Universidad Zaragoza
1542

Trabajo Fin de Grado

Título del trabajo:
Transferencia de textura y estilo entre imágenes utilizando redes neuronales convolucionales: un enfoque orientado a la conservación de contenido

English tittle:
Content-aware texture and style transfer using convolutional neural networks

Autor/es

Santiago Calvo Fantova

Director/es
Ana Serrano Pacheu
Belén Masiá Corcoy

Escuela de Ingeniería y Arquitectura

2016- 2017

TRABAJOS DE FIN DE GRADO / FIN DE MÁSTER

Escuela de Ingeniería y Arquitectura Universidad Zaragoza

DECLARACIÓN DE AUTORÍA Y ORIGINALIDAD

(Este documento debe acompañar al Trabajo Fin de Grado (TFG)/Trabajo Fin de Máster (TFM) cuando sea depositado para su evaluación).

D./Dª. Santiago Calvo Fantova,

con nº de DNI 25204316L en aplicación de lo dispuesto en el art. 14 (Derechos de autor) del Acuerdo de 11 de septiembre de 2014, del Consejo de Gobierno, por el que se aprueba el Reglamento de los TFG y TFM de la Universidad de Zaragoza,

Declaro que el presente Trabajo de Fin de (Grado/Máster) Grado, (Título del Trabajo) Transferencia de textura y estilo entre imágenes utilizando redes neuronales convolucionales: un enfoque orientado a la conservación de contenido,

es de mi autoría y es original, no habiéndose utilizado fuente sin ser citada debidamente.

Zaragoza, 02 / FEBRERO / 2017

Fdo: SANTIAGO CALVO FANTOVA

Resumen

Este proyecto se sitúa en el campo de la computación gráfica y el procesado digital de imagen.

La computación gráfica o gráficos por ordenador es un campo de la informática que estudia métodos de digitalización y manipulación de imagen y contenidos visuales. Esta disciplina comprende el estudio de gráficos computacionales y el procesado de imagen tanto en dos como en tres dimensiones. El procesado digital de imágenes es el conjunto de técnicas que se aplican a las imágenes digitales con el objetivo de mejorar la calidad o facilitar la búsqueda de información.

El proyecto se centra en el planteamiento de un sistema de procesado de imagen mediante el uso de redes neuronales convolucionales. Una red neuronal convolucional es un tipo de red neuronal artificial donde las neuronas corresponden a campos receptivos de una manera muy similar a las neuronas en la corteza visual primaria (V1) de un cerebro biológico. Este tipo de red es una variación de un perceptron multicapa, pero su funcionamiento las hace mucho más efectivas para tareas de visión artificial, especialmente en la clasificación de imágenes. Usos comunes de estas redes son, por ejemplo, la detección de patrones en imagen (ampliamente utilizado en imagen médica para detectar enfermedades) o el reconocimiento de objetos y entidades en imagen (usado para clasificar imágenes según el contenido de las mismas).

Uno de los campos dentro de la computación gráfica y el procesado de imagen, objeto de investigación en los últimos años ha sido la transferencia de texturas y estilos entre imágenes mediante la aplicación y utilización de diferentes técnicas. Una de estas técnicas nace con la introducción de redes neuronales en el campo de la imagen computacional. Dichas redes son entrenadas para detección de patrones en imagen y con la finalidad de desarrollar sistemas de clasificación de imágenes a través del análisis de contenido que hay en ellas. En el año 2016, el equipo de Leon A. Gatys de la Universidad de Tubingen, Alemania [1], descubre que partiendo como base de este tipo de redes neuronales convolucionales de detección de patrones en imagen, y añadiendo unas determinadas funciones de análisis de imagen, era posible conseguir un nuevo método de transferencia de estilo entre dos imágenes. Este método consiste básicamente en, partiendo de dos imágenes, tratar de transferir la información de estilo o textura de una de ellas a la otra. Así la que recibe dicha textura o estilo sera entendida como la imagen que aporta la información de contenido y la portadora de la textura o estilo, será entendida como la imagen que aporta el estilo. La imagen final será entonces una combinación de la información de contenido de la imagen-contenido y del estilo o textura de la imagen-estilo.

Nos encontramos entonces, con una técnica muy reciente y por tanto con un estado del arte muy inexplorado. Este método por tanto, presenta diferentes problemas o deficiencias por resolver. Uno de ellos, es lo impredecible del método, esto es, la dificultad o imposibilidad de predecir como va a ser la salida del sistema. Nos planteamos entonces hacer un estudio/análisis del mismo e intentar ajustar los distintos parámetros de la red para de alguna manera poder

tener una intuición o predicción de la salida que queremos obtener.

Otra carencia que presenta el sistema, es la degradación o pérdida de información, de la información de contenido de la imagen-contenido, que además de ser muy poco predecible como señalábamos antes, tiende a distorsionar los detalles mas pequeños, así como los ejes, bordes y líneas finas. Aquí nace uno de los objetivos principales del proyecto y es el conseguir una transferencia de textura o estilo de forma que la información de la imagen receptora de dicha textura o estilo, o lo que es lo mismo de la imagen que aporta la información de contenido, sufra el menor número de modificaciones y distorsiones posibles, poniendo el foco para ello, en conservar los detalles mas pequeños así como los ejes y bordes.

Para conseguir nuestro objetivo, el proyecto se ha basado en la siguiente estrategia: en primer lugar la identificación de los problemas comentados anteriormente, estudiaremos las causas de los mismos y plantearemos distintos análisis con el objetivo de clarificar el origen y clasificar dichos problemas, dichos análisis se centrarán tanto en el estudio de los diferentes parámetros y funciones de la red neuronal convolucional utilizada como en la propuesta de diferentes técnicas de análisis estadístico de imagen. Una vez realizado dicho estudio, se proponen distintas soluciones a los problemas encontrados y se procede a la implementación de las mismas, seguidamente se lleva a cabo un proceso de evaluación y validación de los resultados obtenidos con las soluciones propuestas e implementadas. Habrá una cierta realimentación en el proceso, pues en función de la evaluación y validación de los resultados se plantearán modificaciones y mejoras a las soluciones implementadas.

Por último, como será explicado en profundidad a lo largo de la memoria, tras asumir como satisfactorios los distintos resultados obtenidos en imagen tomamos la decisión de intentar adaptar el sistema y llegar a una transferencia de textura o estilo en vídeo. Dicha adaptación supuso además de modificaciones en las diferentes implementaciones, la aparición de nuevos problemas, muy significativos fueron los de tiempo de procesado y para ellos también propusimos diferentes soluciones, destacando sobre todas ellas una implementación de un sistema de estimación de movimiento de píxel entre *frames* contiguos, para la predicción de un número considerable de los mismos reduciendo así el número de *frames* a procesar en la red y con ello el tiempo de procesado.

Agradecimientos

Quiero expresar mis mas sincero agradecimiento a todas las personas que han hecho posible la realización de este proyecto, un proyecto basado en el procesado de imagen y la computación gráfica que no habría sido posible sin los conocimientos y experiencia del grupo Graphics and Imaging Lab de la Universidad de Zaragoza.

Por ello en primer lugar me gustaría mostrar mi gratitud al director del grupo, Diego Gutierrez, por permitirme trabajar y aprender durante los últimos 6 meses en este gran grupo de investigación.

A Belén Masiá y Ana Serrano, especialmente, por dirigir, coordinar y realizar conmigo este proyecto. Por ayudarme cada semana a construir lo que hoy es mi trabajo de fin de grado, y por brindarme la oportunidad de aprender de ellas, y de trabajar juntos durante todo este periodo.

A mis compañeros y amigos del grado de telecomunicaciones, pues con este proyecto se cierra una de las etapas mas importantes y significativas de mi vida, y ellos, su amistad y todo el tiempo que hemos compartido, son lo mejor que me llevo de esta etapa.

A todos los profesores del grado, por compartir conmigo y transmitirme sus conocimientos y su experiencia.

A mi familia y amigos, a mis padres, mis abuelos, tíos y primos, por su ayuda, motivación y cariño durante todo este tiempo.

A José Antonio Calvo, mi mentor, mi guía, mi tío y mi mejor amigo, por ser mi apoyo mas importante, por su ayuda y cariño.

