Escuela Superior
de Ingeniería y Tecnología
Universidad de La Laguna

Trabajo de Fin de Grado

Grado en Ingeniería Informática

Deep Learning en Videojuegos: Superresolución

Deep Learning in Video Games: Super-Resolution

Imar Abreu Díaz

La Laguna, *10* de *septiembre* de 2019

D. **Jesús Miguel Torres Jorge**, con N.I.F. 43.826.207-Y Profesor Contratado Doctor adscrito al Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna, como tutor

D. **José Demetrio Piñeiro Vera**, con N.I.F. 43.774.048-B Profesor Titular de Universidad adscrito al Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna, como cotutor

C E R T I F I C A (N)

Que la presente memoria titulada:

"Deep Learning en Videojuegos: Superresolución"

ha sido realizada bajo su dirección por D. **Imar Abreu Díaz**,

con N.I.F. 78.648.660-E.

Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firman la presente en La Laguna a 10 de septiembre de 2019

Agradecimientos

Quisiera agradecer en primer lugar a mi familia y amigos por apoyarme incondicionalmente durante todos estos años, tanto en los buenos como en los malos momentos.

A mis compañeros con los que he compartido estos años y que han hecho el camino más ameno.

Y, por último, pero no menos importante, a los profesores que han dedicado su tiempo en mi formación. En especial a mi tutor, Jesús Miguel Torres Jorge, y cotutor, José Demetrio Piñeiro Vera, por hacer este trabajo posible.

Licencia

Resumen

En la actualidad, una gran parte de la población mundial consume videojuegos, ya sea mediante contenido en streaming o jugando directamente. El tipo de juego más consumido son los de tipo competitivo, debido al gran impacto que han tenido los eSports a nivel mundial. Esto ha producido que los jugadores se preocupen cada vez más por tener un mejor ordenador, consola o smartphone; para poder tener un mejor rendimiento y mejorar como jugadores. Por otra parte, hay jugadores que lo que buscan es disfrutar visualmente de los videojuegos, jugando a altas resoluciones y con un alto nivel de detalle gráfico. Para poder satisfacer estos requisitos las empresas han empezado a buscar en la Inteligencia Artificial, en concreto en las técnicas de deep learning, un aliado para mejorar el rendimiento y la calidad de los videojuegos. En este informe se abordará uno de los métodos que recientemente se ha empezado a implementar en los videojuegos, la Superresolución. Esta técnica de procesamiento de imágenes mediante deep learning permite aumentar la resolución de una imagen sin pérdida de información.

Palabras clave: Videojuegos, Superresolución, Inteligencia Artificial (AI), Redes Neuronales, Redes Neuronales Convolucionales (CNN), Deep Learning, Python, TensorFlow, Keras, Nvidia, DLSS

Abstract

Nowadays, a large part of the world's population consumes video games, whether via streaming or playing. The most consumed kind of game is those of a competitive type, due to the great impact that eSports have had worldwide. This has caused players to become increasingly concerned about having a better computer, console or smartphone; to be able to perform better and improve as a player. On the other hand, there are players who are looking to visually enjoy video games, playing at high resolutions and with a high level of graphic detail. In order to meet these requirements, companies have started looking at Artificial Intelligence, specifically deep learning techniques, an ally to improve the performance and quality of video games. This report will address one of the features that has recently begun to be implemented in video games, Super-Resolution. This deep learning image processing technique allows you to increase the resolution of an image without loss of information.

Keywords: Video Games, Super-resolution, Artificial Intelligence (AI), Neural Network, Convolutional Neural Network (CNN), Deep Learning, Python, TensorFlow, Keras, Nvidia, DLSS

Índice general

Capítulo 1 Antecedentes y estado del arte 9
1.1 Videojuegos 9
1.2 Inteligencia artificial en los videojuegos 11
Capítulo 2 Motivación y Objetivos 13
2.1 Motivación 13
2.2 Objetivos principales 13
2.3 Objetivos específicos 13
Capítulo 3 Desarrollo del proyecto 14
3.1 Entorno de desarrollo y herramientas utilizadas 14
3.2 Elección del modelo 14
3.3 Generación del dataset 17
3.4 Pre-procesado del dataset 17
3.5 Parámetros de entrenamiento 19
Capítulo 4 Resultados y evaluación del modelo 20
4.1 Resultados con factor de escalado x2 20
4.2 Resultados con factor de escalado x4 22
4.3 Comparativa con métodos tradicionales 25
4.4 Pruebas con imágenes fuera de contexto 27
Capítulo 5 Conclusiones y líneas futuras 30
Capítulo 6 Summary and Conclusions 31
Apéndice A 32
Repositorio del proyecto 32
Bibliografía 33

Índice de figura

Figura 1: Datos de audiencia de los eventos de eSports a nivel mundial según Newzoo Research [3] 10
Figura 2: Datos económico de los eSports a nivel mundial según Newzoo Research 10
Figura 3: Comparativa de imagen de textura remasterizada mediante deep learning [11] 12
Figura 4: Comparativa de resolución 4k con TAA y 4k generada con DLSS .12
Figura 5: Esquema de un modelo de una GAN.....15
Figura 6: Representación de un bloque residual 15
Figura 7: Esquema del modelo de ResNet 16
Figura 8: Muestra de imágenes del dataset 17
Figura 9: Muestra de las subimágenes generadas 18
Figura 10: Gráfico que representa la precisión del modelo x2 20
Figura 11: Gráfico que representa la pérdida del modelo x2 21
Figura 12: Demostración de la generación de imágenes del modelo x2.....22
Figura 13: Gráfico que representa la precisión del modelo x4 22
Figura 14: Gráfico que representa la pérdida del modelo x4 23
Figura 15: Zoom sobre imagen x4 24
Figura 16: Demostración de reescalado con factor x4 24
Figura 17: Comparativa métodos tradicionales 1.....25
Figura 18: Comparativa métodos tradicionales 2.....25
Figura 19: Comparativa métodos tradicionales 3.....26
Figura 20: Zoom diferencia.....26
Figura 21: Pruebas con imágenes fuera de contexto 1 27
Figura 22: Zoom Forza Horizon 4.....27
Figura 23: Pruebas con imágenes fuera de contexto 2 28
Figura 24: Pruebas con imágenes fuera de contexto 3 28
Figura 25: Pruebas con imágenes fuera de contexto 4 29
Figura 26: Imagen con ruido 29

Capítulo 1 Antecedentes y estado del arte

1.1 Videojuegos

Durante las últimas décadas la industria de los videojuegos ha crecido sin descanso hasta alcanzar cifras que han hecho que se incluya a los videojuegos como un nuevo arte moderno de entretenimiento, comparable al cine o la música. Esto ha sido propiciado por varios factores:

El primer factor, y más importante, es la llegada de internet a nivel mundial. Con la aparición de las redes sociales, plataformas de streaming y revistas digitales, los videojuegos han conseguido establecerse como un producto de consumo más allá de ser simplemente un juego. Convirtiendo a toda la industria de los videojuegos en un mercado accesible por cualquier persona con acceso a internet. Haciendo que los eventos y convenciones de videojuegos, como el E3 o la Gamescom, tengan repercusión mundial con retransmisiones en directo a través de internet.

Otro factor, que está vinculado al anterior, es el nacimiento y consolidación de los creadores de contenido, youtubers, streamers e influencers, como medio de visualización de videojuegos.

El tercer factor, que también se encuentra conectado a los anteriores, es la irrupción de los deportes electrónicos como evento social. Los deportes electrónicos, conocidos como eSports, han aumentado su popularidad en estos últimos años, llegando a ser reconocidos a nivel mundial como competiciones equiparables a las competiciones de deportes tradicionales [1]. Y en donde actualmente un profesional de los eSports puede ser tan popular o ingresar tantos millones como los deportistas de élite de otros deportes [2].

Y, por último, todo esto no podría ser posible sin la evolución que ha tenido el hardware en la última década. Dando la posibilidad de tener motores gráficos capaces de crear juegos de una calidad gráfica y un nivel de detalle inimaginable hace unos años. Y, también, ha abierto un nuevo mercado para los videojuegos, como lo es el de la telefonía móvil. En donde los smartphones se han convertido en la consola portátil definitiva, con potencia suficiente y con un catálogo bastante amplio.

