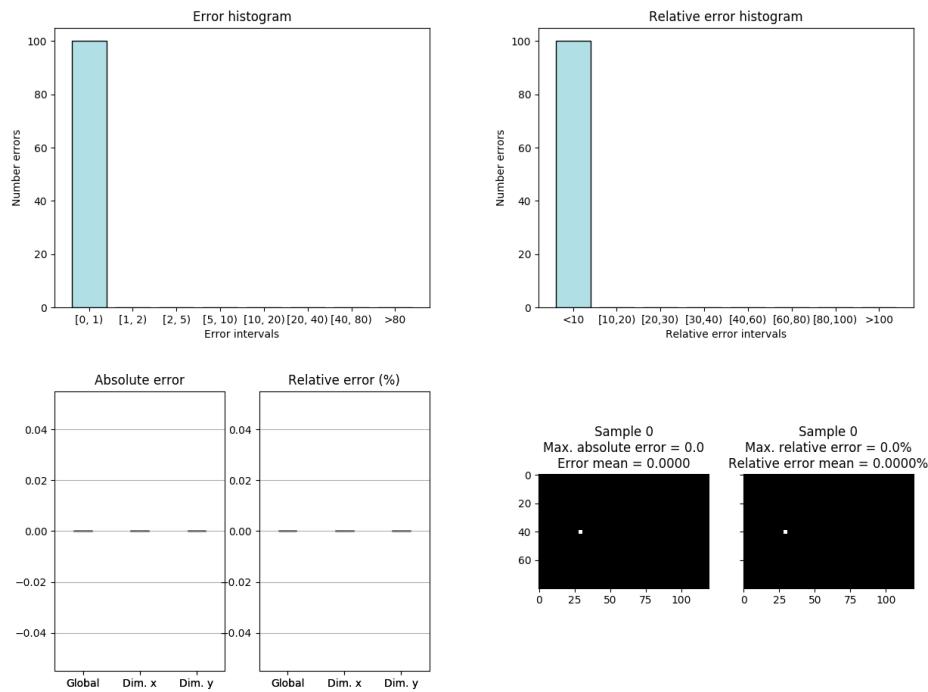


Figura 6.12: Resultados de CNN con dinámica lineal, pendiente nula y altura fija (100 muestras de *test*).

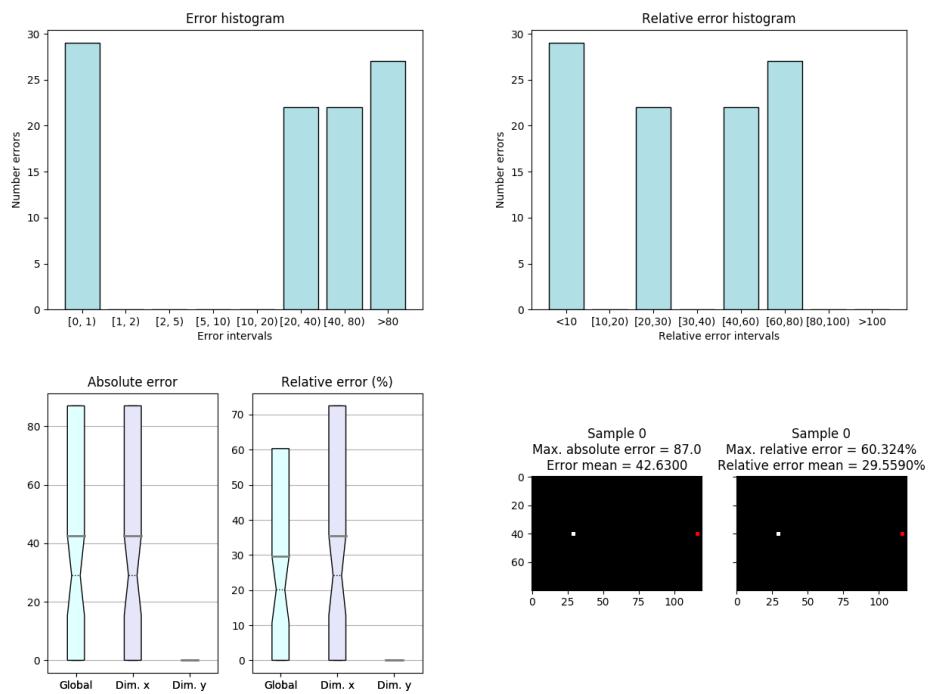


Figura 6.13: Resultados de CNN+LSTM con dinámica lineal, pendiente nula y altura fija (100 muestras de *test*).

Con estos resultados se puede concluir que analizar las relaciones temporales de forma independiente a las espaciales no parece una buena estrategia.

### 6.2.2. Extensión gradual del punto activo

Una de las posibles causas de que esta estrategia de combinar una CNN con una LSTM no funcione correctamente es que se están empleando imágenes con un único píxel activo, complicando a la red la tarea de encontrar correlaciones espaciales. Para analizar este hecho se decide ampliar la zona activa de la imagen, reduciendo de forma gradual el nivel de intensidad de los píxeles alrededor del píxel activo, haciendo uso de una función gaussiana isotrópica. En la Figura 6.14 se muestra tanto el píxel utilizado hasta ahora, que denominaremos píxel discreto, como el resultado de convolucionar un filtro gaussiano con centro en dicho píxel y entorno activo 5x5.

0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	255	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

(a)

0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	7	28	42	28	7	0
0	28	113	170	113	28	0
0	42	170	255	170	42	0
0	28	113	170	113	28	0
0	7	28	42	28	7	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

(b)

Figura 6.14: Ejemplos de píxel: (a) Discreto y (b) Expandido.

La intuición subyacente es que, al extender el píxel, los algoritmos que hacen uso de correlación espacial pueden tener valores gradualmente crecientes al acercarse a una correlación máxima, y tal vez durante el aprendizaje se vean ayudados a converger hacia ese máximo. Con un píxel discreto las correlaciones entre los píxeles de la imagen son todas nulas excepto en el máximo, lo que no ayuda a los algoritmos de aprendizaje (que se suelen apoyar en gradientes) a encontrarlo: en las cercanías del píxel activo las correlaciones valen igual que si se está lejos del mismo.

Para analizar la repercusión que tiene en evaluación el aprendizaje con extensión del píxel, se emplea el mismo conjunto de dinámica lineal utilizado anteriormente, aplicándole a las imágenes una convolución con el filtro gaussiano antes de su entrada a la red. En la Figura 6.15 se muestran los resultados de este nuevo experimento.

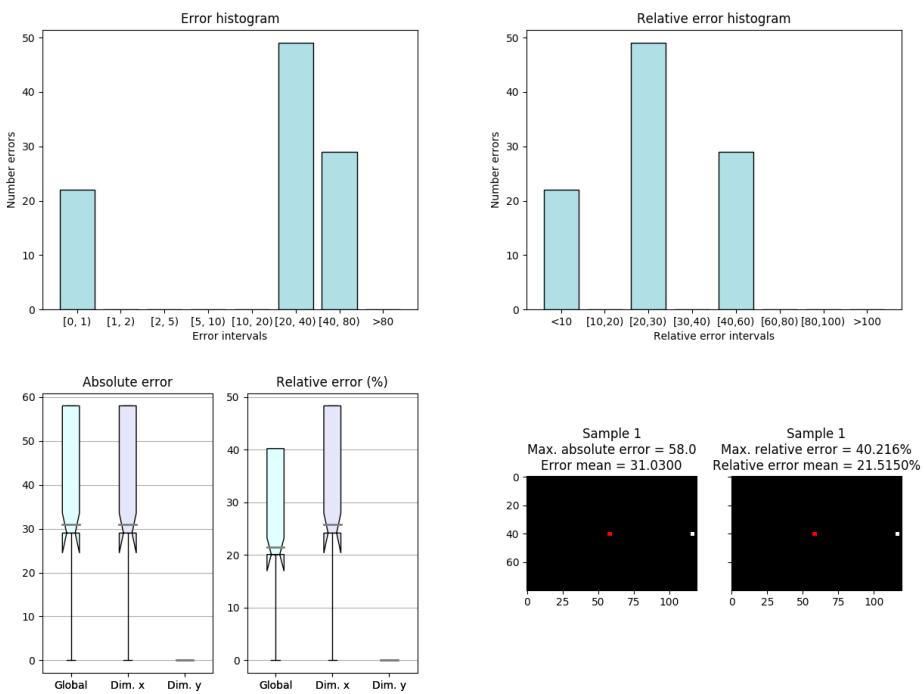


Figura 6.15: Resultados de CNN+LSTM con píxel extendido en dinámica lineal, pendiente nula y altura fija (100 muestras de *test*).