A todos vosotros, muchas gracias.

Índice general

Índice de figuras

Capítulo 1

Introducción

1.1. Objetivo, alcance y contexto del proyecto

La transferencia de textura y estilo entre imágenes da forma a un campo artístico dentro del procesado digital de imagen. Esta transferencia propone la generación de una imagen a partir de otras dos, de forma que una de estas últimas aportará la estructura de contenido de la imagen nueva y la otra aportará el estilo o textura que tendrá la imagen final. En este caso se entiende como estilo, un conjunto identificable de rasgos (por ejemplo, tipo de trazo, color o luminosidad) que confieren cierta apariencia característica a una imagen.

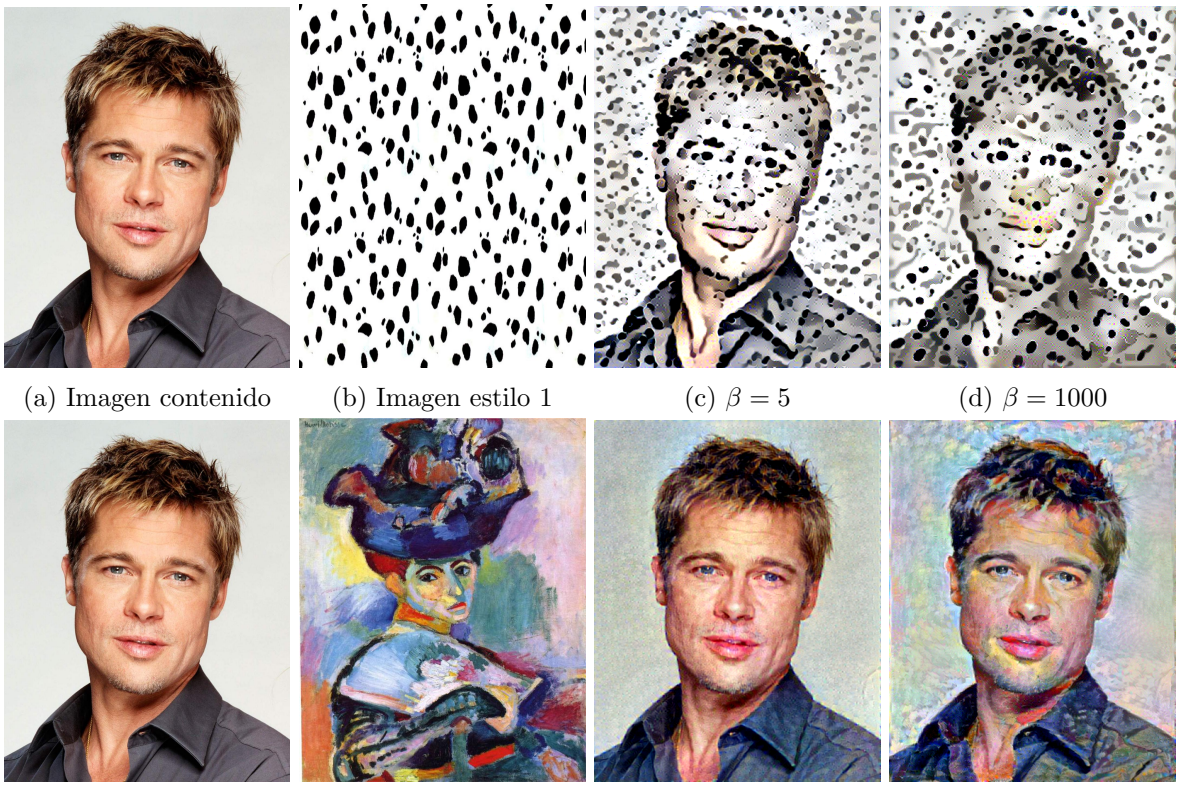
Así dentro de la disciplina del procesado de imagen se sitúa la tarea de la transferencia de textura y estilo entre imágenes. Esta tarea ha sido abordada desde diferentes puntos y a través de distintos métodos a lo largo del tiempo. El inicio del proyecto consistió, entre otras cosas, en realizar un estudio del estado del arte en este área. Estos métodos, difieren en planteamiento, programación y procesado pero todos comparten ciertos denominadores comunes. Así el fundamento básico de los mismos, consiste en la extracción de las características de estilo y textura de una imagen x, la extracción de los parámetros de la estructura de contenido de otra imagen y y la posterior reconstrucción de una tercera imagen z cuya estructura de contenido la aporta la imagen y estilizada mediante la información extraída de la imagen x.

De entre todas las opciones que nos encontramos, decidimos analizar la mas novedosa representada en el estado del arte del momento, la cual, utiliza redes neuronales convolucionales para la extracción de características de estilo y contenido.

Tras la aparición de las redes neuronales convolucionales entrenadas y utilizadas para la detección de patrones en imagen y clasificación de imágenes, nace la idea de la mano de Gatys et al [1], de utilizar dichas redes para intentar conseguir una transferencia de estilo entre 2 imágenes, aprovechando el filtrado en cada capa de dichas redes que a través de distintas operaciones convolucionales lleva a cabo la detección de diferentes patrones y a distintos niveles en imágenes.

El proyecto nace con el objetivo de conseguir, partiendo de estos sistemas de transferencia de estilo que hacen uso de redes neuronales convolucionales, un sistema que realice dichas transferencias de textura y estilo entre imágenes pero que permita una conservación considerablemente mayor y regulable de la información semántica o de contenido de la imagen original. Las transferencias conseguidas hasta el momento no se preocupaban de mantener dicha información y se produce una distorsión considerable en la información de contenido de la imagen a la que queremos transferir la textura o estilo. Además de este problema nos encontramos también con una importante imprecisión a la hora de seleccionar los parámetros del sistema en función de la salida que queríamos obtener. La imagen obtenida como salida del sistema varía gravemente en función de los parámetros seleccionados de la red, y no encontramos un

estudio que nos aportase información a cerca de cuales debíamos utilizar. Nos dimos cuenta que dependiendo de que imágenes introducíamos al sistema necesitábamos unos parámetros diferentes, por tanto, concluimos que era necesario realizar un estudio en esta dirección que permitiese predecir los parámetros que mas favorecen a nuestras exigencias. En la Figura (1.1) se muestra un ejemplo donde se puede observar la diferente influencia del parámetro de estilo β para dos imágenes estilo diferentes aplicadas sobre la misma imagen contenido.



(a) Imagen contenido (b) Imagen estilo 1 (c) $\beta = 5$ (d) $\beta = 1000$

(e) Imagen contenido (f) Imagen estilo 2 (g) $\beta = 5$ (h) $\beta = 1000$

Figura 1.1: Podemos observar como dando diferentes pesos al término que regula la transferencia de estilo (β) a diferentes imágenes estilo el resultado es notoriamente diferente.

Antes de continuar profundizando en la materia preservación de la información de contenido, es necesario aclarar, que no habiendo encontrado dicha concienciación en sistemas de transferencia de textura o estilo que hacen uso de redes neuronales convolucionales, si que existen intentos aplicados a otro tipo de transferencia de estilo, por ello nos centramos en el estudio de que hacía diferentes a estos sistemas y como podríamos plantear una solución.

Así pues, comenzamos un estudio profundo del estado del arte en materia de transferencia de estilos y texturas entre imágenes. Necesitábamos comprender con precisión los sistemas existentes, cuales eran y como funcionaban, además de intentar analizar las carencia o problemas que presentaban. Una de ellas era la ausencia de concienciación en preservar la información de contenido. Una vez detectada dicha carencia debíamos estudiar las diferentes posibilidades y opciones que podían conducirnos hacia una solución. En primer lugar se hizo un estudio en profundidad de las funciones que dentro de la red neuronal generaban la transferencia de estilo o textura, así pues, como se explicará en la Sección 4 llegamos a la conclusión de que no solo nos bastaba con ajustar dichas funciones, sino que era necesaria la implementación de otro sistema concurrente con la red neuronal convolucional, que hiciese un post-procesado de la imagen salida de la misma.

