

Universidad Politécnica  
de Madrid



**Escuela Técnica Superior de  
Ingenieros Informáticos**

Máster Universitario en Inteligencia Artificial

Trabajo Fin de Máster

**Predicción de crecimiento de colonias de  
bacterias mediante técnicas de  
generación de fotogramas**

Autor(a): Silvia Martín Suazo  
Tutor(a): Alfonso Rodríguez-Patón

Madrid, Julio 2021

Este Trabajo Fin de Máster se ha depositado en la ETSI Informáticos de la Universidad Politécnica de Madrid para su defensa.

*Trabajo Fin de Máster*  
*Máster Universitario en Inteligencia Artificial*

**Título:** Predicción de crecimiento de colonias de bacterias mediante técnicas de generación de fotogramas

Julio 2021

**Autor(a):** Silvia Martín Suazo  
**Tutor(a):** Alfonso Rodríguez-Patón  
Inteligencia Artificial  
ETSI Informáticos  
Universidad Politécnica de Madrid

# **Resumen**

La aplicación del Aprendizaje Automático para la predicción de fotogramas en vídeos es uno de los campos más innovadores y desconocidos dentro de la IA, a pesar de que numerosas redes neuronales han surgido a partir de un intento de dar solución a dicha problemática. En este proyecto se exploran las técnicas más destacadas de las propuestas durante los últimos años, estudiando el funcionamiento de su arquitectura y sus aportaciones a las soluciones anteriormente propuestas. Dicho estudio se desarrolla desde dos enfoques distintos: las soluciones que a partir de una secuencia generan un fotograma y las soluciones que a partir de una secuencia generan otra secuencia. Para ello, en primer lugar se ha realizado un estudio detallado de las redes y estructuras básicas del Aprendizaje Profundo, de forma que se ofrece un marco teórico a los términos utilizados en este documento.

A continuación, se ha propuesto una aplicación práctica a este estudio y relacionada con la Biología Sintética. De esta forma, se propone la utilización de redes predictoras de fotogramas para que, a partir de vídeos de colonias de bacterias en crecimiento generen los siguientes fotogramas. Es decir, trata de predecir como se comportará la colonia en los siguientes instantes con el fin de ofrecer un apoyo a los simuladores y facilitar la simulación de los comportamientos de las colonias bacterianas en la naturaleza. En concreto, se han utilizado secuencias de fotogramas obtenidas del simulador GRO como datos de entrenamiento. Esto ha supuesto tanto un estudio de las interacciones dentro de las colonias de bacterias entre las distintas cepas y los factores que influyen en su crecimiento, como una elección de qué redes predictoras de fotogramas serán más adecuadas para la tarea. Una vez realizada dicha elección, se ha procedido a la implementación y estudio de los resultados obtenidos desde un punto de vista individual y desde otro comparativo.



# **Abstract**

The application of Machine Learning for frame prediction in videos is one of the most innovative and unknown fields in AI despite the fact of many neural networks emerging from attempts of solving his problem. In this project the most outstanding techniques within last year's proposals are explored, studying the operation of their architecture and the contributions to previous contributions. The study is developed from two different approaches: the solutions that generate one frame from a sequence of frames and the solutions that generate a sequence from another sequence of frames. First a detailed study of the basic Deep Learning networks and structures has been carried out so that offers a theoretical framework for the terms used in this document is offered.

Next, a practical application to this study and related to Synthetic Biology has been proposed. Thus, the use of frame predictor networks is proposed, so that from videos of growing bacteria colonies they are able to generate the next frames. In other words, it tries to predict how the colony will behave in the next instants in order to offer support to simulators and facilitate the simulation of the behaviors of bacterial colonies in nature. In particular, frame sequences obtained from the GRO simulator have been used as a synthetically generated dataset. This has led to both a study of the interactions within the bacterial colonies between the different strains and the different factors that influence their growth, as well as a choice of which frame predictor networks will be more suitable for the task. Once this choice was made, the implementation and the study of the obtained results from a individual point of view and fom a comparative one was achieved.



# Tabla de contenidos

|  |           |
|--|-----------|
| <b>1. Introducción</b>   | <b>1</b>  |
| 1.1. Contexto . . . . .  | 1         |
| 1.2. Motivación y justificación . . . . .                              | 2         |
| 1.3. Objetivos . . . . .   | 2         |
| <b>2. Marco teórico</b>  | <b>5</b>  |
| 2.1. Aspecto biológico . . . . .                                       | 5         |
| 2.2. Aspecto computacional . . . . .                                   | 7         |
| 2.2.1. Red neuronal convolucional (CNN) . . . . .                      | 9         |
| 2.2.2. Red neuronal recurrente (RNN) . . . . .                         | 13        |
| 2.2.2.1. Memorias largas a corto plazo (LSTM) . . . . .                | 15        |
| 2.2.3. Autoencoder . . . . .   | 17        |
| 2.2.4. Redes generativas antagónicas (GAN) . . . . .                   | 19        |
| 2.2.4.1. Red generativa antagónica profunda (DCGAN) . . . . .          | 21        |
| 2.2.4.2. Redes generativas antagónicas condicionales (CGAN) . .        | 21        |
| <b>3. Estado del arte</b>  | <b>23</b> |
| 3.1. Predicción de fotogramas . . . . .                                | 23        |
| 3.1.1. Predicción de secuencia a un fotograma . . . . .                | 23        |
| 3.1.2. Predicción de secuencia a secuencia de fotogramas . . . . .     | 34        |
| <b>4. Definición del problema</b>                                      | <b>44</b> |
| <b>5. Redes de predicción del crecimiento en colonias de bacterias</b> | <b>46</b> |
| 5.1. Elección del <i>dataset</i> . . . . .                             | 46        |
| 5.1.1. Generación del <i>dataset</i> . . . . .                         | 46        |
| 5.1.2. Tratamiento del <i>dataset</i> . . . . .                        | 47        |
| 5.2. Elección de las redes neuronales . . . . .                        | 48        |
| 5.2.1. GAN multiescalar con pérdida GDL . . . . .                      | 49        |
| 5.2.2. MCNet con conexiones residuales . . . . .                       | 50        |
| 5.2.3. Autoencoder con VRNN . . . . .                                  | 52        |
| <b>6. Resultados y conclusiones</b>                                    | <b>54</b> |
| 6.1. Resultados de la implementación . . . . .                         | 54        |
| 6.1.1. GAN multiescalar con pérdida GDL . . . . .                      | 54        |
| 6.1.2. MCNet con conexiones residuales . . . . .                       | 56        |
| 6.1.3. Autoencoder con VRNN . . . . .                                  | 57        |
| 6.2. Conclusiones . . . . .  | 57        |
| 6.2.1. Trabajo futuro . . . . .  | 59        |

---

**TABLA DE CONTENIDOS**

|              |           |
|--------------|-----------|
| <b>Anexo</b> | <b>64</b> |
|--------------|-----------|

# **Capítulo 1**

## **Introducción**

### **1.1. Contexto**

La Inteligencia Artificial (IA) es uno de los campos en auge actualmente, lo que implica su utilización en un mayor número de campos con el fin de facilitar o automatizar distintas tareas. Uno de estos campos es el de la Biología Sintética, que consiste en la aplicación de la ingeniería a la Biología de forma que se crean sistemas con elementos biológicos, pero que realizan funciones que no existen en la naturaleza.

La idea de generar un sistema sintético a partir de una *Escherichia coli*, uno de los tipos de bacteria más utilizados en Biología Sintética, surgió ya en 1961. En esa época aún no se contaba ni con los conocimientos ni con las técnicas o medios suficientes para investigar en dicha línea. No fue hasta los años 70 y 80 con el descubrimiento del PCR (Reacción en cadena de la polimerasa) y la clonación molecular cuando se comenzaron a realizar modificaciones génicas limitadas a esta técnica. A mediados de los años 90 con la aparición de novedosas herramientas computacionales y de la secuenciación automática se empezó a trabajar con la secuenciación microbiana, es decir, el orden de los nucleótidos(Adenina, Citosina, Guanina y Timina) en las cadenas de ADN. De esta forma se consiguió la secuencia completa de la bacteria *E. coli* y del genoma humano. Además, surgieron estudios sobre las distinciones entre los componentes biológicos y las interacciones entre ellos, lo que permitió buscar similitudes con los sistemas electrónicos.

Es a partir del inicio del milenio cuando surgió la disciplina propiamente dicha y se comenzaron a imitar circuitos electrónicos, como por ejemplo: el *Toggle switch* o interruptor palanca, que permite cambiar entre dos estados distintos a una célula; el represilador, un circuito oscilador; o las puertas lógicas, base de los dispositivos electrónicos. Así mismo, se realizaron avances en la comunicación entre células, generando el primer circuito con *Quorum sensing*. Tras el año 2006 y hasta la actualidad, se han comenzado a realizar aplicaciones más complejas con posibles aplicaciones médicas como la invasión de células cancerosas o la industria combustible, produciendo biocombustible.

Como se puede apreciar es un campo muy novedoso, con numerosos avances en los últimos años, en el que aún queda un amplio terreno por explorar y que puede aportar grandes beneficios en diversas disciplinas, especialmente combinado con la IA y las facilidades que esta aporta.

## 1.2. Motivación y justificación

Las fases de diseño y experimentación en la Biología Sintética son de gran importancia, dado que los gastos derivados de implementar los distintos circuitos diseñados son elevados. Por esto apenas hay lugar para el error, dado que un mal diseño puede llevar a un coste experimental innecesario. Lo que se busca es reforzar la fase de diseño ya sea con simuladores que representen los experimentos en la vida real, o como es el caso de este proyecto, aplicando la Inteligencia Artificial para realizar predicciones de los posibles resultados basándose en experimentos anteriormente realizados.

De esta forma la motivación de este proyecto es diseñar un sistema que sustituya o complemente las simulaciones, evitando gastos innecesarios de tiempo (ya sea haciendo experimentos similares o relacionados con los contemplados en los sistemas) y de presupuesto (por el gasto proveniente de implementaciones fallidas en la vida real debido a un mal diseño o a la falta de información que podría ser aportada por estos sistemas). Además, otro interés es la aceleración del simulador GRO, el cual está en desarrollo en el grupo LIA de la UPM. Para ello, se busca sustituir el motor gro que actualmente va añadiendo las bacterias a la simulación según la colonia va creciendo, suponiendo un alto coste computacional.

En un ámbito más personal, existe una motivación por ampliar los conocimientos en el dominio del *Deep Learning*, explorando algunas de las redes neuronales más utilizadas y técnicas novedosas. Especialmente a la hora de trabajar con vídeos y tratar de predecir el próximo *frame*, es decir, una de las imágenes que componen un video. Estos conocimientos complementarán los adquiridos durante mi formación de cara a mejorar como profesional.

## 1.3. Objetivos

Tal como se ha mencionado en el apartado anterior, la meta principal de este proyecto es dar soluciones de soporte al diseño de sistemas multicelulares en Biología Sintética, valiéndose de datos de experimentos previos y de un *dataset* generado. Para ello se deberá cumplir los siguientes objetivos:

- Crear unos datos de entrenamiento o *dataset* en formato vídeo sobre el crecimiento de una o varias cepas de *E. coli* durante un tiempo determinado utilizando el simulador GRO.
- Realizar un estudio de las implementaciones existentes para predicción de fotogramas, evaluando sus diferencias y resultados.
- Con el *dataset* sintético generado entrenar varias redes neuronales para que a partir de un vídeo de longitud variable sea capaz de predecir el próximo *frame* o imagen. Dichas redes serán escogidas entre las soluciones estudiadas siguiendo un criterio determinado. Además, será necesario ajustar los parámetros de dicha red mediante un algoritmo a escoger.
- Realizar un análisis de los resultados, evaluando la precisión de las redes y las posibles mejoras.

Con el fin de cumplir dichos objetivos será necesario, y se consideran subobjetivos:

## **Introducción**

---

el estudio de las distintas redes neuronales y su aplicación, además de las nuevas implementaciones propuestas de estas que puedan ayudar al cumplimiento de los objetivos, y la capacidad de generar un *dataset* consistente que abarque el mayor número de posibilidades factible.



## **Capítulo 2**

### **Marco teórico**

#### **2.1. Aspecto biológico**

Para entender completamente el objetivo de este proyecto es necesario comprender el funcionamiento de las poblaciones bacterianas. Una bacteria es un organismo unicelular que pertenece al reino procariota que se caracterizan por la ausencia de núcleo. En concreto, en este proyecto se utilizan las *Escherichia coli*, bacteria propia del tracto gastrointestinal. Cada bacteria individual realiza distintas acciones: crecer, reproducirse, alimentarse, comunicarse o morir. La reproducción consiste en la fisión binaria, que es la realización de una copia por parte de la propia bacteria. Esta reproducción es a su vez dependiente de la alimentación, puesto que en ausencia de nutriente no pueden crecer. Además en presencia de distintas cepas que interaccionen de forma negativa se genera una competición por el nutriente y otros metabolitos. En el caso de interacción positiva se genera una situación de cooperación. Por otro lado, la comunicación consiste en la liberación de señales químicas al entorno o en el intercambio de mensajes genéticos conocidos como plásmidos. Los plásmidos son fragmentos circulares de ADN que ofrecen algún tipo de ventaja al huésped. También ocurre el fenómeno conocido como empuje por el cual las bacterias empujan a aquellas nuevas hacia las zonas exteriores de la colonia donde hay mayor cantidad de nutriente[M E17].



Figura 2.1: Colonia de *E-coli* real

Los ecosistemas bacterianos han sido estudiados durante los últimos 50 años con el fin de conocer los procesos que se llevan a cabo y su impacto en la estabilidad del ecosistema. Entendemos por estabilidad la tendencia del ecosistema a perdurar en el tiempo [Mae19]. Las distintas cepas de bacteria que componen un ecosistema bacteriano interactúan entre ellas basándose en distintas variables.

- Factor de conectividad: número de interacciones. No se consideran autointeracciones de las cepas.
- Fuerza de interacción: a mayor es este número mayor es la cantidad de metabolitos absorbidos, y a su vez mayor es el crecimiento. Este valor puede ser negativo, de tal forma que en vez de aumentar el crecimiento lo afecta negativamente. En la siguiente fórmula podemos observar este efecto:

$$\mu = \mu_0 + F \times [m]$$

$\mu$  es la tasa de crecimiento y  $\mu_0$  la tasa de crecimiento base, que se ve afectada por la fuerza de interacción  $F$  y la concentración absorbida del metabolito  $[m]$ .

Estas interacciones se pueden representar de dos formas distintas: un grafo dirigido sin relaciones reflexivas o una matriz sin valores en la diagonal principal.

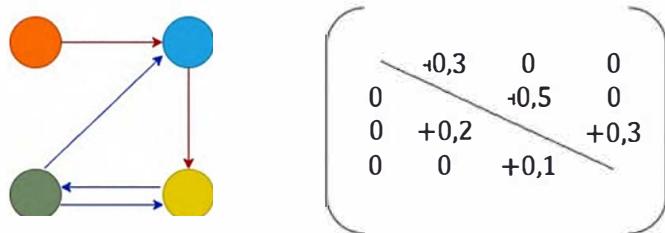


Figura 2.2: Grafo y matriz de las interacciones en una colonia de distintas cepas

En el grafo se pueden observar el número de interacciones y la naturaleza de estas. Mientras que en la matriz correspondiente a dicho grafo también se indica la fuerza de la interacción entre cada cepa, siendo 0 la no existencia de esta. Al no existir interacciones consigo mismas, la diagonal es nula. De esta forma se pueden representar las distintas interacciones y realizar un estudio de como evoluciona dicha colonia, además de su estabilidad. La estabilidad es la capacidad de la colonia de mantenerse en el tiempo a pesar de la ocurrencia de perturbaciones. Entendemos por perturbación una modificación al entorno que causa un efecto drástico. En un caso de ejemplo, si por una perturbación en una colonia mueren el 7% de las bacterias existentes, dicha colonia sería estable si la colonia se mantiene en el tiempo. Para ello la densidad de las distintas cepas deberá mantenerse constante.

Numerosos simuladores han sido creados para reproducir el crecimiento de estas colonias de forma artificial. Entre ellos destacan algunos como Bactosim[A15] o gro. Este último es el simulador utilizado principalmente durante este proyecto debido a su intuitivo motor gráfico, a su gran velocidad en comparación con otras herramientas similares y su constante actualización por parte del grupo de investigación LIA-UPM.

## **2.2. Aspecto computacional**

Las redes neuronales pertenecen al Aprendizaje profundo o **Deep Learning(DL)**, un área dentro del Aprendizaje automático. Esta última, también conocida como **Machine Learning(ML)**, es la rama de la IA que combina computación y estadística aplicada con el fin de aprender de los datos, encontrando patrones entre ellos para generalizarlos y obtener soluciones a distintas problemáticas[GBC16]. Se puede decir que trata de simular el aprendizaje del cerebro humano. Las restantes ramas de la IA se centran en otros aspectos de la inteligencia humana como el procesamiento del lenguaje natural o la lógica difusa, con el objetivo final de crear máquinas capaces de aprender, pensar y actuar como los seres humanos.

El ML sigue un proceso en tres fases: en primer lugar recuerda las medidas que se tomaron la última vez que se encontró una problemática, se adaptan nuevas medidas si las anteriores no funcionaron y por último, se comparan las situaciones para encontrar patrones. Para ello se ejecuta un proceso denominado entrenamiento, en el cual a partir de unos datos, conjunto de datos que son divisibles en casos individuales, encuentra las relaciones ocultas entre los datos para generalizarlas y aplicarlas a nuevas situaciones. Los datos, además de ser utilizados durante la fase de entrenamiento, pueden ser utilizados durante una fase posterior conocida como fase de prueba o *testing*. De forma similar al *testing* realizado en desarrollo software, esta fase es útil para comprobar que el modelo ha sido entrenado correctamente y genera los datos esperados. En ocasiones se toma una porción de los datos de entrenamiento para realizar la validación. El proceso de entrenamiento puede clasificarse basándose en la metodología utilizada para llevarlo a cabo[Gér17]:

- Aprendizaje supervisado: el aprendizaje se basa en la utilización de los valores a obtener a partir de cada uno de los casos de entrenamiento. Es decir, se utiliza la solución a la aplicación del modelo sobre los datos. A estos datos se les denominan etiquetas.
- Aprendizaje no supervisado: al contrario del caso anterior no se utilizan etiquetas. En su lugar se buscan patrones inherentes en los datos, sin hacer uso de la solución.
- Aprendizaje semi-supervisado: combina ambos tipos anteriores, trabajando con casos en el *dataset* que cuentan con etiquetas, como casos que no.
- Aprendizaje multi-instancia: es similar al aprendizaje supervisado, pero una etiqueta se asocia a un número  $n$  de ejemplos y no a cada caso en particular.
- Aprendizaje reforzado: establece un ciclo entre el sistema y el entorno, durante el que recibe *feedback* de este último para modificar sus parámetros y mejorar su rendimiento.
- Aprendizaje evolutivo: se replican patrones conocidos de la naturaleza en el ámbito computacional como la reproducción, la mutación o la selección natural.

Los modelos de ML entrenados se pueden entrenar con distintos fines u objetivos, según la problemática que se quiera abordar. Algunos de los casos[GBC16] son:

- Clasificación: cuando el *dataset* consiste en casos que pertenecen a distintas clases (por ejemplo, un *dataset* múltiples fotos de gatos y perros donde las clases son gato o perro). Dichas clases pueden ser tanto disjuntas como intersec-

cionar. En este caso el modelo es entrenado para predecir con precisión la clase a la que pertenecen los datos de entrada. También se puede utilizar para predecir la probabilidad de que un caso pertenezca a cada una de las clases. Una circunstancia especial es en la que no todos los casos de entrada son conocidos. En esta situación se necesitará inferir múltiples funciones.

- Regresión: en este caso se predice un valor, a partir de la aproximación mediante una función durante el entrenamiento. Este es un problema matemático recurrente de aproximación.
- Transcripción: se basa en la transformación de datos no estructurados en un formato de texto concreto. Un ejemplo recurrente es el reconocimiento de voz a texto.
- Traducción: transforma los datos o símbolos de un lenguaje a los de otro idioma distinto.
- Detección de anomalías: a partir de unos datos que representan diversos eventos, este tipo de modelo es capaz de identificar aquellos que son distintos al resto.
- Muestreo y síntesis: se generan datos a partir de unos datos de entrada similares, procurando que sean lo más realistas posible. Este tipo de tarea necesita una gran cantidad de datos o ejemplos.
- Determinación de valores ausentes: el objetivo es obtener los valores que faltan en una entrada incompleta.
- Eliminación ruido: la entrada en esta tarea son unos datos con ruido, es decir, datos que se mezclan con los datos originales y no aportan información. Estos datos se transforman para obtener la información limpia con la que se pueda operar.
- Estimación de probabilidad: el aprendizaje en esta tarea consiste en una función de distribución. Para ello utiliza puntos conocidos con anterioridad como datos (*dataset*).

Las propiedades que definen cada uno de los casos que componen el *dataset* son las características (*features*). Con el fin de encontrar la mejor generalización se modifican unos valores denominados hiperparámetros en distintos ciclos de entrenamiento hasta encontrar aquellos que ofrecen mejores resultados.

Para medir el rendimiento del modelo de ML implementado se pueden utilizar distintas métricas durante el proceso de *testing*. La más común es la precisión, que mide el porcentaje de acierto en las predicciones sobre un número variable de casos. Esta métrica no es útil para todas las problemáticas que se pueden presentar. Por ejemplo para *datasets* no etiquetados sería inservible, puesto que la solución es desconocida. Otra medida común es la curva ROC, que representa la relación entre los ratios de verdaderos positivos y falsos positivos. Cuando estas métricas ofrecen resultados muy altos para el *dataset* de entrenamiento, pero con los datos de prueba son bajos se afirma que el modelo está sobreajustado o que ha caído en *overfitting*. Esto significa que durante el entrenamiento la función determinada por el modelo se ha ajustado demasiado a los datos de entrada, pero no tiene la habilidad de generalizar para otros casos. Asimismo existe el caso contrario(*underfitting*), en el que el modelo

## Marco teórico

es incapaz de aprender de los datos de entrada, ya sean para entrenamiento o para pruebas.

La idea de implementar una estructura que imite el funcionamiento neuronal humano surge en 1943 de la mano de Warren McCulloch y Walter Pitts[MP43]. En sus principios la idea no resultó especialmente atractiva, lo que combinado con una computación aun en sus inicios hizo que hasta los 90 no hubiera grandes avances. La primera implementación conocida es el perceptrón simple en 1958[Ros58]. El funcionamiento[Sha18] de un perceptrón simple consiste en la aplicación de una

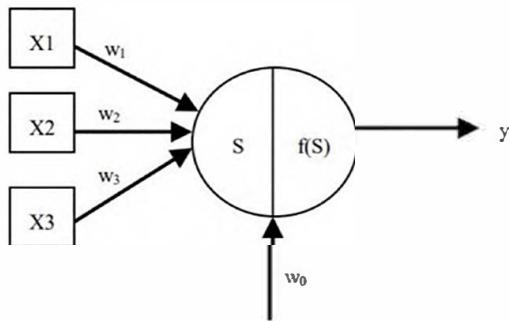


Figura 2.3: Estructura de un perceptrón simple

función de activación sobre el sumatorio de las entradas  $x_i$  multiplicada por su peso correspondiente, siendo estos  $w_i$ . A lo largo de los años las estructuras se han ido haciendo más complejas mediante la adición de capas, característica principal del DL. Un ejemplo, es el Perceptrón multicapa(MLP) o *Feed Forward*(FF). En la figura

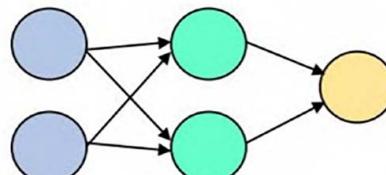


Figura 2.4: Estructura de una red *Feed Forward*

2.2 se puede observar la estructura de un *Feed Forward*, donde la capa intermedia se denomina capa oculta. La característica principal de estas redes es que los datos únicamente se mueven hacia adelante. En las siguientes secciones se explica el origen y funcionamiento de las estructuras más relevantes para el desarrollo de este proyecto.

### 2.2.1. Red neuronal convolucional (CNN)

Las redes neuronales convolucionales surgen a partir de la idea del Neocognitrón[Fuk80], el cual fue la primera estructura utilizada para el reconocimiento de patrones. El neocognitrón utiliza distintos tipos de células para el tratamiento de *features*. En 1989 se propone la primera CNN[LeC+89]. La red propuesta por Lecun et al. presentó la idea de las capas convolucionales, las capas de *pooling*, y las capas completamente conectadas. El nombre oficial que recibió esta red es LeNet y está compuesta por 8 capas, incluida la de entrada.

Estas redes son normalmente utilizadas con aprendizaje supervisado, pero también puede utilizarse con no supervisado. Son las redes más utilizadas para clasificación y reconocimiento de imágenes, lo que implica trabajar con las matrices correspondientes a cada imagen. En el caso de las imágenes de blanco negro se obtiene una matriz por imagen, donde cada número representa un píxel. Sin embargo, las imágenes de color (tanto RGB como BGR) comprenden tres matrices distintas superpuestas. En estas matrices cada número corresponde a la cantidad del color de la matriz a la que pertenece. Por ejemplo, si el formato de los canales utilizados es RGB, los píxeles de la matriz correspondiente a  $R$  son la cantidad de rojo en dichos píxeles.