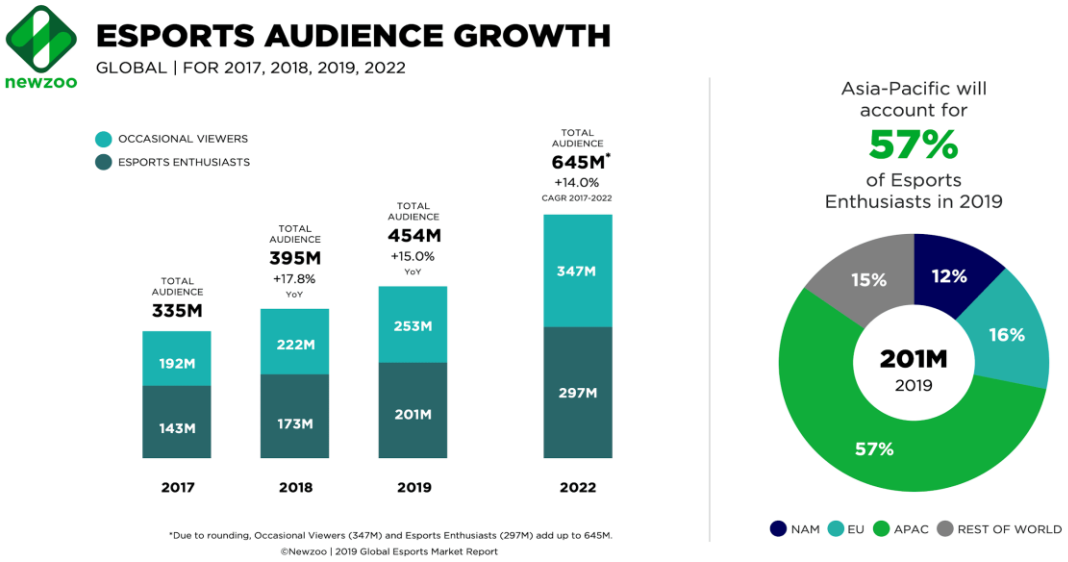

Figura 1: Datos de audiencia de los eventos de eSports a nivel mundial según Newzoo Research [3]

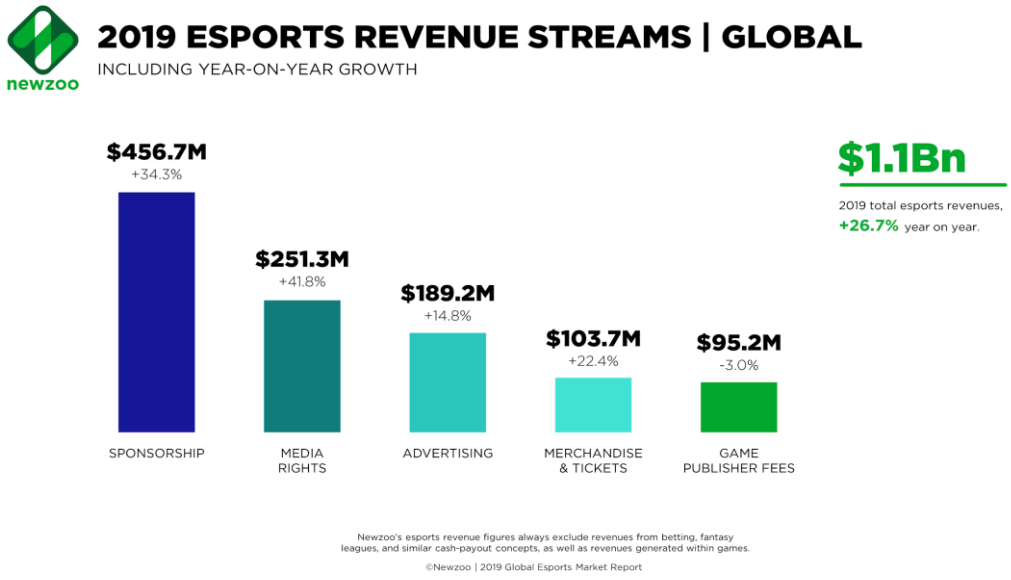

Figura 2: Datos económico de los eSports a nivel mundial según Newzoo Research

1.2 Inteligencia artificial en los videojuegos

Desde la creación de los primeros videojuegos, la inteligencia artificial ha estado ligada a estos. En un principio, la inteligencia artificial se usaba en los videojuegos para describir patrones de comportamiento, definiendo acciones de los NPCs (non-player characters) y para crear mapas de manera procedural [4]. Estas técnicas se implementan mediante redes bayesianas y máquinas de estado. Durante los años se han ido puliendo, añadiendo nuevas funcionalidades como:

- Dinamismo: Que el comportamiento de los NPCs varíe en función de lo que esté ocurriendo en el juego.
- Movimiento: Los NPCs tienen la capacidad de moverse libremente. Ya sea un movimiento de desplazamiento como gesticular.

La implementación de estas funcionalidades ha llegado a un nivel de realismo bastante alto. El problema es que, aun siendo movimientos o comportamientos que imitan a los reales, la mayoría de las veces resultan algo robotizados o irreales.

En la actualidad, podemos encontrar juegos que cuentan con inteligencia artificial avanzada como Pine [5]. Los desarrolladores de Pine implementan la inteligencia artificial en los enemigos del protagonista de manera que estos aprenden en cada combate y en consecuencia modifican sus ataques y movimientos para oponerse al jugador.

También se ha desarrollado nuevas técnicas de generación de movimiento utilizando redes neuronales, como podemos ver en el trabajo de Daniel Holden, Taki Komura y Jun Saito [6]. En este trabajo desarrollan una red neuronal capaz de recibir como entrada el movimiento del jugador, el movimiento previo y el estado del entorno en el que se encuentra para producir un movimiento realista que se adapte a estas circunstancias. Siendo así capaz de moverse con cierta naturalidad ya sea caminando o corriendo, en superficies planas o con relieve, llegando a realizar saltos para descender o superar un obstáculo.

En el sector del procesamiento de imágenes, las desarrolladoras y modders de videojuegos utilizan la inteligencia artificial con técnicas como la superresolución [7] para remasterizar videojuegos antiguos. Esta técnica se basa en una red neuronal, que recibe una imagen y, a partir de ella, genera la misma imagen a una resolución mayor con la menor pérdida posible. En el caso de la remasterización de videojuegos, la red neuronal recibe como entrada las imágenes que componen las texturas del videojuego y generan nuevas texturas a mayor resolución. De esta forma, al generar texturas de mayor resolución, la calidad visual del videojuego aumenta considerablemente.

Por otra parte, empresas de hardware como Nvidia, líderes a nivel mundial en fabricación y desarrollo de hardware dedicado a gráficos, están invirtiendo en esta misma técnica, para mejorar el rendimiento de los videojuegos. Este año han lanzado el Deep Learning Super Sampling (DLSS) [8]. Esta nueva tecnología, hace uso de la superresolución, implementando una red neuronal convolucional [9] con un modelo auto-encoder la cual recibe cada imagen del videojuego generada por la tarjeta gráfica y, a partir de esta, genera una nueva imagen a mayor resolución y con un filtro antialiasing, para suavizar posibles bordes de la imagen y

eliminar los dientes de sierra. Esta red neuronal es ejecutada en los Tensor Cores [10], núcleos que ha añadido Nvidia a sus nuevas tarjetas gráficas. Estos Tensor Cores están diseñados para las tareas de cómputo que se usan en la inteligencia artificial, siendo más eficientes y obteniendo mejor rendimiento. Esta tecnología consigue una mejora de rendimiento considerable ya que el videojuego se renderiza a una resolución inferior en los núcleos normales de la tarjeta gráfica y una vez obtenida la imagen, son los Tensor Cores los que se encargan de realizar el re-escalado a la resolución especificada.

Figura 3: Comparativa de imagen de textura remasterizada mediante deep learning [11]

Figura 4: Comparativa de resolución 4k con TAA y 4k generada con DLSS

Capítulo 2 Motivación y Objetivos

2.1 Motivación

Este trabajo viene motivado por los últimos avances de la inteligencia artificial en el ámbito de los videojuegos. Específicamente en una de la últimas tecnologías lanzadas por Nvidia junto con su nueva generación de tarjetas gráficas, el Deep Learning Super Sampling (DLSS). Como ya hemos comentado, DLSS hace uso de una red neuronal que recibe las imágenes renderizadas por la tarjeta gráfica, las reescala a una resolución mayor, con la menor pérdida posible, y aplica un efecto antialiasing. Todo esto se realiza en tiempo real.

Actualmente, con menos de un año desde su lanzamiento, DLSS no ha logrado convencer a los jugadores. El aumento de rendimiento, teniendo DLSS activado, aumenta considerablemente. Pero, por otra parte, la imagen generada no cumple con las expectativas de calidad prometidas por Nvidia. Las imágenes no alcanzan el mismo nivel de detalle que la imagen original en esa resolución, llegándose a ver algo borrosas en algunas zonas. Aun teniendo mucho que mejorar, esta tecnología podría ser esencial para el salto definitivo a los juegos a resolución 4K (3840 x 2160) y, en un futuro cercano, para la llegada de monitores y televisores 8K (7680 x 4320).

Es por ello, que hemos querido realizar este proyecto como una toma de contacto con el deep learning y la superresolución en videojuegos.