De esta manera, nos damos cuenta de que ajustando unos u otros parámetros de la red

se consiguen realizar diferentes transferencias de estilo que proporcionan imágenes salida con mayor o menor presencia de estilo en las mismas, la cual es inversamente proporcional a la preservación de la información de contenido, esto es, cuanto mas estilo sea transferido mas difuminada quedará la información de contenido, y viceversa (como podíamos observar en la Figura (1.1)). Dando lugar a dos casos extremos, uno de ellos una imagen con mucha presencia del estilo que estamos transfiriéndole y con una gran distorsión o pérdida de la información de contenido y el otro caso, sería una imagen con escasa presencia de dicho estilo y con una preservación de la información de contenido prácticamente intacta. Entre medio de estos casos extremos existe un amplio rango de pares estilo-contenido. Nos dimos cuenta de que la información que mas rápido desaparecía o se perdía era aquella perteneciente a detalles pequeños, como líneas finas, o pequeños objetos en la imagen, así como también los diferentes bordes y transiciones entre distintos objetos de la imagen, en cambio las áreas mas grandes y redundantes de la imagen, recibían la transferencia de estilo sin presentar una alta alteración del contenido de las mismas, como por ejemplo un suelo o el cielo. Visto esto, se plantea una solución que permite transferir el estilo a la imagen de contenido de forma selectiva en zonas, es decir, que permita transferir con mas intensidad a zonas mas grandes y redundantes y con mucha menos a los detalles mas pequeños. Para ello, planteamos un sistema que dividía la imagen contenido en diferentes zonas en función de la información contenida en ellos y a cada una se le sometería a una transferencia de estilo distinta, con diferente valor del parámetro que regula la transferencia de estilo (β). Dichas zonas recibirán el nombre de parches.

Posteriormente, una vez implementado el sistema mencionado, se estudian sus carencias y se incluyen mejoras. Uno de los problemas mas importantes, es la ausencia de una transición entre los parches generados con distintas intensidades de transferencia de estilo, como hemos mencionado anteriormente, al transferir a dos zonas colindantes intensidades de estilo distintas se produce en su frontera de píxeles una transición en escalón muy brusca que es incómoda para la vista. Así pues, se propone una solución para este problema, dicha solución es un método que detecta dichos píxeles pertenecientes a la frontera y mediante una serie de operaciones algebraicas se computa una media entre los píxeles frontera de las imágenes que participan en la misma, produciéndose así una transición mucho mas suave, haciendo así dicha frontera imperceptible para el ojo humano.

Es en este momento, cuando logramos unos resultados competentes con el estado del arte, no solo con aquellos que transfieren estilo con redes neuronales convolucionales, que como hemos mencionado desde el principio, no se centran en la máxima conservación de la información de contenido, sino también con otros que utilizando otros métodos de transferencia de estilo y textura si lo hacen poniendo el foco en esta preservación del contenido de la imagen que recibe el estilo o textura. Es entonces cuando nos planteamos la idea de someter esta sistema, hasta ahora aplicado solo a imágenes individuales, a una adaptación que permita la transferencia de estilo y textura a vídeo. En principio el problema a solucionar no dista mucho del original, pues en definitiva un vídeo no es mas que una sucesión de imágenes, así pues bastaría con extraer cada una de las imágenes o fotogramas (a partir de aquí utilizaremos el término en inglés *frames*) que componen el vídeo y aplicar a cada una de las mismas nuestro método de transferencia de estilo y textura y volver a componer el vídeo. No obstante, como se explicará en el Sección 7, nos tuvimos que enfrentar a varios problemas que este nuevo sistema introducía, y para los cuales propusimos diferentes soluciones que fueron analizadas y validadas, hasta llegar a una solución final, como queda explicado en el apartado mencionado anteriormente.

Finalmente, nos propusimos afrontar el problema del gran aumento del tiempo de procesado que el nuevo sistema de transferencia de textura o estilo a vídeo acarreaba. Cuando trabajábamos con imágenes individuales, este tiempo no era un gran problema, pues procesar una imagen en la maquina utilizada a lo largo del proyecto tenía un coste de unos 5 minutos aproximadamente. El problema llega con la introducción del vídeo, pues si por ejemplo

queríamos realizar una transferencia de una textura a un vídeo de 12 segundos de duración a 30 *frames* por segundo (*fps*), son 360 *frames* que procesar que multiplicado por 5 minutos que cuesta cada una hace un total de unas 30 horas. La solución propuesta también se explica en la Sección 7. Estos tiempos han sido conseguidos con la máquina utilizada para procesar el sistema, cuyas características son: procesador Intel(R)Core(TM) i7-4770 CPU@3.40 GHz, tarjeta gráfica NDIVIA GeForce GTX760 y memoria RAM 16 GB.

Como punto adicional del alcance del proyecto, resulta interesante señalar que se esta trabajando en la elaboración de un artículo científico a partir del trabajo realizado durante este proyecto.

1.2. Organización del proyecto

La memoria al igual que el proceso del proyecto sigue una organización escalonada desde lo mas general a lo mas específico, así en los primeros apartados se intentan explicar los conceptos mas generales utilizados referentes a redes neuronales convolucionales y se evoluciona progresivamente hacia las soluciones específicas implementadas para llegar a la solución buscada. Así en la Sección 2 se analiza el contexto mas actual y el estado del arte en cuanto a redes neuronales convolucionales utilizadas en transferencia de estilo y textura entre imágenes. En la Sección 4 se ahonda en el sistema de redes neuronales convolucionales utilizado finalmente. En la Sección 6 se explica la implementación de estos dos métodos, desde como surgieron las ideas, que necesidades había, que problemas resolvían hasta la implementación y el código de los mismos. En la Sección 5 se explican todos los métodos y conceptos puestos en práctica para llevar a cabo el estudio de las componentes frecuenciales de las imágenes estilo o textura y el posterior análisis de las correlaciones entre las salidas del sistema y la imagen contenido original, así como las diferentes motivaciones que nos condujeron a este estudio.

1.3. Planificación

En la realización de este proyecto han sido invertidos aproximadamente 6 meses. El proyecto se ha dividido en diferentes tareas, y cada una ha supuesto una inversión diferente de tiempo. Algunas tareas se llevaron a cabo de forma paralela, no obstante la mayoría se afrontaron de forma sucesiva. A continuación analizaremos y expondremos de forma aproximada, como fueron sucediendo dichas tareas.

Las tareas realizadas son las siguientes: en primer lugar era necesaria la documentación y aprendizaje del marco teórico de redes neuronales, para ello se realizó un curso *on-line* impartido por la Universidad de Stanford, *Convolutional Neural Networks for Visual Recognition* [2], en dicho curso se aprendía a crear desde cero una red neuronal convolucional para reconocimiento de patrones en imagen. Una vez terminado el curso, se procedió a la documentación y estudio del estado del arte en los ámbitos de redes neuronales convolucionales aplicadas a reconocimiento de patrones en imagen y transferencia de texturas o estilos entre imágenes a través de diversos métodos. Posteriormente y tras haber estudiado estos sistemas, se comenzó con la instalación de la red neuronal convolucional de 19 capas implementada por el *Visual Geometry Group (VGG)* de la Universidad de Oxford. El siguiente paso fue utilizar el sistema implementado por Gatys et al. para transferencia de estilo entre imágenes y que hacía uso de la misma red neuronal mencionada. Se invirtieron entonces, varias semanas en la puesta a punto del sistema, se realizaron las primeras pruebas con distintas imágenes de entrada, distintos pares de imágenes-estilo e imágenes-contenido. Se hizo después un estudio-análisis de como avanzaban y evolucionaban las imágenes dentro de la red, para ello realizamos un análisis de las funciones de pérdidas en las distintas capas y en función del número de iteraciones que se daban, tanto de contenido ($L_{content}$), como de estilo (L_{style}) y totales (L_{total}). Tras adquirir una

clara idea del funcionamiento de la red y basándonos en un primer análisis cuantitativo de la evolución de la imagen procedemos a la búsqueda y adaptación de los parámetros del sistema con el objetivo de intentar obtener como salida una transferencia de textura o estilo bajo la concienciación de la preservación de la información de contenido. Tras un estudio extenso de las diferentes posibilidades que podíamos obtener con el sistema y tras tomar ciertas decisiones de diseño, se decide implementar un sistema de transferencia selectiva de textura y estilo para aplicarlo como post-procesado a la salida de la red, dicho sistema se explica en la Sección 6.