Las capas que comenzaron a utilizarse con LeNet se siguen utilizando en las CNN actuales como se puede observar en la figura 2.4. El funcionamiento[GBC16] de dichas capas se explica a continuación:

- Convolución: se realiza una operación denominada convolución sobre las ma-

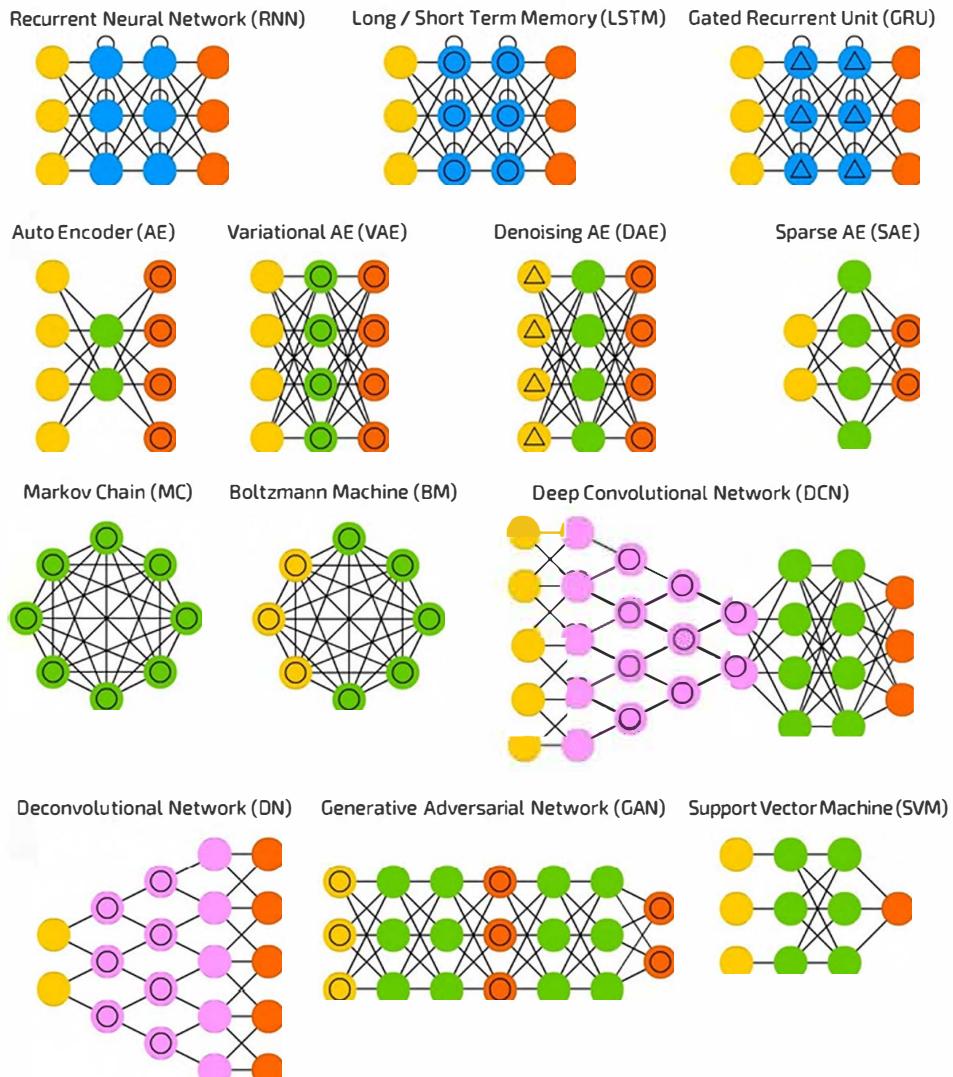


Figura 2.5: Estructura de distintas redes neuronales existentes

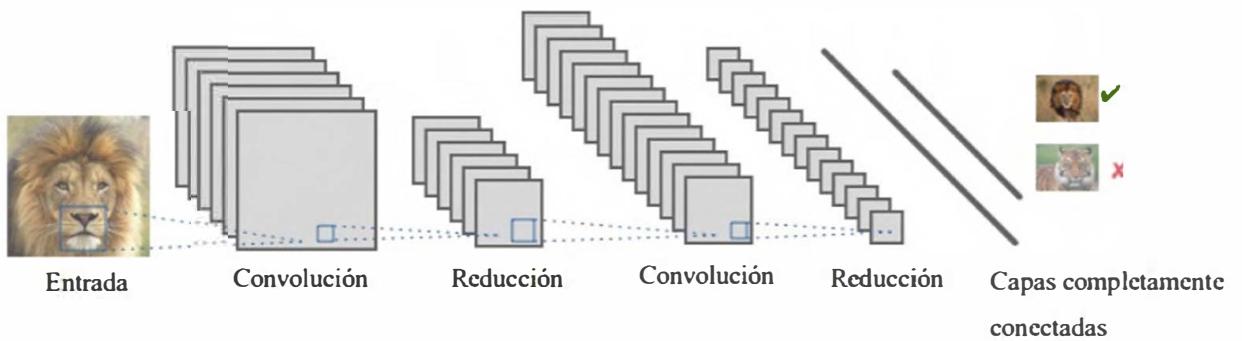


Figura 2.6: Estructura de una CNN

trices de entrada utilizando unas matrices auxiliares. Estas últimas son los denominados filtros o *kernel* cuyo tamaño es especificado previa ejecución y se considera un hiperparámetro. Las operaciones se realizan tomando submatrices del mismo tamaño de los filtros sobre la matriz de entrada. Cada submatriz estará separada de la anterior un número de filas y columnas indicado por el hiperparámetro *strides*. Habitualmente, los filtros son matrices cuadradas. El resultado de la operación es una matriz de menor tamaño que la original, el mapa de características.

Por otro lado, el número de mapas de características se puede modificar aumentando su hiperparámetro. Esto aumenta a su vez la precisión. Una cantidad  $n$  de mapas de características implica la realización de un número  $n$  de operaciones cada una con un filtro distinto sobre la misma submatriz.

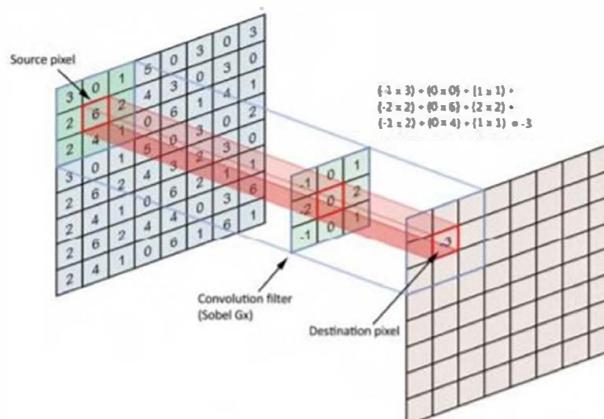


Figura 2.7: Capa convolucional

Hay casos en los que las filas y columnas de la matriz de entrada no son divisibles por el tamaño de los filtros, como ocurre en la figura 2.5. Para poder operar con dichas matrices es necesario utilizar relleno o *padding*. Generalmente se utiliza en modo *same*, que añade columnas y filas de ceros a ambos lados la matriz de tal forma que se pueda operar y mantenga la divisibilidad o no por 2 de las dimensiones originales. El fin de esta capa es extraer las características

de cada imagen.

- No linealidad: a los mapas de características resultantes de la capa anterior se les aplica una función de activación que transforma los datos a no lineales. La función de activación habitual para este tipo de redes es la unidad lineal rectificada (ReLU)[NH10], cuya función da valores en el rango  $[0, \infty)$ . Como se ve en la figura 2.6 se eliminan los negativos, obteniendo los mapas de características rectificados.

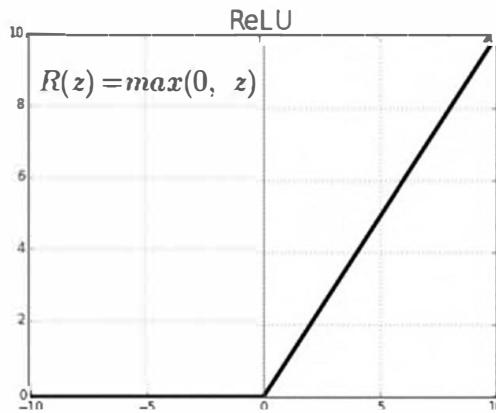


Figura 2.8: Gráfica de una ReLU

- Reducción o pooling: en esta capa se utilizan operaciones comunes en la estadística como el promedio o el máximo con el fin de reducir el tamaño de los mapas de características rectificados y sus parámetros. Sin embargo, la información se mantiene. Como ocurría en la capa de convolución cuenta con distintos hiperparámetros: el tamaño de *kernel*, *stride* y *padding*.

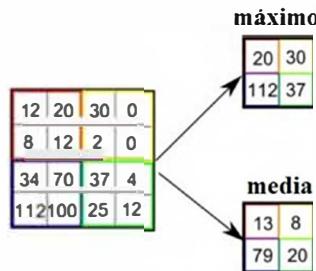


Figura 2.9: Pooling mediante máximo y promedio

- Capas completamente conectadas: son las capas donde la red realiza el aprendizaje de la función a aplicar sobre los datos. Además, es la capa encargada de la clasificación. Se caracteriza porque cada neurona está conectada a todas las neuronas de la capa siguiente. Es necesario reducir la dimensionalidad de las imágenes recibidas de las capas anteriores, transformándolas en vectores. El número de neuronas en estas capas equivale a un hiperparámetro.

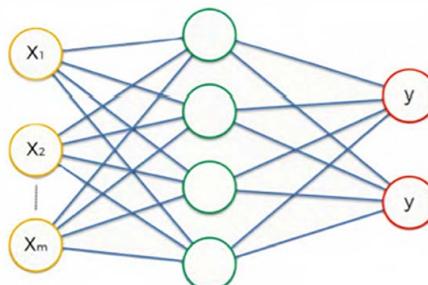


Figura 2.10: Capas completamente conectadas

Una vez una imagen pasa por todo el modelo, que puede tener numerosas capas de las anteriores dependiendo de la complejidad del problema, se compara con su etiqueta. Sobre la base del error o diferencia entre el valor real y el predicho, se obtiene el **coste**. El coste puede ser calculado con diferentes algoritmos como por ejemplo la entropía cruzada binaria o el error cuadrático medio (MSE). El objetivo del entrenamiento es minimizar el coste, es decir, llevarlo a 0 mediante sucesivas iteraciones de entrenamiento (**epochs**). En consecuencia se utilizan optimizadores como el Adam o RMSProp. También es importante el hiperparámetro conocido como **tasa de aprendizaje**, recurrente en otros algoritmos de ML. Su función consiste en dar una medida a la velocidad con la que el modelo se adapta a los ratos. Es habitual utilizar números muy bajos, puesto que una tasa muy alta introduce ruido en los datos de salida.

## 2.2.2. Red neuronal recurrente (RNN)

En 1982 surgen las redes Hopfield[Hop82], un prototipo de las RNN donde cada unidad o neurona cuenta con dos estados: 0 y 1 o -1 y 1, dependiendo del formato utilizado. Estas redes sirven para el aprendizaje de un patrón concreto. Es una red unicapa donde las relaciones entre las unidades no son cíclicas y deben ser simétricas. Las unidades se actualizan siguiendo la siguiente fórmula o sustituyendo 0 por

-1:

$$v_i \leftarrow \begin{cases} 1 & \text{si } \sum_j w_{ij} s_j > \theta_i \\ 0 & \text{en caso contrario.} \end{cases}$$

Donde  $V_i$  es el vector de entrada cuyos valores sirven como estado inicial,  $w_{ij}$  son los pesos de cada relación posible entre las unidades,  $s_j$  es el estado actual del nodo  $j$  y  $\theta$  es el límite. Además se utiliza la función de energía:

$$E = -\frac{1}{2} \sum_{i,j} w_{ij} v_i v_j + \sum_i \theta_i v_i$$

El entrenamiento finalizará cuando se realicen todas las iteraciones indicadas o la función de energía deje de decrecer y se mantenga estable.

En 1986, David Rumelhart et al. proponen la primera RNN[RHW86]. Estas redes se utilizan para aprender datos en formato secuencial, como por ejemplo predicciones de bolsa o procesamiento del lenguaje natural (NLP).

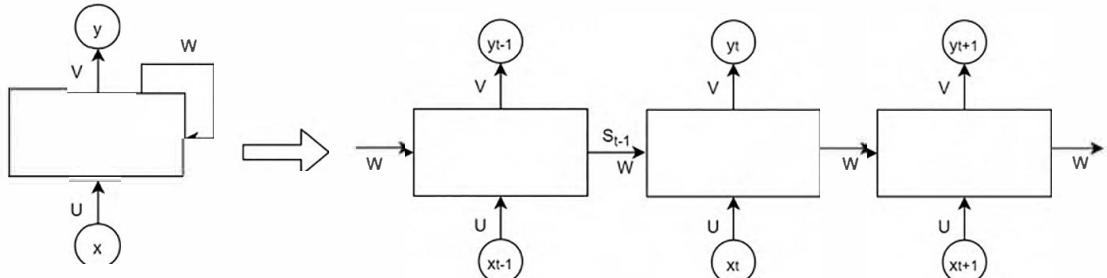


Figura 2.11: Estructura general de una RNN

Cada secuencia estará compuesta por un número de instancias denominadas **timestamps**. En la figura 2.9 está representada la existencia del bucle que permite utilizar la información de casos pasados para realizar la predicción actual. A pesar de que se visualice como una secuencia lineal, siempre se está ejecutando el mismo modelo en bucle.

La fórmula que se aplica para obtener el estado interno  $h$  en el momento  $t$  y que funciona como entrada en el timestamp siguiente, es la indicada a continuación[Ati20]:

$$h_t = \tanh(b + Wh_{t-1} + Ux_t)$$

Se entiende como  $b$  el *bias*, tanto  $W$  como  $U$  son los pesos de la red y  $x$  es la entrada. A continuación, se aplica la siguiente fórmula:

$$y_t = \text{softmax}(c + Vh_t)$$

Como ocurría anteriormente  $c$  corresponde a un *bias*, que es un valor que se mantiene constante y se utiliza para ajustar con mayor precisión el modelo, y  $V$  un peso, siendo todos los pesos hiperparámetros.

El problema de estas redes es que a largo plazo sufre el problema de desvanecimiento de gradiente. Como solución surgen las redes LSTM.

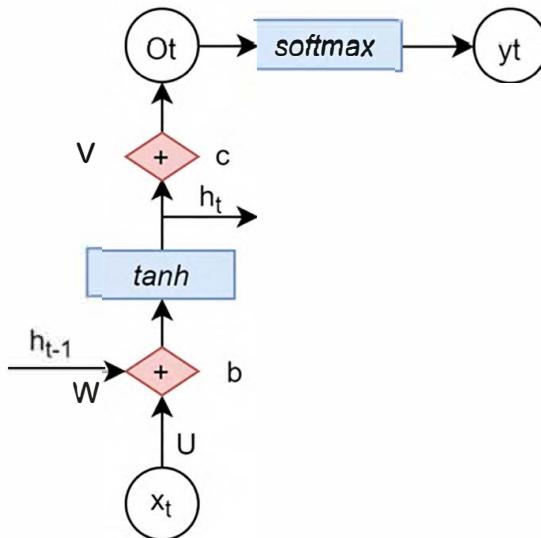


Figura 2.12: Estructura interna de una RNN

### 2.2.2.1. Memorias largas a corto plazo (LSTM)

Las memorias largas a corto plazo(LSTM) fueron propuestas por Sepp Hochreiter y Jürgen Schmidhuber en 1997[HS97]. Esta clase de redes son un subtipo de RNN que resuelven el problema de la memoria a largo plazo y sus dependencias para la predicción actual. Utilizan unidades especiales y celdas de memoria que permiten el mantenimiento de la información a largo plazo.

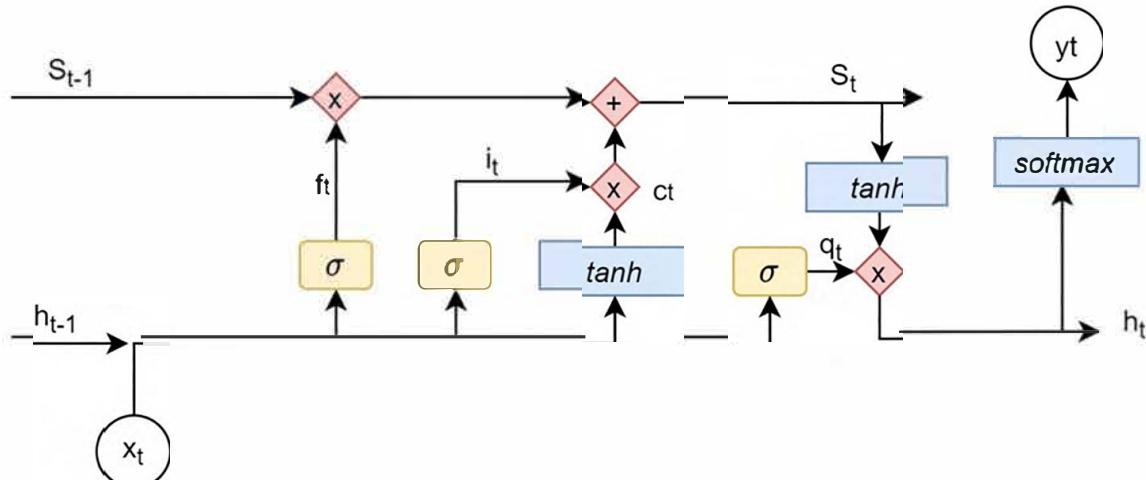


Figura 2.13: Estructura de una LSTM sin pesos

A continuación se explica el funcionamiento general de esta red[Ati20]. La línea  $s_t$  indica el estado de la celda de memoria, actuando de forma similar a una cinta transportadora. La información de esta celda a penas sufre cambios, al no añadir ni borrar información. Únicamente se deja pasar la información a través de las puertas. Estas puertas funcionan con números binarios, donde 0 significa que no dejan fluir la información y 1 que la deja pasar toda. Las puertas y su funcionalidad son las siguientes:

- Puerta de olvido: Se encarga del almacenamiento de la información a largo plazo, olvidando las partes irrelevantes. Su fórmula es la siguiente:

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f)$$

Como ocurría en las RNN  $b$  es el *bias* y los pesos son  $W$  y  $U$ . Por otro lado,  $\sigma$  corresponde a la función de activación sigmoidea.

- Puerta de entrada: se obtiene multiplicando  $i_t$  y  $c_t$ .

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i)$$

$$c_t = \tanh(W_c x_t + U_c h_{t-1} + b_c)$$

$i_t$  decide que valores se deben actualizar, ignorando el resto. En cambio  $c_t$  crea los nuevos valores a recordar. El resultado son los valores seleccionados que contienen la nueva información para actualizar la celda. Esta información se suma con el resultado de multiplicar el estado anterior y la salida de la puerta de olvido, dando lugar al nuevo estado de la celda.

- Puerta de salida: calcula los datos a almacenar a corto plazo, también conocidos como estado oculto.

$$q_t = \sigma(W_o x_t + U_o h_{t-1} + b_o)$$

El resultado de  $q_t$  se multiplica por la solución de aplicar la función  $\tanh$  sobre  $S_t$ , dando lugar a  $h_t$ .

La función de activación sigmoidea transforma los valores de entrada en valores en el rango  $[0, 1]$ [Sha18]. Por este motivo se utiliza para las puertas. Como se mencionó con anterioridad con un valor 0 no pasa la información. En el caso contrario las multiplicaciones se hacen por 1, manteniendo la información intacta.

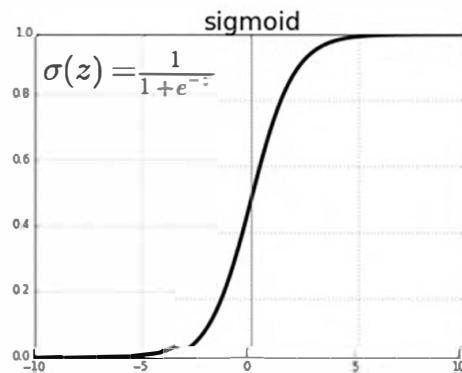


Figura 2.14: Gráfica de la función de activación sigmoidea

La función  $\tanh$  funciona de forma similar pero con el rango  $[-1, 1]$ [Sha18]. El fin de utilizar esta función es mantener los valores en un rango para que no tomen valores demasiado altos con las operaciones.

A pesar de que este tipo de red supone una mejora en términos de memoria en comparación con las RNN, es muy exigente computacionalmente. Además son complicadas de entrenar por su tendencia al sobreajuste.

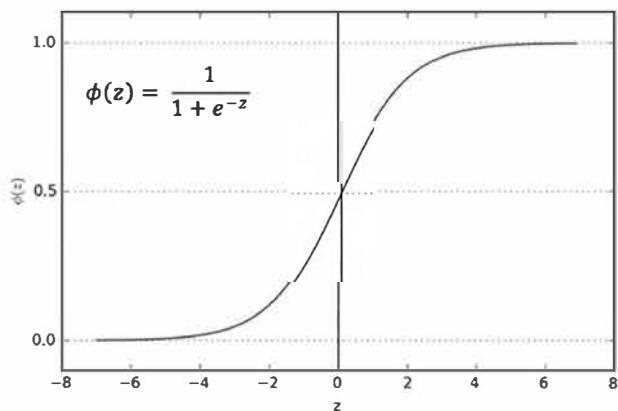


Figura 2.15: Gráfica de la función de activación  $\tanh$

### 2.2.3. Autoencoder

Los *autoencoder*(AE) aceptan distintos tipos de datos de entrada: texto, imágenes, vídeo u otros. Su objetivo general es la realización de transformaciones a estos datos[ZKM17]. De tal forma es útil para tareas de eliminación de ruido o de detección de anomalías. Una vez entrenada la red solo funcionará con el tipo de datos utilizado en el aprendizaje. Utiliza aprendizaje no supervisado, por lo tanto no utiliza etiquetas. Se sabe que la idea original de estas redes surgió de forma temprana, aunque no tiene constancia de un origen claro. Uno de los primeros artículos conocidos que describió un AE fue el publicado por Dana H. Ballard en 1987[Bal87].

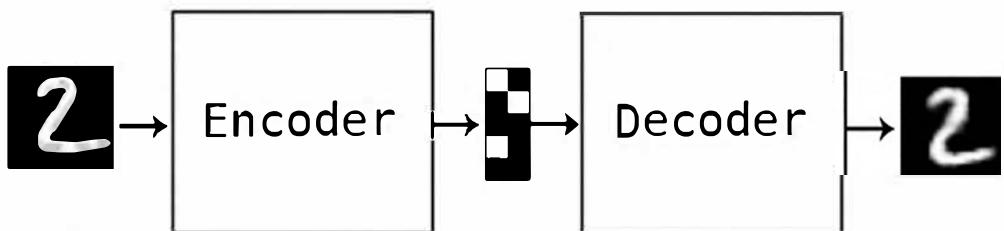


Figura 2.16: Estructura de un AE

Un AE codifica la entrada en un vector que comprime la información relevante y posteriormente la devuelve a su tamaño original, con la perdida de calidad que implica. Esta reducción de las dimensiones de los datos se lleva a cabo para eliminar el ruido y posteriormente reconstruir la entrada. Al vector reducido se denomina vector latente. En los casos en los que el vector latente es demasiado grande, más que los datos de entrada, el AE se limita a copiar los datos recibidos. Las dos partes que componen un *autoencoder* son[Ati20]:

- Codificador: transforma la entrada  $x$  en un vector  $z$ . El número de nodos por capa es decreciente de entrada al vector latente. De esta forma se reducen las dimensiones de los datos de entrada.

$$z = f(x)$$

- Decodificador: trata de obtener la información original, o el  $\tilde{x}$  más similar. El número de nodos por capa según se avanza en las capas es creciente de forma simétrica al codificador, con el fin de devolver la información a su dimensión original.

$$g(z) = \tilde{x}$$

Ambas partes pueden ser implementadas con distintos tipos de red (por ejemplo, las anteriormente explicadas CNN o MLP). Es destacable el uso de la función de pérdida (que en este caso calcula la diferencia entre la entrada y la salida, de forma que se minimice) e hiperparámetros (tamaño del vector, número de capas y nodos por capa) adecuados para recrear con precisión los datos de entrada sin caer en *overfitting*. Las funciones de pérdida utilizadas regularmente son la entropía cruzada binaria, en los casos en los que se opera con datos formados por valores binarios, y el error cuadrático medio (MSE) para otras circunstancias. A continuación se indica la fórmula del MSE para un AE, donde  $m$  corresponde al área de los datos de entrada. Por ejemplo, para una imagen  $m$  será el ancho multiplicado por el alto y por los canales.

$$MSE = \frac{1}{m} \sum_{i=1}^{i=m} (x_i - \tilde{x}_i)^2$$

Es común utilizar propagación hacia atrás[LeC+89] para la minimización del coste. Este algoritmo ajusta los pesos y *bias* de la red de tal forma que se minimice el coste[ZKM17]. Consiste en, tras recorrer los datos el modelo y calcular el coste, pasar la información del coste de forma inversa a la red. Es decir, se pasa de salida a entrada.

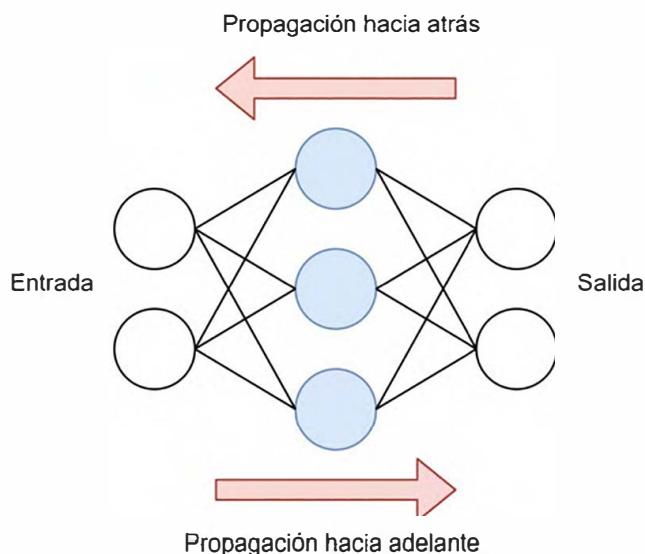


Figura 2.17: Propagación hacia atrás

Para optimizar dicho coste se utiliza el gradiente descendente[Lem12]. Un gradiente es un vector cuyo sentido indica la dirección en la que la función crece. Las proyecciones del gradiente sobre los ejes corresponden a las derivadas parciales de la

función. Por lo tanto, el gradiente descendiente consiste en calcular el gradiente del coste respecto a cada parámetro, de tal manera que se modifiquen los parámetros para que el gradiente sea negativo (el coste decrezca)[ZKM17].

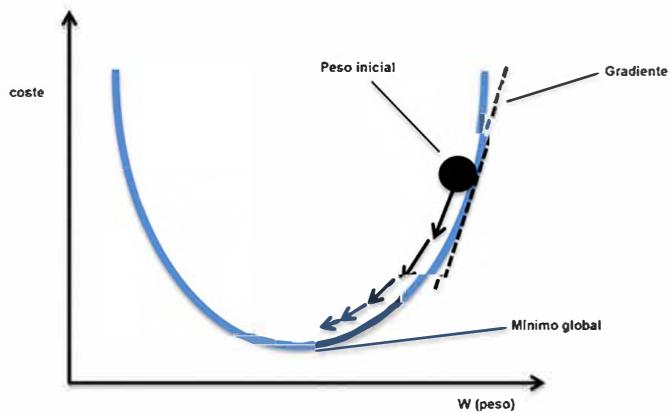


Figura 2.18: Algoritmo del gradiente descendiente

#### 2.2.4. Redes generativas antagónicas (GAN)

Desde su reciente creación en 2014[Goo+14], este tipo de redes ha tenido una gran exposición por su utilidad en distintos ámbitos. Las GAN aprenden a modelar la distribución de entrada mediante dos redes que compiten entre ellas. Se componen de dos subredes:

- Generador: es una red dentro de la GAN cuyo objetivo es generar datos, normalmente imágenes, falsos que sean realistas.
- Discriminador: es una red encargada de distinguir si los datos generados son reales o falsos.

El objetivo general es crear datos que sean tan realistas que el discriminador no sea capaz de distinguirlas de las reales. Cuando se llega a este punto la red ha finalizado su entrenamiento y se puede utilizar el generador por si solo. En consecuencia, la relación de competencia es solo para el entrenamiento. El problema principal que emerge de esta relación entre las redes es encontrar el equilibrio: si el discriminador convergiera demasiado rápido el entrenamiento podría finalizar antes de que el generador obtenga suficientes actualizaciones a sus parámetros como para generar datos realistas. Por otro lado, si el discriminador no está bien entrenado clasificará datos de mala calidad como reales lo que influirá directamente sobre el realismo de los datos generados.

El generador recibe ruido como entrada. Es decir, un vector producido por una distribución (normalmente gaussiana) que influye en las características a representar. Cada vector de ruido posible genera unos datos distintos. El discriminador recibe tanto datos reales como los sintéticos. Los reales reciben una etiqueta de 1.0 (100 % de probabilidad de ser reales) y los sintéticos de 0.0. A pesar de que se utilizan etiquetas, son autogeneradas. En consecuencia, se considera aprendizaje no supervisado.