2.2 Objetivos principales

Adentrarnos en el mundo del deep learning implementando un modelo de red neuronal para superresolución. Y así, conocer los diferentes tipos de modelos que existen actualmente, aprender a usar herramientas como TensorFlow y Keras, además de las técnicas de tratamiento de imágenes.

2.3 Objetivos específicos

Al concluir este proyecto se pretenden cumplir los siguientes objetivos:

1. Aprender las técnicas de generación de datasets de entrenamiento y prueba.
2. Estudiar los distintos modelos de redes neuronales.
3. Generar, limpiar, ordenar y cargar las imágenes que forman el dataset.
4. Implementar y entrenar un modelo de red neuronal basado en superresolución.
5. Mejorar el modelo hasta generar imágenes con un mejor reescalado que con los algoritmos tradicionales (bicubic, nearest, …).

Capítulo 3 Desarrollo del proyecto

3.1 Entorno de desarrollo y herramientas utilizadas

El proyecto se ha desarrollado en un equipo con estas especificaciones:

- CPU: i5 7600
- GPU: GeForce GTX 1070 8 Gb
- RAM: 16 GB
- SO: Windows 10

Se han utilizado los drivers específicos de Nvidia para proyectos de deep learning y computación de altas prestaciones en GPU en sus versiones:

- Nvidia Drivers v431.60
- Nvidia GPU Computing CUDA Toolkit v10.0 [12]
- Nvidia cuDNN v7.6.3 [13]

El proyecto ha sido realizado enteramente en el lenguaje de programación Python, concretamente en su versión 3.6.8 [14]. Se ha elegido Python como lenguaje de programación por ofrecer gran flexibilidad y proporcionar las herramientas y librerías necesarias para el manejo de imágenes, vectores y para la creación de redes neuronales.

Las librerías que se han utilizado para la implementación de la red neuronal han sido:

- Keras v2.2.4 [15]
- TensorFlow-gpu v1.14.0 [16]

También se ha utilizado otras librerías como Skimage [17] y Pillow [18] para el manejo y procesado de imágenes, y Numpy [19] para el manejo de vectores.

3.2 Elección del modelo

Durante la fase de planificación del proyecto se barajaron distintos tipos de modelos de red neuronal. De ellos, destacaban dos tipos de modelo, GAN (Generative Adversarial Network) [20] y ResNet (Residual Neural Network) [21].

Los modelos GAN se componen de dos redes neuronales, una red generativa y otra discriminativa. Estas dos redes compiten entre sí. La red generativa, generalmente una red deconvolucional, construye una imagen y es la red discriminativa la que decide si la imagen generada es correcta o no. El objetivo de este tipo de redes es que las imágenes creadas por la red generativa sean capaces de engañar a la red discriminativa. Este tipo de redes se usan principalmente para generar imágenes hiperrealistas. En el caso de los videojuegos, son las

utilizadas para realizar la remasterización de las texturas de los videojuegos.

Los modelos ResNet o redes neuronales residuales, son redes convolucionales profundas que cuentan con skip connections o bloques residuales [22]. Un bloque residual es un conjunto de capas convolucionales en las cuales la salida de cada capa es la entrada de su capa consecutiva y, además, esa salida se utilizada como entrada de otra capa no consecutiva. Estas conexiones entre capas solucionan el problema de la dificultad a la hora de entrenar una red convolucional profunda, reduciendo el tiempo de entrenamiento antes de obtener resultados aceptables.

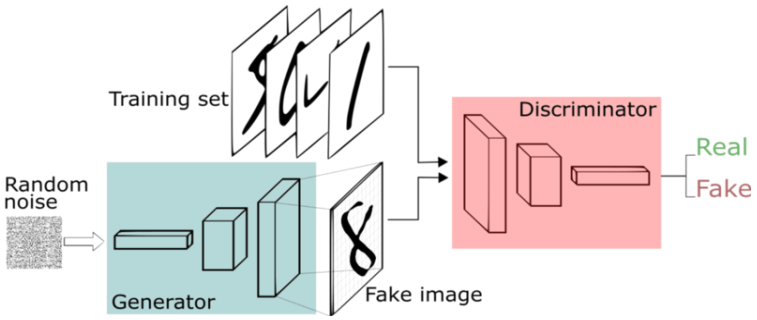

Figura 5: Esquema de un modelo de una GAN

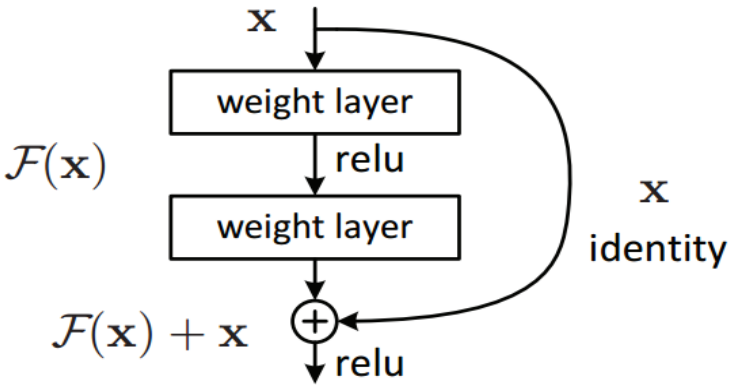

Figura 6: Representación de un bloque residual

Estos dos modelos se implementaron en forma de prototipo de pruebas. Pero, por limitaciones de hardware y de tiempo se terminó descartando el modelo GAN. Un modelo GAN se compone de dos redes neuronales, el generador y el discriminador, lo cual supone un mayor impacto en la GPU a la hora de entrenar el modelo, además, de requerir de una mayor precisión en los parámetros de entrenamiento para que ninguna de las dos redes se vuelva demasiado superior a la otra.

Una vez definido que el modelo utilizado sería ResNet, se implementaron distintas versiones hasta dar con la versión definitiva:

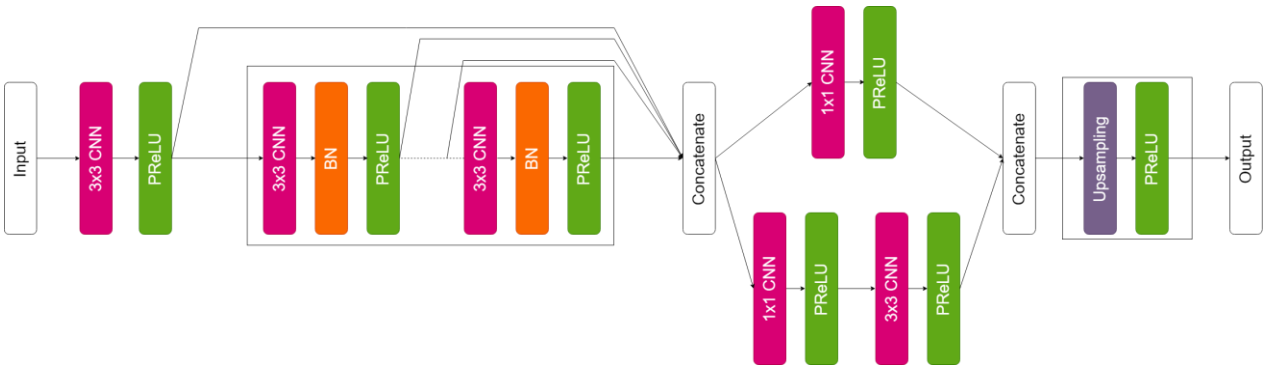

Figura 7: Esquema del modelo de ResNet

Este modelo está compuesto por dos etapas:

- La primera etapa se puede describir como la fase de extracción de datos de la imagen. Esta fase está compuesta por una primera capa de convolución con un kernel de tamaño 3 x 3, y una capa de activación de la función PReLU. La salida de esta capa está conectada directamente con un bloque compuesto de una convolución con kernel de tamaño 3 x 3, seguida de una capa de normalización del batch y una capa de activación PReLu. Este bloque se puede repetir *n* veces. Por último, cada salida de bloque es conectado mediante skip connections y son concatenados.
- La segunda fase, sería la fase de reconstrucción o de reescalado. En esta etapa se realiza, por un lado, una convolución con kernel de tamaño 1 x 1 y una capa de activación PReLU, seguida de otro bloque con las mismas capas, pero con un kernel de tamaño 3 x 3. La salida de esta última capa es concatenada con la salida del bloque formado por una convolución de kernel tamaño 1 x 1 y una capa de activación PReLU. Para finalizar, se realiza el reescalado mediante una convolución transpuesta con un kernel de tamaño 3 x 3 y una capa de activación PReLU. Este último bloque de reescalado se puede repetir *n* veces, siendo 1 un reescalado x2, 2 un reescalado x4, etc.

Este tipo de modelo, el cual realiza el reescalado o upsampling al final, tienen un menor impacto en el tiempo de entrenamiento y en la carga de la GPU. Esto se debe a que las convoluciones se aplican directamente a las imágenes de entrada, las cuales aún no han sido reescaladas y por lo tanto son de menor tamaño. Esto, entre otras cosas, ha beneficiado a la realización de este proyecto.