Aunque los resultados han mejorado en términos de valores medios, la red continúa sin ser capaz de predecir correctamente. Además, la caja del *boxplot* se ha visto desplazada hacia arriba, dejando los valores más bajos de error como *outliers* de los resultados. Con este análisis se concluye que, por un lado, la estrategia de combinar por separado ambas relaciones no es adecuada, y por el otro, extender el píxel para facilitar el análisis de las correlaciones espaciales en la imagen no es una mejora.

### 6.2.3. Resumen de resultados

A modo de resumen, la Tabla 6.2 se muestran los resultados de los experimentos relacionados con la estructura que combina una capa convolucional con una LSTM posterior a modo de resumen.

Dinámica		CNN	CNN+LSTM
Lineal	Discreto	0.0 %	29.6 %
	Extendido		21.5 %

Tabla 6.2: Promedio del error relativo en *test* al evaluar la red CNN+LSTM con imágenes modeladas y distintas dinámicas (100 muestras de *test*).

### 6.3. Arquitectura recurrente: ConvLSTM-1

Otra vía a explorar, en cuanto al efecto de la recurrencia en la capacidad de predicción de las redes, es el uso de las redes *ConvLSTM* presentadas en el Apartado 1.2.2.2. Para ello, se define la estructura de red de la Figura 6.16.

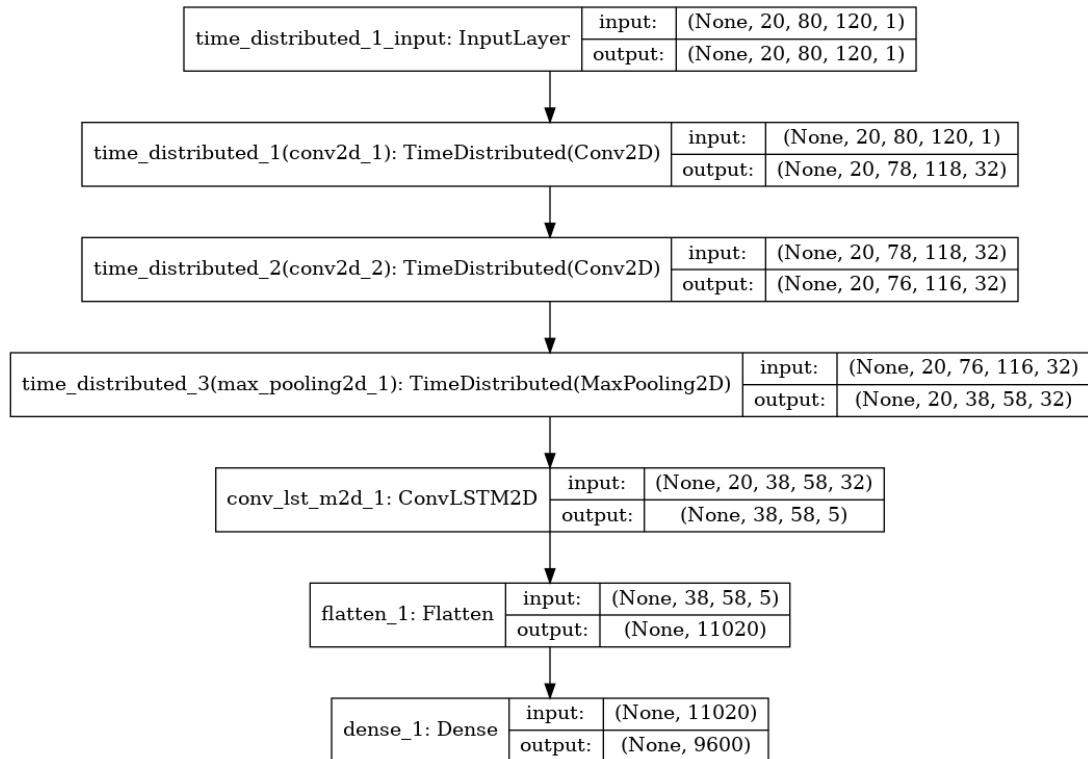


Figura 6.16: Estructura de ConvLSTM-1 para imágenes crudas.

Debido a la cantidad de valores en cada una de las secuencias de entrada, 20 imágenes con  $80 \times 120$  píxeles, es necesario realizar una reducción de la dimensionalidad para que la máquina sea capaz de procesar todos los valores. Para ello se introduce tras la entrada de la red dos capas convolucionales, de 32 neuronas cada una, seguidas de su capa

de *pooling*, que reducen la cantidad de valores a procesar. Posteriormente, se utiliza una única capa *ConvLSTM* de 5 neuronas que será la encargada de analizar las correlaciones espacio-temporales de las entradas.

Con la estructura de red definida se realizan experimentos sobre todas las dinámicas propuestas, aumentando de forma progresiva la complejidad de cada una de ellas.

### 6.3.1. Predicción con dinámicas lineales

Para la predicción en imágenes cuyo píxel se mueve siguiendo una dinámica lineal con un solo DOF, se utiliza el mismo conjunto que en el caso no recurrente: un total del 10000 muestras repartidas en 8000 para entrenar, 1000 para validar y 1000 para evaluar. La Figura 6.17 muestra los resultados obtenidos con dicho conjunto.

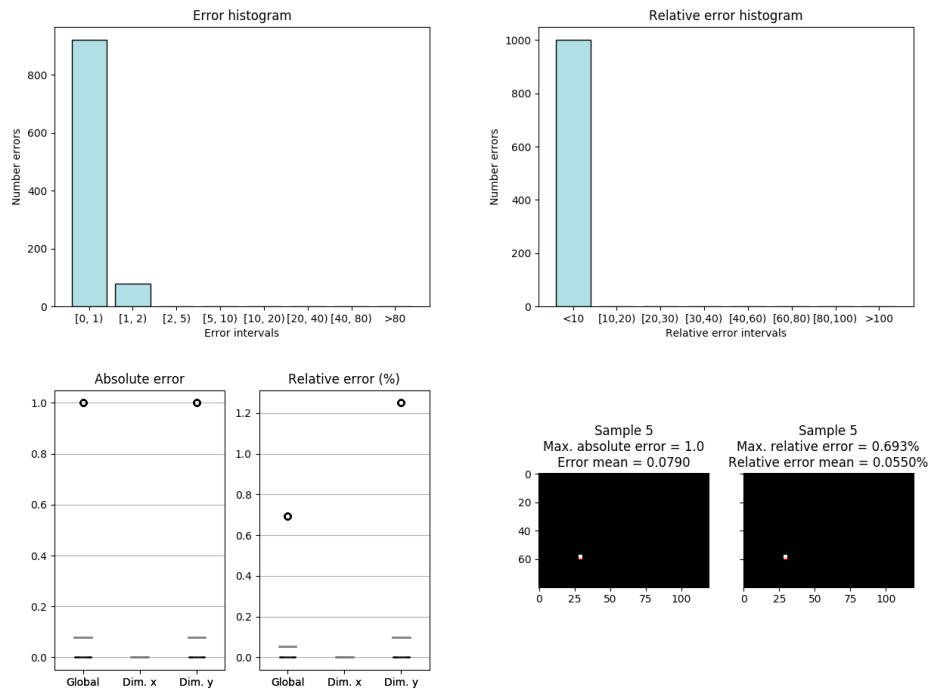


Figura 6.17: Resultados de ConvLSTM-1 con dinámica lineal de 1 DOF (1000 muestras de *test*).

Los resultados obtenidos para esta dinámica con un DOF son similares a los del caso no recurrente, proporcionando una muy buena capacidad predictiva para ambos tipos de redes. Estos resultados indican que la nueva estructura de red recurrente no deteriora las

prestaciones de la estructura no recurrente, como sí ocurría con la estructura combinada de CNN y LSTM posterior.