Durante el entrenamiento del discriminador se actualizan sus parámetros como clasificador binario. Por otro lado, el optimizador calcula los parámetros del generador

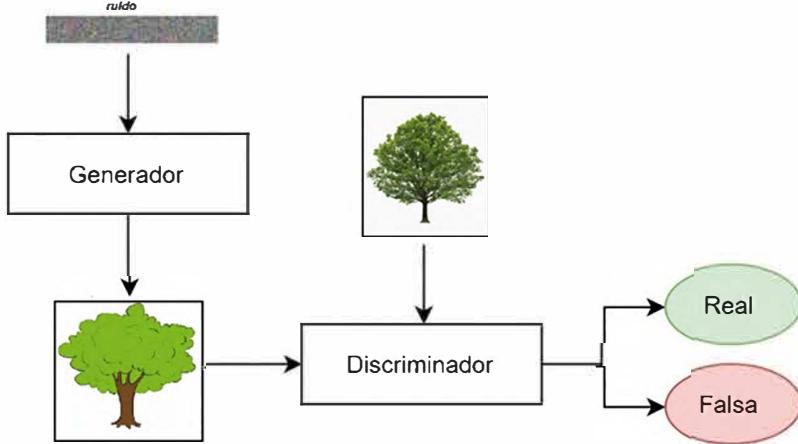


Figura 2.19: Estructura de una GAN

basándose en la predicción presentada por el discriminador. En resumen, se actualizan los pesos de ambas redes mediante propagación hacia atrás. Para calcular dicho coste se utiliza la fórmula de la entropía cruzada:

$$\min_G \max_D V(D, G) = \mathbb{E}_{x \sim p_{\text{data}}(x)}[\log D(x)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))]$$

En esta fórmula  $D(x)$  representa el resultado del discriminador y  $G(x)$  el resultado del generador. El discriminador procura maximizar la función objetivo, mientras que el generador intenta minimizarla. Por consiguiente se puede aplicar el gradiente ascendente en el primer caso y el gradiente descendente en el segundo.

$$\max_D V(D) = \mathbb{E}_{x \sim p_{\text{data}}(x)}[\log D(x)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))]$$

El primer sumando de la fórmula del discriminador trata de que los datos reales sean reconocidas con más precisión, mientras que el segundo procura que reconozca los datos generados mejor.

$$\min_G V(G) = \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))]$$

Por otro lado se optimiza el generador de tal forma que el discriminador pueda ser engañado. La fórmula de optimización del generador es modificada ligeramente para evitar problemas de convergencia a continuación:

$$\min_G V(G) = \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))]$$

Los gradientes se pasan del discriminador al generador alternativamente. En primer lugar se actualizan los pesos del discriminador cada paso de una iteración mediante el gradiente estocástico ascendente. Una vez terminados los pasos y antes de finalizar la iteración, se actualizan los pesos del generador con el gradiente estocástico descendente. Ambas redes pueden utilizar distintas arquitecturas. Por ejemplo, CNN para imágenes o RNN/LSTM para secuencias de una dimensión. El mayor problema de este tipo de redes es la dificultad de entrenar correctamente ambas redes. Además se necesita un *dataset* de gran tamaño para generalizar correctamente. En los años recientes, han surgido diferentes tipos de GAN que proponen mejoras para solucionar estos problemas, mejorar el rendimiento y aplicarlos a distintos problemas.

### 2.2.4.1. Red generativa antagónica profunda (DCGAN)

Estas redes surgidas en 2016[RMC15] son redes GAN compuesta por capas convolucionales. Sin embargo, al contrario de la explicación realizada anteriormente para las redes CNN, estas no se suceden con una capa de *pooling* ni con capas completamente conectadas. En su lugar se implementan las siguientes cuestiones:

- Utiliza *strides* en las convoluciones para el discriminador y *strides* fraccionadas para el generador.
- Se utiliza normalización de lote.
- Las capas completamente conectadas se eliminan.
- Utiliza funciones de activación ReLU en todas las capas del generador, excepto en la última. En la capa de salida se utiliza la función *tanh*.
- Las capas del discriminador utilizan la función de activación LeakyReLU.

La normalización de lotes se basa en la aplicación de normalización por cada lote en el que se divide el *dataset*, como su nombre indica. Dicha normalización sirve para estabilizar los datos de forma que todos estén en el mismo rango de valores. Esto mejora el resultado del entrenamiento y previene problemas como el desvanecimiento de gradiente o la explosión de gradientes. Hay distintas fórmulas, pero la más común utiliza la media y la varianza.

$$x^* = \frac{x - E(x)}{\sqrt{\text{var}(x)}}$$

Por otro lado, la función LeakyReLU[MHN13] es similar a la ReLU, pero sí que toma valores negativos aunque son muy cercanos a 0[Sha18].

### 2.2.4.2. Redes generativas antagónicas condicionales (CGAN)

En las GAN originales, no hay control sobre los datos que se generan. El resultado es dependiente del ruido, pero este es complicado de descifrar. En las CGAN, al contrario de las GAN estándar, si hay control sobre los datos generados. Su idea surge de la mano de M. Mirza y S. Osindero en 2014[MO14]. En el artículo se propone la utilización de un vector en formato *one-hot*, donde cada posición únicamente puede tomar valores 0 o 1. Se asocian cada uno de estos valores a una característica de la

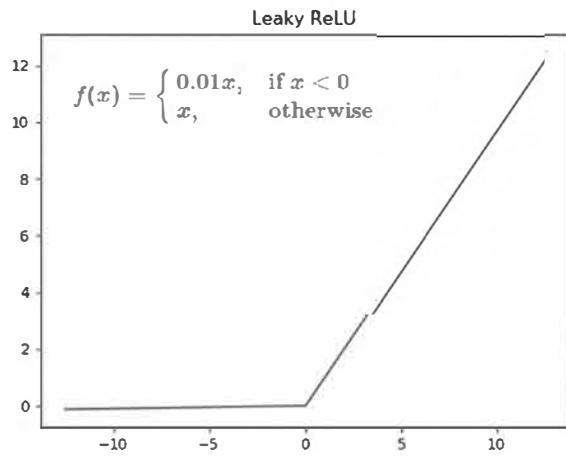


Figura 2.20: Gráfica de la función de activación LeakyReLU

imagen generada. Por ejemplo, si se están generando imágenes de camisetas algunas de las características pueden ser el color o el largo de manga.

Para llevarlo a cabo, la entrada se concatena con el vector latente. El resto de la red sigue la misma estructura que una DCGAN, excepto las capas que sean necesarias para concatenar o redimensionar la entrada de datos.

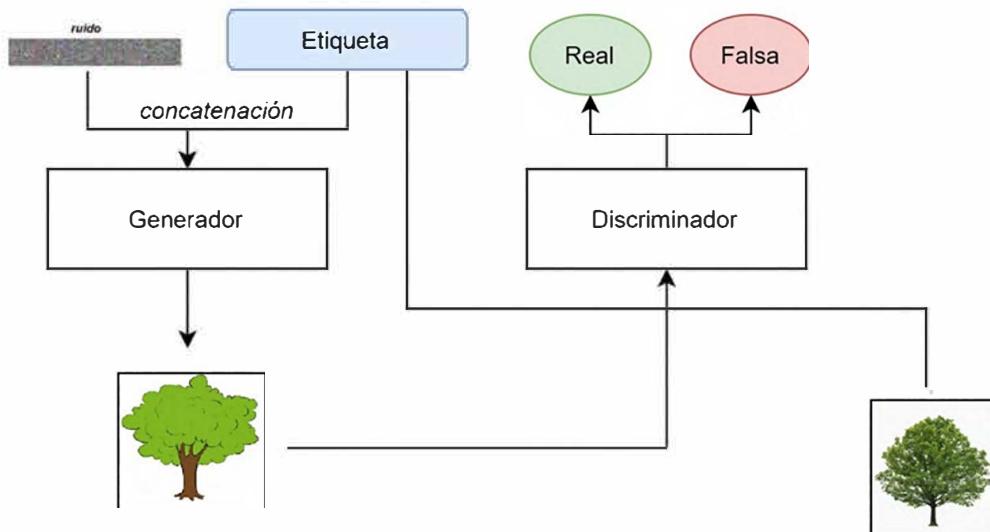


Figura 2.21: Estructura de una CGAN

# Capítulo 3

## Estado del arte

### 3.1. Predicción de fotogramas

Como se ha expuesto anteriormente, las redes neuronales son utilizadas con distintos fines. Entre ellos, se pueden destacar las tareas realizadas sobre imágenes como la clasificación, generación o modificación. Estas problemáticas se han conseguido llevar a cabo con relativo éxito y se pueden utilizar consistentemente. Sin embargo, se ha tratado de expandir su uso a nuevas dificultades como la predicción sobre vídeos. Esta predicción consiste en a partir de uno o más fotogramas de un vídeo obtener los  $n$  siguientes, siendo  $n > 0$ .

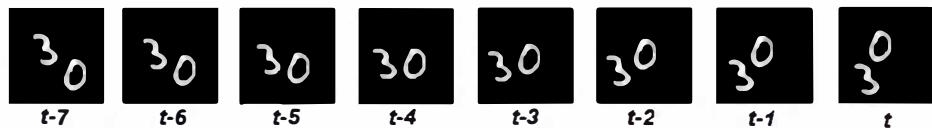


Figura 3.1: Fotogramas pertenecientes al *dataset* MNIST de números en movimiento

El problema principal de esta tarea es capturar correctamente la relación temporal entre las distintas imágenes. Además, también es complicado captar la iluminación y profundidad en las sucesivas imágenes 3D. Al ser un campo relativamente reciente, aún no se ha dado con una solución extendida y universal como si ha ocurrido con otras tareas anteriormente mencionadas. No obstante, se han propuesto numerosas soluciones que ofrecen cada vez mejores resultados. Tomando como ejemplo la figura 3.1, el objetivo es que a partir de un número de fotogramas de los 8 presentados obtener el siguiente en  $t+1$  o los sucesivos  $t+n$ . Las imágenes de esta figura de ejemplo se corresponden al *dataset* de números en movimiento del MNIST, cuyo acceso es gratuito de forma online.

A continuación se enumeran las distintas soluciones propuestas durante estos años, junto a su estructura y el número de fotogramas utilizados.

#### 3.1.1. Predicción de secuencia a un fotograma

En esta primera colección todos los modelos propuestos predicen el fotograma siguiente a la secuencia de entrada. Por lo tanto, entra una secuencia (o un único

### 3.1. Predicción de fotogramas

| Nombre                    | Modelo                           | Fotogramas de entrada | Fotogramas salida |
|---------------------------|----------------------------------|-----------------------|-------------------|
| Goroshin et al.[GML15]    | Conv. AE y <i>phase pooling</i>  | 2/3                   | 1                 |
| Klein et al.[KWA15]       | Capa convolucional dinámica      | 4                     | 1                 |
| Shi et al.[Shi+15]        | ConvLSTM                         | 10                    | 10                |
| Zhao et al.[Zha+15]       | GBM/GAE y modulación contextual  | 2                     | 1                 |
| Brabandere et al.[Bra+16] | Red de Filtros dinámicos         | 3/10                  | 3/10              |
| Cricri et al.[Cri+16]     | Red escalera de video            | 10                    | 10                |
| Lotter et al.[LKC16]      | Conv. AE + LSTM y Discriminador  | 5-15/5                | 1                 |
| Mathieu et al.[MCL16]     | GAN multiescalar con pérdida GDL | 4-8                   | 1-8               |
| Xue et al.[Xue+16]        | Red convolucional cruzada        | 1                     | 1                 |
| Zhou et al.[ZB16]         | Conv. AE / Conv. AE + LSTM       | 1/2/1                 | 1/1/4             |
| Liang et al.[Lia+17]      | <i>Dual motion GAN</i>           | 2-10                  | 1                 |
| Liu et al.[Liu+17]        | Conv. AE y flujo voxel           | 2                     | 1                 |
| Villegas et al.[Vil+17b]  | LSTM + Conv. AE                  | 10                    | 32/64             |
| Villegas et al.[Vil+17a]  | Conv. AE + ConvLSTM y CNN        | 4/10                  | 1                 |
| Vondrick et al.[VTI17]    | CNN                              | 4                     | 12                |
| Vukotic et al.[Vuk+17]    | Conv. AE                         | 1                     | 1                 |
| Denton et al.[DF18]       | AE + convLSTM                    | 5/10/2                | 10/10/23          |
| Liu et al.[W+18]          | <i>U-Net</i>                     | 4                     | 5                 |
| Reda et al.[Red+18]       | SDC-Net                          | 5                     | 5                 |
| Wichers et al.[Wic+18]    | LSTM + Conv. AE                  | 3/5                   | 16/32             |
| Ho et al.[Ho+19]          | SME-net                          | 3/4                   | 1/5               |
| Hu et al.[HW19]           | VPGAN                            | 10                    | 30                |
| Kwon et al.[KP19]         | CycleGAN                         | 4                     | 1                 |
| Ye et al.[Ye+19]          | AE                               | 1                     | 14                |
| Li et al.[Li+20]          | DMMNet                           | 2                     | 1                 |
| Niklaus et al.[NMW21]     | U-Net+Conv.                      | 2                     | 1                 |

Cuadro 3.1: Soluciones propuestas de secuencia a fotograma

fotograma) y se obtiene  $t+1$ . Sin embargo, mediante concatenación de los fotogramas obtenidos a los anteriormente utilizados como entrada se pueden ir obteniendo más de uno en sucesivas iteraciones. Otros modelos son capaces de predecir un fotograma en cualquier momento del futuro y no necesariamente el fotograma siguiente a los conocidos.

#### ***Learning to linearize under uncertainty***

Goroshin et al.[GML15] presentan un AE convolucional con capas de *phase-pooling* y *unpooling* entre codificador y decodificador. Esta capa de *phase-pooling* realiza un *max pooling* (máximo) para identificar la presencia de una característica y el *argmax pooling* (valor que es el máximo) para obtener la posición de la característica. De esta forma se consigue lidiar con la incertidumbre y facilitar la linearización. Por último utiliza similitud del coseno, que se utiliza una vez realizado el *phase-pooling*, y mide la diferencia entre dos vectores, en este caso el resultado de la codificación y el objetivo. De esta forma se complementa la función de pérdida *L2*.

#### ***A dynamic convolutional layer for short range weather prediction***

En el caso del artículo propuesto por Klein et al. [KWA15] se realiza una modificación a la capa convolucional de forma que los filtros no son constantes. Con este fin se utilizan dos entradas: una para los mapas de características (ya utilizada en las capas convolucionales tradicionales) y otra para los filtros. Para obtener estos filtros se entrenará una subred convolucional que producirá los filtros a partir de las imágenes de entrada.

### **Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting**

Shi et al.[Shi+15] introducen el concepto de LSTM convolucional nombrado anteriormente. Este es una extensión del *peephole* LSTM, cuya principal diferencia con el LSTM clásico es la utilización del estado de la celda en las puertas. La nueva característica son la agregación de convoluciones entre estados y entre entrada y estado. Además, para evitar los problemas espacio-temporales existentes en el *peephole* LSTM, la entrada y la salida de la celda, las puertas y los estados ocultos son tensores 3D.

### **Predictive encoding of contextual relationships for perceptual inference, interpolation and prediction**

La arquitectura propuesta por Zhao et al.[Zha+15] consiste en un *autoencoder* con modulación contextual, es decir, se tiene en cuenta la influencia del contexto para el objetivo. Para ello se sintetiza la imagen predicha por el modelo principal, en este caso un *autoencoder*, y se retro-alimenta para que coincida con la entrada. La diferencia entre el resultado de la retroalimentación y la imagen original se denomina señal residual. Esta señal residual se utiliza para actualizar el modelo. En la figura 3.2 se puede apreciar este funcionamiento, donde  $x_i$  es la entrada,  $y_i$  son las capas ocultas,  $\hat{x}_i$  el resultado o predicción cuya diferencia con  $x_i$  se calcula y denominada  $z$ . Por último  $z$  se propaga para actualizar el modelo. En las pruebas utilizan máquinas de Boltzmann con puertas(GBM) y *autoencoder* con puertas(GAE).

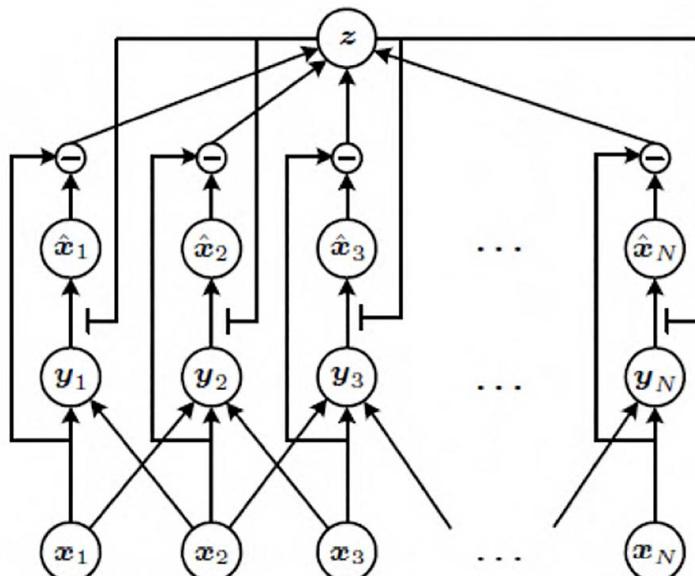


Figura 3.2: Arquitectura propuesta por Zhao et al.[Zha+15]

### **Dynamic filter networks**

La denominada red de filtros dinámicos fue propuesta por Brabandere et al.[Bra+16] en 2016. Esta red se divide en dos partes: un generador de filtros, cuya función es producir filtros para cada entrada de forma dinámica al contrario de lo que ocurre en

las redes convencionales que utilizan los mismos filtros para todas las entradas del modelo, y una capa de filtrado dinámico que aplica los filtros a la imagen de entrada. Para la primera parte puede utilizarse tanto MLP o CNN, como *autoencoder* si es necesario mejorar la generación, mientras que para la segunda parte es frecuente la utilización de CNN dado que su entrada son imágenes. La capa de filtrado dinámico puede a su vez tomar dos formas distintas. La primera es la capa convolucional dinámica, donde se genera un único filtro para cada imagen de entrada y este es aplicado a cada posición de dicha imagen  $I$ . La otra forma existente es la capa de filtrado local dinámica la cual aplica un filtro distinto por cada posición de  $I$ .

#### **Videoladder networks**

Cricri et al.[Cri+16] plantean una red a la que nombran red escalera de vídeo. Esta consiste en una estructura *autoencoder* donde el codificador utiliza capas convolucionales dilatadas y el decodificador capas convolucionales clásicas. Además utiliza normalización de lote, la función de activación *LeakyReLU* y bloques residuales recurrentes. Las capas convolucionales dilatadas [YK16] son aquellas que añaden un parámetro denominado tasa de dilatación que indica los espacios que se dejan entre las celdas tomadas de la entrada al operar con el filtro. Por otro lado, los bloques residuales recurrentes consisten en la utilización tanto de conexiones laterales recurrentes, como de conexiones de alimentación hacia adelante con el fin de mejorar la invariancia a pequeños cambios. El primer tipo de conexiones se encarga del residuo, mientras que la segunda se encarga del salto.

#### **Unsupervised learning of visual structure using predictive generative networks**

Lotter et al.[LKC16] proponen un modelo no supervisado compuesto por 2 CNN (cada una compuesta por una capa convolucional, una capa rectificadora y una capa de *max-pooling*), una LSTM y una capa deconvolucional(deCNN). Utiliza el error medio cuadrado para esta parte generadora y la pérdida adversaria para la discriminadora. Esta red discriminadora, inspirada por las redes GAN, consiste en una CNN y una LSTM que se aplican sobre la secuencia de entrada, y una CNN y una capa completamente conectada que toman como entrada el fotograma generado por la red principal. Por último, la concatenación de los resultados de ambas partes se da como entrada a un MLP.

#### **Deep multi-scale video prediction beyond mean square error**

Mathieu et al.[MCL16] proponen en 2016 tres soluciones distintas para solucionar el problema de la borrosidad en las imágenes predichas por el uso de la función de pérdida MSE. Estas soluciones son: la utilización de multiescala, la implementación de un discriminador y la utilización de una nueva función de perdida denominada diferencia de la pérdida de gradiente(GDL). La multiescala soluciona el problema de las dependencias cortas en el CNN, añadiendo la posibilidad de utilizar imágenes de distintos tamaños. Esto se utiliza en la parte generativa de la estructura GAN propuesta basándose en la siguiente fórmula, donde  $s_k$  corresponde a la escala de la imagen,  $u_k$  al factor de escalado,  $X_k^i$  y  $Y_k^i$  son las imágenes con escala  $s_k$  y  $G'_k$  es la red que predice  $Y_k - u_k (Y_{k-1})$ .

$$\hat{Y}_k = G_k(X) = u_k(\hat{Y}_{k-1}) + G'_k(X_k, u_k(\hat{Y}_{k-1}))$$

De esta forma se obtiene una fórmula recursiva que utiliza las distintas escalas. Se puede visualizar con mayor claridad en la figura 3.4. Por otro lado, se implementa una red discriminadora que dada una secuencia de imágenes, realice una predicción de la probabilidad de que la última imagen de la secuencia haya sido generada por la red generadora anteriormente expuesta. Esto se lleva a cabo para mantener la dependencia temporal y evitar la borrosidad generada por las funciones de perdida  $l_1$  y  $l_2$ . Por último, implementan una nueva función de pérdida GDL combinable con otras existentes. En las pruebas realizadas en el artículo se utilizan tanto GDL- $l_1$ , GDL- $l_2$ , adversaria y adversaria+GDL.

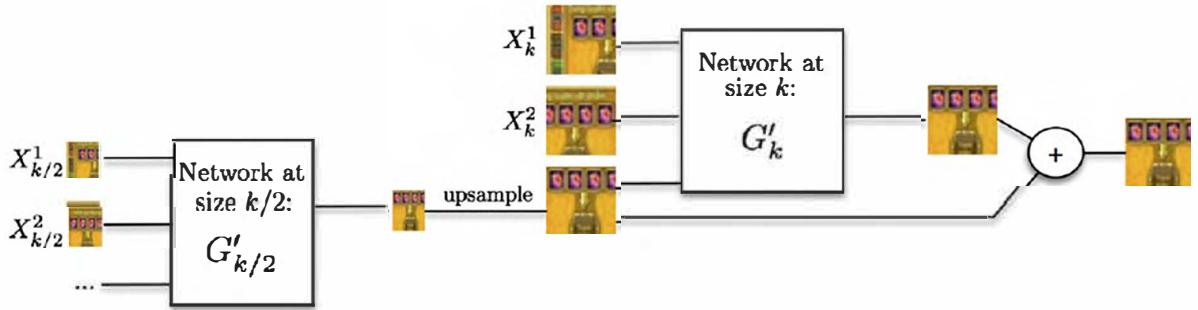


Figura 3.3: Multiescala en Mathieu et al.[MCL16]

### **Visual Dynamics: Probabilistic Future Frame Synthesis via Cross Convolutional Networks**

En el caso del artículo propuesto por Xue et al.[Xue+16] el objetivo es modelizar la distribución condicional de la imagen en  $t + 1$  dada la imagen en  $t$ . Para ello utiliza una combinación de predicción de movimiento determinista y movimiento previo. La arquitectura utilizada consiste en un codificador de movimiento (en este caso utiliza un *autoencoder* variacional o VAE), un decodificador de *kernel*, un codificador de imagen, una capa de convolución cruzada y un decodificador de movimiento.

### **Learning Temporal Transformations From Time-Lapse Videos**

Zhou et al.[ZB16] proponen tres soluciones basadas en distintos enfoques para abordar esta problemática: por parejas, por generación en dos pilas o por generación recurrente. El primer caso se trata de predecir el estado en  $t + 1$  a partir del estado en  $t$  utilizando un *autoencoder* convolucional. Por otro lado, en el segundo caso se utiliza un *autoencoder* convolucional con dos entradas: la imagen en el instante  $t$  y la imagen en el instante  $t + m$ . De esta forma se trata de predecir el *frame* en  $t + 2m$ . Por último, la generación recurrente añade un LSTM al *autoencoder* convolucional que toma como entrada el vector latente y es capaz de generar el siguiente estado. Un VAE[KW14] consiste en la introducción de regularización en un *autoencoder* para evitar el sobreajuste y asegurar que se den las condiciones para realizar la generación.

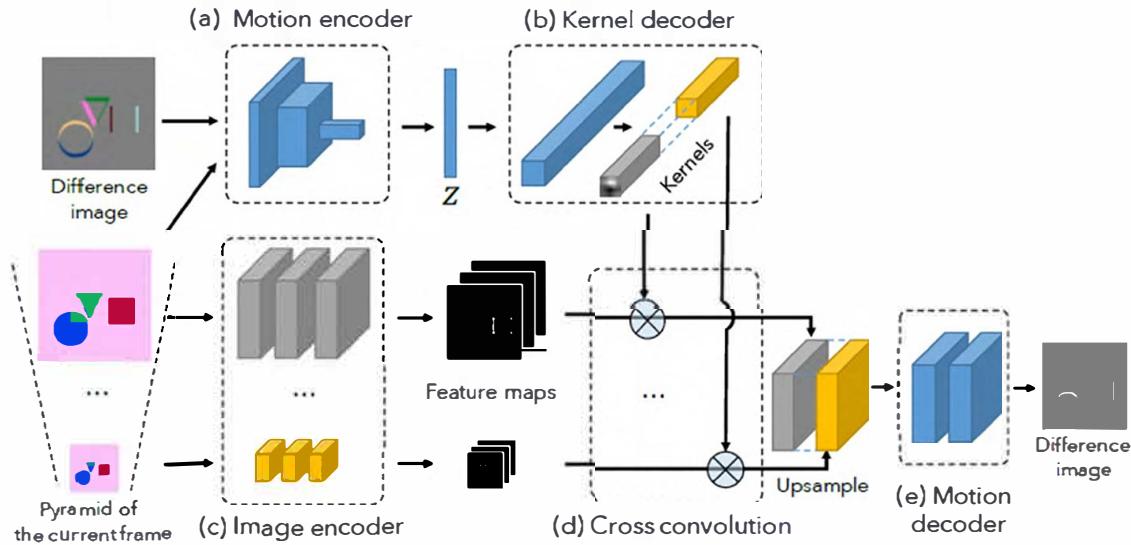


Figura 3.4: Red convolucional cruzada[Xue+16]

El proceso se fundamenta en la codificación de una distribución a partir de la cual se estima un punto que es decodificado. Por lo tanto, la diferencia es que no se codifica un punto directamente. Por otro lado, la capa de convolución cruzada se basa en una capa convolucional que toma como entrada tanto los pesos como los mapas de características. Es decir, no utiliza los mismos pesos para todas las entradas.

#### **Dual Motion GAN for Future-Flow Embedded Video Prediction**

En cuanto a la estructura propuesta por Liang et al.[Lia+17], en general se corresponde a una red GAN donde el generador es un VAE compuesto a su vez por un codificador de movimiento y dos generadores paralelos para fotogramas y flujo que dan salida a  $\tilde{I}_{t+1}$  y  $\tilde{F}_{t+1}$  respectivamente. El codificador de movimiento está formado por 4 capas convolucionales, un ConvLSTM, y dos ConvLSTM paralelos que producen la media y la varianza. El resultado se muestrea y pasa a ambos generadores, donde se decodifica. Por su parte, el discriminador está compuesto por dos discriminadores separados. El primero es  $D_I$  cuyo objetivo es distinguir si  $I_{t+1}$ ,  $\tilde{I}_{t+1}$  (obtenido en la capa de deformación de flujo  $Q_{F \rightarrow I}$  a partir de  $I_t$  e  $\tilde{I}_{t+1}$ ) y  $\tilde{I}_{t+1}$  son imágenes reales.  $D_F$  clasifica si  $\tilde{F}_{t+1}$  (resultado del estimador de flujo  $Q_{I \rightarrow F}$ ),  $\tilde{F}_{t+1}$  y  $F_t + 1$  son imágenes reales o falsas.

#### **Video Frame Synthesis using Deep Voxel Flow**

En 2017 Liu et al.[Liu+17] proponen un modelo denominado flujo voxel profundo. Esta arquitectura consiste en la entrada de 2 fotogramas a un *autoencoder* convolucional, de donde se obtiene una predicción en forma de flujo voxel (flujo óptico de la imagen objetivo a la siguiente). Posteriormente, una capa de muestreo de volumen sintetiza el fotograma objetivo, ya sea mediante interpolación o extrapolación.