3.3 Generación del dataset

Dado que el proyecto está enfocado en los videojuegos, se ha optado por generar el dataset de entrenamiento y prueba a partir de imágenes de videojuegos, en concreto se ha elegido el F1 2018 [23].

Para generar las imágenes se ha grabado distintas carreras en el F1 2018, variando los vehículos y los circuitos; además de las condiciones climatológicas. Una vez obtenido los vídeos, se han desarrollado distintas herramientas en Python, para obtener cada fotograma del video y guardarlo como imágenes individuales, obtenido un total de 20.000 imágenes.

Figura 8: Muestra de imágenes del dataset

3.4 Pre-procesado del dataset

Para mejorar el manejo de las imágenes y mejorar los tiempos y el resultado del entrenamiento de la red neuronal, se ha aplicado un preprocesamiento a las imágenes del dataset.

En primer lugar, cada imagen es cargada utilizando la librería skimage. Estas imágenes están en formato RGB (240, 426, 3). Este vector tridimensional cuenta con valores en el rango de [0, 255], por lo que es importante normalizarlo en [0, 1]. Normalizar los datos reduce drásticamente el tiempo de entrenamiento de la red neuronal, por lo que es esencial realizarlo.

Luego, a cada imagen normalizada le realizamos un alineamiento. Debido a que para el entrenamiento tenemos que reducir la resolución de la imagen de entrada para que luego la red neuronal vuelva a aumentarla, modificamos la resolución de cada imagen para que sea divisible por el valor de escalado. Si no se realiza podría darse el caso de que al reducir la imagen y luego aumentarla mediante la red neuronal, la resolución de la imagen difiera de la original.

El siguiente paso, teniendo las imágenes normalizadas y alineadas, es dividir cada imagen en subimágenes. El número de subimágenes se calcula según el tamaño que tendrán las nuevas imágenes y el movimiento o *stride* que se realizará entre imagen e imagen. En nuestro caso hemos establecido el tamaño de las imágenes a 48 pixeles, con un *stride* del mismo tamaño, 48 pixeles.

Realizando este proceso cambiamos el dataset de 20.000 imágenes a resolución 240 x 426, a un dataset formado por 800.000 imágenes de tamaño 48 x 48. Además, reducimos la carga de memoria de la GPU y conseguimos que la red neuronal no sea tan dependiente del contexto de las imágenes.

Por último, reducimos cada imagen original según el valor de escalado para obtener las imágenes a baja resolución. Para realizar el *downsampling* se calcula el ancho y alto multiplicando por el factor de escalado. Se utiliza la librería Pillow, que cuenta con una función para reescalar imágenes utilizando distintos algoritmos, en nuestro caso se ha utilizado el algoritmo bicubic.

Una vez hemos realizado estos pasos obtenemos 4 vectores de 4 dimensiones, los vectores de entrenamiento y de prueba, con imágenes en alta resolución y los vectores de entrenamiento y de prueba con las imágenes en baja resolución.

Figura 9: Muestra de las subimágenes generadas

3.5 Parámetros de entrenamiento

Para realizar el entrenamiento se han utilizado los siguientes parámetros:

- Número de imágenes: 20.000 ⇒ 800.000
- Resolución de las imágenes: 240 x 426 ⇒ 48 x 48
- Número de epochs: 5
- Tamaño de batch: 64

Los parámetros de la red neuronal utilizados:

- Primera convolución: 196 filtros
- Bloques de extracción: 11 bloques
- Convoluciones del bloque de extracción: 48 filtros
- Convoluciones de reescalado: 32 filtros

El optimizador utilizado es el Adam con los parámetros por defecto. Adam es un algoritmo de optimización que se puede usar en lugar del procedimiento clásico de descenso de gradiente estocástico para actualizar los pesos de la red de forma iterativa según los datos de entrenamiento.

La función de pérdida utilizada es la función de error cuadrático medio. El error cuadrático medio o mean squared error (MSE) mide el promedio de los cuadrados de los errores, es decir, la diferencia al cuadrado promedio entre los valores estimados y el valor real. En nuestro caso, la diferencia se produce entre las imágenes generadas por la red y las imágenes originales.

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

Capítulo 4 Resultados y evaluación del modelo

En este apartado se muestran los datos y resultados obtenidos durante el entrenamiento del modelo, según los parámetros descritos en el apartado anterior, con el fin de realizar un análisis que ofrezca conclusiones relevantes.

4.1 Resultados con factor de escalado x2

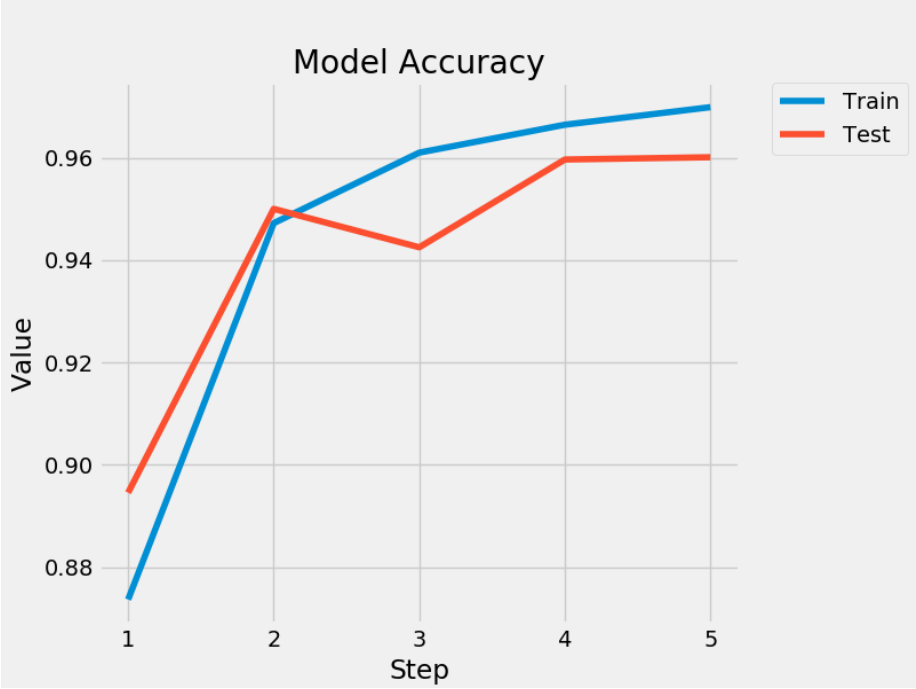

Figura 10: Gráfico que representa la precisión del modelo x2

Como podemos observar, el modelo, tras la primera vuelta o primer epoch, ya obtiene un nivel de precisión por encima del 85% tanto para el dataset de entrenamiento como para el de prueba. Esto puede ser debido a la gran cantidad de imágenes utilizadas (800.000 imágenes), ya que tras los primeros 20 batches en el primer epoch la precisión del modelo ronda el 65%. A partir de ahí, vemos como la precisión en los datos de entrenamiento aumentan de manera logarítmica de la misma forma que decrece el valor de la función de pérdida, hasta alcanzar una precisión del 97%.

Por otro lado, en los datos de validación, observamos que, tras unos valores iniciales sorprendentemente altos, se produce un reajuste. Esto podría ser producido porque las imágenes de validación utilizadas tras el segundo epoch tuvieran elementos similares a las imágenes de entrenamiento y las utilizadas en el siguiente se diferenciaron algo más. Aun así, el valor tras el tercer epoch sigue superando el 94%. Tras ese reajuste observamos que el modelo sigue aumentando la precisión hasta igualarse con los valores de entrenamiento.