Siguiendo el procedimiento establecido hasta el momento, se aumenta la complejidad de la dinámica estableciendo 2 DOF, y se vuelve a entrenar la estructura neuronal propuesta. En este caso se utiliza un conjunto de 100000 muestras, distribuidas en 80000 de entrenamiento, 10000 de validación y 10000 de *test*. En la Figura 6.18 se muestran las prestaciones de dicha red bajo las condiciones mencionadas.

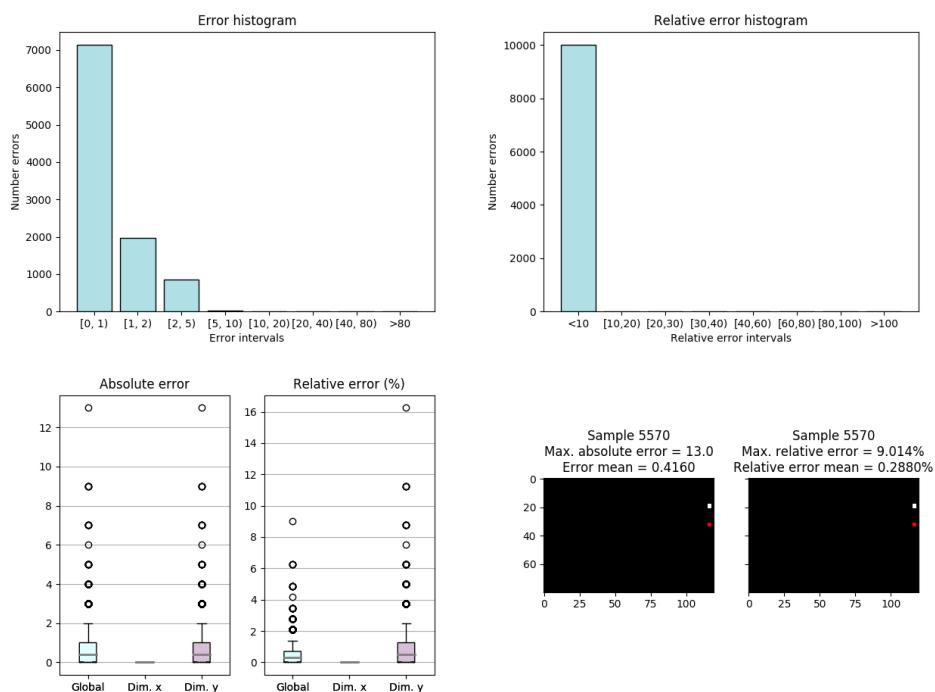


Figura 6.18: Resultados de ConvLSTM-1 con dinámica lineal de 2 DOF (10000 muestras de *test*).

Al comparar los resultados con los de la red convolucional, representados en la Figura 6.5, se observa una gran mejoría de la estructura recurrente (ConvLSTM-1) respecto a la no recurrente (CNN). Se reducen los valores de error relativo tanto en media como en máximo, y la cantidad de *outliers* es también menor. Este hecho refuerza la conclusión extraída con las imágenes modeladas de que la recurrencia, bien aplicada, mejora las prestaciones de las redes como predictores visuales.

### 6.3.2. Predicción con dinámicas parabólicas

En el caso de imágenes cuyo píxel activo sigue una la dinámica parabólica, se emplea un conjunto con 100000 ejemplos, de los que 80000 se emplearán para entrenar, 10000 para validar y otros 10000 para evaluar. En la Tabla 6.3 se muestra la comparación del error relativo obtenido, tanto máximo como medio, con la red sin recurrencia, la CNN, y con la red recurrente *ConvLSTM-1*. En dicha tabla se reafirma el hecho de que introducir de manera adecuada la recurrencia en la estructura de la red conduce a una mejora en las prestaciones.

DOF	CNN		ConvLSTM-1	
	<i>Max.</i>	<i>Mean</i>	<i>Max.</i>	<i>Mean</i>
1 ( <i>a</i> )	0.7 %	0.01 %	0.7 %	0.01 %
2 ( <i>c</i> )	8.3 %	0.07 %	4.9 %	0.03 %
3 ( <i>b</i> )	54.7 %	4.4 %	22.9 %	3.8 %

Tabla 6.3: Error relativo en la dinámica parabólica con CNN y ConvLSTM-1 (10000 muestras de *test*).

A pesar de haber mejorado los resultados obtenidos, en el caso más complejo de la dinámica, 3 DOF, se continúa sin conseguir una buena capacidad predictiva, lo que deja un margen de mejora a estudiar por distintas vías.

### 6.3.3. Predicción con dinámicas sinusoidales

Para la dinámica sinusoidal se emplea para cada caso un conjunto de 100000 muestras, que se reparten en 80000 para entrenamiento, 10000 para validación y 10000 para *test*. La Tabla 6.4 refleja la misma comparación que en el caso parabólico. Los resultados muestran que, a pesar del aporte de la recurrencia en las prestaciones, la mejora obtenida es muy pequeña y se continúa sin ser capaz de predecir en los casos más complejos de la dinámica sinusoidal.

DOF	CNN		ConvLSTM-1	
	<i>Max.</i>	<i>Mean</i>	<i>Max.</i>	<i>Mean</i>
<b>1 (<i>f</i>)</b>	0.7 %	0.01 %	0.7 %	0.01 %
<b>2 (<i>yθ</i>)</b>	64.0 %	1.1 %	44.3 %	1.1 %
<b>3 (<i>A</i>)</b>	×	×	46.5 %	3.4 %
<b>3 (<i>c</i>)</b>	×	×	54.1 %	13 %

Tabla 6.4: Error relativo en la dinámica sinusoidal con ConvLSTM-1 (10000 muestras de *test*).

### 6.3.4. Resumen de resultados

La Tabla 6.5 presenta un resumen de los mejores resultados obtenidos utilizando la red *ConvLSTM-1* para cada una de las dinámicas.

Dinámica		ConvLSTM-1
Lineal	1 DOF	0.06 %
	2 DOF	0.29 %
Parabólica	1 DOF	0.01 %
	2 DOF	0.03 %
	3 DOF	3.76 %
Sinusoidal	1 DOF	0.01 %
	2 DOF	1.12 %
	3 DOF	3.44 %
	4 DOF	13.00 %

Tabla 6.5: Promedio del error relativo en *test* al evaluar la red ConvLSTM-1 con imágenes modeladas y distintas dinámicas (10000 muestras de *test*).

Al igual que en el primer caso, se ha reevaluado la dinámica lineal de 1 DOF con 10000 muestras para que la comparación entre las redes sea equitativa.

## 6.4. Arquitectura recurrente: ConvLSTM-4

Para tratar de obtener una arquitectura capaz de predecir en todas las dinámicas en su caso más complejo se realizan una serie de experimentos que dan lugar a la estructura de red recurrente definida en la Figura 6.19.