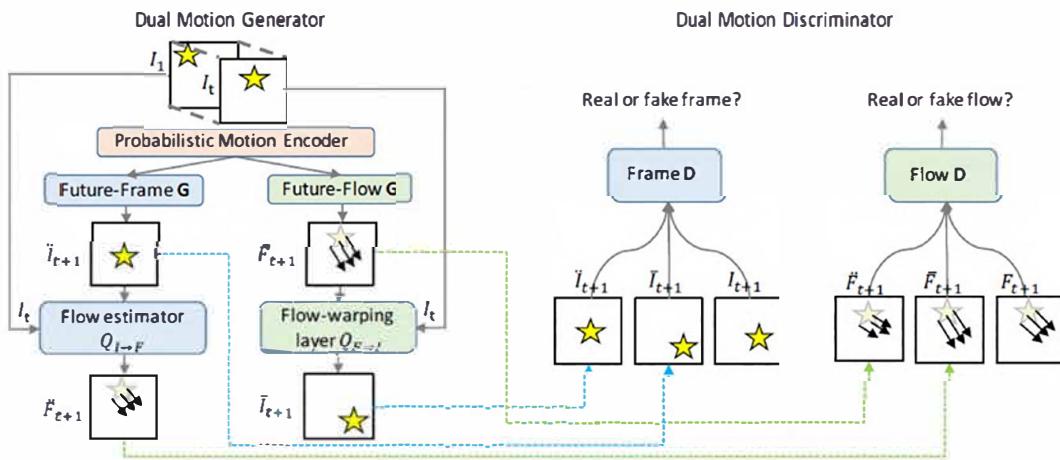


Figura 3.5: Dual motion GAN[Lia+17]

### Learning to Generate Long-term Future via Hierarchical Prediction

Villegas et al.[Vil+17b] proponen una arquitectura LSTM y AE convolucional que realiza la predicción en tres pasos: en primer lugar estima la estructura de la imagen a alto nivel, a continuación predice la estructura de la secuencia siguiente y por último genera los fotogramas correspondientes a dichas estructuras. La estimación de la estructura se realiza mediante una red Hourglass[NYD16]. Tomando como entrada de la red LSTM un número  $n$  de estructuras, se predice la secuencia de estructuras siguiente. Finalmente, se realiza una analogía de la estructura[Ree+15] mediante un *autoencoder* convolucional y se obtiene el fotograma siguiente.

### Decomposing Motion and Content for Natural Video Sequence Prediction

En el caso de Villegas et al.[Vil+17a] el algoritmo de predicción está compuesto por dos codificadores (uno de movimiento y otro de contenido), el contenido residual de movimiento multiescala, las capas de combinación y el decodificador. El codificador de movimiento está integrado por un codificador convolucional y un LSTM convolucional, donde la entrada al codificador es la diferencia entre cada par  $x_t$  y  $x_{t-1}$  de la secuencia de entrada. Por otro lado, el codificador de contenido está formado por un CNN. El objetivo del codificador de movimiento es codificar la relación temporal, mientras que el codificador de contenido codifica el contenido espacial. El resultado de ambos codificadores se concatena y la capa de combinación, compuesta por CNN con capas cuello de botella, proyecta el resultado en un espacio de menor dimensión para a continuación devolverlo a su tamaño original. De esta forma se obtiene el mapa de contenido. Este mapa es la entrada del decodificador deconvolucional, el cual incluye la utilización de conexiones residuales calculadas mediante convoluciones y la rectificación de la concatenación de los resultados de los codificadores en la capa  $l$ . Finalmente se obtiene la predicción del fotograma en  $t + 1$ .

### Generating the Future with Adversarial Transformers

Vondrick et al.[VT17] la red propuesta es una red convolucional donde las primeras 4 capas son capas convolucionales dilatadas, utilizadas para capturar las dependen-

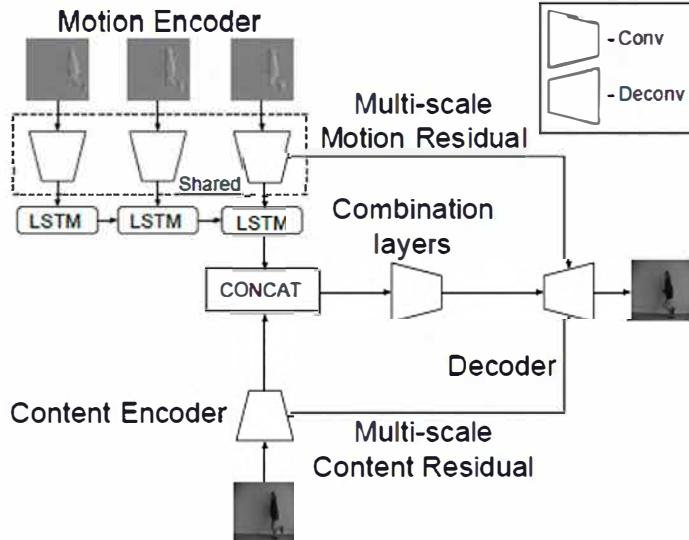


Figura 3.6: Estructura MCnet con relaciones residuales[Vil+17a]

cias temporales a largo plazo, y las 4 siguientes son capas de desconvolucionales. El objetivo de la red es predecir una transformación, que aplicada al fotograma original permitirá obtener el fotograma siguiente. Una vez predicha mediante la red convolucional, se aplicará la transformación sobre el fotograma inicial mediante interpolación de píxeles vecinos. En lugar de aplicarse de forma recurrente para obtener un número  $n$  de fotogramas, las transformaciones siempre se aplican sobre el fotograma inicial. Además, utilizan un discriminador convolucional profundo para distinguir los videos reales de los generados de forma que se entrena la red principal para que la predicción sea verosímil.

#### **One-Step Time-Dependent Future Video Frame Prediction with a Convolutional Encoder-Decoder Neural Network**

Vukotic et al.[Vuk+17] proponen una arquitectura general de AE convolucional cuya principal diferencia es la utilización de dos caminos distintos en el codificador. El primer camino codifica la imagen en  $t$ , mientras que el segundo codifica el desplazamiento temporal  $\Delta t$ . Ambos son concatenados tras sus respectivas capas finales.

#### **Stochastic Video Generation with a Learned Prior**

Denton et al.[DF18] introduce dos enfoques distintos para el entrenamiento de una red AE con un LSTM convolucional. El primero es el SVG-FP, donde la distribución a priori es de forma fija una distribución  $N(0, 1)$ . El problema de este enfoque es que se ignora la dependencia temporal existente entre los fotogramas. El otro enfoque es el SVG-LP, donde la distribución de probabilidad a priori es una distribución normal condicional que varía en el tiempo.

$$\mathcal{N}(\mu_{\text{th}}(\mathbf{x}_{1:t-1}), \sigma_{\text{th}}(\mathbf{x}_{1:t-1}))$$

En el entrenamiento se utiliza un modelo de inferencia que genera la distribución  $q_\phi(\mathbf{z}_t | \mathbf{x}_{1:t})$  de la que se obtiene  $z_t$ , la variable latente. Esta última se fuerza a ser

cercana a  $p(z)$  mediante el término de divergencia KL. Además se penaliza el error entre  $x_t$  y  $\hat{x}_t$ .

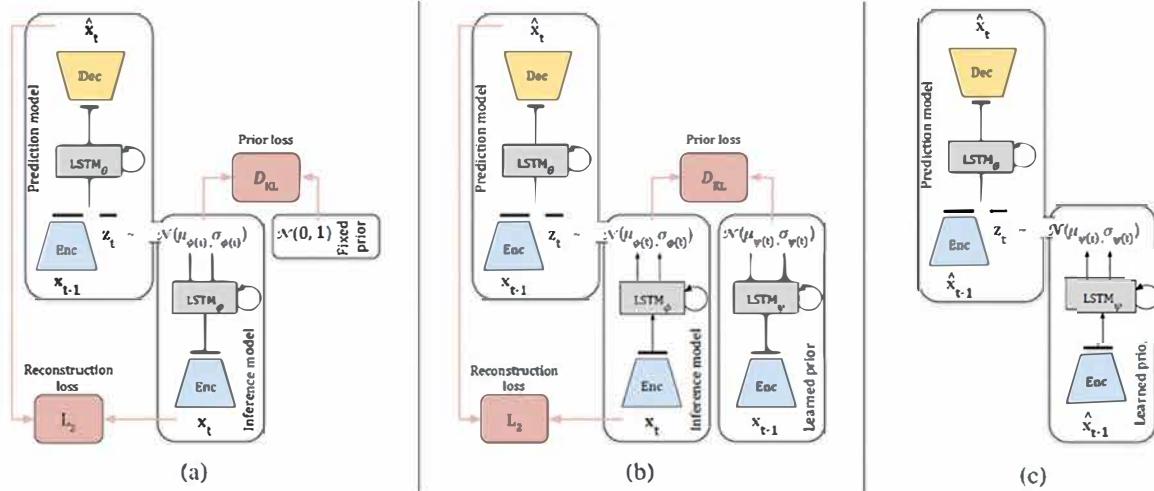


Figura 3.7: Entrenamiento con SVG-FP, entrenamiento con SVG-LP y generación[DF18]

### **Future Frame Prediction for Anomaly Detection – A New Baseline**

En el caso de Liu et al.[W+18] la arquitectura es una red *U-Net*[RFB15] modificada para realizar predicciones como generador y un discriminador de parches. Una U-Net consiste en un *autoencoder* con conexiones entre codificador y decodificador. Además, utiliza distintas limitaciones con el fin de mejorar los resultados. Una de ellas es la diferencia entre intensidad y gradiente, que utiliza la función  $l_2$  y garantiza la similitud de los píxeles y la agudeza de la imagen generada. También utiliza un límite de movimiento, calculando la diferencia del flujo óptico de la imagen predicha y la real. Para estimar estos flujos ópticos utiliza una red Flownet[Dos+15].

### **SDC-Net: Video prediction using spatially-displaced convolution**

Por otro lado, Reda et al.[Red+18] presentan una nueva red convolucional denominada convolución espacialmente desplazada(SDC). Este tipo de convolución consiste en la operación de un kernel adaptativo  $K(x, y)$ , cuyo tamaño es de  $N \times N$ , con un parche de la imagen también de tamaño  $N \times N$ . El centro de este parche se calcula como  $(x + u, y + v)$ , donde  $(u, v)$  es un vector de movimiento. El resultado es un píxel de la imagen en  $t + 1$ .

$$\mathbf{I}_{t+1} = \mathcal{T}(\mathcal{G}(\mathbf{I}_{1:t}, \mathbf{F}_{2:t}), \mathbf{I}_t)$$

Por lo tanto, la imagen siguiente se obtiene mediante la aplicación de  $\mathcal{T}$  con SDC sobre la imagen en el momento  $t$  y la transformación predicha a partir de los fotogramas anteriores y el flujo óptico desde 2 hasta  $t$ . El flujo óptico se calcula con FlowNet2[Ilg+17].  $\mathcal{G}$  es una red convolucional 3D que genera los kerneles y vectores de movimiento.

### Hierarchical Long-term Video Prediction without Supervision

Wichers et al.[Wic+18] utiliza una arquitectura de LSTM con una red de analogía visual (VAN) de forma similar a la anteriormente explicada con Villegas et al. en 2017. El LSTM calcula el vector de características  $\hat{e}_t$  y  $H_t$  a partir del anterior  $\hat{e}_{t-1}$ , obtenido por el codificador, y  $H_{t-1}$ . Por otro lado, el VAN calcula los siguientes fotogramas a partir de los primeros, de forma contraria a lo que ocurría en el artículo de Villegas et al. Se proponen tres métodos de entrenamiento distintos: el entrenamiento de fin a fin, creando analogías (donde se limitan las características predichas por el LSTM de forma que sean más cercanas a las de entrada) o EPVA con pérdida adversaria (el discriminador trata de identificar si la codificación de características ha sido generada por el predictor o el codificador).

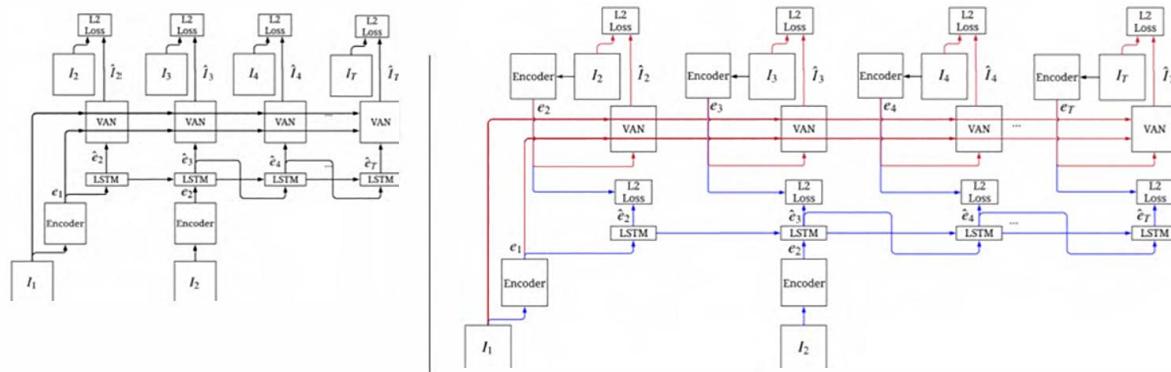


Figura 3.8: Entrenamiento con fin a fin y entrenamiento EPVA con pérdida adversaria[Wic+18]

### SME-Net: Sparse Motion Estimation for Parametric Video Prediction through Reinforcement Learning

Ho et al.[Ho+19] utiliza dos redes distintas: una red posicional y otra de estimación del movimiento. La red posicional está compuesta por una CNN de 6 capas, donde la entrada son los fotogramas de contexto  $I_c$  y el fotograma predicho en el paso anterior  $\hat{I}_n$ . El resultado es la distribución sobre la localización de los píxeles críticos  $s_i$ . A continuación, entra en la red de estimación de movimiento también compuesta por capas convolucionales. Esta red tiene dos entradas, el nuevo píxel crítico y dos parches de  $I_c$  en  $s_i$  de tamaño 64x64 y 128x128 respectivamente. Ambas entradas siguen caminos separados y pasan 3 capas convolucionales distintas para posteriormente concatenarse. Después de 4 capas convolucionales más y un mapa de probabilidad, se obtiene el vector de movimiento  $v(s_i)$  para el píxel  $s_i$ . Por último, se realiza el POBMC[CP12] a partir del fotograma anterior y todos los  $v(s)$ . De esta forma se obtiene la predicción del fotograma siguiente para el píxel  $i$ .

$$\hat{I}_n^{(i)}(s) = \sum_{j \in M(s)} w_j^* I_{n-1}(s + v(s_j)), \forall s \in I_n$$

Tras  $K$  iteraciones, que operan con cada uno de los píxeles críticos, se obtiene la predicción final del fotograma.

### **A Novel Adversarial Inference Framework for Video Prediction with Action Control**

A su vez, Hu et al.[HW19] proponen un *framework* denominado VPGAN que utiliza un modelo adversario de inferencia y un ciclo de consistencia de la función de pérdida. Para ello utiliza dos generadores distintos:  $p_\psi$  y  $q_\theta$ . El primero obtiene  $Z_t$ , el vector latente que contiene el movimiento de los objetos y las variaciones del entorno, estimándolo a partir de los fotogramas  $(X_{t-n:t-1})$  y los vectores latentes anteriores( $Z_{t-n:t-1}$ ).

$$Z_t \sim p_\psi(Z | X_{t-n:t-1}, Z_{t-n:t-1}) \quad p_\psi(Z | X_{t-n:t-1}, Z_{t-n:t-1}) \sim N(\mu_\psi(X, Z), \sigma_\psi(X, Z)\mathbf{I})$$

El segundo generador estima el fotograma siguiente  $X'_t$  a partir del fotograma en el momento  $t$  y  $Z'_t$ .

$$X'_t \sim q_\theta(X | Z'_t, X_{t-1}) \quad q_\theta(X | Z_t, X_{t-1}) \sim N(\mu_\theta(X, Z), \sigma_\theta(X, Z)\mathbf{I})$$

Durante el entrenamiento ambos generadores estiman un par  $(X, Z)$  y  $(X', Z')$ , teniendo el discriminador diferenciar si son verdaderos o falsos. Una vez finalizado el entrenamiento el generador  $p_\psi$  estimará  $z_t$ , que será utilizado en  $q_\theta$  para predecir  $x_t$ .

### **Predicting Future Frames using Retrospective Cycle GAN**

Kwon et al.[KP19] utilizan una red CycleGAN[Zhu+17], cuya diferencia con una GAN clásica es que es una GAN condicional que realiza una transformación de una imagen en otra distinta de forma no supervisada. El *dataset* utilizado no tiene que ser de imágenes emparejadas, sino que se pueden utilizar dos tipos distintos. Esta red utiliza dos generadores y dos discriminadores para realizar la transformación de forma bidireccional. En el artículo propuesto la entrada del único generador es un número de fotogramas ordenado entre  $m$  y  $n$ . Este generador produce el fotograma  $X_{n+1}$ . Además utiliza dos discriminadores distintos:  $D_a$  que distingue si un fotograma concreto es falso y  $D_b$  que distingue si una secuencia de fotogramas contiene un fotograma falso. Este modelo es capaz de generar fotogramas tanto en el pasado ( $X_{m-1}$ ) o en el futuro ( $X_{n+1}$ ).

### **Compositional Video Prediction**

En el caso de Ye et al.[Ye+19] el modelo propuesto se basa en la idea de que de una imagen se pueden extraer la localización de cada objeto o entidad. De tal forma, una entidad  $\{x_n^t\}_{n=1}^N$ , donde  $N$  es el número de entidades, se compone de la localización  $b_n^t$  y las características o apariencias  $a_n^t$ . Durante el entrenamiento un codificador infiere la distribución de las variables latentes  $z_1, \dots, z_t$ , que capturan las posibles ambigüedades del vídeo, a partir del primer y último fotograma. Posteriormente, un predictor de entidades utiliza las características  $b_n^t$  y  $a_n^t$ , que pueden ser extraídas con un ResNet-18[He+16], y la variable latente  $z_t$ . La salida del predictor son las características del fotograma siguiente  $b_n^{t+1}$  y  $a_n^{t+1}$ . Por último, el decodificador de fotogramas toma estas características y el fotograma inicial(necesario para incorporar el fondo estático), obteniendo el fotograma siguiente. El decodificador debe respetar la localización de las entidades indicadas, que algunas entidades quizá tengan que ser fusionadas y que distintas partes del fondo sean visibles según se muevan las entidades.

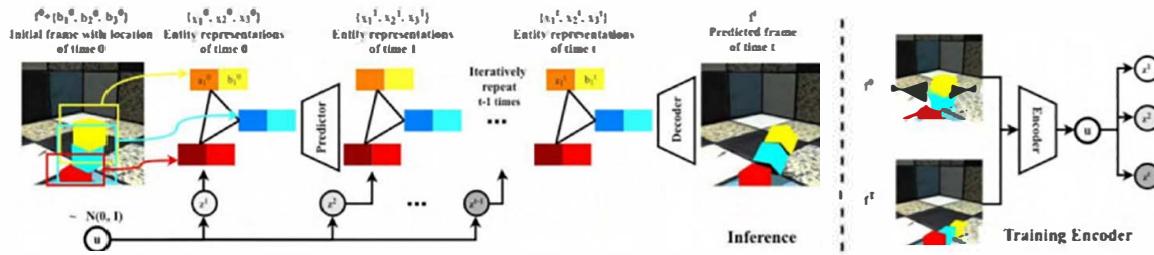


Figura 3.9: Estructura propuesta por Ye et al.[Ye+19]

#### **Video Frame Prediction by Deep Multi-branch Mask Network**

**Li et al.**[Li+20] proponen una red de máscara profunda multi-rama(DMMNet). Esta red se divide en cuatro partes: el codificador de movimiento compartido, el sintetizador de flujo futuro, el sintetizador de apariciones futuras y el sintetizador de fusión. En primer lugar el codificador de movimiento compartido, que está compuesto por cuatro capas convolucionales y sus respectivos *poolings*, transforma la secuencia de entrada en una representación latente del movimiento  $z_{1:t}^m$ . Una copia de esta representación pasará por un convLSTM dando lugar a la representación latente de apariencia  $z_{1:t}^a$ . A continuación, el sintetizador de flujo futuro toma  $z_{1:t}^m$  y lo decodifica con un decodificador convolucional obteniendo tanto el flujo voxel 3D  $F_t = (\delta X, \delta Y, M_3)$  como la máscara  $M_1$ . A continuación se realiza una operación de deformación sobre la secuencia de entrada, donde  $M_3$  indica la cantidad de contenido que se utilizará en el siguiente fotograma y  $M_1$  el rango del flujo óptico, obteniendo  $\hat{I}_{t+1}$ . El sintetizador de apariciones futuras toma  $z_{1:t}^a$  y lo decodifica en  $\hat{I}_{t+1}$  y la máscara  $M_2$ . De esta forma, tanto  $\hat{I}_{t+1}$  como  $M_2$  y  $\hat{I}_{t+1}$  entran en el sintetizador de fusión, que realiza la siguiente operación para fusionar los componentes y obtener como resultado el fotograma final en  $t + 1$ :

$$M_2 \odot \hat{I}_{t+1} + (1 - M_2) \odot \bar{I}_{t+1}$$

#### **Revisiting Adaptive Convolutions for Video Frame Interpolation**

Niklaus et al. abordan esta problemática desde otro ángulo, la interpolación. Por lo tanto en vez de predecir los fotogramas futuros, se trata de predecir un fotograma intermedio  $\hat{I}$ . Es decir, un fotograma entre otros dos  $I_1$  e  $I_2$ . Para ello utiliza una U-Net con bloques residuales y convoluciones escalonadas que generan los filtros unidimensionales que capturan el movimiento y varían espacialmente. Se generan dos kerneles por cada imagen( $K_{1,h}, K_{1,v}, K_{2,h}, K_{2,v}$ ). A continuación, se utilizan los kerneles para procesar las imágenes de entrada mediante una convolución con suma Kahan y obtener  $\hat{I}$ . Además, se utiliza relleno retardado. Es decir, no se rellenan durante las convoluciones de la U-Net, sino que se rellena durante la aplicación de kerneles de forma que se obtenga una imagen con la misma resolución de entrada, a la vez que se mejoran los resultados y se consigue una mayor eficiencia computacional. Por último, se realiza una normalización de kernel para mejorar la síntesis.

#### **3.1.2. Predicción de secuencia a secuencia de fotogramas**

Los modelos propuestos de secuencia a secuencia toman las entradas de una en una, obteniendo el fotograma siguiente en cada caso individual. Suelen ser estructuras

## Estado del arte

---

| Nombre                   | Modelo                            | <i>k</i> | <i>n</i> |
|--------------------------|-----------------------------------|----------|----------|
| Michalski et al.[MMK14]  | GAE piramidal                     | 2/3/5    | 1/2/20   |
| Ranzato et al.[Ran+14]   | k-medias y n-grama/NN/RNN/RCNN    | 12       | 1        |
| Oh et al.[Oh+15]         | Conv. AE / Conv. AE + LSTM        | 4/11     | 1        |
| Srivastava et al.[SMS15] | LSTM AE                           | 10/16    | 10/13    |
| Finn et al.[FGL16]       | ConvLSTM AE                       | 2/10     | 8/10     |
| Patraucean et al.[PHC16] | AE ConvLSTM y flujo óptico        | 10/12    | 1        |
| Denton et al.[DB17]      | Red de representación desenredada | 5/10     | 495/90   |
| Lotter et al.[LKC17]     | PredNet                           | 1        | 1        |
| Wang et al.[Wan+17]      | PredRNN                           | 10       | 10       |
| Oliu et al.[OSE18]       | Red plegada recurrente            | 10       | 10       |
| Wang et al.[Wan+18]      | PredRNN++                         | 10       | 10/20    |
| Castrejon et al.[CBC19]  | vRNN con jerarquía                | 5/2/2    | 10/28/10 |
| Minderer et al.[Min+19]  | AE+VRNN                           | 8        | 8        |
| Le Guen et al.[GT20]     | PhyDNet                           | 4/7      | 4/6/7    |
| Rakhimov et al.[Rak+20]  | LVT                               | 4        | 12       |
| Wu et al.[Wu+20b]        | AE+E                              | 4        | 5        |
| Wu et al.[Wu+20a]        | Mask RCNN+fg-net+bg-net           | 10/10/12 | 20/10/6  |
| Yu et al.[Yu+20]         | CrevNet                           | 10       | 10       |
| Behrmann et al. [BGN21]  | infoNCE                           | 20       | 20       |

Cuadro 3.2: Soluciones propuestas de secuencia a secuencia

que incluyen algún tipo de RNN. De tal forma predicen *n* fotogramas sin necesidad de distintas iteraciones. Tanto los fotogramas de entrada como los de salida utilizados en los experimentos de los artículos están indicados en la tabla 3.2. Seguidamente se incluye una breve explicación del funcionamiento de cada propuesta.

### ***Modeling deep temporal dependencies with recurrent grammar cells***

El primer artículo al que se hace mención por año de publicación es el de Michalski et al.[MMK14]. En él proponen un modelo de predicción basado en un *autoencoder* cerrado(GAE)[Mem08]. A diferencia de un AE tradicional como el explicado en el apartado 2.2.3., utiliza dos entradas distintas *x* e *y*, de forma que trata de reconstruir *y* a partir de *x*. Además, se emplean módulos GAE en cascada de forma que la capa inferior capte las características relacionales de primer orden (transformaciones entre entradas consecutivas) y las capas superiores las características relacionales de mayor orden. Por tanto, utiliza una organización en pirámide para captar la estructura entre los *timesteps* y de esta forma aplicar dichas transformaciones aprendidas para predecir los *n* fotogramas siguientes (Figura 3.2).

### ***Video (language) modeling: a baseline for generative models of natural videos***

En el artículo publicado por Ranzato et al.[Ran+14] se propone la comparación de distintas estructuras inspiradas en el modelado de lenguaje. Estos modelos son el n-grama, NN, RNN y CNN recurrente, siendo esta última la que obtiene mejores resultados. Una RCNN contiene capas recurrentes convolucionales (RCL) que, como su nombre indica, introducen conexiones recurrentes en las capas convolucionales explicadas en la sección 2[LH15]. Además, utiliza el algoritmo k-medias para tomar piezas de cada imagen y las discretiza. De esta forma, la predicción se reduce al pronóstico de a que parte de la imagen pertenece que centroide.

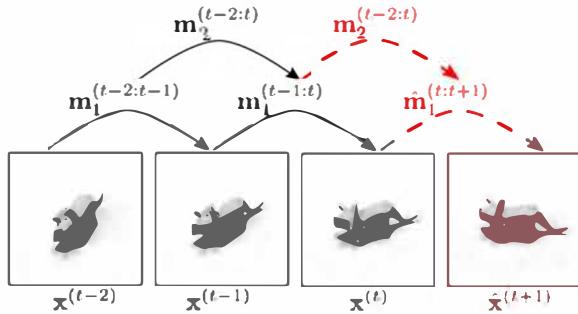


Figura 3.10: Características relacionales en cascada: primer y segundo orden[MMK14]

#### **Action-conditional videoprediction using deep networks in Atari games**

Oh et al.[Oh+15] utilizan tanto un AE convolucional como un AE convolucional con una capa LSTM entre codificador y decodificador. Este AE convolucional[Mas+11], como su nombre indica, utiliza capas convolucionales en el codificador y capas deconvolucionales en el decodificador. En el modelo propuesto la salida del codificador se multiplica por la acción tomada en el videojuego juego de Atari utilizado como *dataset*. Además en la estructura CAE + LSTM se incluye una capa LSTM tras el codificador convolucional para captar con mayor precisión la relación temporal.

#### **Unsupervised learning of video representations using LSTMs**

La estructura propuesta por Srivastava et al.[SMS15] corresponde a un LSTM AE que utiliza capas LSTM apiladas tanto para el codificador como para el decodificador. A continuación, realizan pruebas con tres metodologías distintas: reconstrucción de entrada, donde el codificador toma la secuencia de entrada y el decodificador la devuelve de forma inversa; la predicción de casos futuros, que al contrario de lo ocurrido en el caso anterior devuelve los fotogramas predichos; y el compuesto, que combina ambos enfoques anteriores. Además, utiliza decodificadores tanto condicionales como no condicionales. Los decodificadores condicionales son aquellos que utilizan el último fotograma generado como entrada, mientras que los no condicionales son aquellos decodificadores que no lo reciben.