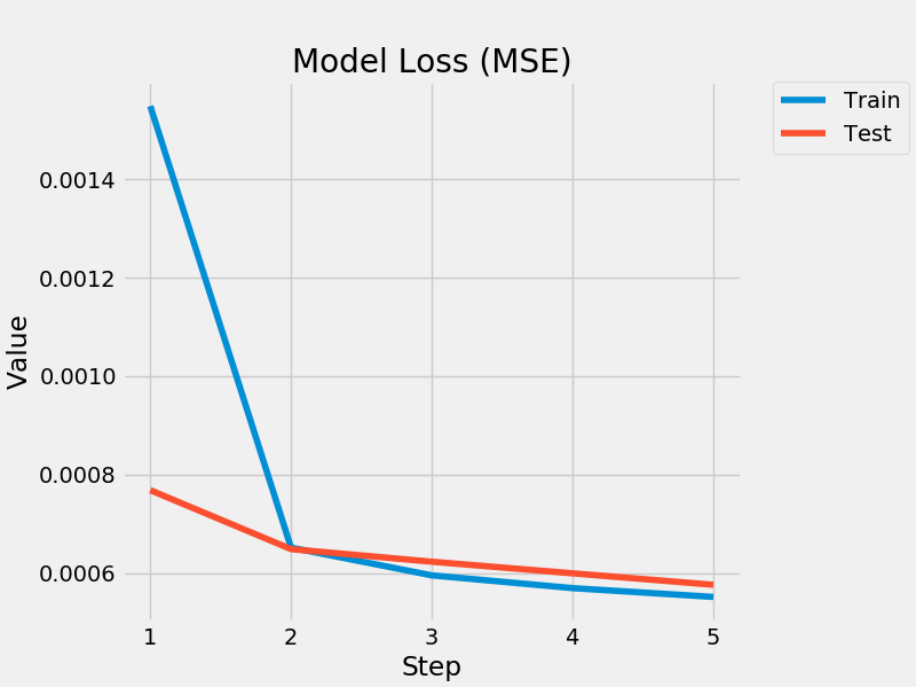

Figura 11: Gráfico que representa la pérdida del modelo x2

Si miramos conjuntamente los gráficos de precisión y los valores de la función de pérdida, vemos que la diferencia de precisión y de pérdida entre el primer epoch y el final del entrenamiento es relativamente baja. Confirmando así que las skip connections y el uso de subimágenes hace que los resultados obtenidos tras un breve periodo de tiempo ya alcanzan un mínimo de calidad en el reescalado de la imagen.

Las imágenes generadas por el modelo tienen una calidad visual similar a las imágenes originales correspondientes. Esta similitud varía de manera directamente proporcional con la variación de la resolución de la imagen de entrada. Cuando el modelo recibe una imagen a muy baja resolución, al tener poca información, genera una imagen parecida a la original, pero con poca definición. Una vez aumentamos la resolución, se puede dar el caso de que la imagen generada y la imagen original sean prácticamente indistinguibles a primera vista. Normalmente, según las diferentes pruebas, las mayores diferencias entre la imagen generada y la original suelen estar en los objetos en segundo plano y carteles con texto.

Figura 12: Demostración de la generación de imágenes del modelo x2

4.2 Resultados con factor de escalado x4

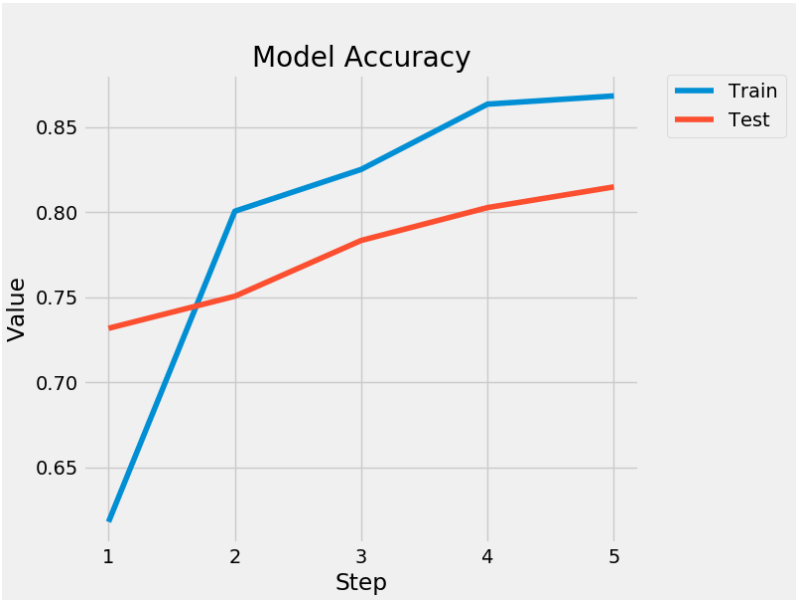

Figura 13: Gráfico que representa la precisión del modelo x4

A diferencia de los datos obtenidos en el apartado anterior, donde el factor de escalado era solamente de x2, la precisión del modelo no es tan alta. Obviamente, al ser mayor el reescalado, la cantidad de pixeles que el modelo tiene que predecir es mayor. Aun así, los resultados superan el 80%, solamente con 5 epochs, lo cual, aun siendo suficiente para el entrenamiento de la red para reescalar con factor x2, es insuficiente en este caso.

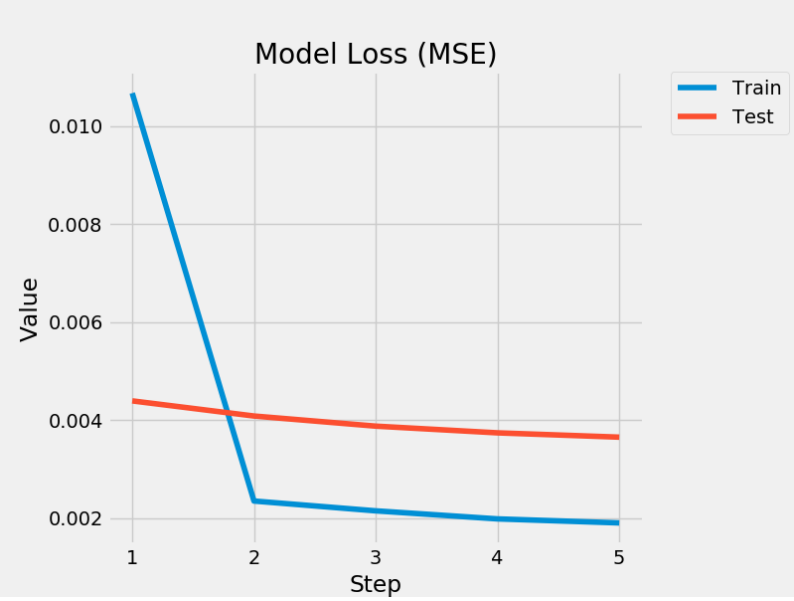

Figura 14: Gráfico que representa la pérdida del modelo x4

El valor de pérdida, del conjunto de prueba, se queda cerca los 0.004, mientras que en el reescalado x2 es inferior a 0.0006. Esto repercute directamente en la generación de imágenes como podemos ver en la Figura 15, en donde no solo la imagen no llega a obtener un nivel de definición suficiente, sino que aparecen diferentes artefactos.

Si miramos con detenimiento una imagen generada con factor x4, podemos ver como la red tiene un patrón de generación de ruido. Esto hace que la imagen cuente con una malla de pixeles que su color no corresponde con la imagen original. A medida que la resolución aumenta, el ruido y artefactos que aparecen en las imágenes queda disimulado gracias a la disminución del tamaño de los pixeles.

Esto demuestra, como ya predecían los datos de pérdida y precisión, que el número de epochs es insuficiente o el modelo necesita optimizarse para este factor de escalado.

Figura 15: Zoom sobre imagen x4

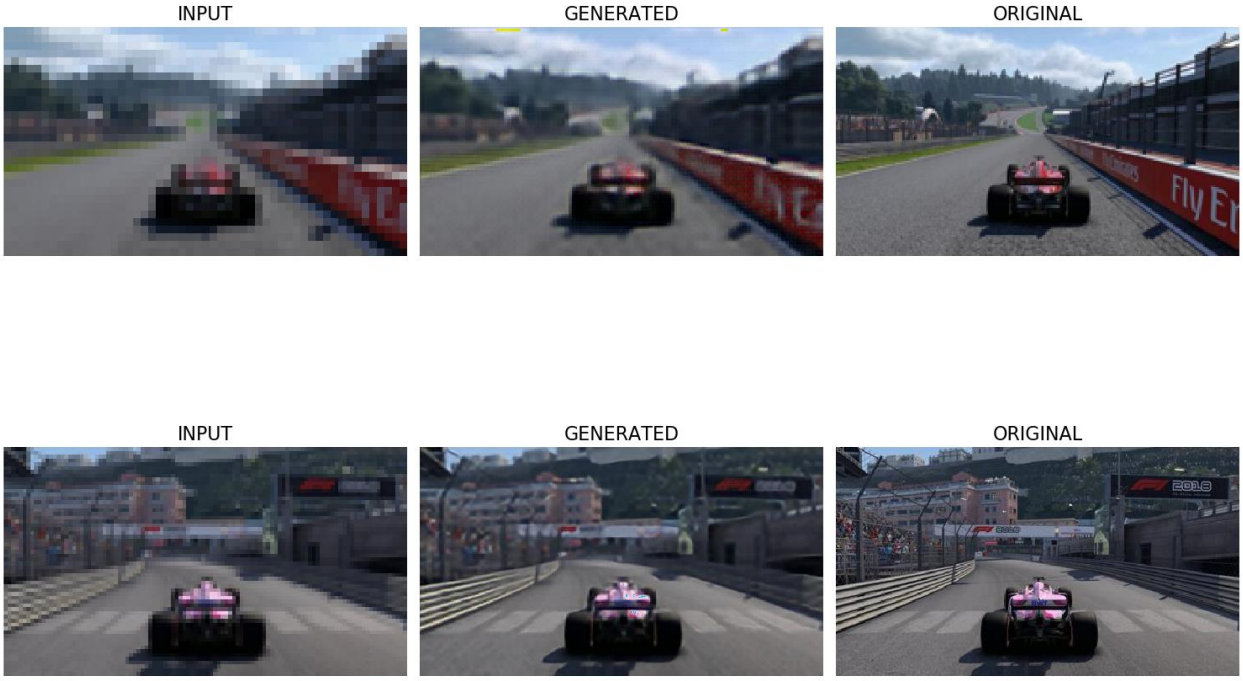

Figura 16: Demostración de reescalado con factor x4

4.3 Comparativa con métodos tradicionales

Uno de los objetivos que se busca conseguir en este proyecto es que el modelo implementado superase, en cuanto a calidad visual, a los métodos tradicionales de reescalado. De entre todos los algoritmos, se han utilizado para la comparativa: nearest, bicubic y lanczos.