Tras la realización de dicho sistema se observan ciertas imperfecciones o deficiencias y se procede a la búsqueda e implementación de una solución a las mismas, dicha solución queda explicada en la Sección 6.3. Con la introducción del transferencia selectiva de textura y estilo se hace necesaria la selección de las distintas imágenes que se utilizarán en el mismo (estas imágenes son todas ellas salida del sistema de transferencia de textura o estilo a través del uso de una red neuronal convolucional). Para ello se lleva a cabo entonces un estudio de las imágenes estilo o textura y de como varía la salida final en función de la variación del parámetro β (este parámetro condiciona el proceso de transferencia de textura o estilo, dándole en función de su valor, mayor o menor peso a la presencia de la información de estilo o textura en la imagen final y por tanto condicionando también la pérdida o distorsión de la información de contenido. Este parámetro será explicado en la Sección 4. El objetivo era hacer una clasificación de dichas imágenes estilo para intentar poder hacer una predicción de dicho comportamiento, el estudio, explicado en la Sección 5 consiste, de forma resumida, en un análisis de las distribuciones de las componentes frecuenciales de las imágenes estilo o textura, así tras hacer pasar el mismo par imagen-contenido, imagen-estilo por la el sistema de transferencia de textura o estilo con diferentes valores de β se procede a un cálculo de la correlación entre dicha salida y la imagen contenido original, así podíamos ahora saber en función de las distribuciones de las componentes frecuenciales de una imagen estilo cuales serían los valores de β necesarios para obtener las imágenes salida óptimas que después se utilizarán como entrada del sistema transferencia selectiva de textura y estilo para conseguir los resultados finales buscados.

Finalmente una vez se obtienen resultados competitivos con el estado del arte actual, se decide intentar adaptar la implementación nuestro sistema con el objetivo de conseguir una transferencia de textura o estilo a un vídeo. Esto dará lugar a la aparición de nuevos y diferentes problemas, para los cuales plantearemos también nuevas soluciones, que serán analizadas, puestas en práctica y evaluadas y si lo requieren sufrirán las modificaciones necesarias, hasta finalmente conseguir el sistema y los resultados buscados.

En la Figura (1.2) se muestra un diagrama de Gantt donde se representa de forma cronológica la planificación llevada a cabo.

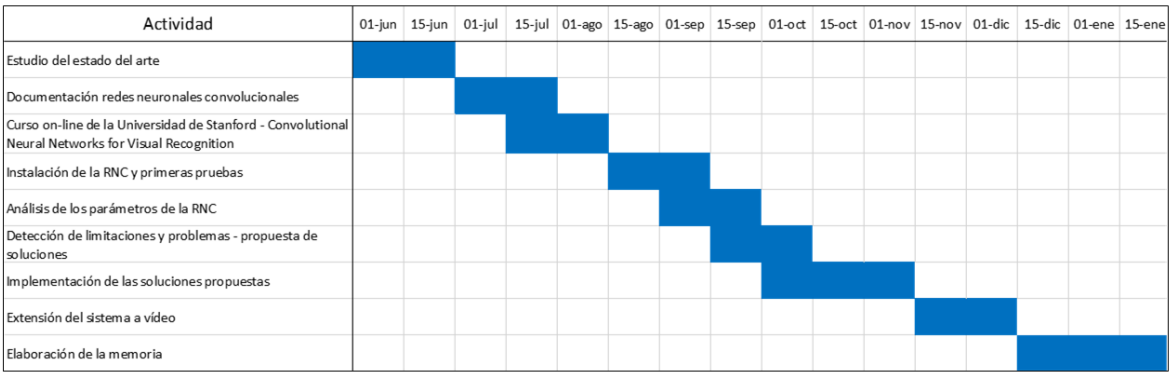


Figura 1.2: Diagrama de Gantt.

Capítulo 2

Trabajo relacionado

Dividimos este apartado en diferentes puntos, en función del tema principal abordado en cada uno de ellos, pues en este proyecto se han acometido estudios, análisis y resolución de problemas en diferentes disciplinas en el ámbito de la imagen computacional y el procesado de imagen y vídeo.

En primer lugar cabe mencionar el carácter innovador del principio fundamental del proyecto -búsqueda de la máxima preservación de la información de contenido en un sistema de transferencia de textura o de estilo a través de una red neuronal convolucional- y por ello no existe un gran número de estudios y pruebas en este campo, si bien, algunos grupos de investigación de diferentes Universidades han abordado problemas similares. Así pues mencionaremos primeramente al grupo de Gatys et al. [1], ellos desarrollaron la idea de utilizar una red neuronal convolucional, entrenada para reconocimiento de patrones y clasificación de imagen, para llevar a cabo una transferencia de estilo o textura entre imágenes. La diferencia entre nosotros y ellos es que ellos no ponen el foco en la preservación de la información de contenido y por ello líneas, bordes y detalles quedan totalmente alterados en sus resultados. Podemos ver este estudio de Gatys en su artículo *Image Style Transfer Using Convolutional Neural Networks*. En segundo lugar presentamos al grupo de Justin Johnson [3] de la Universidad de Stanford, ellos fueron los que aportaron en código libre, una implementación del sistema que aparecía en el artículo de Gatys. Así pues sus resultados son muy similares a los de Gatys y por tanto tampoco hay ningún método incorporado con la intención de preservar la información semántica de la imagen que aporta la información de contenido. Es importante señalar que para la implementación del sistema hacen uso de la red neuronal convolucional que Gatys proponía en su trabajo, y es la VGG-19 (Una red desarrollada por el Visual Geometric Group de la Universidad de Oxford, [4])

Por otro lado Rujie Yin [5] en Enero de 2016 publica un artículo (*Content-Aware Neural Style Transfer*) en el que intenta realizar una transferencia de textura a un objeto con el objetivo de preservar en la mayor medida de lo posible la información semántica de la imagen de partida. Ellos utilizan también el mismo sistema de transferencia de textura (la red neuronal convolucional de 19 capas desarrollada por el VGG de la Universidad de Oxford) y su sistema se centra en la división de la imagen contenido en varias partes correspondientes cada una a un mismo objeto o campo semántico y llevan a cabo la transferencia de textura de forma individual en cada una de dichas partes, utilizando variaciones del sistema en cada caso, finalmente proponen la unión de todas ellas asegurando una transferencia mucho mas precisa y de mayor resolución. Es importante señalar el hecho de que solo proponen un ejemplo, es decir, todo su artículo se basa en una sola transferencia de textura a una imagen, luego no podemos comparar nuestros resultados con los suyos.

El caso mas parecido al nuestro es el que Frigo et al. [6], de Technicolor, Research &

Innovation, France, ellos también proponen, en su artículo *Split and Match: Example-based Adaptive Patch Sampling for Unsupervised Style Transfer*, la introducción de un sistema de transferencia selectiva de textura o estilo. Hacen uso también de una descomposición en árbol cuaternario, mediante la cual se consigue la división de la imagen en parches de tamaño variable en función de la diferencia entre dos píxeles del mismo parche y un umbral. Es importante señalar que no especifican como llevan a cabo la transferencia de textura, aunque si indican, que utilizan un sistema de transferencia de textura o estilo que no hace uso de redes neuronales convolucionales.

Los casos mencionados hasta este punto son los mas cercanos a nuestro proyecto, si bien, otros trabajos del mismo campo y que nos han servido en algún momento del proyecto como fuente de información y complemento, son los que mencionaremos y explicaremos brevemente a continuación.