#### **Unsupervised learning for physical interaction through video prediction**

En el caso de Finn et al.[FGL16] son 3 enfoques distintos los propuestos. Todos los enfoques utilizan una estructura compuesta por una capa convolucional, 8 LSTM convolucionales, otra capa convolucional y una capa de composición. El primer enfoque es la advección neuronal dinámica (ADN), que predice la distribución por cada localización para los píxeles en la nueva imagen. Para ello, se asume que los píxeles no se mueven largas distancias y que solo realizan movimientos en un área reducida. La advección neuronal dinámica convolucional, al contrario del enfoque anterior, predice múltiples distribuciones discretas para toda la imagen a partir de las que se obtiene el movimiento esperado para cada píxel. Por último, los predictores de transformadores espaciales (STP) producen parámetros para realizar transformaciones de

## Estado del arte

imágenes afines en 2D y a continuación los aplica realizando transformaciones bilineales de muestreo de *kernel*.

### **Spatio-temporal video autoencoder with differentiable memory**

La estructura propuesta por Patraucean et al.[PHC16] consiste en dos *autoencoder*, uno temporal anidado dentro de otro espacial. Por lo tanto, el modelo está compuesto por un *autoencoder* que entre codificador y decodificador tiene un LSTM convolucionar. También utiliza flujo óptico, que genera un mapa 2D (sobre la predicción y la imagen original) del movimiento realizado para mejorar el aprendizaje a largo plazo.

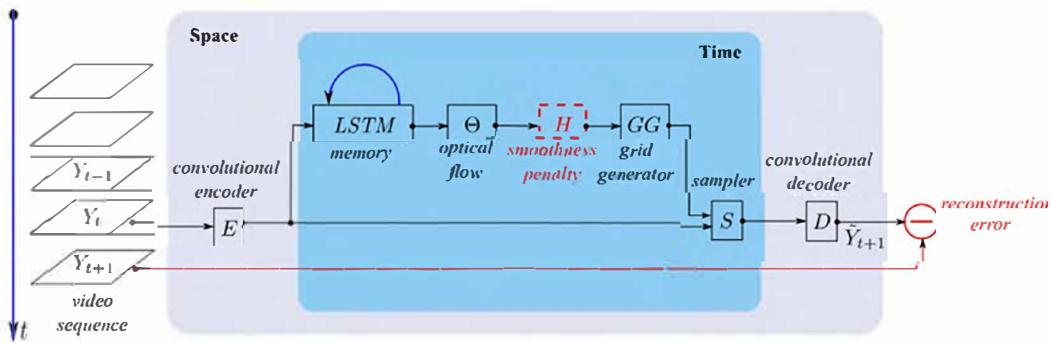


Figura 3.11: Estructura de Patraucean et al.[PHC16]

### **Unsupervised learning of disentangled representations from videos**

Denton et al.[DB17] proponen una red de representación desenredada cuyas estructuras principales son dos codificadores convolucionales, uno para contenido y otro para la pose. El codificador de contenido toma el fotograma de entrada  $x^t$  y lo codifica en  $h_c^t$ , el cual representa el contenido que se mantiene en el tiempo. Por otro lado, el codificador de pose obtiene  $h_p^{t+k}$  (representación de pose) a partir del fotograma  $x^{t+k}$ . La concatenación del resultado de ambos codificadores es la entrada de un LSTM que obtiene  $\tilde{h}_p^{t+k+1}$ , es decir, la pose en el instante siguiente. Por último, la concatenación de  $\tilde{h}_p^{t+k+1}$  y  $h_c^t$  se decodifica para obtener el fotograma siguiente  $\tilde{x}^{t+k+1}$ . La red utiliza la pérdida adversaria para asegurar el correcto funcionamiento de los codificadores y que el codificador de pose no incorpore contenido. Para ello asume que el contenido es constante en un vídeo, pero cambia entre vídeos. También utiliza el error de reconstrucción  $l_2$  para comparar el fotograma predicho y el verdadero, y el error de similitud para penalizar el error cuadrado entre los  $h_c$  de fotogramas vecinos. El discriminador C utiliza entropía cruzada binaria para que dados dos vectores se obtenga la probabilidad de proceder del mismo vídeo.

### **Deep predictive coding networks for video prediction and unsupervised learning**

Lotter et al.[LKC17] plantean, también en 2017, una red que denominan como PredNet y que se basa en la idea de codificación predictiva. Esta red tiene distintos módulos, estando cada uno compuesto por: la capa de representación recurrente, la cual predice  $\hat{A}_l$  (el fotograma siguiente a  $A_l$ ); la capa de representación del error, que calcula el error a partir de la diferencia entre  $\hat{A}_l$  y  $A_l$ ; la capa de entrada convolucional,

que aplica convolución a la entrada; y la capa de predicción. El funcionamiento general consiste en la predicción de  $\hat{A}_l$  por parte de la capa de representación recurrente, cuya estructura es un LSTM convolucional. La diferencia de  $\hat{A}_l$  con  $A_l$ , previa convolución de ambos en la capa de entrada y la de predicción respectivamente, se utiliza para calcular el error  $E_l$ . Este error es a su vez propagado a la capa de representación  $R_l$ , donde también se recibe el error desde  $R_{l+1}$ , de forma que se utiliza el error de los niveles superiores hacia los inferiores.

### **PredRNN: Recurrent neural networks for predictive learning using spatiotemporal LSTMs**

Wang et al.[Wan+17] propone un LSTM modificado, al que denominan ST-LSTM y que añade operaciones sobre la memoria espacio-temporal. Ambas memorias, espacio-temporal y temporal, se combinan en la puerta de salida. A continuación, se apilan distintos ST-LSTM para formar la red PredRNN que tiene una memoria espacio-temporal unificada. También modifica las conexiones respecto a los convLSTM apilados, añadiendo conexiones de actualización en zigzag de forma que la información fluya en horizontal y vertical.

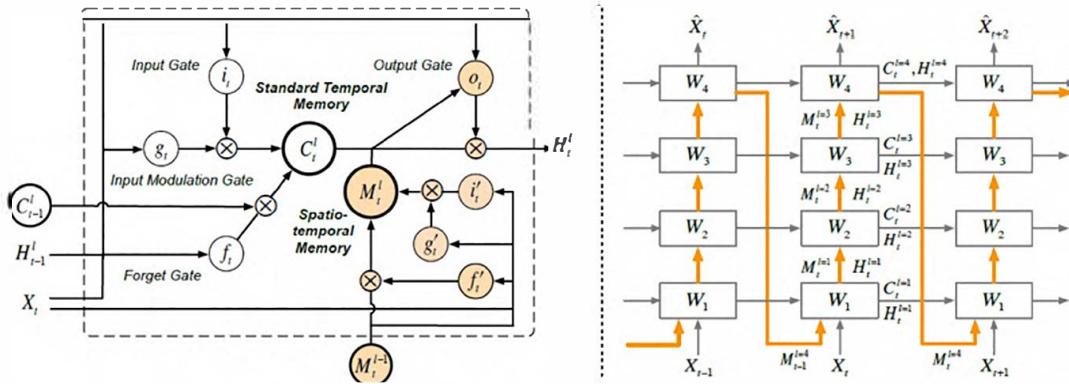


Figura 3.12: Estructura del ST-LSTM y PredRNN[Wan+17]

### **Folded recurrent neural networks for future video prediction**

Oliu et al.[OSE18] plantean en 2018 una red plegada recurrente, que se basa en apilar bGRU para obtener un AE recurrente que comparte estado entre codificador y decodificador. Los bGRU son GRU clásicos (RNN que tienen el estado oculto como salida) con mapeo doble, es decir, flujo bidireccional de información entre entrada y salida. También utiliza capas convolucionales, previa codificación y una vez obtenida la salida del decodificador, para realizar la deconvolución y obtener los siguientes fotogramas.

### **PredRNN++: Towards a resolution of the deep-in-time dilemma in spatiotemporal predictive learning**

También en 2018, Wang et al.[Wan+18] proponen una mejora a la anteriormente mencionada PredRNN, denominada PredRNN++. Esta mejora trata de resolver los problemas de predicción a largo plazo de PredRNN a partir de la utilización de LSTM

causales. Los LSTM causales son LSTM modificados, como su propio nombre indica, que añaden capas no lineales entre transiciones recurrentes y donde las memorias temporales y espacio-temporales están conectadas en forma de cascada a través de distintas puertas de tal forma que la memoria espacial actúa como una función de la memoria temporal. También utiliza una unidad de gradiente de carretera para evitar la propagación hacia atrás del gradiente.

### **Improved Conditional VRNNs for Video Prediction**

Castrejon et al.[CBC19] proponen un RNN variacional(vRNN) con jerarquía. Para ello utiliza vectores latentes  $z_t$  divisibles por niveles de la siguiente forma  $z_t^1, \dots, z_t^L$ , siendo  $L$  el número de niveles. En primer lugar se utilizan todos los fotogramas de contexto (aquellos fotogramas que son conocidos) pasándolos por un codificador con 2D CNN, bloques ResNet y *max-pooling*. El resultado son los estados iniciales de a priori, a posteriori y la probabilidad de las redes. A continuación, otro codificador toma el fotograma en el momento  $t$ , que pasa por otro codificador de fotogramas a su vez generando los  $z_t$ . Un decodificador recurrente toma ambas salidas y genera  $x_t$  siguiendo la probabilidad a priori  $p(x_t | z_t, x_{<t}, c)$ . Este decodificador está compuesto por convLSTM y deconvoluciones. EL proceso se repite para generar consecuentes fotogramas, excepto la generación de estados iniciales.

### **Unsupervised Learning of Object Structure and Dynamics from Videos**

Por el contrario, Minderer et al.[Min+19] divide su estructura en dos partes: un detector de puntos clave  $x_t$ , y un modelo dinámico. El detector de puntos clave está a su vez dividido en dos redes que tienen una estructura de AE. La primera red es un detector formado por capas convolucionales y denominado  $\varphi^{\text{det}}$ , que toma el fotograma en  $v_t$  y genera los mapas de características de los puntos clave  $x_t$ . Por lo tanto, produce  $K$  mapas de características(uno por punto clave), los cuales se normalizan y se condensan en coordenadas  $(x, y)$ . Por otro lado, el generador  $\varphi^{\text{rec}}$ , también compuesto por capas convolucionales, reconstruye él  $v_t$  a partir de los mapas de características del primer fotograma  $v_1$  y los mapas gaussianos obtenidos a partir de los puntos claves correspondientes  $(x, y)$ . El modelo dinámico utiliza una VRNN, donde predice  $z_t$  (vector latente sobre los puntos clave) de forma previa a observar la imagen de la siguiente forma:

$$p(z_t | x_{<t}, z_{<t}) = \varphi^{\text{prior}}(h_{t-1})$$

A continuación, se obtiene la creencia a posteriori a partir de las coordenadas de los puntos clave:

$$q(z_t | x_{<t}, z_{<t}) = \varphi^{\text{enc}}(h_{t-1}, x_t)$$

Por último, se codifica  $z_t$  y el estado oculto  $h_{t-1}$  para obtener los mapas de características predichos:

$$p(x_t | z_t, x_{<t}) = \varphi^{\text{dec}}(z_t, h_{t-1})$$

A su vez se actualiza la RNN. Para obtener la imagen totalmente reconstruida, se concatenan los mapas de características de  $v_1$  y los  $\hat{x}_t$ , reconstruyendo el fotograma de la predicción.

### 3.1. Predicción de fotogramas

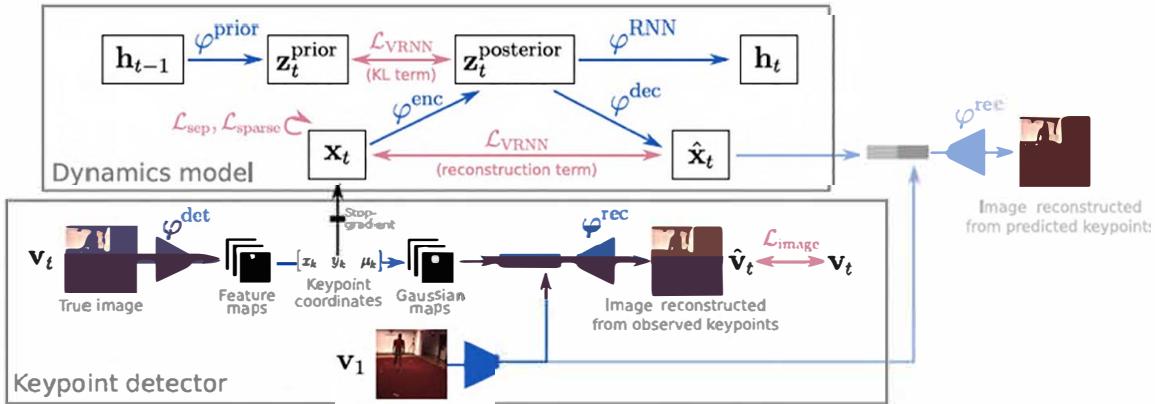


Figura 3.13: Estructura de Minderer et al.[Min+19]

#### **Disentangling Physical Dynamics from Unknown Factors for Unsupervised Video Prediction**

Le Guen et al.[GT20] proponen una arquitectura que consiste en la idea de las representaciones latentes  $h(t, x)$ , donde  $t$  es el instante de tiempo y  $x$  son las coordenadas ( $x, y$ ). Se obtienen representaciones latentes del espacio latente  $\mathcal{H}$  de la siguiente manera:

$$\frac{\delta h(t, x)}{\delta t} = \frac{\delta h^p}{\delta t} + \frac{\delta h^r}{\delta t} = M_p(h^p, u) + M_r(h^r, u)$$

$h^p$  es la representación física y  $h^r$  es la representación residual. Por otra parte,  $M_p(h^p, u)$  y  $M_r(h^r, u)$  son las dinámicas físicas y residuales respectivamente en el espacio  $\mathcal{H}$ . La arquitectura consiste en el mapeo de las secuencias de entrada al espacio latente, como ya ha ocurrido en arquitecturas anteriores. Cada uno de los fotogramas entra en el codificador convolucional y se obtiene  $E(u_t)$ . Este entra tanto a PhyCell, donde se modela la parte física de la fórmula anterior, y un convLSTM que modela la parte residual. Al entender  $h_{t+1}$  como la suma de  $h_{t+1}^p$  y  $h_{t+1}^r$ , se suman los resultados y pasan a un decodificador que genera el fotograma siguiente  $u_{t+1}$ . Por su parte, PhyCell calcula  $M_p(h^p, u)$  de la siguiente manera, donde  $\phi(h)$  es el predictor físico y  $C(h, u)$  es un término corrector:

$$M_p(h^p, u) = \phi(h) + C(h, u)$$

PhyCell se divide en dos partes distintas: el predictor físico y la asimilación de la entrada. El predictor físico calcula  $\phi(h)$  mediante una CNN y realiza la aproximación de  $\tilde{h}_{t+1}$ .

$$\tilde{h}_{t+1} = h_t + \phi(h_t)$$

A continuación, se approxima la ganancia de Kalman ( $K_t$ ) a partir de los kernels  $W_h$  y  $W_u$  y el bias  $b_k$ .

$$K_t = \tanh(W_h * \tilde{h}_{t+1} + W_u * E(u_t) + b_k)$$

Por último, calcula el vector latente en  $t + 1$ .

$$h_{t+1} = \tilde{h}_{t+1} + K_t \odot (E(u_t) - \tilde{h}_{t+1})$$

### **Latent Video Transformer**

Rakhimov et al [Rak+20] proponen un transformador de vídeo latente (LVT). La primera parte está compuesta por un *autoencoder*, en este caso VQ-VAE[OVK18], donde cada fotograma de entrada con dimensiones  $H \times W \times 3$  pasa por el codificador y se divide en  $n_c = 4$  por la dimensión de los canales. A continuación, se mapean los píxeles de cada una de las partes a su integración más cercana según el libro de códigos. El resultado es  $z_e(x)$ , que en el decodificador es mapeado de vuelta al espacio de píxeles. Por otro lado, se utiliza el generador de vídeo latente en forma de generador auto-regresivo[WTU19] pero aplicándolo al espacio latente en lugar de a píxeles. De esta forma se generan los fotogramas no conocidos de la secuencia a partir del  $z$  de los fotogramas conocidos. La entrada al modelo consiste en  $Z \in K^{T \times h \times w \times n_c}$ , donde  $T$  es el número de fotogramas y se utilizan los  $T_0$  primeros. A continuación, se divide el vídeo en porciones mediante el subescalado:

$$\frac{T}{s_t} \times \frac{h}{s_h} \times \frac{w}{s_w}$$

El generador se encarga de, mediante una estructura AE, generar nuevos valores para los píxeles dentro de cada  $Z$ . El decodificador de fotogramas anteriormente explicado es el encargado de decodificar esta representación latente devolviendo cada uno de los fotogramas.

### **Future Video Synthesis with Object Motion Prediction**

Wu et al. [Wu+20b] proponen una red de predicción de fotogramas compuesta por dos partes distintas: el predictor de fondo y el predictor dinámico de movimiento de objetos. Este último consiste en un AE con conexiones omitidas que toma como entrada el fondo y los mapas de instancias. En concreto, el codificador es una estructura ResNet y el decodificador está compuesto por capas deconvolucionales. En primer lugar, la secuencia de fotogramas, los mapas semánticos, los mapas de instancias y el flujo óptico entre los fotogramas entran a un modelo  $M$  de detección dinámica de objetos, generando los fondos y los objetos dinámicos de forma separada. Una vez entran en la predicción del fondo y pasan el AE, las imágenes generadas llegan a un modelo  $I$  donde se rellenan los posibles vacíos generados mediante el método del pintado[Yu+18]. Por otro lado, los objetos dinámicos son tratados por separado en el predictor dinámico de movimiento de objetos, donde junto a los fondos obtenidos en el predictor de fondo, los flujos ópticos y los mapas semánticos pasan a un transformador espacial que actúa como codificador y genera los parámetros de las transformaciones 2D afines. Estos parámetros son tomados por el generador de cuadrículas, junto al último fotograma, que transforma las coordenadas de los objetos. Por último, se componen los objetos y el fondo generados por cada parte y se vuelve a utilizar el método del pintado.

### **Generating Future Frames with Mask-Guided Prediction**

Wu et al. [Wu+20a] proponen una arquitectura compuesta por cuatro partes distintas: la capa de separación, el modelo de fondo, el modelo de primer plano y la capa de unión. En primer lugar, la capa de separación toma la secuencia de fotogramas y divide cada imagen en fondo y primer plano. Para separar el fondo se utiliza la máscara, que es un mapa binario donde el 1 indica si el píxel pertenece a la instancia

### 3.1. Predicción de fotogramas

y 0 que no pertenece. En concreto se utiliza una red Mask RCNN[He+17] para obtener la máscara. A continuación, las máscaras entran en el modelo de primer plano o fg-net. El modelo fg-net está compuesto por un AE, y un convLSTM en los casos de secuencias largas, para predecir las máscaras en los siguientes fotogramas. Una vez predichas, estas máscaras pasan a una red M2F junto al último primer plano de entrada. El M2D sigue una estructura U-Net y genera los primeros planos de los futuros fotogramas. Por otro lado, el modelo de fondo o bg-net toma como entrada los fondos obtenidos en la capa de separación, que pasan a una red similar a la utilizada en fg-net (con convLSTM) y se genera un mapa de flujo. A partir del mapa de flujo y el último fondo de entrada se realiza una interpolación bilinear mediante un AE que aprende dicha transformación espacial y que produce el fondo siguiente. De esta forma se generan los siguientes fondos. Por último, la capa de unión une el primer plano y el fondo, para lo que utiliza una matriz ponderada obtenida al final de la bg-net.

$$\hat{I}_{t+1} = \hat{I}_{t+1}^f * \hat{w}_{t+1} + \hat{I}_{t+1}^b * (1 - \hat{w}_{t+1})$$

A partir de este cálculo (donde  $\hat{I}_{t+1}^f$  es el primer plano predicho,  $\hat{I}_{t+1}^b$  el fondo predicho y  $\hat{w}_{t+1}$  la matriz ponderada) se obtienen los nuevos fotogramas.

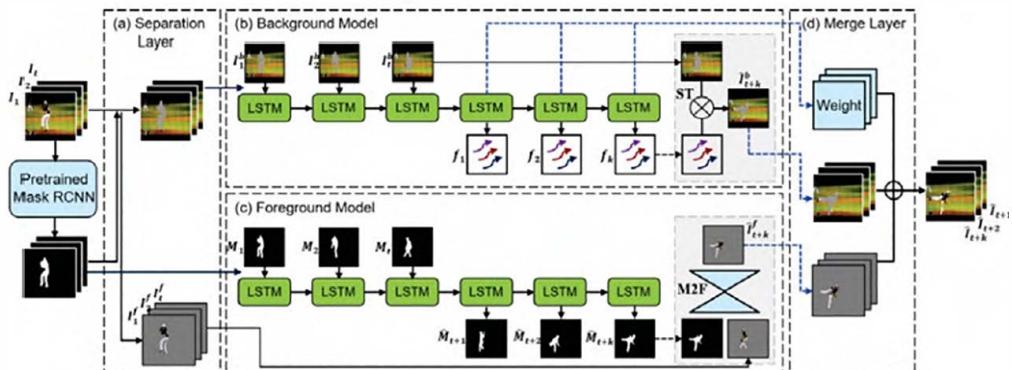


Figura 3.14: Estructura de Wu et al.[Wu+20a]

#### **Efficient and Information-Preserving Future Frame Prediction and Beyond**

Yu et al.[Yu+20] propone CrevNet, una arquitectura compuesta por dos redes: un autoencoder reversible y bidireccional y un predictor recurrente( $\mathcal{P}$ ). El AE está compuesto por un codificador  $\mathcal{E}$  y un decodificador  $\mathcal{D}$ . La secuencia de fotogramas  $x_t$  se divide en dos( $x^1$  y  $x^2$ ) para entrar al AE. Este está compuesto por distintos bloques donde cada uno realiza los siguientes cálculos, siendo  $\mathcal{F}$  una operación de convolución 3D (utiliza tensores de entrada con 4 dimensiones):

$$x^1 = \hat{x}^1 - \mathcal{F}_2(\hat{x}^2) \quad x^2 = \hat{x}^2 - \mathcal{F}_1(x^1)$$

De esta forma se asegura que no se pierda información. Por el funcionamiento bidireccional y reversible de esta estructura no es necesario implementar  $\mathcal{D}$ , sino que se utiliza de forma inversa. Por otro lado,  $\mathcal{P}$  utiliza una estructura similar, donde las convoluciones se sustituyen por ConvLSTM o LSTM espacio-temporales. Además utiliza un módulo de atención calculado como:

$$g_t = \phi(W_2 * \text{ReLU}(W_1 * h_t^1 + b_1) + b_2)$$

A su vez, utiliza suma ponderada:

$$\hat{x}_t^2 = (1 - g_t) \odot x_t^2 + g_t \odot h_t^1$$

Como ocurría con el AE esta estructura es reversible, de tal forma que el cálculo de la secuencia de fotogramas siguiente se puede calcular como:

$$x_{t-1} = \mathcal{E}^{-1} (\mathcal{P}^{-1} (\mathcal{E} (\hat{x}_t) | h_{t-1}))$$

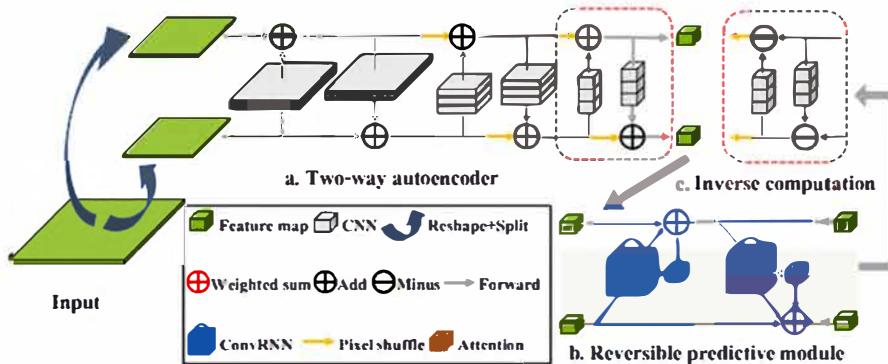


Figura 3.15: Estructura de CrevNet[Yu+20]

### ***Unsupervised Video Representation Learning by Bidirectional Feature Prediction***

El artículo de Behrmann et al.[BGN21] parte de la distinción de fotogramas pasados y futuros dados los fotogramas presentes. A esta red la denominan InfoNCE. En primer lugar se dividen los fotogramas en pasado, presente y futuro, de forma que se puedan explotar las representaciones conjuntas ( $P, F$ ). Esta red bidireccional consiste en la división de los fotogramas en bloques de fotogramas de tal manera que no se superpongan. Cada uno de estos bloques pasa por una transformación espacial  $\mathcal{T}$ , que puede consistir en operaciones como recorte o rotaciones. En el caso de los bloques presentes, se realiza una convolución mediante un ResNet-18 2D-3D ( $\phi$ ). En el caso de pasado y futuro,  $\phi$  es compartido para todos los bloques. Las características extraídas del presente se agregan mediante un convGRU( $\psi$ ) con un kernel de tamaño 1 de forma incremental hasta obtener  $z_v$ . En el caso de las características de pasado y futuro, esta agregación se denota  $\varphi$  y se aplica a las características de pasado y, posteriormente a su resultado, a las características de futuro obteniendo  $z_{pf}$ . Para obtener los negativos se realiza esta agregación a la inversa, obteniendo  $z_{fp}$ .

## Capítulo 4

# Definición del problema

Con el fin de predecir el crecimiento de colonias de bacterias se han desarrollado distintas herramientas que llevan a cabo simulaciones de dicho crecimiento. Estas herramientas simulan el crecimiento de las colonias a cambio de un gran costo computacional debido a las distintas funciones que realiza una bacteria al mismo tiempo (consumición de nutrientes, difusión de señales y estado de la célula) y el gran número de estas, lo que hace complicada su utilización para ciertas tareas.

El objetivo de este proyecto es desarrollo de una herramienta que sea capaz de realizar predicciones de como se desarrollará una colonia de bacterias, en este caso *echoli*, en formato visual y a partir de un estado inicial. De esta forma, con la utilización de redes neuronales, ya sea como herramienta única o complementaria a estos simuladores, se consigue solventar el problema del coste computacional. Además, se consigue ofrecer una alternativa al motor gráfico existente en el simulador GRO, en el cual se centra el proyecto, para que pueda ser utilizada a futuro. Las redes elegidas para estas tareas podrán ser utilizadas con otros *dataset* que cubran mejor las necesidades, ya sea con imágenes más realistas u otras a elección.