Para la comparativa se ha usado la función de Proporción Máxima de Señal a Ruido o PSNR (*Peak Signal-to-Noise Ratio*). Esta función calcula la relación pico señal-ruido para la imagen generada, con la imagen original como referencia.

Como se puede apreciar en las Figuras 17, 18, 19; las imágenes generadas por el modelo siempre están por encima de los métodos tradicionales. El método Lanczos es el que más se acerca, pero con bastante menos definición. Este efecto borroso aparece en los algoritmos tradicionales cuando las imágenes de entrada son de baja resolución.

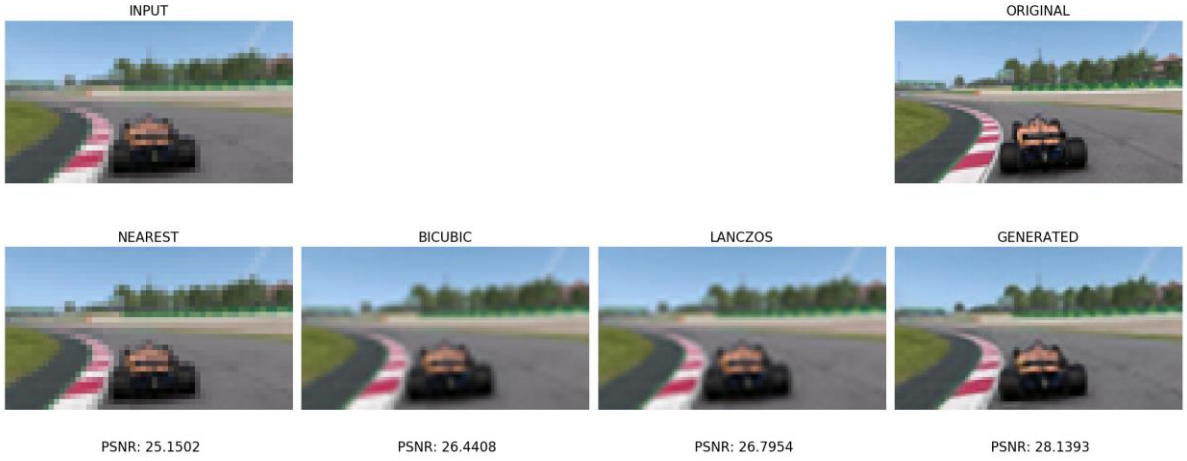

Figura 17: Comparativa métodos tradicionales 1

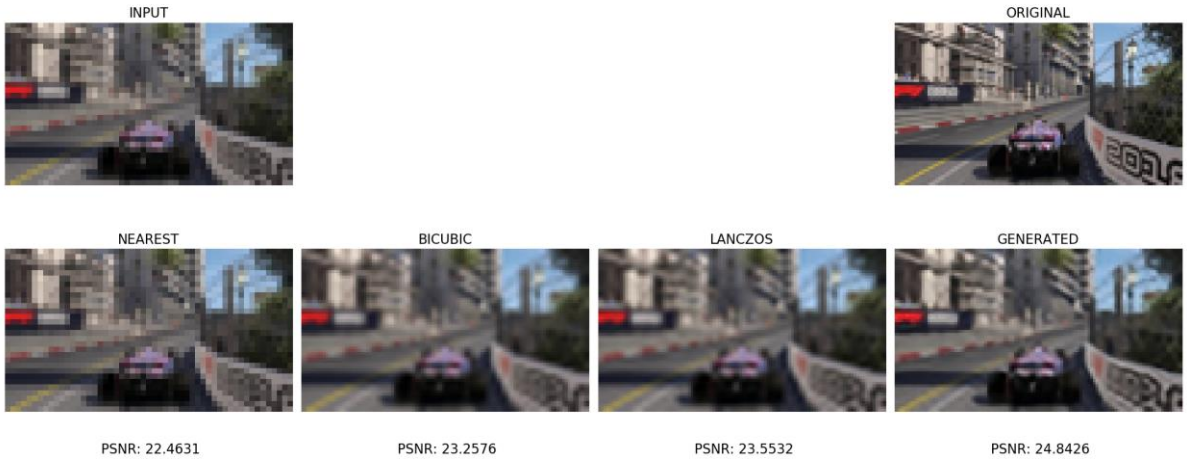

Figura 18: Comparativa métodos tradicionales 2

Como pudimos ver en los análisis anteriores, cuando aumentamos la resolución, las imágenes generadas son indistinguibles de las imágenes originales y con las imágenes reescaladas con algoritmos pasa exactamente lo mismo. Aun así, el modelo sigue siendo superior en cuanto a definición, llegando a solamente diferenciarse por los carteles de publicidad, Figura 20, en donde la red no ha sabido recrear las letras de publicidad, quedando una mancha roja. Las imágenes generadas con algoritmos tampoco son capaces de generar estas letras.

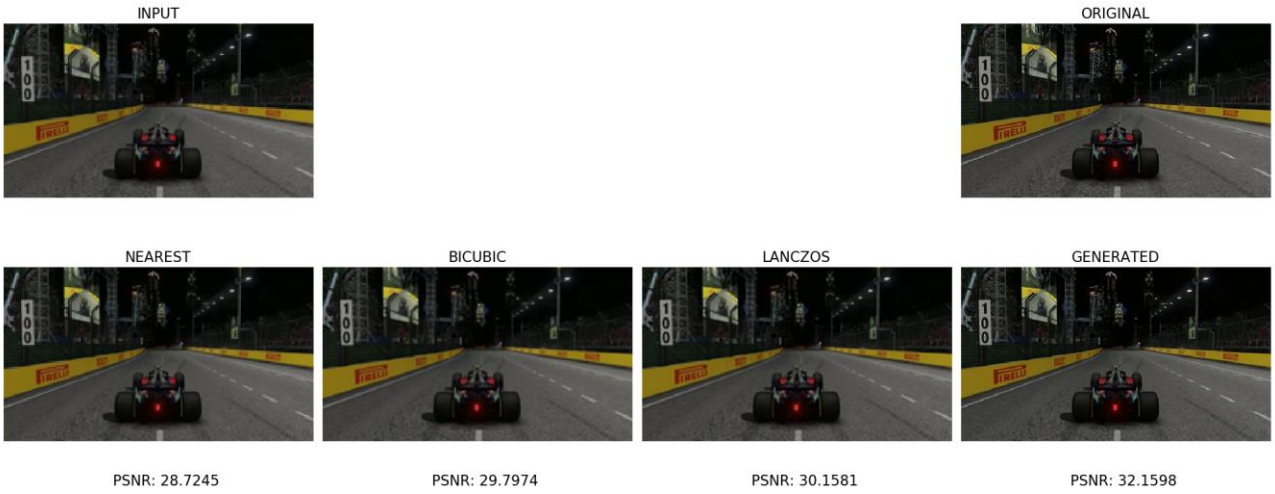

Figura 19: Comparativa métodos tradicionales 3

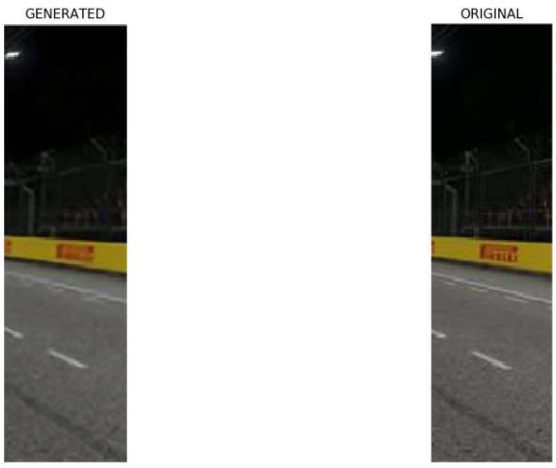

Figura 20: Zoom diferencia

4.4 Pruebas con imágenes fuera de contexto

Por último, se ha realizado una prueba muy interesante, en donde se prueba la precisión de la red neuronal con imágenes con contenido completamente distinto a las imágenes de entrenamiento y de prueba. En este apartado comprobaremos que el modelo no es dependiente del contexto y estructura de las imágenes de entrenamiento, al usar subimágenes para entrenar el modelo.