Destacamos en primer lugar el estudio predecesor al *Image Style Transfer Using Convolutional Neural Networks* de Gatys, dicho estudio, con título *Texture Synthesis Using Convolutional Neural Networks* [7], fue desarrollado por el propio Gatys y el objetivo del mismo era crear un modelo que extrajese y generase texturas partiendo de determinadas imágenes haciendo uso de redes neuronales convolucionales entrenadas para clasificación de imágenes basadas en el reconocimiento de patrones en las mismas. Observaron que al hacer pasar una imagen a lo largo de las capas de la red, si extraían las imágenes que se generaban en dichas capas tras pasar por los filtros correspondientes, las imágenes que extraían iban capturando las propiedades estadísticas que representan la textura de una imagen haciéndolas mas explícitas en las capas mas profundas de la red. Es por tanto, este trabajo claramente la precuela de lo que conseguirían con el siguiente, pues a a partir de este estudio estadístico de como se iban extrayendo en cada capa las propiedades de las texturas de las imágenes que metían en la red neuronal convolucional entrenada para reconocimiento de patrones y clasificación, consiguieron desarrollar el sistema que finalmente conocemos, que consigue transferir dicha textura o estilo de una imagen a otra aplicando las propiedades analizadas de forma paralela a ambas imágenes.

El equipo de Dimitry Ulyanov [8], de Skolkovo Institute of Science and Technology & Yandex, Russia, en su trabajo, *Texture Networks: Feed-forward Synthesis of Textures and Stylized Images*, realizaron un estudio del coste computacional del sistema que proponía Gatys. Ellos proponen el uso de un tipo de red neuronal diferente, que recibe el nombre de F*eed-forward convolutional networks*, aunque sigue siendo una red entrenada para reconocimiento de patrones en imagen y su posterior clasificación, guarda ciertas diferencias con la utilizada por el equipo de Gatys, las cuales según explican en dicho trabajo, se basan fundamentalmente en el tipo de funciones de pérdidas mas eficientes y que suponen un menor tiempo de procesado. Así, afirman reducir considerablemente el coste temporal de todo el proceso.

En otra línea, queremos mencionar algunos de los trabajos que nos encontramos mientras realizábamos el estudio del arte y que tienen un significado característico y se debe a la doble vertiente que presentan y que nos condujo en ocasiones diferentes a encontrarnos con ellos ya que afrontan dos problemas que nosotros enfrentamos en diferentes momentos del proyecto.

Uno de ellos es el estudio llevado a cabo por Efros y Freeman [9], de nombre, *Image Quilting for Texture Synthesis and Transfer*, este estudio nos lo encontramos por primera vez cuando llevamos a cabo el estudio del estado del arte en transferencia de texturas y estilos entre imágenes, la diferencia fundamental es que utilizan otro método de transferencia totalmente distinto, no hacen uso de redes neuronales convolucionales, por lo que no compartimos las competencias fundamentales del proyecto, pero pensamos que resulta interesante mencionar otros métodos. Dicho método se basa en lo que ellos mismos definen como *quilting* que en español, sería algo así como 'bordado' o 'cosido a máquina', mediante el cual, extrayendo mapas de correspondencia entre los píxeles de dos imágenes consiguen realizar una transferencia

de textura o estilo alternativa. Por otro lado, nos encontramos con este trabajo cuando nos enfrentamos al problema de las transiciones entre parches, que resolvimos con nuestro método *blending*, pues Efros y Freeman también se enfrentaron al mismo y lo resolvieron de una forma muy similar, ellos proponen un sistema llamado *Minum error boundary cut*, el cual consiste, al igual que nuestro *blending*, en extender los parches que comparten un borde, provocando un solapamiento entre ellos y sustituir la zona de convergencia por aquella que suponga la menor diferencia entre los píxeles mas próximos a la frontera de las regiones que la comparten.

Encontramos otros trabajos en la misma línea como el de Hardik Modi [10], de título, *Multi Order Minimum Error Boundary Cut Patch Based Algorithm: Innovative Image Inpainting Application* de Charotar University of Science and Technology, India, el cual propone una extensión al trabajo de Efros y Freeman, que se fundamenta en la ampliación de la zona de solape entre parches y en una variación de los cálculos establecidos sobre la misma.

En la misma línea, referente a modelado de texturas, encontramos el trabajo de Javier Portilla y Eero Simoncelli [11], *A Parametric Texture Model Based on Joint Statistics of Complex Wavelet Coefficients* del Center for Neural Science, and Courant Institute of Mathematical Sciences, New York University. Este trabajo escrito en el año 2000, nos presenta otra forma de síntesis de texturas. Desarrollan un algoritmo basado en un modelo parametrizado por un conjunto de funciones estadísticas de imagen basadas en coeficientes de onda compleja, correspondientes a funciones básicas de zonas espaciales adyacentes, orientaciones y escalas.

Otro ejemplo de un diferente intento de síntesis de textura, es el trabajo que desarrollaron David J. Heeger [12], Stanford University y James R. Bergeny, SRI David Sarnoff Research Center, con título *Pyramid-Based Texture Analysis/Synthesis*. En este estudio describen una técnica de correspondencia entre el aspecto o apariencia de una textura dada. Tal y como describen en su estudio, el método se inicia con la imagen textura objetivo y una imagen ruido blanco gaussiano. El algoritmo modifica la imagen ruido blanco con el objetivo de que esta tome la apariencia de la imagen textura *input*. Para ello hacen uso de un método que llaman *image pyramid* y de un *match-histogram*, que busca correspondencias entre los histogramas de ambas imágenes. Este trabajo se llevó a cabo en la misma década que el anteriormente comentado, concretamente en 1995 y como podemos observar guarda una relación muy cercana al método propuesto por Gatys et al., en lo referido a partir de una imagen ruido blanco gaussiano que será la que vaya tomando al apariencia de nuestra imagen *input*, la diferencia está en como se van extrayendo esas características de imagen pues Gatys propone hacer uso de las distintas funciones que aparecen en las redes neuronales convolucionales para extracción de patrones en imagen a diferencia de las diferentes funciones que estos métodos de la década de los noventa utilizaban.

Hemos incluido estos últimos métodos, porque sin tener relación con los métodos de transferencia de textura y estilo que nosotros hemos utilizado, presentan diferentes formas de llevar a cabo dicha tarea y son un buen ejemplo de como diferentes intentos en el campo de la trasferencia y síntesis de texturas han ido desarrollándose y evolucionando hasta llegar a los últimos modelos que actualmente conocemos como los que hacen uso de redes neuronales convolucionales.

Hasta aquí hemos descrito el trabajo relacionado referente a imagen, no obstante queríamos dedicarle una sección a parte a aquellos trabajos referidos a la estilización de vídeo. Esto es así, debido a que hemos encontrado pocos estudios que abarcasen este proceso, de hecho el único que hemos podido ver es el que encontramos en *www.deepart.io* una web que nace de los trabajos de Gatys et al. y que esta dirigida por ellos. En la misma encontramos que el trabajo de adaptación a vídeo fue llevado a cabo por Manuel Ruder y Thomas Brox [13].

Capítulo 3

Introducción a las redes neuronales convolucionales

Esta sección contiene una breve explicación de carácter general de qué son las redes neuronales y redes neuronales convolucionales (nos referiremos a las mismas en este apartado como RNC). Se explicarán y se definirán diferentes conceptos que se utilizarán en los capítulos posteriores. Para su desarrollo se ha utilizado información de los artículos [14] y [15], así como del curso on-line impartido por la Universidad de Stanford, *Convolutional Neural Networks for Visual Recognition* [2].

Una RNA (Red Neuronal Artificial) es un modelo matemático inspirado en el comportamiento biológico de las neuronas y en la estructura del cerebro. Esta también puede ser vista como un sistema inteligente que lleva a cabo tareas de manera distinta a como lo hacen las computadoras actuales. Si bien estas últimas son muy rápidas en el procesamiento de la información, existen tareas muy complejas, como el reconocimiento y clasificación de patrones, que demandan demasiado tiempo y esfuerzo aún en las computadoras más potentes de la actualidad, pero que el cerebro humano es más apto para resolverlas, muchas veces sin aparente esfuerzo. El cerebro puede considerarse un sistema altamente complejo. Su unidad básica, la neurona, esta masivamente distribuida con conexiones entre ellas (se calcula que hay aproximadamente 10 billones de neuronas en la corteza cerebral y 60 trillones de conexiones neuronales).