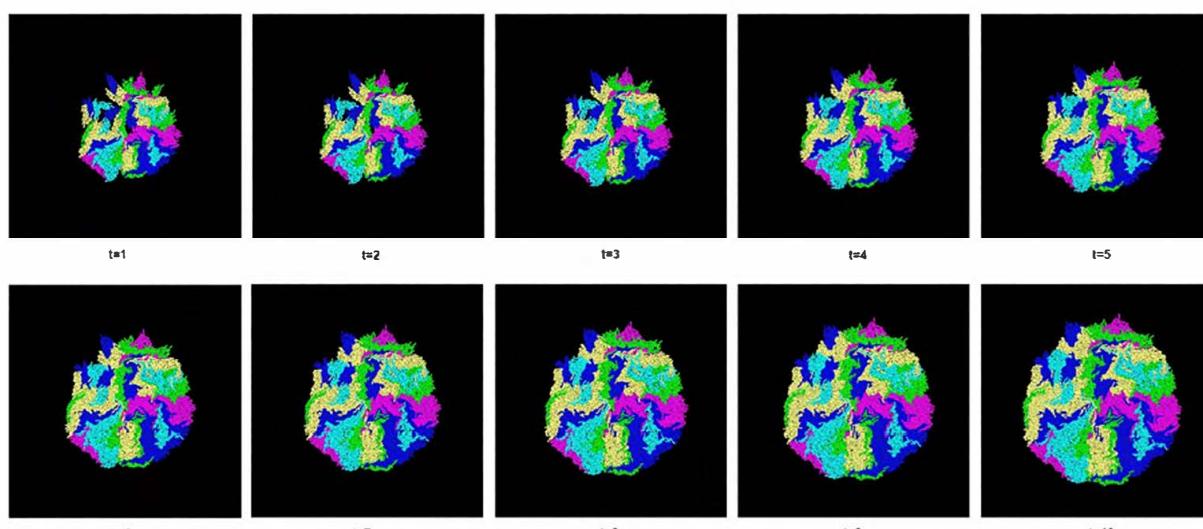


Figura 4.1: Crecimiento en una colonia de bacterias en GRO

## **Definición del problema**

---

Para alcanzar estos objetivos satisfactoriamente, se plantean las siguientes problemáticas:

- Obtención o generación de un *dataset* sintético de colonias de bacterias con interacciones entre ellas. Dicho *dataset* deberá ser equilibrado y completos sobre las distintas posibilidades para evitar el sobreajuste y mejorar la generalización. Para simplificar el rango de posibilidades el número de tipos de bacteria se limitará a 5.
- Elección de las redes a implementar a partir del estudio realizado en la sección 3.2. Será necesario establecer un criterio de elección de forma que las redes elegidas sean apropiadas para la tarea. Se establece el número de redes a implementar en 6 de forma que se cubran distintas metodologías y tipos de redes.
- Implementación y realización de pruebas sobre el *dataset* generado de cada una de las redes elegidas para la comparación de resultados. Dichas redes deberán ser ajustadas y se elegirá la red que ofrezca mejores resultados. Se realizarán las modificaciones necesarias para adaptar estas redes al *dataset* y a la tarea.
- La relación entre crecimiento y la existencia de nutrientes, interacciones y fuerza de las interacciones deberá ser capturada.
- Para considerar los resultados aceptables es necesario que tengan una buena resolución, al menos manteniendo la resolución de entrada, que los colores no sean alterados y no contengan borrosidad.
- Las redes de predicción de videos experimentan problemas a la hora de identificar las distintas posibilidades en el futuro. Este problema puede verse empeorado por el hecho de que las interacciones entre tipos de bacteria no son perceptibles en estadios iniciales del crecimiento. De esta forma, un fotograma en el instante  $t = 1$  puede seguir distinto crecimiento según las interacciones entre los tipos de bacteria o la cantidad de nutriente existente, entre otros. Será necesario encontrar una solución a este problema para que los primeros fotogramas sean predichos con precisión.

# Capítulo 5

## Redes de predicción del crecimiento en colonias de bacterias

En esta sección se expone la implementación de soluciones a las principales problemáticas expuestas en la sección anterior. En concreto se abordan aquellas relativas a la elección del *dataset* y de las redes neuronales a emplear.

### 5.1. Elección del *dataset*

En primer lugar, es necesario establecer el *dataset* sobre el que se realizará el aprendizaje. En este caso un *dataset* que muestre el crecimiento de una colonia con distintos tipos de bacterias. Antes que nada se ha procedido a la búsqueda en bases de datos públicas para reutilizar un *dataset* ya existente y con las imágenes tratadas, pero no se han encontrado ninguno que satisfaga las necesidades de este proyecto. Por eso, se ha procedido a la creación de un *dataset* sintético.

#### 5.1.1. Generación del *dataset*

La herramienta utilizada para la creación del *dataset* es el propio simulador GRO para el que se trata de ofrecer un motor gráfico alternativo. Para ello, se establece un límite de máximo 5 tipos de bacteria en una misma colonia. Por lo tanto, se definen 5 tipos con 5 colores distintos,

| Cepa | Color resultante | Verde | Rojo | Amarillo | Cian |
|------|------------------|-------|------|----------|------|
| b1   | Turquesa         | 500   | 0    | 0        | 500  |
| b2   | Verde            | 500   | 0    | 0        | 0    |
| b3   | Amarillo         | 250   | 400  | 100      | 100  |
| b4   | Rosa             | 0     | 500  | 0        | 500  |
| b5   | Azul             | 0     | 0    | 0        | 500  |

Cuadro 5.1: Cepas y sus colores junto a la combinación utilizada en GRO

Cada colonia, cuenta con  $n$  cepas (siendo  $n \leq 5$  y  $n > 0$ ), y donde habrá un factor de interacción  $i = n^2 - n$ . El número de interacciones positivas  $p$  cumplirá que  $p \leq i$ . La

## Redes de predicción del crecimiento en colonias de bacterias

---

fuerza de estas interacciones pertenecerá al intervalo [0'1,0'5].

| Nº Cepas | Factor interacción | Interacciones positivas | Fuerza interacciones |
|----------|--------------------|-------------------------|----------------------|
| 1        | 0                  | 0                       | [0'1,0'5]            |
| 2        | [0,2]              | [0,2]                   | [0'1,0'5]            |
| 3        | [0,6]              | [0,6]                   | [0'1,0'5]            |
| 4        | [0,12]             | [0,12]                  | [0'1,0'5]            |
| 5        | [0,20]             | [0,20]                  | [0'1,0'5]            |

Cuadro 5.2: Número de cepas y las posibles interacciones entre ellas

Con el fin de hacer el *dataset* robusto y equilibrado se adoptan las siguientes limitaciones de cara a la generación de casos, entendiendo por caso cada vídeo:

- Se abarcan todas las posibles combinaciones de cepas para cada número de cepas en una colonia. Por ejemplo en el caso de 2 cepas tendremos los siguientes pares: b1-b2, b1-b3, b1-b4, b1-b5, b2-b3, b2-b4, b2-b5, b3-b4, b3-b5 y b4-b5. Por claridad a estas combinaciones se las denominará *cc* (combinación de cepas) a partir de este punto.
- Se incluirá al menos un caso por cada posibilidad de *cc* e interacciones. Por ejemplo, en la situación del *cc* b1-b2 se incluirá mínimo un caso de 0, 1 y 2 interacciones respectivamente.
- Solo podrá existir un caso de interacción 0 por cada *cc*.
- Todos los números de cepas contarán con el mismo número de casos (excepto para 1 número de cepas, puesto que es inviable al no existir interacciones) distribuido entre los distintos *cc*. Este número se establece en 100.
- Para cada uno de los casos, durante la simulación en GRO se tomarán imágenes durante el crecimiento con periodo 10 hasta  $t = 300$ . Este  $t$  es el tiempo de referencia en GRO, no se corresponde al tiempo real puesto que éste último es variable dependiendo de la exigencia computacional de cada caso particular. El resultado serán 300 fotogramas por cada uno de los 405 casos.
- También se tomarán los datos poblacionales de cada caso en formato .xlsx.

De esta forma se generan los distintos archivos GRO de forma automatizada mediante la modificación del código generador de matrices[Lóp19]. Todos los casos y sus variables han sido documentados en el anexo de este documento. A continuación, se ha ejecutado en GRO el código generado para cada caso de forma individual y manual.

### 5.1.2. Tratamiento del dataset

De cara a preparar el *dataset* para el entrenamiento de las distintas redes neuronales, se realiza un preprocesamiento. En primer lugar se tratarán las imágenes individualmente, recortando píxeles superiores para que no se introduzca ruido en forma del texto informativo en GRO. También se redimensionarán las imágenes con el fin de reducir el tiempo de entrenamiento y adecuarlas a los recursos computacionales existentes para este proyecto. El tamaño elegido para estas imágenes es de 400x400 desde un 800x800. A continuación, también con el fin de adecuar el *dataset* a los recursos disponibles, se establecen los fotogramas que compondrán cada caso a 20,

que en lugar de ser consecutivos tendrán un intervalo de 5. Es decir, para formar el *dataset* final se tomarán 20 fotogramas con 5 fotogramas de separación entre cada uno sobre el *dataset* original. Sin embargo, el primer fotograma de la secuencia no será siempre  $t = 0$ , sino que se tomará un fotograma aleatorio entre 0 y 250. De esta forma también se consigue una secuencia en la que el crecimiento se observa de forma más rápida.

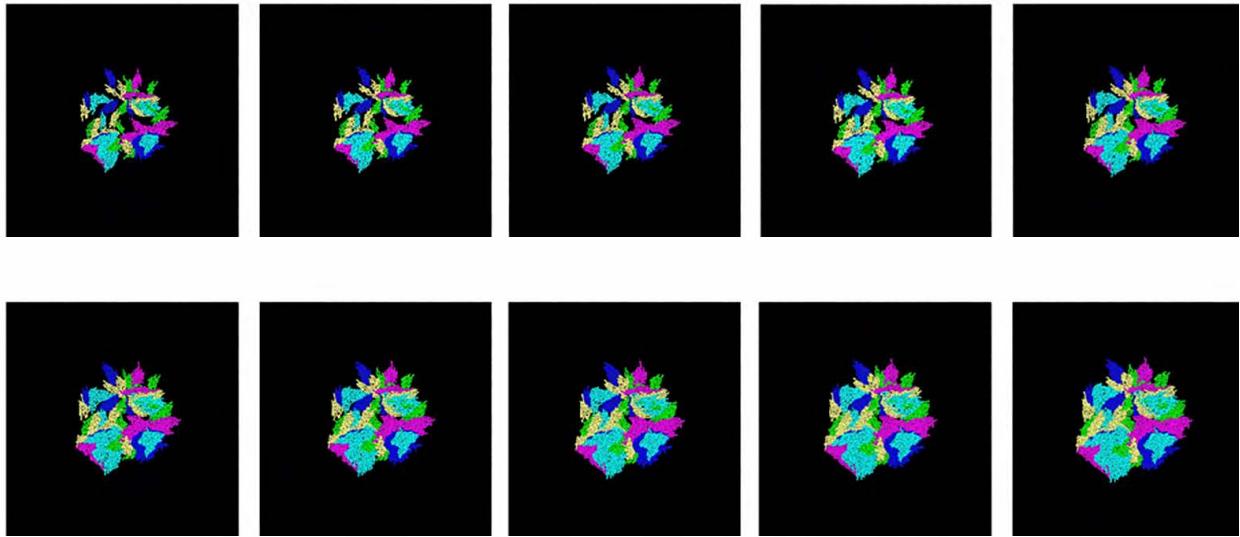


Figura 5.1: Fotogramas 178, 183, 188, 193, 198, 203, 208, 213, 218 y 223 del caso 46 del cc b1-b2-b3-b4-b5.

Los píxeles de los fotogramas serán normalizados ya sea al intervalo  $[-1, 1]$  o al  $[0, 1]$ . dependiendo de la red en la que se vaya a utilizar y la función de activación que contienen. Además es necesario asegurarse de que se extraen los 3 canales RGB de las imágenes a color de forma correcta a formato numérico. Por último, es necesario dividir los casos entre los de entrenamiento y los de *testing*. Para *testing* únicamente se utilizará un caso, de forma que en las pruebas de todas las redes sean sobre el mismo caso y se puedan comparar los resultados obtenidos. El resto se destinan al entrenamiento. Para aquellos casos en los que el aprendizaje sea supervisado y se necesiten etiquetas, dado un fotograma en  $t$  se considerará la etiqueta el fotograma en  $t + 1$ .

## 5.2. Elección de las redes neuronales

Como se expuso en la sección anterior, es necesario seleccionar unos criterios para la elección de las redes neuronales a implementar. Se van a implementar diversas redes y no una única para valorar distintos resultados. Por lo tanto, se establecen los siguientes criterios:

- No se utilizarán redes muy complejas debido a dos factores. El primero es que la mayoría de los *dataset* utilizados en los *paper* estudiados están compuestos por imágenes con profundidad y objetos con distintas formas y tamaños, que además se mueven de forma variable. En cambio, en el *dataset* generado son imágenes 2D (sin profundidad), donde solamente hay un tipo de objeto(la

## Redes de predicción del crecimiento en colonias de bacterias

---

bacteria) y su movimiento es mínimo. El segundo factor es por la limitación de recursos computacionales destinados a éste proyecto.

- Se procurará en la medida de lo posible utilizar soluciones basadas en ideas y estructuras distintas, y que haya un equilibrio entre el número de implementaciones secuencia a fotograma y secuencia a secuencia.
- En los casos de duda entre dos soluciones similares, se elegirá la que haya mostrado mejores resultados en las pruebas realizadas en el artículo.
- El código fuente debe utilizar TensorFlow y python.

Las soluciones elegidas son: Mathieu et al.[MCL16], Villegar et al.[Vil+17b] y Minderer et al.[Min+19].

### 5.2.1. GAN multiescalar con pérdida GDL

Siguiendo la estructura expuesta en Mathieu et al.[MCL16] se utiliza una red GAN. La red generadora o G tiene una arquitectura compuesta por convoluciones con *padding* y no linearidad ReLU. Además, añade una función tanh al final para que los valores estén en el rango de -1 a 1. Por otro lado, la red discriminadora D utiliza convoluciones sin *padding*, no linearidad ReLU y capas completamente conectadas.



Figura 5.2: Arquitectura detallada de la GAN multiescalar implementada

A parte de la multiescala anteriormente utilizada, emplea la proporción de señal máxima a ruido(PSNR):

$$\text{PSNR}(Y, \hat{Y}) = 10 \log_{10} \frac{\max_{\hat{Y}}^2}{\frac{1}{N} \sum_{i=0}^N (Y_i - \hat{Y}_i)^2}$$

En esta fórmula  $Y$  corresponde al fotograma real,  $\hat{Y}$  al fotograma predicho y  $\max_{\hat{Y}}^2$  al valor máximo posible de las intensidades. Este cálculo se utiliza para evaluar la calidad de la imagen predicha. Además de esta medida utiliza el índice de similitud estructural(SSIM)[Sim04]. Por otro lado, para el cálculo de la nitidez se utiliza la

siguiente fórmula:

$$\text{Sharp. diff.}(Y, \hat{Y}) = 0 \log_{10} \frac{\max_{\hat{Y}}^2}{\frac{1}{N} \left( \sum_i \sum_j \left| (\nabla_i Y + \nabla_j Y) - (\nabla_i \hat{Y} + \nabla_j \hat{Y}) \right| \right)}$$

Esta red ha sido elegida para cubrir la estructura GAN, que es una de las más interesantes de cara a llevar a cabo esta tarea, y por ser una de las soluciones con más impacto de cara a los avances en este campo. La implementación utilizada ha sido obtenida del repositorio oficial del artículo, la cual se puede encontrar en el siguiente enlace: [https://github.com/dyelax/Adversarial\\_Video\\_Generation](https://github.com/dyelax/Adversarial_Video_Generation). Dicha implementación ha sido modificada para adaptarla a Python 3, a la versión actual de Tensorflow (2.5.0) y al *dataset* sintético utilizado. Para esto último ha sido necesario modificar la lectura y entrada del *dataset*. El resto del código se ha mantenido.

El objetivo de esta red consiste en la predicción de los 10 fotogramas siguientes dados 10 fotogramas de una secuencia. Los parámetros utilizados se ilustran en la figura 5.2. En dicha figura se indican las capas convoluciones de color azul y las completamente conectadas de verde. Los valores dentro de cada capa corresponden al tamaño de los kerneles en el caso de las capas convolucionales, y al tamaño de la capa en el caso de las completamente conectadas. Los valores indicados al lado derecho son la forma de los mapas de características obtenidos. Las funciones de activación ReLU han sido obviadas.

Se han realizado 60000 *epoch* con tamaño de *batch* 1 por problemas de memoria. Por eso mismo, en este modelo las imágenes utilizadas como *dataset* han sido redimensionadas a 200 píxeles de ancho y 200 píxeles de alto. Otros datos de interés son el uso de guardados progresivos cada 5000 *epoch* para resultados del *testing* y el modelo, y de 100 *epoch* para los resultados de *summaries*, que son estructuras de Tensorflow que almacenan datos del entrenamiento para su posterior visualización. La tasa de aprendizaje utilizada es de 0.00004 en el generador y de 0.02 para el discriminador. Además, no se utiliza *padding* para las convoluciones del discriminador, pero si para las del generador.

### 5.2.2. MCNet con conexiones residuales

La estructura propuesta por Villegas et al.[Vil+17a], como se expuso en la sección 3, es un *autoencoder* con capas ConvLSTM. Esta red ha sido escogida por la capacidad temporal de las redes LSTM, y su posterior mejora con la aparición de las ConvLSTM. Además, los *autoencoder* han demostrado tener buenos resultados en el campo de la predicción de fotogramas como se ha podido comprobar en el estudio previo. Otra ventaja de esta red es la separación en movimiento y contenido, que funciona de forma similar a la separación en pose y contenido [DB17], y que ofrece un enfoque distinto a la hora de resolver esta problemática.

La aportación más interesante de esta red, aparte de la separación entre movimiento y contenido, es la incorporación de conexiones residuales[Sun15]. La función de las conexiones residuales es captar información la información tanto de contenido como de movimiento que habría sido perdida tras las operaciones de *pooling*, y se pasan de vuelta al decodificador tras las operaciones *unpooling*.

## Redes de predicción del crecimiento en colonias de bacterias

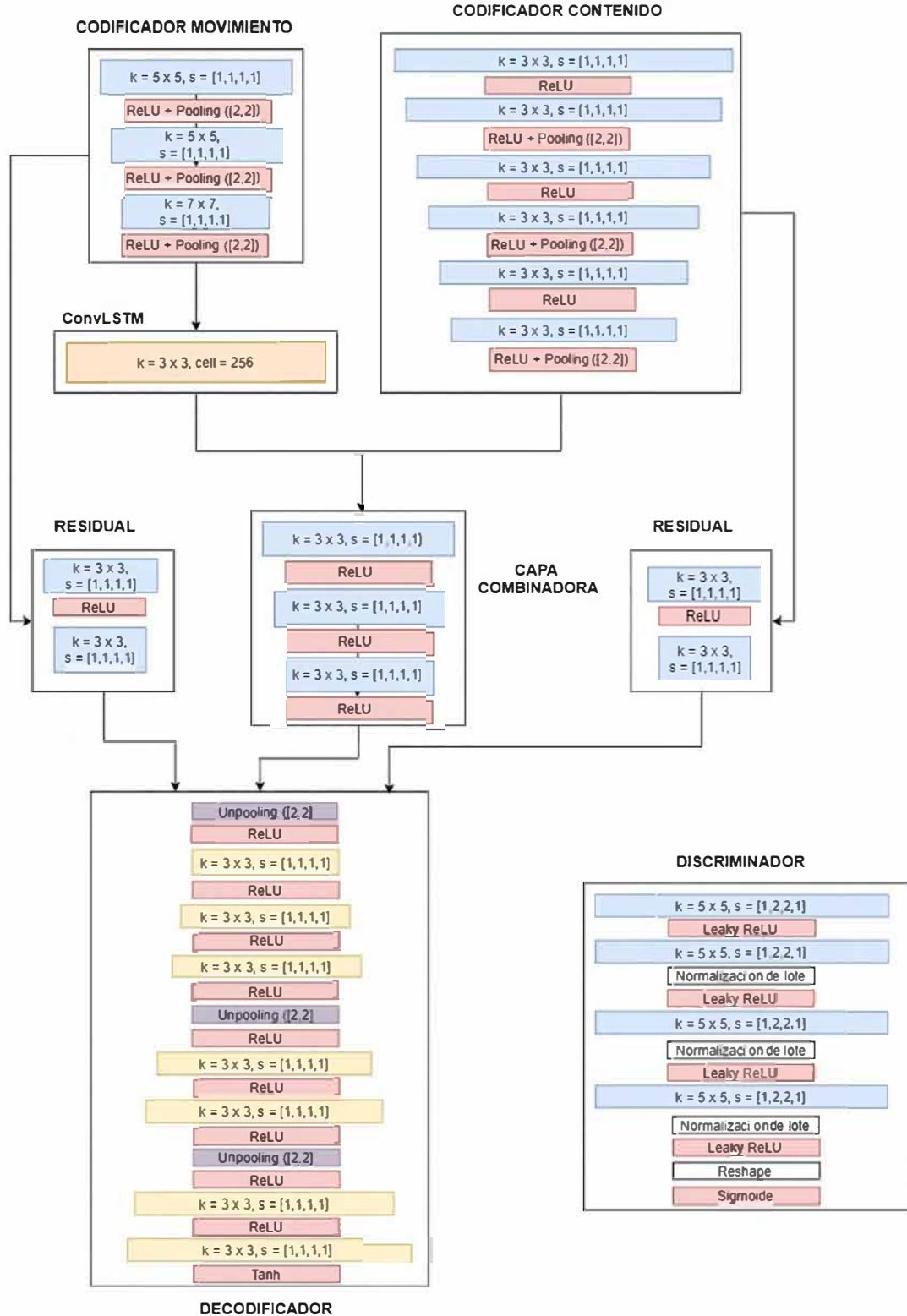


Figura 5.3: Arquitectura detallada de la MCnet implementada

La arquitectura de la red implementada ha sido detallada en la figura 5.3. En dicha figura las capas azules indican las capas convolucionales, las amarillas las deconvolucionales, las rojas las funciones de activación y el *pooling*, las moradas el *unpooling*,

las naranjas las capas ConvLSTM y las blancas otras operaciones. Todas las capas de convolución y convolución cuentan con el tamaño de los filtros y con las *strides*. Por otro lado, la capa ConvLSTM también indica la profundidad de la celda. Todas las convoluciones utilizan *padding*. El *pooling* también va acompañado del tamaño de su filtro.

El código para esta implementación se ha obtenido del repositorio <https://github.com/rubenvillegas/iclr2017mcnet>. En esta red la tasa de aprendizaje base utilizada es del 0.0001, mientras que el tamaño del lote es de 1 al no contar con suficientes recursos computacionales como para aumentar dicho número. Por esta misma razón, el gran consumo de recursos de esta red, se han realizado únicamente 5000 *epoch* o iteraciones de entrenamiento. Al contrario de lo que ocurría en la red anterior, se ha mantenido el tamaño de las imágenes del *dataset* en 400 x 400. También ha sido necesario modificar la entrada de datos, y la red entera para que fuera capaz de operar con imágenes a color. Además, la implementación ha sido modificada para adaptarla a Python 3 y a la versión actual de Tensorflow (2.5.0).

### 5.2.3. Autoencoder con VRNN

La red propuesta por Minderer et al.[Min+19] ha sido la tercera y última elegida por la utilización de la VRNN, poco utilizada en los artículos estudiados. Además, es el único modelo de los escogidos que forma parte de las soluciones secuencia a secuencia. También ha sido elegido porque es relativamente nuevo y, aparte de estar más actualizado, hace uso de herramientas más novedosas.

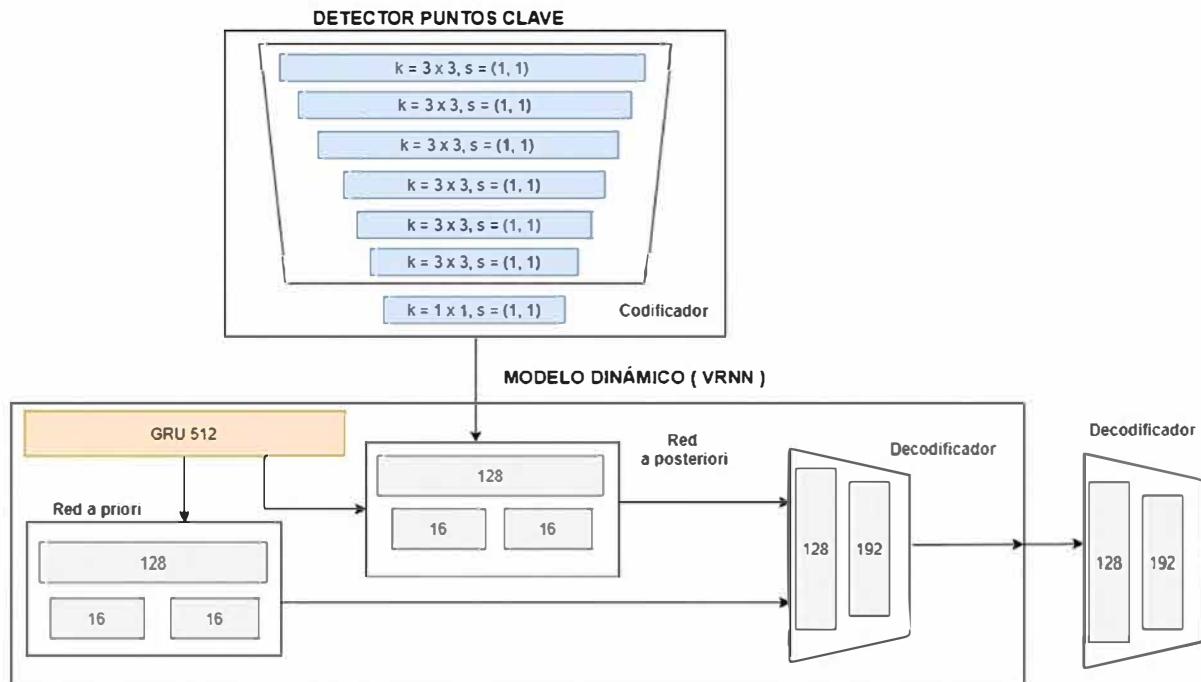


Figura 5.4: Arquitectura detallada de la AE + VRNN implementada

La idea general de la red es que en primer lugar se extraigan los puntos clave de la secuencia de imágenes de entrada a través de un codificador. Dichos puntos clave pasan al modelo dinámico donde se utilizan para predecir los puntos clave futuros.

## **Redes de predicción del crecimiento en colonias de bacterias**

---

Estos puntos clave se utilizan únicamente en el primer paso. En este modelo dinámico la red a priori calcula el conocimiento latente a priori a partir del estado anterior de la red GRU. Por otro lado, el decodificador toma como entrada el conocimiento latente y se obtienen los puntos clave. Por último, la red a posteriori calcula el conocimiento latente a posteriori a partir de los puntos clave y el estado anterior de la red GRU.

La arquitectura se puede observar en la figura 5.4, donde las capas azules son de convolución y se indica el tamaño de sus filtros y las *strides*. Las capas grises son capas densas donde el número indica sus unidades, al igual que ocurre con la GRU. Siguiendo esta imagen el detector de puntos clave recibiría la secuencia de imágenes y extraería los puntos clave que pasarían al modelo dinámico. Allí se calcula el conocimiento latente a priori a partir del estado de la RNN y el conocimiento latente a posteriori (si procede) a partir de los puntos clave y el estado anterior de la RNN. El conocimiento latente es muestreado y utilizado para obtener los puntos clave decodificándolos. Por último, estos puntos clave salen del modelo dinámico y son decodificados de nuevo para obtener la secuencia de imágenes.

Este código ha sido el más modificado de los tres, puesto que ha sido necesario cambiar completamente la entrada de datos y adaptar algunas partes del código. Ahora se emplean generadores para alimentar el *dataset* y se ha descartado el *chunking* del *dataset*. En este caso los parámetros utilizados han sido: lotes de tamaño 4, 2000 *epoch* (por problemas de recursos, dado que cada *epoch* consume una alta cantidad de tiempo), 81 pasos por *epoch*, tasa de aprendizaje del 0.001 y se toman 64 puntos clave por imagen. Además las redes convolucionales utilizan *padding*, función de activación Leaky ReLU y regularización L2. Por otro lado, las capas densas utilizan función de activación ReLU. El código base está disponible en: [https://github.com/google-research/google-research/tree/34444253e9f57cd03364bc4e50057a5abe9bcf17/video\\_structure](https://github.com/google-research/google-research/tree/34444253e9f57cd03364bc4e50057a5abe9bcf17/video_structure).

# Capítulo 6

## Resultados y conclusiones

### 6.1. Resultados de la implementación

En primer lugar, se van a evaluar los resultados del entrenamiento y validación individualmente por cada red implementada con TensorBoard.

#### 6.1.1. GAN multiescalar con pérdida GDL

Esta implementación se divide tanto en generador como discriminador, como toda GAN. Por lo tanto, se pueden observar los resultados de ambas redes por separado.

Se han tomado distintos datos, como el PSNR (Proporción Máxima de Señal a Ruido) que es una medida que valora la calidad de la reconstrucción, en este caso de una imagen.