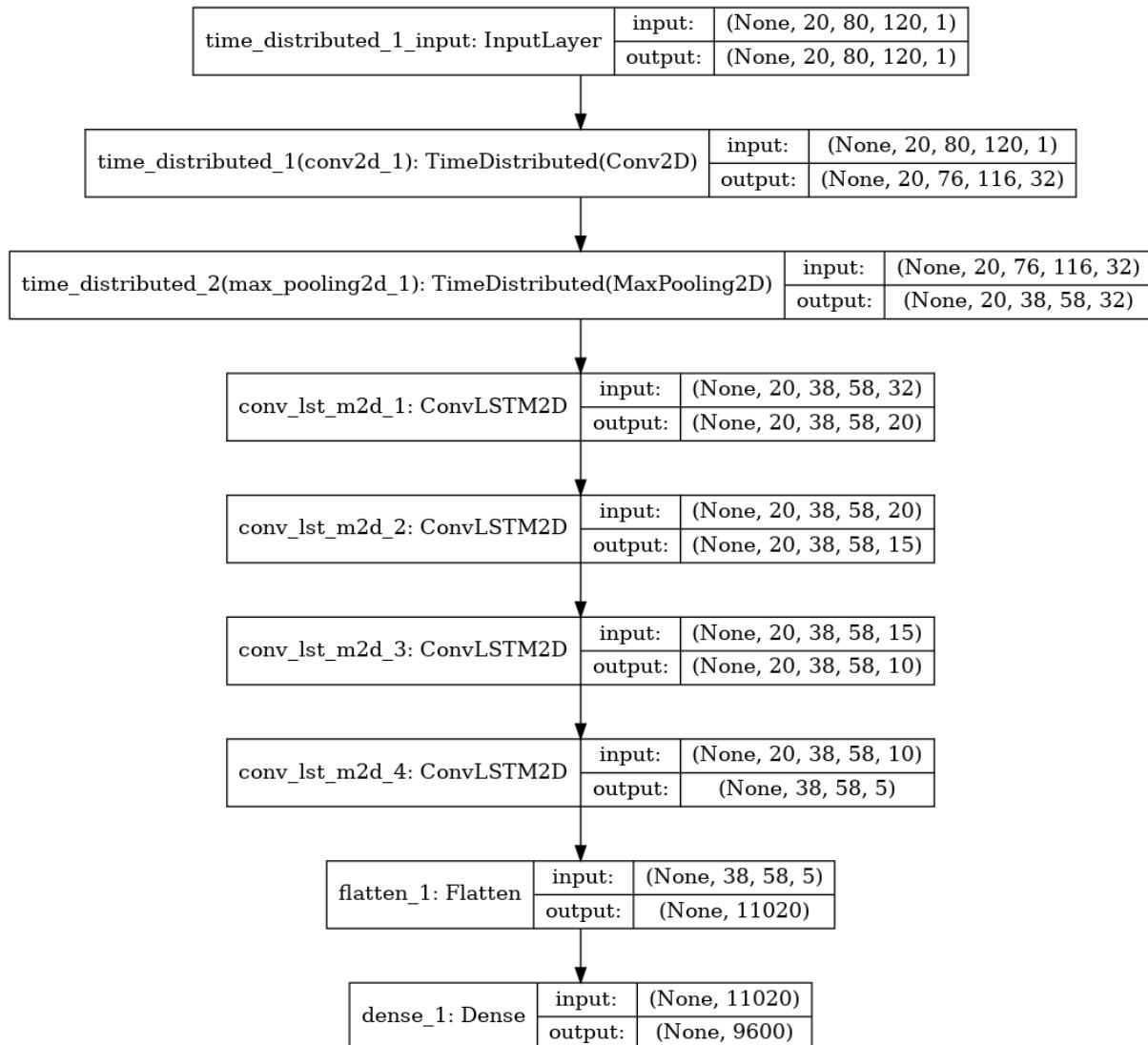


Figura 6.19: Estructura de ConvLSTM-4 para imágenes crudas.

La nueva arquitectura se compone de 4 capas *ConvLSTM* de 20, 15, 10 y 5 neuronas que vienen precedidas de una única estructura convolucional (capa convolucional y de *pooling*) para la reducción de la cantidad de valores de entrada a la red.

Con el objetivo de mejorar los resultados, por un lado se reduce el número de capas convolucionales previas a la recurrencia, para evitar perder demasiadas correlaciones y perjudicar a la predicción. Por otro lado, y conforme a las conclusiones extraídas de las imágenes modeladas, se opta por aumentar el número de capas *ConvLSTM* que se introducen en la estructura.

#### 6.4.1. Aumento de capas

En línea con el estudio del Apartado 5.3.2, se explora dónde situar el límite de número de capas a añadir a la estructura para obtener las mejores prestaciones. Para ello se ha utilizado el conjunto que sigue una dinámica sinusoidal de máxima complejidad, 4 DOF, con un total de 100000 muestras: 80000 de entrenamiento, 10000 de validación y 10000 de *test*. En la Figura 6.20 se muestra el resultado de este estudio con la evolución del error relativo según el número de capas utilizado.

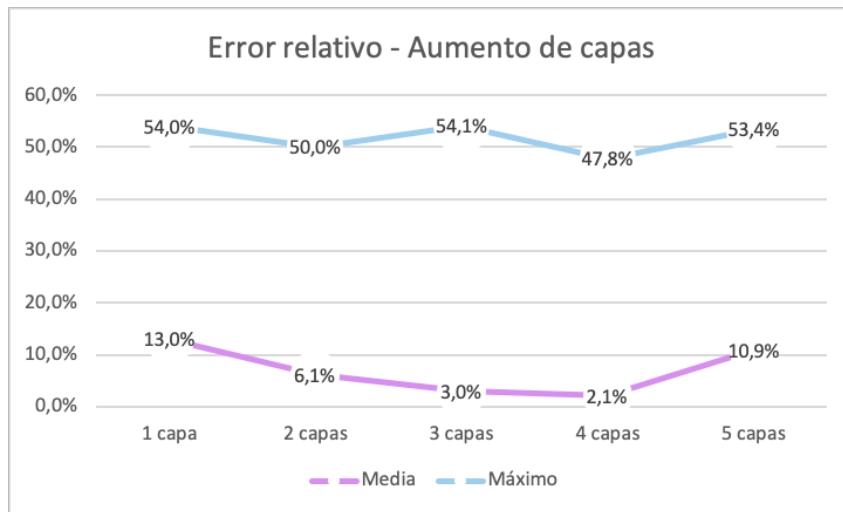


Figura 6.20: Comparación del error relativo al aumentar el número de capas *ConvLSTM* (Sinusoidal, 4 DOF, 10000 muestras de *test*).

El gráfico ilustra que a medida que se aumenta el número de capas *ConvLSTM* en la estructura, las prestaciones de la misma mejoran en promedio. Sin embargo, al llegar a 4 capas se obtiene el mínimo en el valor de la media del error relativo, pues al pasar a 5 capas los resultados vuelven a empeorar. Este hecho hace que la estructura escogida esté formada por 4 capas *ConvLSTM*, pues es la que mejores prestaciones ofrece.

### 6.4.2. Predicción con dinámicas lineales

En la Figura 6.21 se muestran los resultados obtenidos utilizando un conjunto que sigue la dinámica lineal de un DOF. Este conjunto consta de 100000 muestras en total, de las que 80000 se usan para el entrenamiento, 10000 para la validación y 10000 para la evaluación.

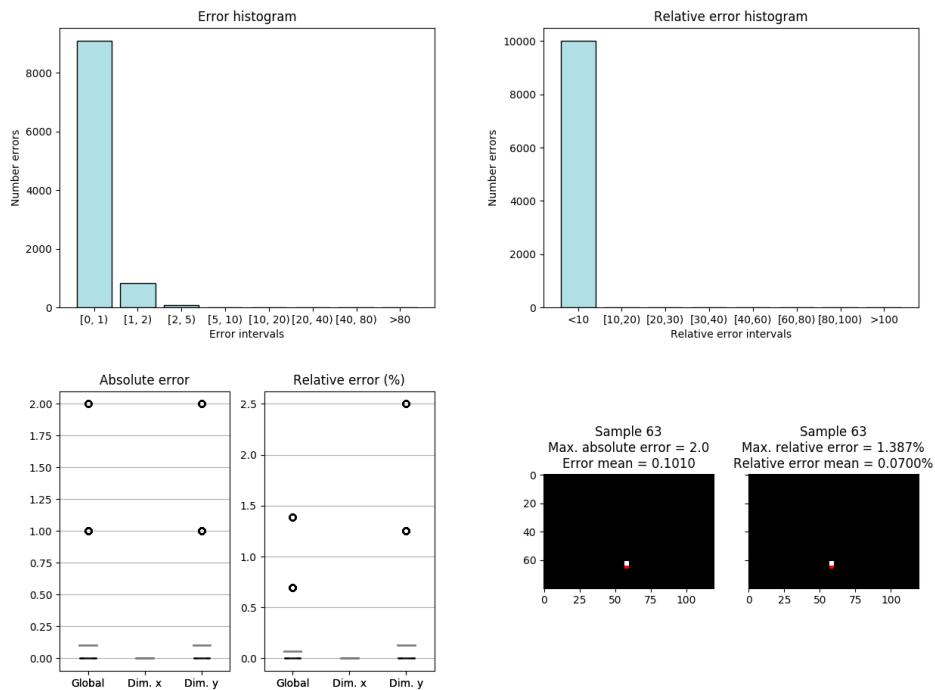


Figura 6.21: Resultados de ConvLSTM-4 con dinámica lineal de 1 DOF (10000 muestras de *test*).

El aumento de número de capas tiene un claro efecto positivo en el caso más complejo de esta dinámica, reduciendo considerablemente el error cometido y el número de *outliers*.