Las primeras imágenes que se han probado son del videojuego Forza Horizon 4. Este juego sigue la misma estructura visual del juego usado de entrenamiento, F1 2018. Un coche visto desde detrás, en un carril y con un fondo diferenciado. Fijándonos en los resultados, pasa algo curioso. Aunque el valor de psnr de la imagen generada sea inferior al de las imágenes reescaladas mediante el algoritmo bicubic y lanczos, visualmente las imágenes son prácticamente iguales. La diferencia en este caso se encuentra en el HUD (Head-Up Display) del juego, la información que se muestra en pantalla. Si lo miramos con detenimientos, vemos que la red no ha sabido interpretarlo bien y genera ruido tanto en el contador de vueltas como en el velocímetro.

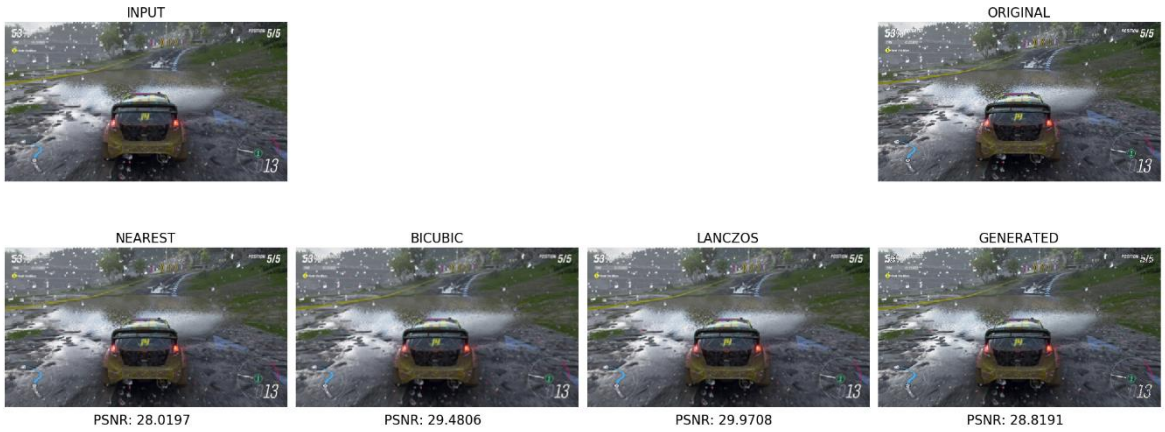

Figura 21: Pruebas con imágenes fuera de contexto 1

Figura 22: Zoom Forza Horizon 4

El siguiente juego que se ha probado es el Metro Exodus. Este juego difiere completamente de la estructura de imagen. Es un juego en primera persona en donde se lleva siempre un arma en las manos, los escenarios varían entre sitios cerrados (bukers), desiertos y zonas boscosas.

Como vemos los resultados obtenidos están a favor del modelo. Se puede apreciar como la definición de las imágenes generadas es bastante superior. Sin embargo, en ciertas imágenes, como es el caso de la Figura 23, se generan algunos pixeles de ruido. En este caso alrededor de las ramas del árbol seco del fondo y en la propia arma del personaje.

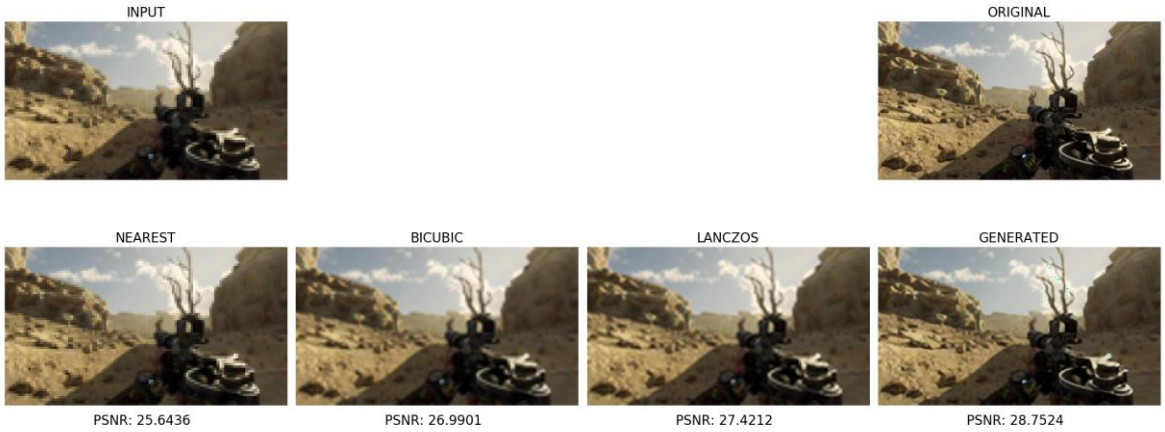

Figura 23: Pruebas con imágenes fuera de contexto 2

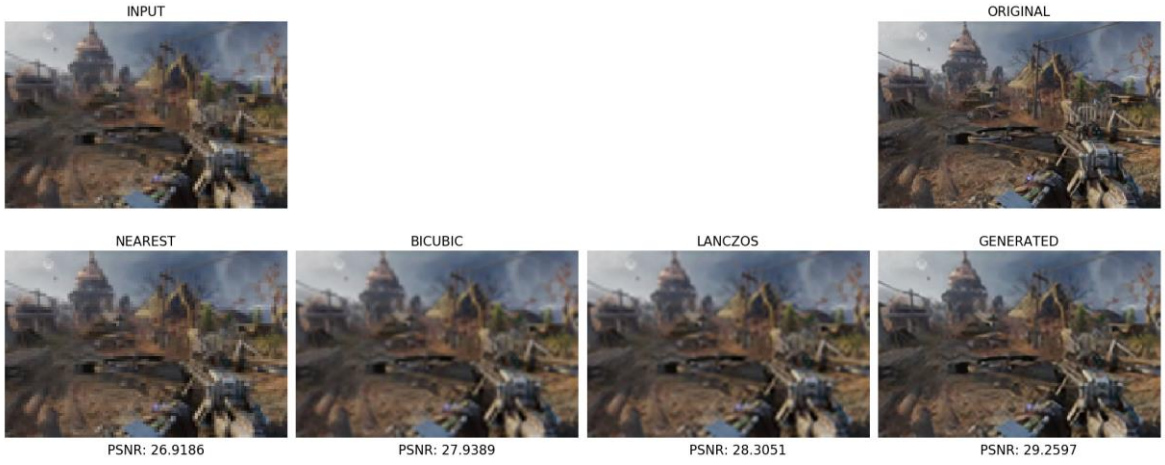

Figura 24: Pruebas con imágenes fuera de contexto 3

Por último, se ha elegido el videojuego The Witcher 3, un juego en tercera persona con escenarios boscosos. Al igual que con los juegos anteriores, el resultado es bastante satisfactorio, pero, al tener un mayor número de efectos visuales, hay imágenes donde el resultado se podría considerar desastroso, Figura 26.

Este juego, a diferencia del F1 2018, tiene un mayor número de elementos visuales, es por ello que las imágenes donde el personaje o algún enemigo utiliza fuego o rayos, la red no consigue interpretarlo bien. Los efectos de luz, en este caso el atardecer donde los rayos del sol atraviesan los árboles, tampoco los consigue interpretan, dando como resultado una imagen no aceptable.

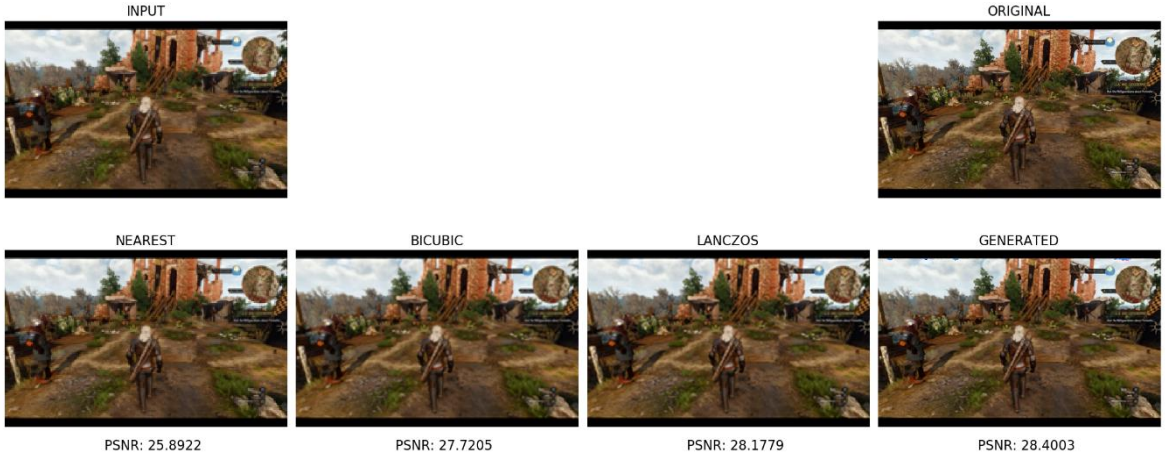

Figura 25: Pruebas con imágenes fuera de contexto 4

Figura 26: Imagen con ruido

Capítulo 5 Conclusiones y líneas futuras

Tras analizar los resultados obtenidos, podemos afirmar que se ha cumplido el objetivo inicial de implementar un modelo de red neuronal para superresolución capaz de aumentar la resolución de las imágenes con un nivel de pérdida mínimo. Aun así, creemos que el modelo podría ser mejorado y optimizado, ya que en este proyecto no se ha tenido en cuenta, como variable a reducir, el tiempo de ejecución.