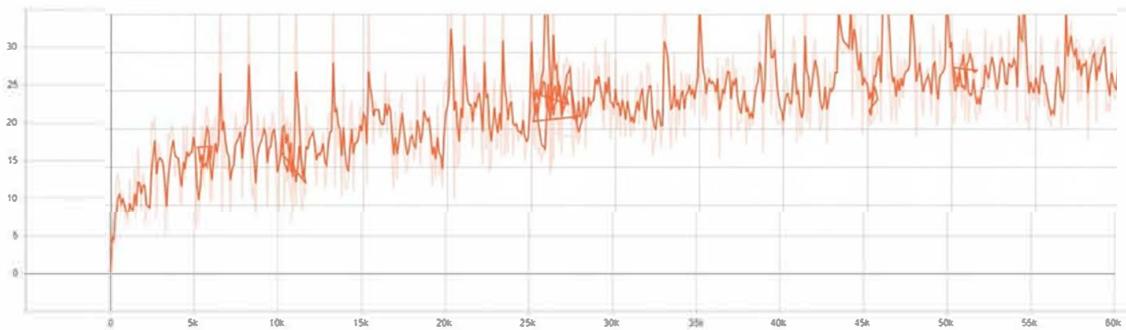


Figura 6.1: PSNR en el tiempo del entrenamiento en la red de Mathieu

Como se puede observar la calidad aumenta con el tiempo, lo que es una buena señal de que la red está aprendiendo a generar imágenes similares a las reales o las etiquetas. Por lo tanto, los resultados obtenidos del PSNR en el entrenamiento son favorables.

En cambio, en el *testing* se realizan 12 pruebas distintas representadas en los puntos de la gráfica cada 5000 *epoch*. Se puede observar como la calidad aumenta hasta llegar últimos dos puntos, en 55000 y 60000, donde decae repentinamente. Esto podría ser un signo de sobreajuste y de que la red ya no sabe generalizar a un caso que no haya visto anteriormente.

## Resultados y conclusiones

---

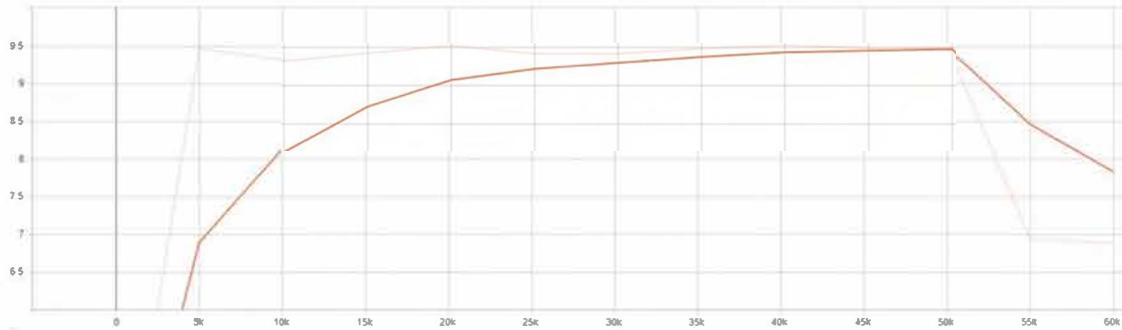


Figura 6.2: PSNR en el tiempo del *testing* en la red de Mathieu

Otra medida utilizada es la diferencia de nitidez entre la imagen predicha y la real. Esta medida también se ha utilizado tanto para entrenamiento como para pruebas.

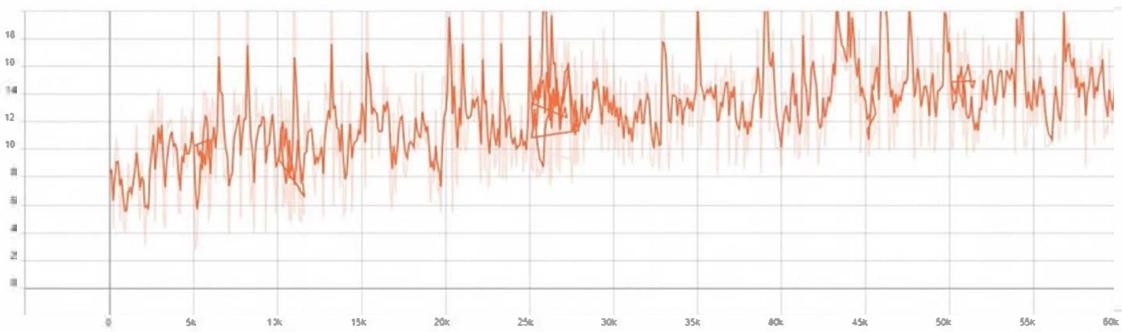


Figura 6.3: Error de nitidez en el tiempo del entrenamiento en la red de Mathieu

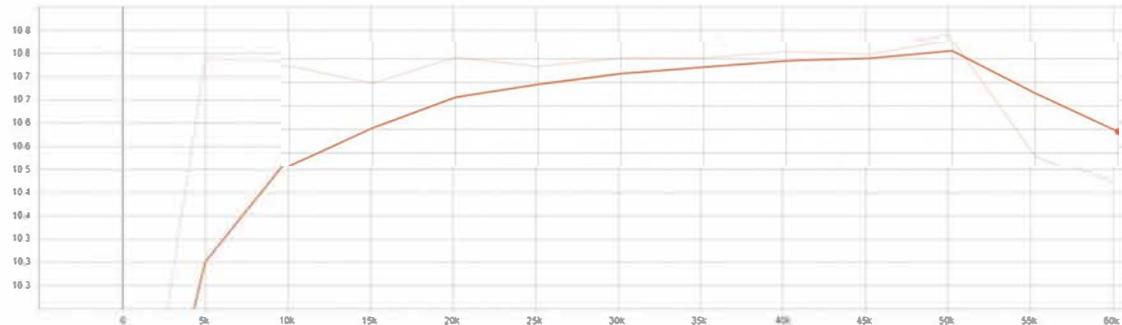


Figura 6.4: Error de nitidez en el tiempo del *testing* en la red de Mathieu

Al igual que ocurría con el PSNR, el error de nitidez parece mejorar constantemente durante el entrenamiento, mientras que en las pruebas decae bruscamente en los dos últimos puntos. Por último, se evalúa la última y más común métrica: la pérdida. Esta consiste en una función que evalúa la desviación entre los valores reales y los predichos. En esta implementación se utiliza una pérdida combinada de pérdida adversaria (con entropía cruzada binaria),  $\text{lp}(l1 \text{ o } l2)$  y GDL para el generador. El GDL es la pérdida de diferencia de gradiente explicada e implementada en el artículo.

Como es normal en la pérdida de una red neuronal, la pérdida baja hasta un punto en el que se vuelve relativamente constante, a pesar de los picos que muestra la gráfica. Esto es un buen indicativo de que la red ha sido correctamente entrenada.

## 6.1. Resultados de la implementación

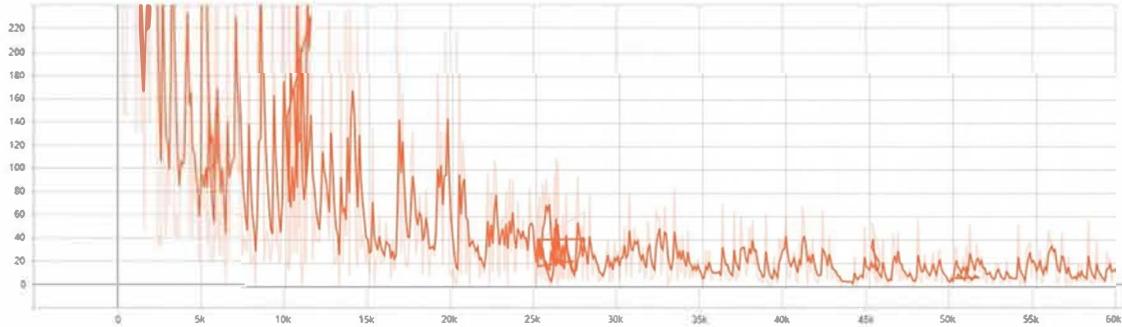


Figura 6.5: Pérdida del generador durante el entrenamiento en la red de Mathieu

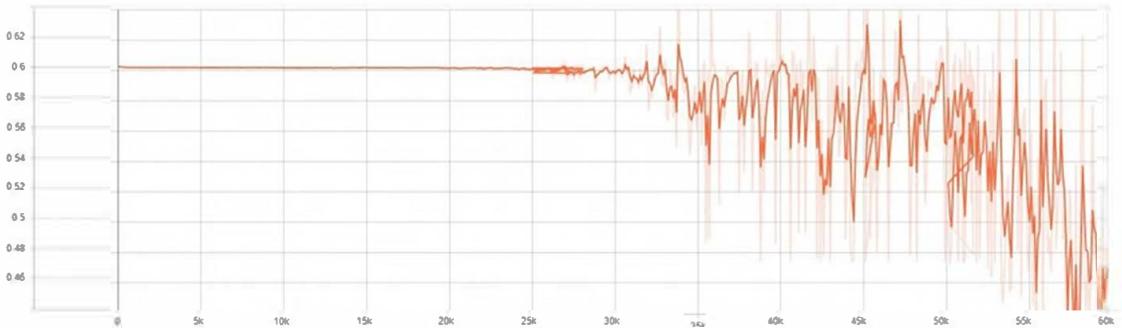


Figura 6.6: Pérdida del discriminador durante el entrenamiento en la red de Mathieu

De igual manera, cuando la pérdida del generador empieza a minimizarse la del discriminador también. Aunque en este caso no llega a ser constante y tiene cambios drásticos en forma de picos.

### 6.1.2. MCNet con conexiones residuales

Los parámetros a estudiar en este caso son todos perdidas: la pérdida lp, la pérdida del discriminador y la pérdida de la red generadora.

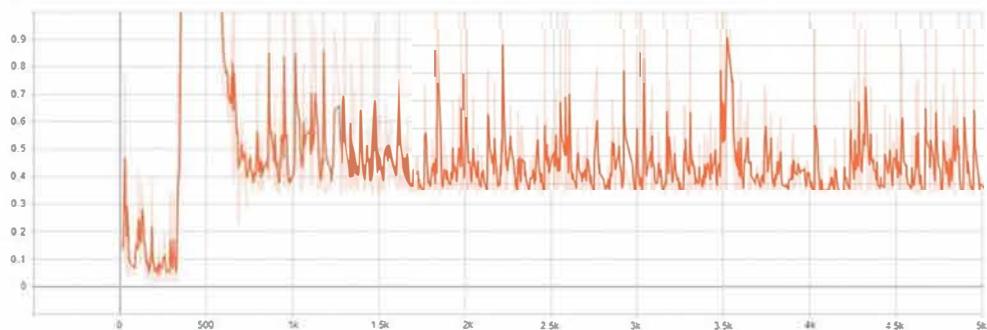


Figura 6.7: Pérdida lp durante el entrenamiento en la red de Villegas

Como se puede apreciar en la figura 6.7 la pérdida lp a penas disminuye, de hecho aumenta en los primeros *epoch* y a continuación vuelve a decrecer manteniéndose relativamente constante. Esto es un signo de que la red no está consiguiendo aprender correctamente la información.

## Resultados y conclusiones

---

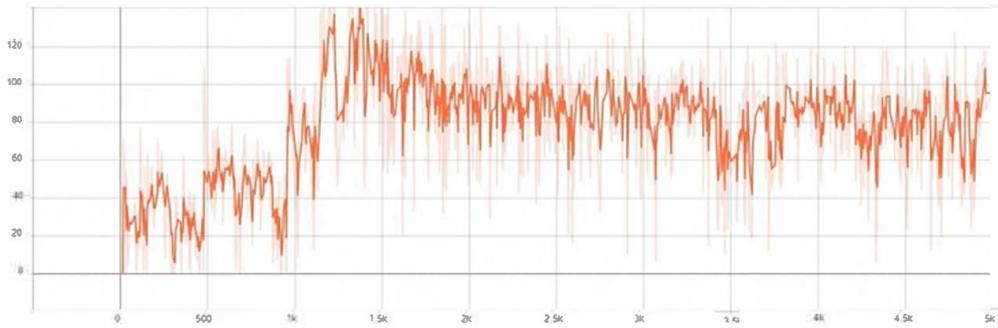


Figura 6.8: Pérdida de la red generadora durante el entrenamiento en la implementación de Villegas

Similar a lo que ocurría en la figura anterior, en la 6.8 se puede observar la pérdida de la red generadora. Esta aumenta bruscamente sobre la iteración 500 y posteriormente decrece ligeramente.

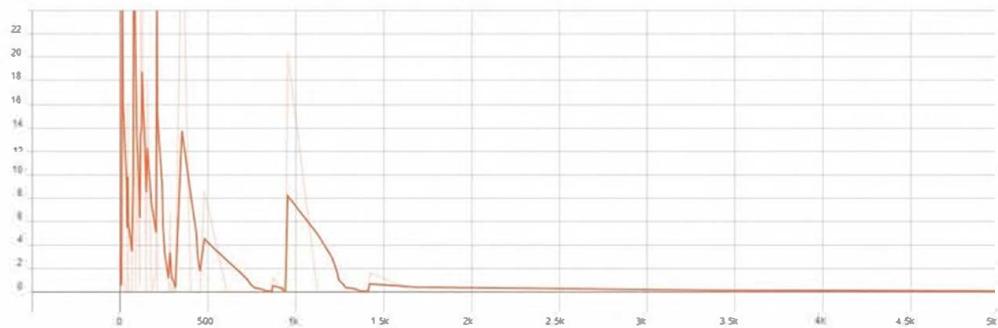


Figura 6.9: Pérdida de la red discriminadora durante el entrenamiento en la implementación de Villegas

En la última figura, al contrario de las otras dos, la pérdida si decrece correctamente por lo que podemos asegurar que el discriminador está funcionando como se esperaba. En conclusión, los resultados de esta red no son prometedores, puesto que los valores de la pérdida en el generador son demasiado altos y no decrecen. Además, modificar la tasa de aprendizaje no mejora estos resultados.

### 6.1.3. Autoencoder con VRNN

En la estructura propuesta por Minderer solo se ha tomado como métrica la pérdida durante el entrenamiento de toda la red.

Al igual que ocurría en la red anterior, no se obtienen buenos resultados. La pérdida aumenta en lugar de disminuir y no se consigue minimizarla. Esto se traduce en resultados muy alejados de los deseados y los reales.

## 6.2. Conclusiones

En este proyecto se ha llevado a cabo la búsqueda de una solución para facilitar las simulaciones del crecimiento de colonias de bacterias compuestas por diversas cepas.

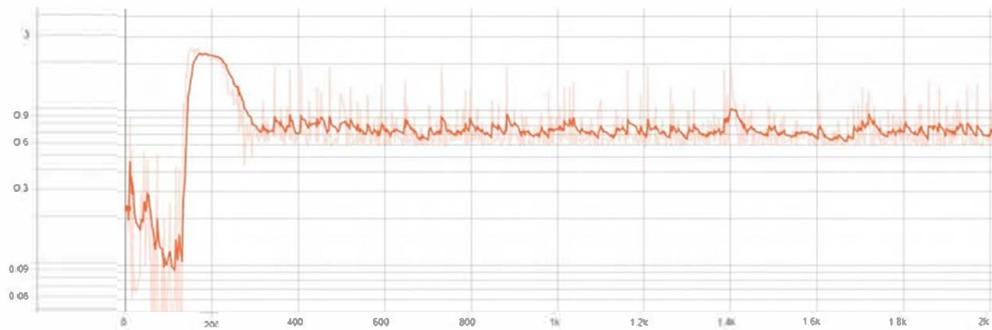


Figura 6.10: Pérdida de la red propuesta por Minderer

Estos simuladores constituyen un problema tanto a la hora de replicar correctamente la naturaleza de las bacterias, como de que dicha simulación no consuma una alta cantidad de recursos computacionales. Para ello, se ha recurrido a las técnicas de predicción de fotogramas por ser una aproximación de gran interés y generalizable a otras problemáticas, como por ejemplo a una sustitución de motor gráfico. Para llevar a cabo el proyecto se han llevado a cabo las siguientes tareas:

- La creación de un *dataset* sintético a base de imágenes extraídas del simulador GRO. Con este fin se han elegido parámetros de forma aleatoria, pero tratando de que sean equilibrados, para realizar distintas simulaciones que cubran todas las posibilidades de crecimiento. Se ha impuesto un máximo de 5 cepas de bacteria para simplificar la creación de dicho *dataset*. Dichas simulaciones por cada vídeo han sido ejecutadas a mano, mientras que los fotogramas han sido extraídos automáticamente.
- Análisis del estado del arte de las técnicas de predicción de fotogramas, desde el año 2014 hasta el actual. Para ello previamente se ha realizado un estudio del funcionamiento de las distintas redes neuronales básicas. Las técnicas de predicción de fotogramas han sido agrupadas desde dos enfoques: aquellas que reciben una secuencia de fotogramas y generan el siguiente fotograma, y aquellas que reciben una secuencia de fotogramas y general los siguientes  $t$  fotogramas.
- Elección e implementación de 3 técnicas estudiadas de forma que se abarquen distintas posibilidades y métodos. Los 3 métodos elegidos han sido la GAN multiescalar con pérdida GDL, la MCnet y el *autoencoder* con VRNN.
- Realización de pruebas sobre las redes implementadas y estudio de los resultados obtenidos, incluyendo la comparación de resultados entre estas.

De esta forma, se puede concluir que este campo del *Deep Learning* es aún demasiado inconsistente para obtener unos resultados consistentes y aplicables, como se ha podido apreciar tanto en los resultados de los artículos del estado del arte como en los propios resultados obtenidos. Aun así es una primera aproximación a un campo prometedor y con multitud de innovaciones cada año. También es importante tener en cuenta las limitaciones existentes, tanto de recursos computacionales para la realización de un entrenamiento lo suficientemente duradero como a la hora de generar los suficientes casos para que el *dataset* sea todo lo completo posible.

### **6.2.1. Trabajo futuro**

Tras los resultados obtenidos, es evidente que es necesario mejorarlos. Una posible vía de llevarlo a cabo sería obteniendo más recursos, de forma que se puedan realizar entrenamientos más exhaustivos y *datasets* más completos sin que genere problemas de memoria. Otra posibilidad es la exploración de distintas soluciones, o incluso aquellas nuevas propuestas que surgen continuamente. Como último recurso, sería interesante considerar otros enfoques, sobre todo de cara a la sustitución del motor físico que era uno de los subobjetivos del proyecto.

# Bibliografía

- [A15] Rodriguez Paton A. Prestes García; A. «Bactosim-an individual-based simulation environment for bacterial conjugation». En: *International Conference on Practical Applications of Agents and Multi-Agent Systems* (2015).
- [Ati20] R. Atienza. *Advanced Deep Learning with Tensorflow 2 and Keras*. Second. Packt Publishing Ltd., 2020. ISBN: 978-1-83882-165-4.
- [Bal87] D. H. Ballard. «Modular Learning in Neural Networks». En: *AAAI-87 Proceedings* 1 (1987), págs. 279-284.
- [BGN21] N Behrmann, J Gall y M Noroozi. «Unsupervised Video Representation Learning by Bidirectional Feature Prediction». En: WACV (2021).
- [Bra+16] B. De Brabandere et al. «Dynamic filter networks». En: *NIPS* (2016), págs. 667-675.
- [CBC19] L. Castrejon, N. Ballas y A. Courville. «Improved Conditional VRNNs for Video Prediction». En: *ICCV* (2019).
- [CP12] Y. Chen y W. Peng. «Parametric OBMC for Pixel-adaptive Temporal Prediction on Irregular Motion Sampling Grids». En: *IEEE Transactions on Circuits and Systems for Video Technology* (2012).
- [Cri+16] F. Cricri et al. «Videoladder networks». En: (2016).
- [DB17] E. Denton y V. Birodkar. «Unsupervised learning of disentangled representations from videos». En: *NIPS* (2017), págs. 4414-4423.
- [DF18] E. Denton y R. Fergus. «Stochastic Video Generation with a Learned Prior». En: *ICML* (2018).
- [Dos+15] A. Dosovitskiy et al. «Learning optical flow with convolutional networks». En: *ICCV* (2015).
- [FGL16] C. Finn, I. Goodfellow y S. Levine. «Unsupervised learning for physical interaction through video prediction». En: *NIPS* (2016), págs. 64-72.
- [Fuk80] K. Fukushima. «Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position». En: *Biological Cybernetics* 36.4 (1980), págs. 193-202.
- [GBC16] I. Goodfellow, Y. Bengio y A. Courville. *Deep Learning*. <http://www.deeplearningbook.org>. MIT Press, 2016.
- [Gér17] A. Géron. *Hands-On Machine Learning with Scikit-Learn and TensorFlow*. Ed. por Nicole Tache. First. O'Reilly Media, Inc, 2017. ISBN: 978-1-491-96229-9.
- [GML15] R. Goroshin, M. Mathieu e Y. LeCun. «Learning to linearize under uncertainty». En: *NIPS* (2015), págs. 1234-1242.
- [Goo+14] I. J. Goodfellow et al. «Generative adversarial networks». En: *Communications of the ACM* 63.11 (2014).
- [GT20] V Le Guen y N Thome. «Disentangling Physical Dynamics from Unknown Factors for Unsupervised Video Prediction». En: *CVPR* (2020).

## BIBLIOGRAFÍA

---

- [He+16] K. He et al. «Deep residual learning for image recognition». En: *CVPR* (2016).
- [He+17] K He et al. «Mask r-cnn». En: *ICCV* (2017).
- [Ho+19] Y. Ho et al. «SME-Net: Sparse Motion Estimation for Parametric Video Prediction through Reinforcement Learning». En: *ICCV* (2019).
- [Hop82] J. J. Hopfield. «Neural networks and physical systems with emergent collective computational abilities». En: *Proceedings of the National Academy of Science of the United States of America* 79.8 (1982), págs. 2554-2558.
- [HS97] S. Hochreiter y J. Schmidhuber. «Long Short-term Memory». En: *Neural Computation* 9.8 (1997), págs. 1735-1780.
- [HW19] Z. Hu y J. T. L. Wang. «A Novel Adversarial Inference Framework for Video Prediction with Action Control». En: *ICCVW* (2019).
- [Ilg+17] E. Ilg et al. «Evolution of optical flow estimation with deep networks». En: *CVPR* (2017).
- [KP19] Y. Kwon y M. Park. «Predicting Future Frames using Retrospective Cycle GAN». En: *CVPR* (2019).
- [KW14] D. P. Kingma y M. Welling. «Auto-Encoding Variational Bayes». En: *ICLR* (2014).
- [KWA15] B. Klein, L. Wolf e Y. Afek. «A dynamic convolutional layer for short range weather prediction». En: *2015 IEEE Conference on Computer Vision and Pattern Recognition* (2015), págs. 4840-4848.
- [LeC+89] Y. LeCun et al. «Backpropagation applied to handwritten zip code recognition». En: *Neural Computation* 4.1 (1989), págs. 541-551.
- [Lem12] C. Lemaréchal. «Cauchy and the Gradient Method». En: (2012).
- [LH15] M. Liang y X. Hu. «Recurrent Convolutional Neural Network for Object Recognition». En: *CVPR* (2015).
- [Li+20] S Li et al. «Video Frame Prediction by Deep Multi-branch Mask Network». En: *IEEE Transactions on Circuits and Systems for Video Technology* (2020).
- [Lia+17] X. Liang et al. «Dual Motion GAN for Future-Flow Embedded Video Prediction». En: *ICCV* (2017).
- [Liu+17] Z. Liu et al. «Video Frame Synthesis using Deep Voxel Flow». En: *ICCV* (2017), págs. 4473-4481.
- [LKC16] W. Lotter, G. Kreiman y D. Cox. «Unsupervised learning of visual structure using predictive generative networks». En: *ICLR* (2016).
- [LKC17] W. Lotter, G. Kreiman y D. Cox. «Deep predictive coding networks for video prediction and unsupervised learning». En: *ICLR* (2017), págs. 1-18.
- [Lóp19] Víctor Aarón Maestro López. «Estudio del paradigma complejidad-estabilidad de la ecología en ecosistemas microbianos mediante modelos basados en individuos». En: (2019).
- [M E17] Gutiérrez Pescarmona M. E. «A new agent-based platform for simulating multicellular biocircuits with conjugative plasmids». Tesis doct. 2017.
- [Mae19] V. A. Maestro. «Estudio del paradigma complejidad-estabilidad de la ecología en ecosistemas microbianos mediante modelos basados en individuos». Tesis de mtría. 2019.
- [Mas+11] J. Masci et al. «Stacked Convolutional Auto-Encoders for Hierarchical Feature Extraction». En: *ICANN* (2011), págs. 52-59.
- [MCL16] M. Mathieu, C. Couprie e Y. LeCun. «Deep multi-scale video prediction beyond mean square error». En: *CoRR* (2016), págs. 1-14.

- [Mem08] R. Memisevic. «Non-linear latent factor models for revealing structure in high-dimensional data». En: (2008).
- [MHN13] A. L. Maas, A. Y. Hannun y A. Y. Ng. «Rectifier Nonlinearities Improve Neural Network Acoustic Models». En: *Proceedings of the 30 th International Conference on Machine Learning* 28 (2013).
- [Min+19] M. Minderer et al. «Unsupervised Learning of Object Structure and Dynamics from Videos». En: *NeurIPS* (2019).
- [MMK14] V. Michalski, R. Memisevic y K. Konda. «Modeling deep temporal dependencies with recurrent grammar cells». En: *NIPS* (2014), págs. 1925-1933.
- [MO14] M. Mirza y S. Osindero. «Conditional Generative Adversarial Nets». En: *CoRR* (2014).
- [MP43] W. S. McCulloch y W. Pitts. «A logical calculus of the ideas immanent in nervous activity». En: *The bulletin of mathematical biophysics* 5 (1943), págs. 115-133.
- [NH10] V. Nair y G. E. Hinton. «Rectified linear units improve restricted boltzmann machines». En: *ICML'10* (2010), págs. 807-814.
- [NMW21] S Niklaus, L Mai y O Wang. «Revisiting Adaptive Convolutions for Video Frame Interpolation». En: *WACV* (2021).
- [NYD16] A. Newell, K. Yang y J. Deng. «Stacked hourglass networks for human pose estimation». En: *ECCV* (2016).
- [Oh+15] J. Oh et al. «Action-conditional videoprediction using deep networks in Atari games». En: *NIPS* (2015), págs. 2863-2871.
- [OSE18] M. Oliu, J. Selva y S. Escalera. «Folded recurrent neural networks for future video prediction». En: *ECCV* (2018), págs. 745-761.
- [OVK18] A van den Oord, O Vinyals y K Kavukcuoglu. «Neural discrete representation learning». En: *NIPS* (2018).
- [PHC16] V. Patraucean, A. Handa y R. Cipolla. «Spatio-temporal video autoencoder with differentiable memory». En: *ICLR* (2016).
- [Rak+20] R Rakhimov et al. «Latent Video Transformer». En: *VISIGRAPP* (2020).
- [Ran+14] M. Ranzato et al. «Video (language) modeling: a baseline for generative models of natural videos». En: (2014).
- [Red+18] F. A. Reda et al. «SDC-Net: Video prediction using spatially-displaced convolution». En: *ECCV* (2018).
- [Ree+15] S. Reed et al. «Deep visual analogy-making». En: *NIPS* (2015).
- [RFB15] O Ronneberger, P Fischer y T Brox. «U-Net: Convolutional Networks for Biomedical Image Segmentation». En: *MICCAI* (2015).
- [RHW86] D .E. Rumelhart, G. E. Hinton y R. J. Williams. «Learning representations by back-propagating errors». En: *Nature* 323 (1986), págs. 533-536.
- [RMC15] A. Radford, L. Metz y S. Chintala. «Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks». En: *ICLR* (2015).
- [Ros58] F. Rosenblatt. «The perceptron: A probabilistic model for information storage and organization in the brain.» En: *Psychological Review* 65.6 (1958), págs. 386-408.
- [Sha18] R. Shanmugamani. *Deep Learning for Computer Vision*. Packt Publishing, 2018. ISBN: 978-1-78829-562-8.
- [Shi+15] X. Shi et al. «Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting». En: *NIPS* (2015).