Si bien hay distintos tipos de neuronas biológicas, en la Figura (3.1) se muestra un esquema simplificado de un tipo particular que es muy común. Vemos que la misma está compuesta por:

1. El cuerpo central, llamado soma, que contiene el núcleo celular
2. Una prolongación del soma, el axón
3. Una ramificación terminal, las dendritas
4. Una zona de conexión entre una neurona y otra, conocida como sinapsis

La función principal de las neuronas es la transmisión de los impulsos nerviosos. Estos viajan por toda la neurona comenzando por las dendritas hasta llegar a las terminaciones del axón, donde pasan a otra neurona por medio de la conexión sináptica. La manera en que respondemos ante los estímulos del mundo exterior y nuestro aprendizaje del mismo está directamente relacionado con las conexiones neuronales del cerebro, y las RNAs son un intento de emular este hecho.

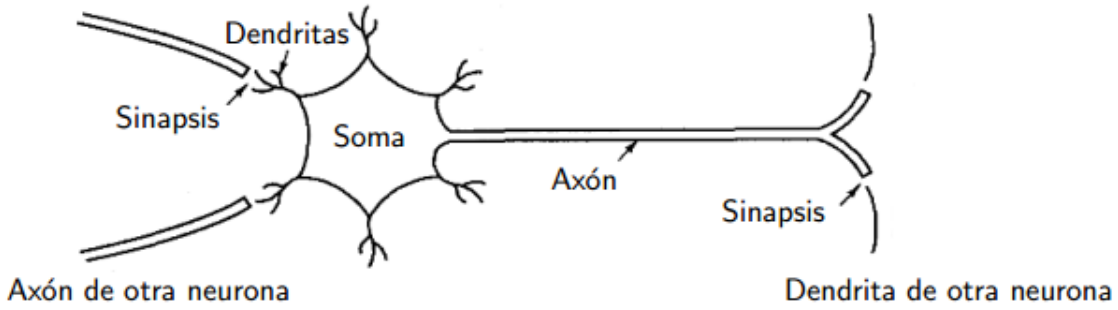


Figura 3.1: Esquema básico de una neurona biológica.

3.1. Modelo neuronal de McCulloch-Pitts

Tras la breve introducción del concepto de red neuronal y su analogía con las neuronas biológicas, vamos a introducir el concepto de red neuronal artificial y comenzaremos por explicar el primer modelo que se llevo a cabo.

El primer modelo matemático de una neurona artificial, creado con el fin de llevar a cabo tareas simples, fue presentado en el año 1943 en un trabajo conjunto entre el psiquiatra y neuroanatomista Warren McCulloch y el matemático Walter Pitts. Un ejemplo de modelo neuronal con dos entradas x e y es representado en la Figura (3.2).

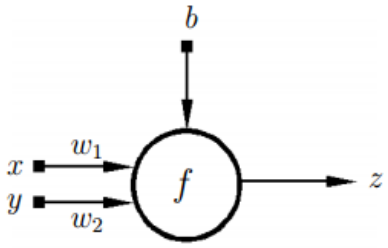


Figura 3.2: Esquema básico de una neurona artificial.

El mismo consta de:
1. Las entradas x e y
2. Los pesos sinápticos $w1$ y $w2$ correspondientes a cada entrada
3. Un término aditivo b
4. Una función de activación f
5. Una salida z

Las entradas x e y son el estímulo que la neurona artificial recibe del entorno que la rodea, y la salida z es la respuesta a tal estímulo. La neurona se adapta al medio circundante y aprende de él modificando el valor de sus pesos sinápticos $w1$ y $w2$ y su término aditivo b. Estos son conocidos como los parámetros libres del modelo, pues los mismos pueden ser modificados y adaptados para realizar una tarea determinada. En este modelo, la salida neuronal z está dada por la Ecuación 3.1.

$$z = f(w1 * x + w2 * y + b) \tag{3.1}$$

La función de activación f es seleccionada de acuerdo a la tarea realizada por la neurona.

3.2. Red neuronal convencional

Tras la creación del primer modelo de red neuronal, se desarrollaron diferentes modelos mas avanzados que conforman la evolución de las redes neuronales a lo largo de su historia hasta los modelos actuales (por ejemplo la creación del perceptrón de Frank Rosenblat(1958) o las investigaciones de aprendizaje automático llevadas a cabo por Marvin Minsky y Seymour Papert (1969)).

Esta sección se centra en una introducción básica de las redes neuronales convencionales tal y como se entienden en la actualidad.

Las RNAs están compuestas de gran cantidad de procesadores conectados entre sí y actuando en paralelo. Una red neuronal se compone de un gran número de neuronas que se conectan entre sí y que están distribuidas en diferentes capas. El comportamiento de la red estará entonces determinado por su topología, los pesos de las conexiones y la función característica de las neuronas.

Veamos ahora una definición mas concreta y técnica:
Una neurona, o unidad procesadora, sobre un conjunto de nodos N, es una tripleta (X, f, Y), donde X es un subconjunto de N, Y es un único nodo de N y f es una función neuronal (también llamada función de activación) que calcula un valor de salida para Y basado en una combinación lineal de los valores de las componentes de X, es decir, $Y = f(\sum_{x_i \in X} w_i * x_i)$. Los elementos X, Y y f se denominan conjunto de nodos de entrada, conjunto de nodos de salida, y función neuronal de la unidad neuronal, respectivamente.

Siguiendo la ilustración de la Figura 3.3, por ejemplo, el nodo X_4 tendría como entradas los nodos X_1, X_2 y X_3. Dicho nodo X_4, generaría una salida y_4 que conformaría parte de la información de entrada de los nodos X_7 y X_8.

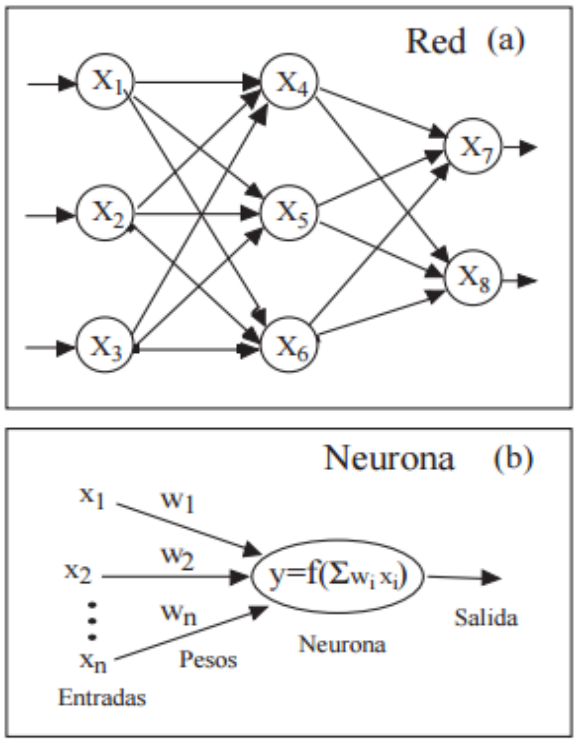


Figura 3.3: Ejemplo red neuronal(a) y conexiones en una neurona aislada (b). Cada uno de los nodos en (a) tendría la forma de una neurona como la mostrada en (b).

3.2.1. Aprendizaje y prueba

Existen dos fases en toda aplicación de las redes neuronales: la fase de aprendizaje o entrenamiento y la fase de prueba.

1. Fase de Aprendizaje: Uno de los conceptos mas importantes ligado a las redes neuronales es el proceso de aprendizaje de la red. Este proceso es el que definirá y caracterizará la red una vez este completo. Una característica de las redes neuronales es su capacidad de aprender. Aprenden por la actualización o cambio de los pesos sinápticos que caracterizan a las conexiones (w_i). Los pesos son adaptados de acuerdo a la información extraída de los patrones de entrenamiento nuevos que se van presentando. Normalmente, los pesos óptimos se obtienen optimizando (minimizando o maximizando) alguna función de energía. Por ejemplo, un criterio popular en el entrenamiento supervisado (ver Sección 3.2.1) es minimizar el error cuadrático medio entre el valor deseado y el valor de salida de la red.

2. Fase de Prueba: Una vez calculados los pesos de la red, las neuronas de la última capa se comparan con la salida deseada para determinar la validez del diseño.

Tipos de entrenamiento

Existen, fundamentalmente, dos tipos de entrenamiento para las RNA, el entrenamiento supervisado y el entrenamiento no supervisado.