### 6.4.3. Predicción con dinámicas parabólicas

Para la dinámica parabólica se emplea el mismo conjunto utilizado con la estructura de una capa, manteniendo todos los grados de libertad. En la Figura 6.22 se muestran los resultados obtenidos para este caso. Se puede comprobar que, aunque sí hay una mejora en cuanto al valor del promedio del error relativo, se produce una mayor dispersión en los resultados que reduce la capacidad predictiva de la red.

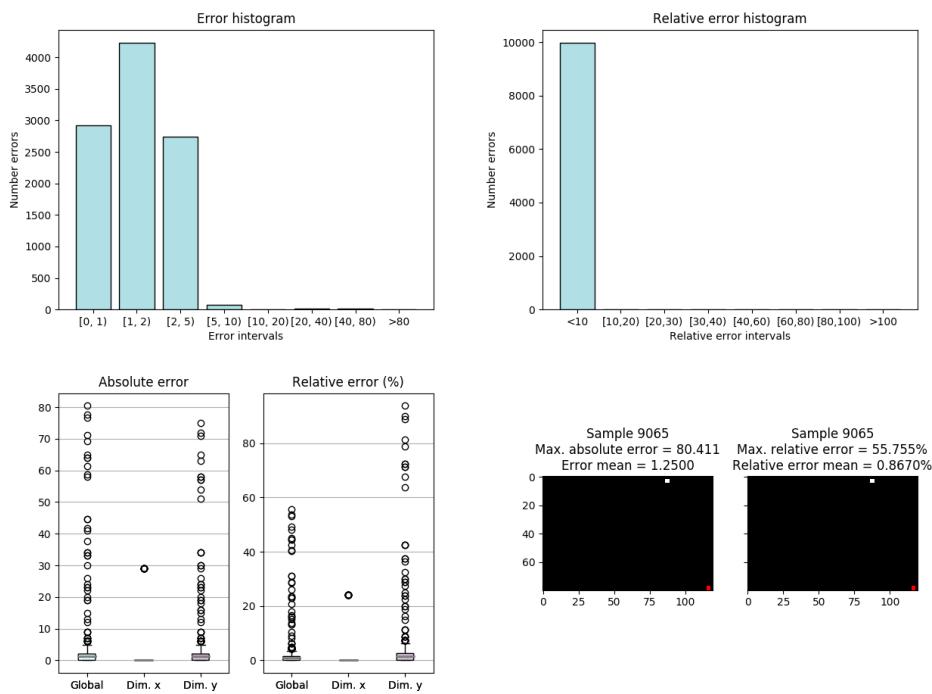


Figura 6.22: Resultados de ConvLSTM-4 con dinámica parabólica de 2 DOF (10000 muestras de *test*).

#### 6.4.4. Predicción con dinámicas sinusoidales

En la última dinámica propuesta, la sinusoidal, con las redes anteriores no se conseguía predecir bien con varios DOF. Por este motivo la nueva estructura de red se aplica, no solo al caso más complejo de 4 DOF, sino también a los casos de 2 y 3 DOF cuyos resultados no fueron satisfactorios.

En la Figura 6.23 se muestran los resultados obtenidos para la dinámica con 2 DOF. Con la nueva estructura propuesta se consigue dominar esta dinámica en términos de media del error relativo, pero sigue existiendo un gran número de *outliers* que reducen la capacidad de predicción de la red.

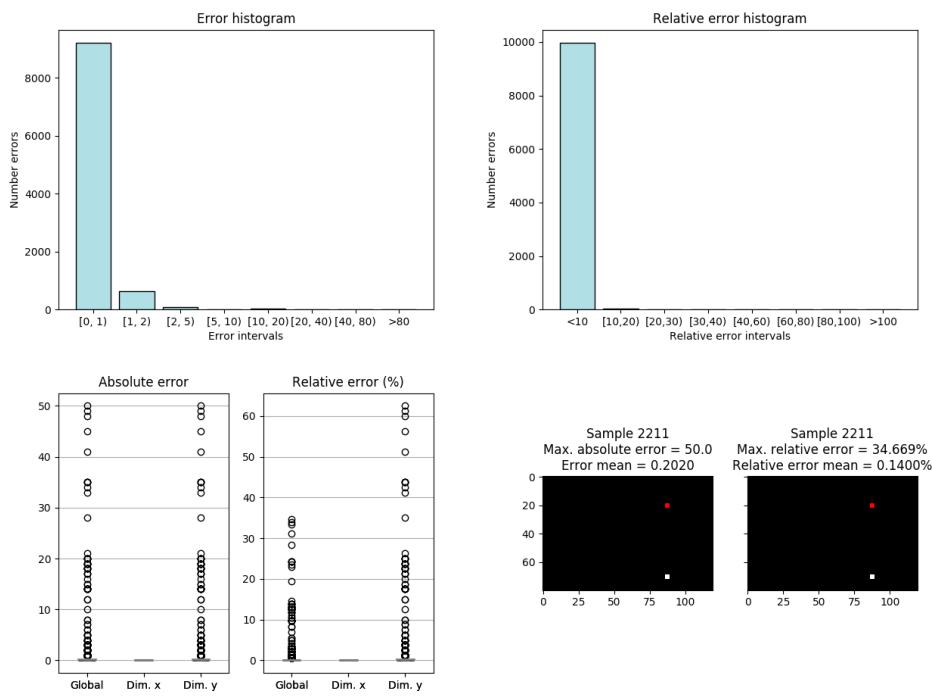


Figura 6.23: Resultados de ConvLSTM-4 con dinámica sinusoidal de 2 DOF (10000 muestras de *test*).

Aumentando la complejidad con un grado de libertad más, 3 DOF, se obtienen los resultados mostrados en la Figura 6.24. La situación es muy similar al caso anterior, se mejoran los valores de promedio del error relativo, pero la presencia de múltiples *outliers* hace que las prestaciones de la red se vean reducidas.

Para terminar con la dinámica sinusoidal, en la Figura 6.25 se muestran los resultados obtenidos para la dinámica con 4 DOF. Como ocurría en el caso parabólico, a pesar de una clara mejora en los valores, la existencia de *outliers* de valores elevados empeoran los resultados y perjudican a la capacidad de predicción de la red.

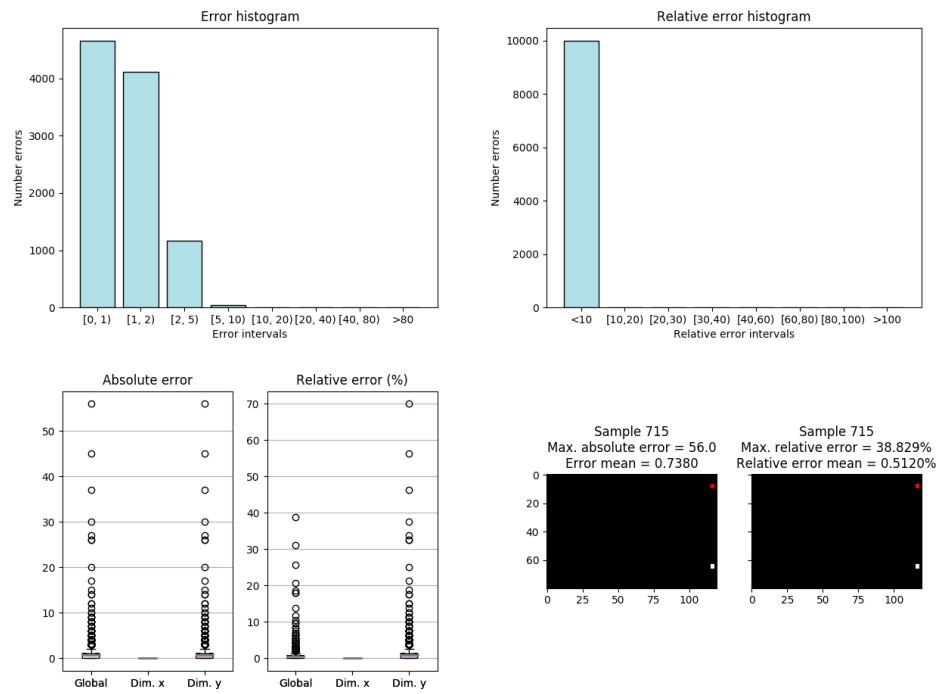


Figura 6.24: Resultados de ConvLSTM-4 con dinámica sinusoidal de 3 DOF (10000 muestras de *test*).