## BIBLIOGRAFÍA

---

- [Sim04] Z. Wang; A.C. Bovik; H.R. Sheikh; E.P. Simoncelli. «Image quality assessment: from error visibility to structural similarity». En: *IEEE Transactions on Image Processing* 13.4 (2004).
- [SMS15] N. Srivastava, E. Mansimov y R. Salakhutdinov. «Unsupervised learning of video representations using LSTMs». En: *ICML* (2015), págs. 1-10.
- [Sun15] K. He; X. Zhang; S. Ren; J. Sun. «Deep residual learning for image recognition». En: *CoRR* (2015).
- [Vil+17a] R. Villegas et al. «Decomposing Motion and Content for Natural Video Sequence Prediction». En: *ICLR* (2017).
- [Vil+17b] R. Villegas et al. «Learning to Generate Long-term Future via Hierarchical Prediction». En: *ICML* (2017), págs. 3560-3569.
- [VT17] C. Vondrick y A. Torralba. «Generating the Future with Adversarial Transformers». En: *CVPR* (2017), págs. 2992-3000.
- [Vuk+17] Vedran Vukotic et al. «One-Step Time-Dependent Future Video Frame Prediction with a Convolutional Encoder-Decoder Neural Network». En: *ICIAP* (2017), págs. 140-151.
- [W+18] Liu W. et al. «Future Frame Prediction for Anomaly Detection – A New Baseline». En: *CVPR* (2018).
- [Wan+17] Y. Wang et al. «PredRNN: Recurrent neural networks for predictive learning using spatiotemporal LSTMs». En: *NIPS* (2017), págs. 879-888.
- [Wan+18] Y. Wang et al. «PredRNN++: Towards a resolution of the deep-in-time dilemma in spatiotemporal predictive learning». En: *Proceedings of the 35th International Conference on Machine Learning* (2018), págs. 5123-5132.
- [Wic+18] N. Wichers et al. «Hierarchical Long-term Video Prediction without Supervision». En: *ICML* (2018).
- [WTU19] D Weissenborn, O Täckström y J Uszkoreit. «Scaling autoregressive video models». En: (2019).
- [Wu+20a] Q Wu et al. «Generating Future Frames with Mask-Guided Prediction». En: *ICME* (2020).
- [Wu+20b] Y. Wu et al. «Future Video Synthesis with Object Motion Prediction». En: *CVPR* (2020).
- [Xue+16] T. Xue et al. «Visual Dynamics: Probabilistic Future Frame Synthesis via Cross Convolutional Networks». En: *NIPS* (2016), págs. 91-99.
- [Ye+19] Y. Ye et al. «Compositional Video Prediction». En: *ICCV* (2019).
- [YK16] F. Yu y V. Koltun. «Multi-Scale Context Aggregation by Dilated Convolutions». En: *ICLR* (2016).
- [Yu+18] J Yu et al. «Generative image inpainting with contextual attention». En: *CVPR* (2018).
- [Yu+20] W Yu et al. «Efficient and Information-Preserving Future Frame Prediction and Beyond». En: *ICLR* (2020).
- [ZB16] Y. Zhou y T. L. Berg. «Learning Temporal Transformations From Time-Lapse Videos». En: *ECCV* (2016).
- [Zha+15] M. Zhao et al. «Predictive encoding of contextual relationships for perceptual inference, interpolation and prediction». En: *ICLR* (2015).
- [Zhu+17] J. Zhu et al. «Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks». En: *ICCV* (2017).
- [ZKM17] G. Zaccone, Md. Rezaul Karim y A. Menshawy. *Deep Learning with Tensorflow*. Packt Publishing, 2017. ISBN: 978-1-78646-978-6.

# Anexo

## Experimentos 2 tipos de bacteria

| cc    | Nº Experimento | Nº interacciones | Interacciones positivas | Fuerza interacciones |
|-------|----------------|------------------|-------------------------|----------------------|
| b1,b2 | 1              | 0                | 0                       | 0,1                  |
|       | 2              | 1                | 1                       | 0,4                  |
|       | 3              | 2                | 2                       | 0,3                  |
|       | 4              | 1                | 0                       | 0,5                  |
|       | 5              | 2                | 1                       | 0,4                  |
|       | 6              | 2                | 1                       | 0,3                  |
|       | 7              | 1                | 1                       | 0,5                  |
|       | 8              | 2                | 1                       | 0,2                  |
|       | 9              | 2                | 0                       | 0,5                  |
|       | 10             | 2                | 2                       | 0,1                  |
| b1,b3 | 1              | 2                | 1                       | 0,2                  |
|       | 2              | 2                | 2                       | 0,3                  |
|       | 3              | 2                | 0                       | 0,2                  |
|       | 4              | 1                | 1                       | 0,1                  |
|       | 5              | 1                | 0                       | 0,3                  |
|       | 6              | 0                | 0                       | 0,2                  |
|       | 7              | 1                | 0                       | 0,4                  |
|       | 8              | 1                | 1                       | 0,3                  |
|       | 9              | 1                | 0                       | 0,1                  |
|       | 10             | 2                | 1                       | 0,5                  |
| b1,b4 | 1              | 0                | 0                       | 0,2                  |
|       | 2              | 1                | 1                       | 0,3                  |
|       | 3              | 2                | 1                       | 0,3                  |
|       | 4              | 1                | 0                       | 0,2                  |
|       | 5              | 1                | 0                       | 0,5                  |
|       | 6              | 2                | 0                       | 0,2                  |
|       | 7              | 1                | 1                       | 0,1                  |
|       | 8              | 1                | 1                       | 0,5                  |
|       | 9              | 2                | 0                       | 0,3                  |
|       | 10             | 2                | 2                       | 0,3                  |
| b1,b5 | 1              | 2                | 2                       | 0,2                  |
|       | 2              | 1                | 0                       | 0,2                  |
|       | 3              | 1                | 1                       | 0,3                  |
|       | 4              | 0                | 0                       | 0,2                  |
|       | 5              | 1                | 1                       | 0,2                  |
|       | 6              | 1                | 0                       | 0,3                  |
|       | 7              | 2                | 1                       | 0,1                  |
|       | 8              | 1                | 1                       | 0,1                  |
|       | 9              | 2                | 0                       | 0,3                  |
|       | 10             | 2                | 2                       | 0,5                  |

Anexo

---

| cc    | Nº Experimento | Nº interacciones | Interacciones positivas | Fuerza interacciones |
|-------|----------------|------------------|-------------------------|----------------------|
| b2,b3 | 1              | 1                | 0                       | 0,4                  |
|       | 2              | 0                | 0                       | 0,5                  |
|       | 3              | 1                | 1                       | 0,1                  |
|       | 4              | 2                | 0                       | 0,2                  |
|       | 5              | 2                | 2                       | 0,3                  |
|       | 6              | 1                | 0                       | 0,5                  |
|       | 7              | 1                | 0                       | 0,3                  |
|       | 8              | 2                | 2                       | 0,4                  |
|       | 9              | 2                | 1                       | 0,4                  |
|       | 10             | 2                | 0                       | 0,5                  |
| b2,b4 | 1              | 1                | 0                       | 0,3                  |
|       | 2              | 0                | 0                       | 0,5                  |
|       | 3              | 2                | 1                       | 0,4                  |
|       | 4              | 2                | 0                       | 0,5                  |
|       | 5              | 2                | 2                       | 0,3                  |
|       | 6              | 1                | 1                       | 0,3                  |
|       | 7              | 1                | 0                       | 0,4                  |
|       | 8              | 1                | 0                       | 0,1                  |
|       | 9              | 2                | 0                       | 0,2                  |
|       | 10             | 1                | 1                       | 0,4                  |
| b2,b5 | 1              | 0                | 0                       | 0,2                  |
|       | 2              | 1                | 0                       | 0,2                  |
|       | 3              | 2                | 0                       | 0,5                  |
|       | 4              | 2                | 1                       | 0,2                  |
|       | 5              | 2                | 2                       | 0,4                  |
|       | 6              | 2                | 2                       | 0,3                  |
|       | 7              | 1                | 0                       | 0,1                  |
|       | 8              | 2                | 0                       | 0,2                  |
|       | 9              | 1                | 1                       | 0,3                  |
|       | 10             | 1                | 0                       | 0,5                  |
| b3,b4 | 1              | 2                | 2                       | 0,5                  |
|       | 2              | 0                | 0                       | 0,1                  |
|       | 3              | 1                | 1                       | 0,3                  |
|       | 4              | 2                | 1                       | 0,2                  |
|       | 5              | 1                | 0                       | 0,4                  |
|       | 6              | 2                | 2                       | 0,3                  |
|       | 7              | 1                | 1                       | 0,4                  |
|       | 8              | 2                | 0                       | 0,4                  |
|       | 9              | 2                | 1                       | 0,3                  |
|       | 10             | 1                | 0                       | 0,3                  |
| b3,b5 | 1              | 1                | 0                       | 0,3                  |
|       | 2              | 2                | 2                       | 0,3                  |
|       | 3              | 2                | 2                       | 0,5                  |
|       | 4              | 2                | 1                       | 0,3                  |
|       | 5              | 1                | 1                       | 0,5                  |
|       | 6              | 2                | 0                       | 0,5                  |
|       | 7              | 0                | 0                       | 0,2                  |
|       | 8              | 2                | 1                       | 0,5                  |
|       | 9              | 1                | 0                       | 0,1                  |
|       | 10             | 1                | 1                       | 0,3                  |

## Experimentos 3 tipos de bacteria

| cc       | Nº Experimento | Nº interacciones | Interacciones positivas | Fuerza interacciones |
|----------|----------------|------------------|-------------------------|----------------------|
| b1,b2,b3 | 1              | 3                | 3                       | 0,1                  |
|          | 2              | 0                | 0                       | 0,1                  |
|          | 3              | 4                | 1                       | 0,1                  |
|          | 4              | 6                | 6                       | 0,3                  |
|          | 5              | 2                | 2                       | 0,3                  |
|          | 6              | 2                | 0                       | 0,2                  |
|          | 7              | 1                | 0                       | 0,5                  |
|          | 8              | 3                | 2                       | 0,5                  |
|          | 9              | 4                | 0                       | 0,4                  |
|          | 10             | 5                | 1                       | 0,2                  |
| b1,b2,b4 | 1              | 4                | 2                       | 0,4                  |
|          | 2              | 1                | 0                       | 0,4                  |
|          | 3              | 3                | 1                       | 0,2                  |
|          | 4              | 4                | 1                       | 0,3                  |
|          | 5              | 6                | 6                       | 0,2                  |
|          | 6              | 6                | 1                       | 0,4                  |
|          | 7              | 5                | 0                       | 0,2                  |
|          | 8              | 2                | 1                       | 0,4                  |
|          | 9              | 5                | 1                       | 0,2                  |
|          | 10             | 0                | 0                       | 0,2                  |
| b1,b2,b5 | 1              | 3                | 0                       | 0,4                  |
|          | 2              | 0                | 0                       | 0,1                  |
|          | 3              | 3                | 1                       | 0,2                  |
|          | 4              | 6                | 2                       | 0,5                  |
|          | 5              | 4                | 4                       | 0,3                  |
|          | 6              | 4                | 0                       | 0,3                  |
|          | 7              | 5                | 2                       | 0,4                  |
|          | 8              | 6                | 1                       | 0,4                  |
|          | 9              | 1                | 1                       | 0,3                  |
|          | 10             | 2                | 2                       | 0,2                  |
| b1,b3,b4 | 1              | 5                | 3                       | 0,4                  |
|          | 2              | 0                | 0                       | 0,1                  |
|          | 3              | 1                | 0                       | 0,5                  |
|          | 4              | 4                | 0                       | 0,4                  |
|          | 5              | 1                | 1                       | 0,4                  |
|          | 6              | 2                | 2                       | 0,4                  |
|          | 7              | 4                | 3                       | 0,3                  |
|          | 8              | 5                | 4                       | 0,4                  |
|          | 9              | 3                | 0                       | 0,3                  |
|          | 10             | 6                | 1                       | 0,2                  |
| b1,b3,b5 | 1              | 4                | 0                       | 0,4                  |
|          | 2              | 3                | 1                       | 0,2                  |
|          | 3              | 6                | 2                       | 0,4                  |
|          | 4              | 5                | 1                       | 0,2                  |
|          | 5              | 4                | 1                       | 0,3                  |
|          | 6              | 4                | 3                       | 0,3                  |
|          | 7              | 2                | 2                       | 0,4                  |
|          | 8              | 0                | 0                       | 0,5                  |
|          | 9              | 4                | 0                       | 0,5                  |
|          | 10             | 1                | 0                       | 0,4                  |
| b1,b4,b5 | 1              | 4                | 4                       | 0,5                  |
|          | 2              | 1                | 0                       | 0,5                  |
|          | 3              | 3                | 1                       | 0,4                  |
|          | 4              | 3                | 0                       | 0,4                  |
|          | 5              | 0                | 0                       | 0,4                  |
|          | 6              | 4                | 3                       | 0,3                  |
|          | 7              | 2                | 2                       | 0,4                  |
|          | 8              | 5                | 3                       | 0,3                  |
|          | 9              | 6                | 4                       | 0,3                  |
|          | 10             | 2                | 1                       | 0,3                  |

## Anexo

---

| cc       | Nº Experimento | Nº interacciones | Interacciones positivas | Fuerza interacciones |
|----------|----------------|------------------|-------------------------|----------------------|
| b2,b3,b4 | 1              | 5                | 4                       | 0,3                  |
|          | 2              | 2                | 1                       | 0,4                  |
|          | 3              | 2                | 2                       | 0,2                  |
|          | 4              | 3                | 3                       | 0,3                  |
|          | 5              | 6                | 0                       | 0,2                  |
|          | 6              | 1                | 1                       | 0,3                  |
|          | 7              | 6                | 6                       | 0,3                  |
|          | 8              | 3                | 2                       | 0,2                  |
|          | 9              | 0                | 0                       | 0,2                  |
|          | 10             | 4                | 0                       | 0,3                  |
| b2,b3,b5 | 1              | 2                | 1                       | 0,3                  |
|          | 2              | 0                | 0                       | 0,3                  |
|          | 3              | 6                | 2                       | 0,1                  |
|          | 4              | 6                | 1                       | 0,1                  |
|          | 5              | 1                | 1                       | 0,3                  |
|          | 6              | 4                | 3                       | 0,4                  |
|          | 7              | 5                | 5                       | 0,3                  |
|          | 8              | 3                | 3                       | 0,5                  |
|          | 9              | 2                | 0                       | 0,3                  |
|          | 10             | 5                | 1                       | 0,4                  |
| b2,b4,b5 | 1              | 6                | 2                       | 0,2                  |
|          | 2              | 4                | 0                       | 0,4                  |
|          | 3              | 0                | 0                       | 0,1                  |
|          | 4              | 4                | 1                       | 0,2                  |
|          | 5              | 6                | 0                       | 0,5                  |
|          | 6              | 5                | 2                       | 0,4                  |
|          | 7              | 2                | 0                       | 0,1                  |
|          | 8              | 5                | 4                       | 0,4                  |
|          | 9              | 1                | 0                       | 0,5                  |
|          | 10             | 3                | 3                       | 0,3                  |
| b3,b4,b5 | 1              | 1                | 0                       | 0,1                  |
|          | 2              | 4                | 4                       | 0,2                  |
|          | 3              | 5                | 1                       | 0,1                  |
|          | 4              | 4                | 3                       | 0,3                  |
|          | 5              | 1                | 1                       | 0,4                  |
|          | 6              | 5                | 5                       | 0,4                  |
|          | 7              | 0                | 0                       | 0,5                  |
|          | 8              | 2                | 2                       | 0,5                  |
|          | 9              | 3                | 3                       | 0,4                  |
|          | 10             | 6                | 1                       | 0,4                  |

## Experimentos 4 tipos de bacteria

| cc          | Nº Experimento | Nº interacciones | Interacciones positivas | Fuerza interacciones |
|-------------|----------------|------------------|-------------------------|----------------------|
| b1,b2,b3,b4 | 1              | 7                | 3                       | 0.4                  |
|             | 2              | 9                | 5                       | 0.3                  |
|             | 3              | 8                | 8                       | 0.4                  |
|             | 4              | 0                | 0                       | 0.3                  |
|             | 5              | 7                | 4                       | 0.4                  |
|             | 6              | 1                | 0                       | 0.5                  |
|             | 7              | 8                | 3                       | 0.2                  |
|             | 8              | 3                | 1                       | 0.1                  |
|             | 9              | 12               | 11                      | 0.1                  |
|             | 10             | 9                | 4                       | 0.2                  |
|             | 11             | 1                | 0                       | 0.2                  |
|             | 12             | 6                | 1                       | 0.4                  |
|             | 13             | 2                | 1                       | 0.5                  |
|             | 14             | 4                | 1                       | 0.2                  |
|             | 15             | 9                | 7                       | 0.1                  |
|             | 16             | 6                | 3                       | 0.4                  |
|             | 17             | 2                | 1                       | 0.2                  |
|             | 18             | 10               | 0                       | 0.4                  |
|             | 19             | 7                | 0                       | 0.2                  |
|             | 20             | 11               | 5                       | 0.5                  |
|             | 21             | 1                | 1                       | 0.2                  |
|             | 22             | 8                | 7                       | 0.3                  |
|             | 23             | 4                | 3                       | 0.2                  |
|             | 24             | 10               | 4                       | 0.3                  |
|             | 25             | 5                | 3                       | 0.4                  |
| b1,b2,b3,b5 | 1              | 7                | 1                       | 0.2                  |
|             | 2              | 0                | 0                       | 0.1                  |
|             | 3              | 9                | 8                       | 0.3                  |
|             | 4              | 6                | 1                       | 0.5                  |
|             | 5              | 10               | 2                       | 0.4                  |
|             | 6              | 2                | 2                       | 0.5                  |
|             | 7              | 4                | 2                       | 0.3                  |
|             | 8              | 12               | 12                      | 0.4                  |
|             | 9              | 8                | 6                       | 0.5                  |
|             | 10             | 12               | 2                       | 0.5                  |
|             | 11             | 1                | 0                       | 0.4                  |
|             | 12             | 10               | 10                      | 0.2                  |
|             | 13             | 3                | 2                       | 0.4                  |
|             | 14             | 5                | 3                       | 0.5                  |
|             | 15             | 8                | 7                       | 0.3                  |
|             | 16             | 3                | 1                       | 0.4                  |
|             | 17             | 11               | 2                       | 0.3                  |
|             | 18             | 3                | 0                       | 0.4                  |
|             | 19             | 11               | 11                      | 0.4                  |
|             | 20             | 8                | 0                       | 0.4                  |
|             | 21             | 6                | 3                       | 0.3                  |
|             | 22             | 4                | 3                       | 0.1                  |
|             | 23             | 9                | 3                       | 0.3                  |
|             | 24             | 7                | 7                       | 0.4                  |
|             | 25             | 8                | 3                       | 0.3                  |

## Anexo

---

| cc          | Nº Experimento | Nº interacciones | Interacciones positivas | Fuerza interacciones |
|-------------|----------------|------------------|-------------------------|----------------------|
| b1,b3,b4,b5 | 1              | 10               | 1                       | 0.4                  |
|             | 2              | 3                | 0                       | 0.3                  |
|             | 3              | 11               | 8                       | 0.2                  |
|             | 4              | 0                | 0                       | 0.2                  |
|             | 5              | 4                | 1                       | 0.3                  |
|             | 6              | 5                | 2                       | 0.1                  |
|             | 7              | 1                | 0                       | 0.4                  |
|             | 8              | 9                | 1                       | 0.3                  |
|             | 9              | 2                | 1                       | 0.3                  |
|             | 10             | 11               | 0                       | 0.2                  |
|             | 11             | 5                | 3                       | 0.2                  |
|             | 12             | 4                | 3                       | 0.4                  |
|             | 13             | 11               | 6                       | 0.2                  |
|             | 14             | 12               | 12                      | 0.1                  |
|             | 15             | 10               | 3                       | 0.3                  |
|             | 16             | 11               | 4                       | 0.3                  |
|             | 17             | 3                | 2                       | 0.3                  |
|             | 18             | 3                | 3                       | 0.2                  |
|             | 19             | 5                | 4                       | 0.4                  |
|             | 20             | 8                | 2                       | 0.4                  |
|             | 21             | 6                | 6                       | 0.5                  |
|             | 22             | 7                | 0                       | 0.4                  |
|             | 23             | 1                | 0                       | 0.3                  |
|             | 24             | 2                | 0                       | 0.3                  |
|             | 25             | 12               | 9                       | 0.3                  |
| b2,b3,b4,b5 | 1              | 5                | 1                       | 0.4                  |
|             | 2              | 2                | 2                       | 0.4                  |
|             | 3              | 4                | 1                       | 0.3                  |
|             | 4              | 8                | 1                       | 0.5                  |
|             | 5              | 3                | 3                       | 0.1                  |
|             | 6              | 12               | 9                       | 0.3                  |
|             | 7              | 12               | 8                       | 0.5                  |
|             | 8              | 7                | 1                       | 0.2                  |
|             | 9              | 0                | 0                       | 0.2                  |
|             | 10             | 8                | 7                       | 0.1                  |
|             | 11             | 5                | 3                       | 0.4                  |
|             | 12             | 6                | 4                       | 0.1                  |
|             | 13             | 10               | 0                       | 0.4                  |
|             | 14             | 12               | 11                      | 0.1                  |
|             | 15             | 4                | 3                       | 0.2                  |
|             | 16             | 12               | 2                       | 0.4                  |
|             | 17             | 11               | 0                       | 0.3                  |
|             | 18             | 9                | 1                       | 0.2                  |
|             | 19             | 8                | 2                       | 0.1                  |
|             | 20             | 6                | 2                       | 0.3                  |
|             | 21             | 3                | 0                       | 0.2                  |
|             | 22             | 1                | 0                       | 0.4                  |
|             | 23             | 4                | 4                       | 0.2                  |
|             | 24             | 10               | 1                       | 0.3                  |
|             | 25             | 1                | 1                       | 0.3                  |

## Experimentos 5 tipos de bacteria

| cc             | Nº Experimento | Nº interacciones | Interacciones positivas | Fuerza interacciones |
|----------------|----------------|------------------|-------------------------|----------------------|
| b1.b2.b3.b4.b5 | 1              | 18               | 4                       | 0.4                  |
|                | 2              | 3                | 3                       | 0.3                  |
|                | 3              | 7                | 5                       | 0.1                  |
|                | 4              | 3                | 0                       | 0.2                  |
|                | 5              | 10               | 8                       | 0.2                  |
|                | 6              | 10               | 9                       | 0.1                  |
|                | 7              | 3                | 3                       | 0.4                  |
|                | 8              | 16               | 8                       | 0.4                  |
|                | 9              | 12               | 3                       | 0.1                  |
|                | 10             | 0                | 0                       | 0.3                  |
|                | 11             | 3                | 3                       | 0.2                  |
|                | 12             | 13               | 13                      | 0.5                  |
|                | 13             | 5                | 0                       | 0.4                  |
|                | 14             | 4                | 0                       | 0.4                  |
|                | 15             | 9                | 3                       | 0.4                  |
|                | 16             | 19               | 18                      | 0.3                  |
|                | 17             | 20               | 16                      | 0.2                  |
|                | 18             | 9                | 6                       | 0.4                  |
|                | 19             | 10               | 5                       | 0.2                  |
|                | 20             | 2                | 0                       | 0.4                  |
|                | 21             | 11               | 0                       | 0.3                  |
|                | 22             | 17               | 17                      | 0.1                  |
|                | 23             | 8                | 3                       | 0.3                  |
|                | 24             | 16               | 1                       | 0.4                  |
|                | 25             | 10               | 4                       | 0.2                  |
|                | 26             | 17               | 14                      | 0.2                  |
|                | 27             | 8                | 1                       | 0.1                  |
|                | 28             | 8                | 3                       | 0.4                  |
|                | 29             | 15               | 8                       | 0.2                  |
|                | 30             | 2                | 1                       | 0.2                  |
|                | 31             | 11               | 7                       | 0.2                  |
|                | 32             | 8                | 6                       | 0.3                  |
|                | 33             | 5                | 3                       | 0.2                  |
|                | 34             | 7                | 6                       | 0.2                  |
|                | 35             | 7                | 1                       | 0.4                  |
|                | 36             | 16               | 1                       | 0.3                  |
|                | 37             | 11               | 2                       | 0.2                  |
|                | 38             | 2                | 1                       | 0.3                  |
|                | 39             | 13               | 5                       | 0.5                  |
|                | 40             | 12               | 10                      | 0.3                  |
|                | 41             | 7                | 1                       | 0.2                  |
|                | 42             | 7                | 6                       | 0.5                  |
|                | 43             | 6                | 3                       | 0.4                  |
|                | 44             | 4                | 1                       | 0.1                  |
|                | 45             | 9                | 0                       | 0.3                  |
|                | 46             | 16               | 2                       | 0.2                  |
|                | 47             | 6                | 4                       | 0.4                  |
|                | 48             | 4                | 2                       | 0.2                  |
|                | 49             | 16               | 13                      | 0.5                  |
|                | 50             | 10               | 7                       | 0.4                  |
|                | 51             | 1                | 1                       | 0.3                  |
|                | 52             | 4                | 3                       | 0.3                  |
|                | 53             | 19               | 13                      | 0.2                  |
|                | 54             | 8                | 7                       | 0.4                  |
|                | 55             | 19               | 18                      | 0.2                  |
|                | 56             | 20               | 13                      | 0.3                  |
|                | 57             | 11               | 8                       | 0.2                  |
|                | 58             | 2                | 2                       | 0.2                  |
|                | 59             | 9                | 1                       | 0.4                  |
|                | 60             | 11               | 6                       | 0.2                  |

## Experimentos 5 tipos de bacteria

| cc             | Nº Experimento | Nº interacciones | Interacciones positivas | Fuerza interacciones |
|----------------|----------------|------------------|-------------------------|----------------------|
| b1.b2.b3.b4.b5 | 61             | 13               | 6                       | 0.4                  |
|                | 62             | 14               | 7                       | 0.1                  |
|                | 63             | 7                | 7                       | 0.4                  |
|                | 64             | 16               | 9                       | 0.2                  |
|                | 65             | 9                | 4                       | 0.3                  |
|                | 66             | 19               | 3                       | 0.3                  |
|                | 67             | 4                | 4                       | 0.3                  |
|                | 68             | 2                | 1                       | 0.5                  |
|                | 69             | 17               | 15                      | 0.4                  |
|                | 70             | 3                | 2                       | 0.3                  |
|                | 71             | 8                | 2                       | 0.3                  |
|                | 72             | 19               | 13                      | 0.3                  |
|                | 73             | 4                | 4                       | 0.1                  |
|                | 74             | 11               | 10                      | 0.2                  |
|                | 75             | 20               | 19                      | 0.3                  |
|                | 76             | 18               | 7                       | 0.4                  |
|                | 77             | 8                | 0                       | 0.4                  |
|                | 78             | 8                | 3                       | 0.2                  |
|                | 79             | 18               | 7                       | 0.3                  |
|                | 80             | 9                | 1                       | 0.3                  |
|                | 81             | 5                | 4                       | 0.3                  |
|                | 82             | 17               | 12                      | 0.4                  |
|                | 83             | 4                | 4                       | 0.4                  |
|                | 84             | 11               | 10                      | 0.4                  |
|                | 85             | 20               | 2                       | 0.3                  |
|                | 86             | 16               | 15                      | 0.4                  |
|                | 87             | 19               | 6                       | 0.2                  |
|                | 88             | 11               | 4                       | 0.3                  |
|                | 89             | 1                | 0                       | 0.2                  |
|                | 90             | 11               | 8                       | 0.1                  |
|                | 91             | 2                | 2                       | 0.5                  |
|                | 92             | 15               | 7                       | 0.3                  |
|                | 93             | 12               | 6                       | 0.2                  |
|                | 94             | 8                | 1                       | 0.4                  |
|                | 95             | 15               | 10                      | 0.4                  |
|                | 96             | 7                | 4                       | 0.4                  |
|                | 97             | 19               | 4                       | 0.2                  |
|                | 98             | 12               | 8                       | 0.2                  |
|                | 99             | 3                | 2                       | 0.4                  |
|                | 100            | 8                | 2                       | 0.2                  |

Los archivos .gro y el dataset pueden encontrarse en <https://app.box.com/s/jry72gx43lbg125gn3w>