Como balance personal, cabe destacar que este proyecto ha sido todo un reto personal dado mi inexperiencia en el campo del deep learning. Tras realizarlo, he obtenido un gran conocimiento en este campo que me permitirá, en el futuro, mejorar este proyecto o iniciar nuevos proyectos de mayor magnitud. Además, he adquirido nuevos conocimientos sobre Python y sus librerías, tanto para el procesamiento de imágenes como las utilizadas para inteligencia artificial.

En cuanto a posibles líneas futuras del proyecto, se podría experimentar más a fondo con diferentes tipos de modelos de redes neuronales. En la fase de elección se estudiaron diferentes proyectos de superresolución y se eligió el modelo que mejor se adaptaba a la magnitud de este proyecto. Pero, se podría analizar en profundidad diferentes modelos y comparar los resultados.

Por otro lado, este proyecto está íntegramente implementado usando Keras. Esta API de alto nivel utiliza de base la API de TensorFlow. Podría ser una buena práctica migrar el proyecto de Keras a TensorFlow aprovechando que, durante la realización de este proyecto, Google, empresa a la que pertenece TensorFlow, ha lanzado la TensorFlow 2.0 [24]. Esta nueva versión incorpora importantes cambios, se eliminarán diferentes APIs, por lo que la versión actual de Keras no será compatible con esta versión. Se añadirán importantes mejoras en los métodos y se añaden nuevos métodos como la incorporación de las funcionalidades de Keras dentro del propio TensorFlow 2.0. Esto haría que el proceso de migración fuese relativamente poco costoso y, además, daría la posibilidad de utilizar las herramientas que trae TensorFlow 2.0.

Por último, como se ha explicado previamente, este proyecto surge a partir de la presentación de la tecnología Deep Learning Super Sampling (DLSS) de Nvidia. Esta tecnología es capaz de ejecutarse en tiempo real durante la ejecución de un videojuego. Esto significa que cada segundo es capaz de aumentar la resolución de más de 60 imágenes cada segundo, en resoluciones superiores a 1280 x 720.

Para nosotros, realizar tal cosa sería demasiado complicado ya que DLSS se implementa directamente en los drivers de la tarjeta gráfica. Pero, una propuesta de futuro es poder mejorar la red para trabajar, en primer lugar, con video y, si es viable, con video en tiempo real.

Capítulo 6 Summary and Conclusions

After analysing the results, we can say that the initial objective of implementing a neural network model for super-resolution capable of increasing the resolution of the images with a minimum level of loss has been met. However, we believe that the model could be improved and optimized, because we have not considered the runtime.

As a personal balance, it should be noted that this project has been a personal challenge given my inexperience in the field of deep learning. After doing so, I have gained great knowledge in this field that will allow me to improve this project or start new projects. In addition, I have acquired new knowledge about Python and its libraries.

As for possible future lines, we could experiment more thoroughly different types of neural network models. In the election phase, different super-resolution projects were studied and the model that best suited the magnitude of this project was chosen. But you could analyse in depth different models and compare the results.

On the other hand, this project is fully implemented using Keras. This high-level API uses the TensorFlow API as a base. It might be a good practice to migrate the project from Keras to TensorFlow taking advantage of the being that, during the realization of this project, Google, the company to which TensorFlow belongs, has launched TensorFlow 2.0 [24]. This new version incorporates major changes, different APIs will be removed, so the current version of Keras will not be compatible with this version. Significant improvements to the methods will be added and new methods are added such as incorporating Keras functionalities within TensorFlow 2.0 itself. This would make the migration process relatively inexpensive and would also give you the ability to use the tools that TensorFlow 2.0 brings.

Finally, as explained previously, this project arises from the presentation of Nvidia's Deep Learning Super Sampling (DLSS) technology. This technology can run in real time while running a video game. This means that every second it can increase the resolution of more than 60 images every second, in resolutions greater than 1280 x 720.

For us, doing such a thing would be too complicated as DLSS is implemented directly in the graphics card drivers. But, a proposal for the future is to be able to improve the network to work, first, with video and, if feasible, with real-time video.

Apéndice A

Repositorio del proyecto

El código del proyecto ha sido subido a un repositorio público junto con algunas de las imágenes de resultados y los archivos de los modelos con sus pesos en formato .h5:

- Repositorio del proyecto

Bibliografía

[1] Álvaro Guiñón, esport AS, "Más de 200 millones de espectadores vieron la final del Mundial de LoL 2018" https://esports.as.com/worlds-2018/audiencia-espectadores-final-Mundial-LoL_0_1188181179.html

[2] Chema Mansilla, 3DJuegos, "Un jugador de DOTA2 gana casi 5 millones de dólares en un torneo" https://www.3djuegos.com/noticias-ver/196361/un-jugador-de-dota2-gana-casi-5-millones-de-dlares-en-un/

[3] Jurre Pannekeet, Newzoo, "Global Esports Economy Will Top $1 Billion for the First Time in 2019" https://newzoo.com/insights/articles/newzoo-global-esports-economy-will-top-1-billion-for-the-first-time-in-2019/

[4] Georgios N. Yannakakis, "Game AI Revisited" https://web.archive.org/web/20140808052149/http://aida.ii.uam.es/teaching/videojuegos/wp-content/uploads/course_files_12_13/gameAI.pdf

[5] Pine Game https://pine-game.com/

[6] Daniel Holden, Taku Komura, Jun Saito, "Phase-Functioned Neural Networks for Character Control" http://theorangeduck.com/media/uploads/other_stuff/phasefunction.pdf

[7] Nvidia, "DLSS: ¿What Does It Mean for Game Developers?" https://news.developer.nvidia.com/dlss-what-does-it-mean-for-game-developers/

[8] Zhihao Wang, Jian Chen, Steven C.H. Hoi, "Deep Learning for Image Super-resolution: A Survey" https://arxiv.org/abs/1902.06068

[9] Yann LeCun, Patrick Haffner, Léon Bottou, Yoshua Bengio, "Object Recognition with Gradient-Based Learning" http://yann.lecun.com/exdb/publis/pdf/lecun-99.pdf

[10] Nvidia, Tensor Core https://www.nvidia.com/es-es/data-center/tensorcore/

[11] Hidfan, "Doom Neural Upscale 2X" https://www.doomworld.com/forum/topic/99021-v-0-95-doom-neural-upscale-2x/

[12] Nvidia, "Develop, Optimize and Deploy GPU-accelerated Apps" https://developer.nvidia.com/cuda-toolkit

[13] Nvidia, "CUDA Deep Neural Network library (cuDNN)" https://developer.nvidia.com/cudnn

[14] Python, "Release Python 3.6.8" https://www.python.org/downloads/release/python-368/

[15] Keras, "The Python Deep Learning library" https://keras.io/

[16] TensorFlow, "TensorFlow GPU support" https://www.tensorflow.org/install/gpu

[17] Scikit-image, “scikit-image, image processing in python” https://scikit-image.org

[18] Alex Clark and Contributors, “Pillow, Python Imaging Library” https://pillow.readthedocs.io/en/stable/

[19] Numpy, “Fundamental package for scientific computing with Python” https://numpy.org/

[20] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio, “Generative Adversarial Networks” https://arxiv.org/abs/1406.2661

[21] Chao Dong, Chen Change Loy, Kaiming He, Xiaoou Tang, “Image Super-Resolution Using Deep Convolutional Networks” https://arxiv.org/abs/1501.00092

[22] Harshit Kumar, “Skip connections and Residual blocks” https://kharshit.github.io/blog/2018/09/07/skip-connections-and-residual-blocks

[23] Steam, “F1 2018” https://store.steampowered.com/app/737800/F1_2018/?l=spanish

[24] TensorFlow, “TensorFlow 2.0 RC” https://www.tensorflow.org/beta