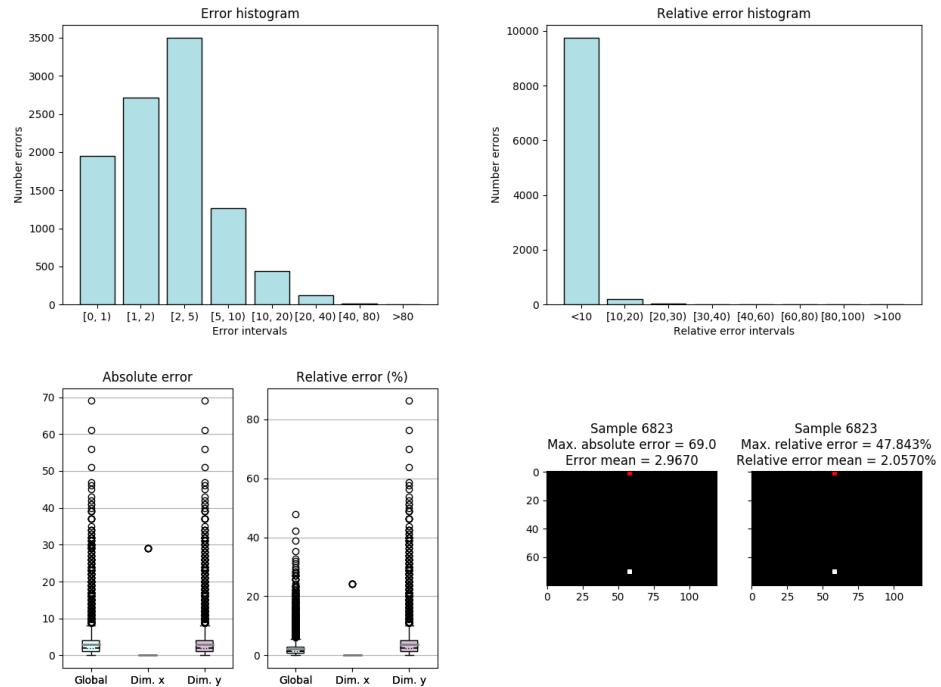


Figura 6.25: Resultados de ConvLSTM-4 con dinámica sinusoidal de 4 DOF (10000 muestras de *test*).

#### 6.4.5. Predicción con dinámica combinada

Por último, se realiza un experimento similar al que se realiza con imágenes modeladas, utilizando un conjunto que combine muestras de los tres tipos. Para ello se utilizan un total 33000 ejemplos que siguen una dinámica lineal, 33000 que siguen una parabólica y 34000 que siguen una sinusoidal, repartidas en una proporción de 80%-10%-10% para los subconjuntos de entrenamiento, validación y *test* respectivamente. Con esto, se obtienen los resultados mostrados en la Figura 6.26.

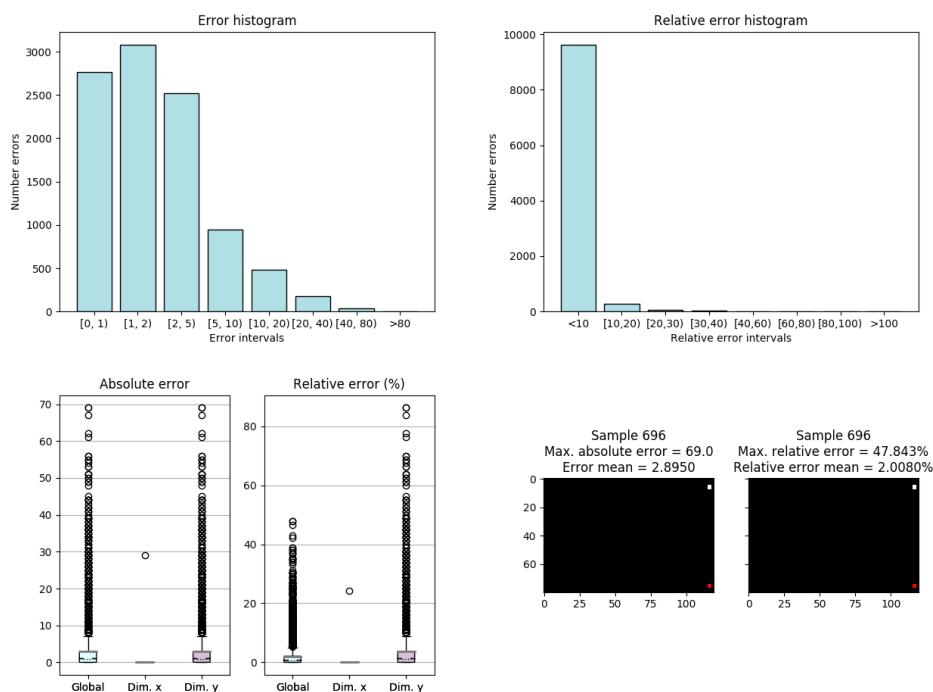


Figura 6.26: Resultados de ConvLSTM-4 con dinámica combinada (10000 muestras de *test*).

Los resultados obtenidos son muy similares a los que se consiguen con la dinámica más compleja de todas, la sinusoidal con 4 DOF. Sin embargo, se obtiene una ligera mejora en el la media del error relativo está propiciada por las buenas prestaciones en el caso lineal.

#### 6.4.6. Resumen de resultados

En la Tabla 6.6 se muestra el resumen de los resultados obtenidos con la red *LSTM-4* para las dinámicas consideradas.

Dinámica		LSTM-4
Lineal	2 DOF	0.07 %
Parabólica	3 DOF	0.87 %
Sinusoidal	2 DOF	0.14 %
	3 DOF	0.51 %
	4 DOF	2.06 %
Combinada		2.01 %

Tabla 6.6: Promedio del error relativo en *test* al evaluar la red ConvLSTM-4 con imágenes modeladas y distintas dinámicas (10000 muestras de *test*).

Como en los casos anteriores, con el objetivo de que la comparación de los resultados sea equitativa, se ha reevaluado la dinámica lineal de 1 DOF con 10000 muestras.

La estructura ConvLSTM-4 consigue mejorar los resultados en cuanto al promedio del error relativo en todos los casos, sin embargo no se consigue solucionar el problema de los *outliers*. Este hecho hace que, aunque se mejoren las prestaciones de la red, éstas no alcanzan unos valores suficientes para concluir que la red es capaz de predecir satisfactoriamente en todas las dinámicas independientemente de su complejidad.

## 6.5. Comparativa global

Tras realizar los experimentos con imágenes crudas se obtiene un conjunto de redes entrenadas para las distintas dinámicas. Para comparar de una forma rápida todas ellas, se ha elaborado la Tabla 6.7, que sigue el mismo código de colores especificado en la Sección 5.4. También se han utilizado los resultados de la dinámica lineal de 1 DOF obtenidos con 10000 muestras para que la comparación sea equitativa.

Dinámica		CNN	ConvLSTM-1	ConvLSTM-4
Lineal	1 DOF	0.07 %	0.06 %	
	2 DOF	0.39 %	0.29 %	0.07 %
Parabólica	1 DOF	0.01 %	0.01 %	
	2 DOF	0.07 %	0.03 %	
	3 DOF	4.4 %	3.76 %	0.87 %
Sinusoidal	1 DOF	0.003 %	0.01 %	
	2 DOF	1.12 %	1.08 %	0.14 %
	3 DOF		3.44 %	0.54 %
	4 DOF		13 %	2.06 %
Combinada				2.01 %

Tabla 6.7: Comparativa del promedio de error relativo en las distintas redes para imágenes crudas con las distintas dinámicas (10000 muestras de *test*).

Los resultados obtenidos reflejan la complejidad de la tarea de predicción con las imágenes en crudo, debido al elevado número de parámetros a encontrar. La red que obtiene mejores resultados es la que incorpora un mayor número de capas y, además, utiliza recurrencia. Sin embargo, no se consigue dominar todas las dinámicas en su grado más complejo, con el máximo número de DOF. Únicamente la dinámica lineal produce unos resultados que indican que la red es capaz de predecir bien.