Supervisado: Los datos están constituidos por varios patrones de entrada y de salida. El hecho de conocer la salida implica que el entrenamiento se beneficia de la supervisión de un maestro (Figura 3.4).

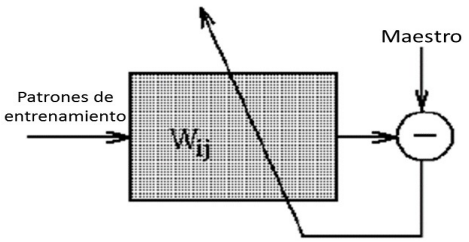


Figura 3.4: Entrenamiento supervisado.

No supervisado: Para los modelos de entrenamiento no supervisado, el conjunto de datos de entrenamiento consiste solo en los patrones de entrada. Por lo tanto, la red es entrenada sin el beneficio de un maestro. La red aprende a adaptarse basada en las experiencias recogidas de los patrones de entrenamiento anteriores.

3.3. Red neuronal convolucional

Las redes neuronales convolucionales son muy similares a las redes neuronales convencionales que se han explicado en las secciones anteriores. Lo que las diferencia es la asunción explícita de que las *input*s son imágenes. Esto conlleva diferencias en la arquitectura y en las funciones. El objetivo fundamental de estas redes es el reconocimiento de objetos en imagen a través de la detección de patrones para su posterior clasificación. Dicho reconocimiento de patrones se basa fundamentalmente en operaciones mediante filtros de convolución, que se llevan a cabo en las diferentes capas de una RCN, sobre la imagen *input* que va a ser analizada y clasificada.

3.3.1. Vista global de la arquitectura

Las redes neuronales convencionales recibían como *input* un único vector que se transformaba a lo largo de las diferentes capas de la red. Estas capas, compuestas por un conjunto de neuronas quedan conectadas entre sí, de forma que cada neurona de una capa determinada está conectada con todas las neuronas de la capa anterior como veíamos en la Figura (3.3). La última capa recibe el nombre de *output layer* y nos da los *scores* o probabilidades de cada clase. Estas redes neuronales convencionales presentan graves limitaciones cuando las *input*s son imágenes, y es aquí donde aparecen las redes neuronales convolucionales, con el objetivo de tratar esta limitación y aportar una solución óptima. Las redes neuronales convolucionales construyen su arquitectura de forma similar a las convencionales pero dado el hecho de que las *input* son imágenes se construyen con ciertas diferencias. En particular, a diferencia de las anteriores, las capas de una RNC tienen las neuronas organizadas en 3 dimensiones: ancho, alto y profundo. Por ejemplo, si nuestras imágenes *input* son de dimensiones 32x32x3, el ancho y el alto serían 32 y la profundidad sería 3.

Otra diferencia importante de las RNC con respecto a las convencionales es que en éstas las neuronas de cada capa no están conectadas con todas las neuronas de la capa anterior, ya que resulta poco práctico y puede llevar a un sobreajuste, que es el efecto de sobreentrenar un algoritmo de aprendizaje con unos ciertos datos para los que se conoce el resultado deseado. Cuando un sistema se entrena demasiado (se sobreentrena) o se entrena con datos extraños, el algoritmo de aprendizaje puede quedar ajustado a unas características muy específicas de los datos de entrenamiento que no tienen relación causal con la función objetivo.). Así se conectan solamente con una parte determinada de la capa anterior. El alcance espacial de esta conexión está determinado por un parámetro llamado campo receptivo de la neurona.

En la Figura (3.5) se muestra un ejemplo comparativo de las conexiones y dimensiones de una red neuronal convencional y una convolucional.

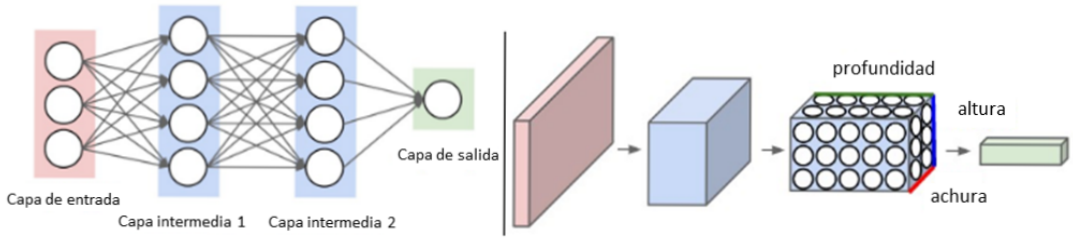


Figura 3.5: Comparativa de un esquema básico de una red neuronal convencional (izquierda) y una RNC (derecha). En el podemos observar fundamentalmente el rasgo característico de las RNC y es la componente de profundidad. Dicha profundidad no debe confundirse con la profundidad referida al número de capas que compone una RNC, sino que, como se explica en la Sección 3.3.2, es consecuencia de los diferentes procesos de convolución que se dan en las capas de la red.

3.3.2. Capas utilizadas en las RNC

Como hemos descrito anteriormente, una RNC es una secuencia de capas, y cada capa transforma un volumen de activaciones en otro a través de una serie de funciones diferenciables. A continuación vamos a describir brevemente las capas mas importantes que forman una RNC, más adelante se llevará a cabo una explicación más detallada de las capas más importantes.
1. INPUT: conformada por la imágenes de entrada, las cuales están compuestas por 3 dimensiones, ancho, alto y profundo (este último correspondiente a los canales de color RGB), por

ejemplo, imágenes de [32x32x3].
2. CONV: es la capa convolucional. Es la capa fundamental del proceso, en ella se computan las operaciones entre los pesos que contiene dicha capa y la región determinada del volumen de datos de *input* con el que esté conectada la capa. Gracias a los filtros de estas capas se consigue llevar a cabo el proceso de detección de patrones en las imágenes. En cuanto a las dimensiones de salida, si por ejemplo, utilizamos 12 filtros, la salida tendrá las dimensiones [32x32x12] (el cambio de tamaño en la tercera dimensión, se debe a una reducción a una única dimensión del canal RGB y su posterior transformación en 12 activaciones correspondientes cada una a cada uno de los 12 filtros utilizados).
3. RELU: la función de esta capa consiste en aplicar un umbral o límite a los valores de salida de la capa CONV. Por ejemplo, max(0, x), donde x sería el valor de umbral determinado. Esta capa no produce cambios en las dimensiones.
4. POOLING: esta capa tiene la función de realizar un submuestreo en las dimensiones alto y ancho de los datos que le llegan como *input*, por ejemplo, si el dato *input* tiene dimensiones [32x32x12], una posible *output* tendría las dimensiones [16x16x12].
5. FC (i.e. *fully-connected*) esta capa es la encargada de computar las probabilidades de cada clase, es la última capa de la red y la única que si que está conectada a todas las neuronas de la capa anterior. El *output* de esta capa consistirá en un vector con tantas componentes como clases hayamos determinado y dicho vector contendrá los valores de probabilidad de que la imagen *input* que hayamos seleccionado pertenezca a una clase u otra.

De este modo, las RNC transforman la imagen *input* original en un vector de probabilidades de clases.

En la Figura (3.6) se muestra un ejemplo de una RNC con todas sus capas, donde la imagen *input* es un coche y las clases definidas para la red son coche, camión, avión, barco y caballo. Como vemos en la imagen, la red ha sido capaz de extraer la información necesaria para determinar que la imagen *input* con mayor probabilidad pertenece a la clase coche.