En cuanto al número de muestras utilizadas en el entrenamiento, su aumento puede aportar una mejora a las prestaciones siempre y cuando no se sobrepase un límite. Este límite viene determinado por la naturaleza de las imágenes a predecir. Si se establece un número muy elevado de muestras en el aprendizaje, se aumenta la complejidad del entrenamiento sin obtener un beneficio del mismo. Algo parecido pasa con el número de capas, incrementar en exceso este valor puede proporcionar una estructura excesivamente compleja que no solo no mejore los resultados, sino que pueda llegar a empeorarlos.

Por otro lado, el uso de capas que analicen las relaciones espaciales junto con las temporales (*ConvLSTM*) introduce una mejora en las prestaciones de la red. Sin embargo, si se separan ambos tipos de relación (CNN+LSTM), no solo no se obtienen mejores resultados que sin recurrencia, se reduce por completo la capacidad predictiva de la red. Además,

## CAPÍTULO 6. PREDICCIÓN DE IMÁGENES CRUDAS

---

el número de capas utilizado repercute en los resultados, introduciendo una mejora en los resultados de promedio del error relativo a medida que se aumentan, hasta 4 capas, pero sin solucionar el problema de los *outliers* que empeora la predicción.

Con todo lo expuesto anteriormente se concluye que, con imágenes crudas, ConvLSTM-4 es la red que mejores resultados obtiene. Esta red consigue dominar por completo únicamente la dinámica lineal. Para el resto de dinámicas se deja la puerta abierta a nuevas investigaciones que permitan reducir el número de valores atípicos y mejorar las prestaciones para los casos más complejos.

# Capítulo 7

## Conclusiones

En este capítulo se exponen las conclusiones alcanzadas con el desarrollo del proyecto, las aportaciones principales y posibles líneas para continuar con este trabajo.

### 7.1. Conclusiones

El desarrollo de este trabajo ha permitido llegar una serie de conclusiones referentes a la tarea de predicción en secuencias de vídeo mediante el uso de redes neuronales profundas. A continuación se exponen las más importantes, desglosadas según los subobjetivos definidos en la Sección 1.4.

#### Desarrollo *software* para ejecución y evaluación de redes

Para realizar el diseño y análisis de una estructura neuronal concreta con un conjunto de imágenes determinado, se han desarrollado dos herramientas *software* en Python. Una de ellas permite la ejecución de las distintas redes neuronales y la otra obtiene las figuras de mérito para comparar las prestaciones de las redes.

#### Creación de las bases de datos

Con la generación de una secuencia a partir de un único píxel activo cuya posición puede cambiar en fotogramas consecutivos conforme a una dinámica determinada, se han creado los conjuntos necesarios para entrenar y evaluar las distintas redes neuronales. Estos conjuntos constan de dos tipos de imágenes, modeladas y crudas, y rigen el movimiento del píxel con tres dinámicas: lineal, parabólica y sinusoidal. Además, se considera una cuarta dinámica combinada que mezcla, en un mismo

conjunto de muestras, ejemplos de cada una de las dinámicas. Para obtener las bases de datos necesarias se ha programado también un generador en Python que crea un determinado conjunto que está gobernado por unos parámetros concretos.

### Estudio de la predicción con imágenes modeladas

Las imágenes modeladas son fotogramas simplificados, resumidos a las coordenadas del píxel activo en cada instante. El análisis de las distintas redes neuronales para la predicción de la posición del píxel en conjuntos que siguen diferentes dinámicas de movimiento ha dado lugar a las siguientes conclusiones:

- Las redes neuronales LSTM-4 predicen satisfactoriamente en todas las dinámicas, incluyendo las más complejas.
- El número de muestras utilizadas en el entrenamiento repercute en los resultados obtenidos. En dinámicas más complejas, se obtienen mejores resultados cuando se utiliza un número de muestras mayor. Sin embargo, un aumento excesivo puede ser contraproducente.
- La elección del tipo de red tiene un efecto directo en la capacidad de predicción. La estructura MLP establece su límite de predicción en la dinámica sinusoidal de 1 DOF, mientras que la red LSTM-1 es capaz de predecir razonablemente bien hasta con 3 DOF de dicha dinámica. La estructura LSTM-4, por el contrario, es capaz de predecir satisfactoriamente en todas las dinámicas.
- La recurrencia aporta una mejora a los resultados, pues es capaz de captar las correlaciones temporales que una red no recurrente no puede procesar.
- El aumento del número de neuronas en la capa LSTM no proporciona una mejora significativa en la capacidad predictiva de la red.
- El aumento del número de capas LSTM en la estructura hace mejorar los resultados. Sin embargo, se establece un límite en este número a partir del cual crece la complejidad disminuyendo las prestaciones
- El incremento del horizonte temporal a predecir complica la tarea, siendo más complicado para la red establecer una correlación temporal entre la última muestra conocida y la que se quiere estimar. A pesar de esto, se logra una predicción razonable cuando se establece un *gap* elevado (50 instantes temporales).

### Estudio de la predicción con imágenes crudas

El análisis de distintas estructuras neuronales como predictores visuales con imágenes crudas, matrices 2D, da lugar a las conclusiones expuestas a continuación. Algunas de ellas coinciden con las extraídas para imágenes modeladas.

- Se consigue predecir con buenos resultados en un gran número de dinámicas. En las dinámicas más complejas se obtienen resultados mejorables que limitan la capacidad de predicción con imágenes crudas.
- El número de muestras utilizadas en el entrenamiento repercute en los resultados obtenidos, de la misma forma que en el caso de imágenes modeladas.
- El análisis separado de las correlaciones espaciales y temporales, concatenando una red convolucional con una LSTM, no proporciona buenos resultados.
- El uso de redes que son capaces de analizar las correlaciones espacio-temporales de forma simultánea (*ConvLSTM*) eleva su calidad como predictores visuales.
- La expansión gaussiana del píxel para facilitar la obtención de correlaciones espaciales en el fotograma, mejora ligeramente las prestaciones. Sin embargo, esta mejora no es suficiente como para que la estructura que procesa independiente de las correlaciones espaciales y temporales (CNN+LSTM) sea considerada una buena estrategia.
- El aumento del número de capas *ConvLSTM* en la estructura hace mejorar los resultados de la misma forma que ocurría en las LSTM con las imágenes modeladas.

Finalmente, en cuanto a la comparación entre la predicción usando imágenes modeladas y crudas, ha sido posible encontrar una red (LSTM-4) que predice satisfactoriamente en todas las casuísticas modeladas. En contraposición, para las imágenes crudas, la mejor red obtenida (ConvLSTM-4) presenta ciertas dificultades de predicción. Este hecho demuestra que para la red es más complicado establecer relaciones entre matrices 2D, formadas por píxeles, que entre un par de valores ( $x, y$ ), pues la complejidad de los datos es mayor en el primer caso.

En resumen, los objetivos planteados al comienzo de este Trabajo Fin de Máster se han alcanzado satisfactoriamente.

## 7.2. Líneas futuras

Para continuar con la investigación abordada en este trabajo se pueden seguir varias vías que permitan obtener más resultados interesantes en el ámbito de la predicción con secuencias de vídeo.

- En cuanto a la generación de muestras, es posible realizar un estudio más amplio sobre los valores que se otorgan a los parámetros que permanecen fijos, como la amplitud en la dinámica sinusoidal de 1 DOF. Además, se puede ampliar el rango en el que se generan los valores aleatorios, como la pendiente de la recta en la dinámica lineal de 1 DOF, incrementando las distintas posibilidades en la dinámica.
- En las imágenes modeladas, se puede realizar un análisis más profundo sobre el efecto de modificar la estructura de red aumentando simultáneamente tanto el número de capas como el de neuronas. De esta forma, se podrían mejorar los resultados obtenidos en las dinámicas más complejas que, a pesar de ser satisfactorios, permiten dicha mejora.
- Respecto a las imágenes crudas, hay aún un amplio campo de exploración en cuanto a la obtención de una estructura neuronal que sea capaz de predecir satisfactoriamente con todas las dinámicas.
- Referente a la expansión del píxel con valores decrecientes de luminancia, dado que sí se obtuvo una mejora, se puede aumentar el campo de exploración modificando el área de expansión y la función que rige la pérdida de intensidad. De esta forma se analiza su efecto en el entrenamiento de las redes y en sus prestaciones finales, y se comprobará si una expansión adecuada puede lograr una predicción satisfactoria.
- Una de las limitaciones de este trabajo es la sencillez de las imágenes en cuanto a tamaño y contenido, pues aparece un único píxel activo. En este aspecto, sería interesante investigar el efecto de aumentar el tamaño de la imagen a predecir, así como la modificación del tamaño y forma del objeto móvil.
- Otra limitación se encuentra en que las dinámicas que rigen el movimiento del píxel no sufren ningún tipo de distorsión o ruido. Este hecho permite profundizar en la investigación de secuencias de movimiento que no sean tan limpias, asemejándose más a un movimiento real.

## CAPÍTULO 7. CONCLUSIONES

---

- La última limitación del trabajo es que el muestreo se realiza de forma regular, con una velocidad constante y sin eliminación de muestras. En este sentido, queda abierta la exploración a secuencias cuyo muestreo se realice de forma no regular, con pérdida de datos en algunas posiciones o introduciendo una aceleración.
- Finalmente, es posible trasladar el estudio a una aplicación en el mundo real, que proporcione ayuda en la solución de problemas que se presentan en el día a día. A modo de ejemplo, se podría estimar la posición de una persona en movimiento, con el objetivo de agilizar su seguimiento si en algún momento se pierde su detección.

# Bibliografía

- [1] Wikipedia. Visión artificial. [https://es.wikipedia.org/wiki/Vision\\_artificial](https://es.wikipedia.org/wiki/Vision_artificial), 2020. [Accedido 24 de Septiembre de 2020].
- [2] Barath Narayanan Narayanan, Manawaduge Supun De Silva, Russell C. Hardie, Nathan K. Kueterman, and Redha Ali. Understanding deep neural network predictions for medical imaging applications. <https://arxiv.org/abs/1912.09621>, 2019. [Accedido 24 de Septiembre de 2020].
- [3] AeroCovid ©. Aplicaciones de aerocovid ©. <https://aerocovid.com/aplicaciones/>, 2017. [Accedido 24 de Septiembre de 2020].
- [4] Istituto Italiano di Tecnologia. Person detection and tracking. <https://pavis.iit.it/projects/people-detection-and-tracking>, 2020. [Accedido 24 de Septiembre de 2020].
- [5] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.
- [6] Joseph Redmon, S. Divvala, Ross B. Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788, 2016.
- [7] Delano Marques, Alex Barradas Filho, Alexandre Romariz, Isabelle Viegas, Djavania Luz, Allan Kardec Barros Filho, Sofiane Labidi, and Antonio Ferrando. Recent developments on statistical and neural network tools focusing on biodiesel quality. *International Journal of Computer Science and Application*, 3:97, 01 2014.
- [8] M.Eng. Dewi Suryani, S.Kom. “Convolutional Neural Network”, *Binus University - School of Computer Science*. <http://socs.binus.ac.id/2017/02/27/convolutional-neural-network/>, 2017. [Accedido 24 de Septiembre de 2020].
- [9] Christopher Olah. Understanding LSTM Networks. <https://colah.github.io/posts/2015-08-Understanding-LSTMs/>, 2015. [Accedido 24 de Septiembre de 2020].

## BIBLIOGRAFÍA

---

- [10] Alexandre Xavier. An introduction to ConvLSTM. <https://medium.com/neuronio/an-introduction-to-convlstm-55c9025563a7>, 2019. [Accedido 24 de Septiembre de 2020].
- [11] Catalin Stoean, Wiesław Paja, Ruxandra Stoean, and Adrian Sand ita. Deep architectures for long-term stock price prediction with a heuristic-based strategy for trading simulations. *PLoS ONE*, 14(10):e0223593, October 2019.
- [12] Carl Vondrick, Hamed Pirsiavash, and Antonio Torralba. Anticipating visual representations from unlabeled video. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 98–106, 06 2016.
- [13] Junhyuk Oh, Xiaoxiao Guo, Honglak Lee, Richard L Lewis, and Satinder Singh. Action-conditional video prediction using deep networks in atari games. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 2863–2871. Curran Associates, Inc., 2015.
- [14] Neda Cvijetic (Nvidia). Drive labs: Predicting the future with rnns. <https://blogs.nvidia.com/blog/2019/05/22/drive-labs-predicting-future-motion/>, 2019. [Accedido 24 de Septiembre de 2020].
- [15] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.
- [16] Peter Zhang. Zhang, g.p.: Time series forecasting using a hybrid arima and neural network model. *neurocomputing* 50, 159-175. *Neurocomputing*, 50:159–175, 01 2003.
- [17] Takaomi Hirata, Takashi Kuremoto, Masanao Obayashi, Shingo Mabu, and Kunikazu Kobayashi. Time series prediction using dbn and arima. In *2015 International Conference on Computer Application Technologies*, pages 24–29, 2015.
- [18] Erol Egrioglu, Ufuk Yolcu, Cagdas Aladag, and Eren Bas. Recurrent multiplicative neuron model artificial neural network for non-linear time series forecasting. *Neural Processing Letters*, 109, 04 2014.
- [19] Filipe Rodrigues, Ioulia Markou, and Francisco C. Pereira. Combining time-series and textual data for taxi demand prediction in event areas: A deep learning approach. *Information Fusion*, 49:120–129, Sep 2019.

## BIBLIOGRAFÍA

---

- [20] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese. Social lstm: Human trajectory prediction in crowded spaces. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 961–971, 2016.
- [21] Seong Park, ByeongDo Kim, Chang Kang, Chung Chung, and Jun Choi. Sequence-to-sequence prediction of vehicle trajectory via lstm encoder-decoder architecture. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 1672–1678, 06 2018.
- [22] Ruben Villegas, Jimei Yang, Seunghoon Hong, Xunyu Lin, and Honglak Lee. Decomposing motion and content for natural video sequence prediction. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.
- [23] Jason Y. Zhang, Panna Felsen, A. Kanazawa, and Jitendra Malik. Predicting 3d human dynamics from video. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 7113–7122, 2019.
- [24] Osamu Shouno. Photo-realistic video prediction on natural videos of largely changing frames. <https://arxiv.org/abs/2003.08635>, 2020. [Accedido 01 de Octubre de 2020].
- [25] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. <http://www.deeplearningbook.org>.
- [26] Microsoft. Project innereye – democratizing medical imaging ai. <https://www.microsoft.com/en-us/research/project/medical-image-analysis>, 2020. [Accedido 24 de Septiembre de 2020].
- [27] Anyu. “RNA - Redes Neuronales Artificiales”, *Bitácoras de un Ingeniero*. <http://andrealezcano.blogspot.com.es/2011/04/rna-redes-neuronales-artificiales.html>, 2011. [Accedido 24 de Septiembre de 2020].
- [28] Missinglink.ai. Perceptrons and multi-layer perceptrons: The artificial neuron at the core of deep learning. <https://missinglink.ai/guides/neural-network-concepts/perceptrons-and-multi-layer-perceptrons-the-artificial-neuron-at-the-core-of-deep-learning/>, 2020. [Accedido 24 de Septiembre de 2020].
- [29] Xingjian Shi, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-Kin Wong, and Wang-chun Woo. Convolutional LSTM network: A machine learning approach for precipitation nowcasting. *CoRR*, abs/1506.04214, 2015.

## BIBLIOGRAFÍA

---

- [30] Jeffrey L Elman. Finding structure in time. *Cognitive science*, 14(2):179–211, 1990.
- [31] Timo Koskela, Mikko Lehtokangas, Jukka Saarinen, and Kimmo Kaski. Time series prediction with multilayer perceptron, fir and elman neural networks. In *Proceedings of the World Congress on Neural Networks*, pages 491–496. INNS Press San Diego, USA, 1996.
- [32] Kevin J. Lang, Alex H. Waibel, and Geoffrey E. Hinton. A time-delay neural network architecture for isolated word recognition. *Neural Networks*, 3(1):23 – 43, 1990.
- [33] John. J. Hopfield. Hopfield network. *Scholarpedia*, 2(5):1977, 2007. revision #91363.
- [34] J. Contreras, Rosario Espinola, Francisco Nogales, and Antonio Conejo. Arima models to predict next-day electricity prices. *Power Engineering Review, IEEE*, 22:57 – 57, 10 2002.
- [35] George E. P. Box, Gregory C. Reinsel, and Gwilym M. Jenkins. *Time series analysis: forecasting and control*. Prentice-Hall, Englewood Cliffs, NJ, 1994.