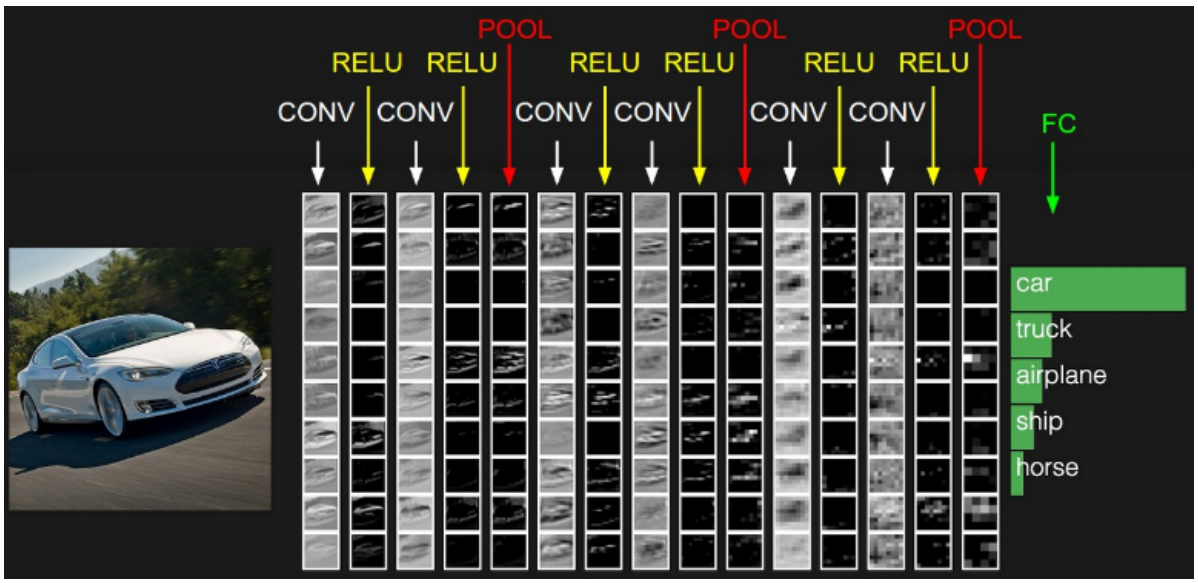


Figura 3.6: Ejemplo de una RCN donde se pueden observar las diferentes capas que conforman el procesado a lo largo de la red. Al final se muestra la capa de decisión, donde aparecen los resultados (*scores*) de cada clase.

Cada una de las imágenes filtradas en cada capa reciben el nombre de activaciones.

Capa convolucional

En primer lugar analizamos que ocurre en las capas convolucionales de una RNC. Estas capas son las que dan nombre a este tipo de redes neuronales y como su propio nombre indica implican un proceso de convolución. Estas capas están compuestas por una serie de filtros que se aplican, mediante el método de convolución de imagen, a la imagen *input* de la red que va a ser analizada y que queremos clasificar. Recordemos que el objetivo es clasificar la imagen *input* dándole una serie de pesos de probabilidades para cada una de las clases para las que ha sido entrenada la red.

Es fundamental destacar la distribución de los tipos de filtros a lo largo de la red. Estos filtros se distribuyen de forma que al principio de la red se sitúan aquellos encargados de la detección de las características más generales y de bajo nivel de la imagen. Al final de la red, se sitúan los filtros encargados de detectar las características de más alto nivel (los detalles mas pequeños y que describen, normalmente, los detalles más característicos de la imagen). Ejemplificando lo anterior con la imagen de un retrato de una persona, las características de bajo nivel que detectarían los filtros del principio de la red serían, por ejemplo, la forma y contorno de la cabeza o el fondo de la imagen. Siguiendo con el mismo ejemplo, las características de alto nivel se corresponderían, por ejemplo, con la detección de los ojos, las pestañas, la boca y la nariz, detalles que caracterizan y diferencian una persona de una animal o de cualquier otro objeto.

Presentamos en la Figura (3.7) un ejemplo de los filtros comentados en el párrafo anterior.

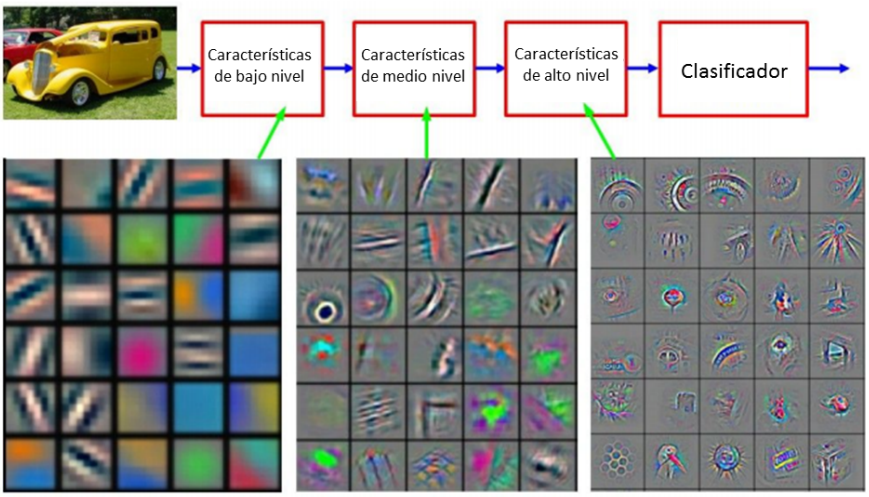


Figura 3.7: Ejemplo básico de tres capas convolucionales. Se observa como las capas de bajo nivel (izquierda) detectan características mas generales que las capas de alto nivel (derecha), que detectan las características más específicas.

Capa de *pooling*

La aportación principal de estas capas es hacer las activaciones mas pequeñas y manejables. El proceso es el siguiente:

1. Se acepta un volumen de tamaño W_1 * H_1 * D_1
2. Se requieren dos parámetros que definen la capa de *pooling*:
   2.1 La extensión espacial, F.
   2.2 El paso, S.
3. Se genera el nuevo volumen, de tamaño W_2 * H_2 * D_2, donde:
   3.1 $W_2 = (W_1 - F)/S + 1$
   3.2 $H_2 = (H_1 - F)/S + 1$
   3.3 $D_2 = D_1$

En la Figura (3.8) se muestra un ejemplo de un proceso de *pooling* sobre una activación de forma aislada.

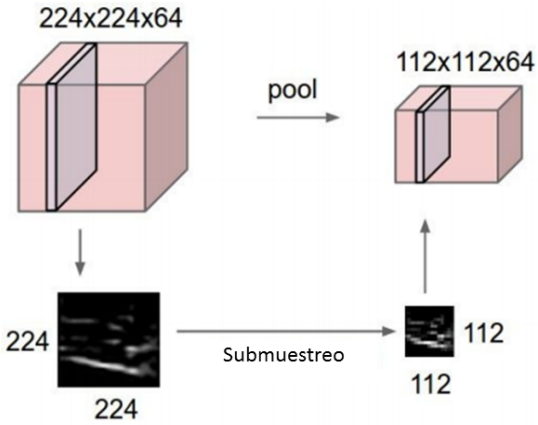


Figura 3.8: Ejemplo de la aplicación de una capa de *pooling* sobre una activación aislada. En el se puede observar como se reduce el tamaño de la activación. Los valores de la extensión espacial y el paso, en este ejemplo, serían respectivamente: $F = 0$ y $S = 1$.

Fully conected layer

La capa de decisión final (mas conocida por su término en inglés *Fully conected layer*), representa la última capa de una RNC. Se caracteriza por, a diferencia de las demás, tener todos sus nodos conectados a la capa que le precede. En ella se procede al cálculo de las probabilidades correspondientes con cada clase. Un ejemplo muy representativo es el mostrado en la Figura (3.6).

Capítulo 4

Sistema principal de transferencia de textura y estilo

Como ya se ha comentado anteriormente, la introducción de las redes neuronales convolucionales en el mundo de los gráficos computacionales ha supuesto una nueva herramienta para llevar a cabo transferencia de texturas o estilos entre imágenes.

Así pues, en este apartado se explica la red neuronal convolucional que se ha utilizado en el proyecto. Se desarrolla también un análisis en profundidad de como esta compuesta dicha red y del método empleado para llevar a cabo nuestro sistema de transferencia de estilo y textura.

Cabe señalar que, como se comentó en la Sección 1, en este apartado se explica y se analiza de forma minuciosa las diferentes partes que componen la red neuronal y su principio de funcionamiento así como el sistema de procesado que utiliza la red, si bien, todos los métodos implementados que forman parte del procesado introducido después de la red se explicarán mas adelante.

Comenzaremos introduciendo la red neuronal convolucional que se ha utilizado. Esta ha sido la conocida, extendida y ampliamente utilizada red 19-VGG, una red de 19 capas creada por el *Visual Geometry Group* [4] de la Universidad de Oxford. Dicha red, consiste en una red entrenada para la detección de patrones en imágenes y está compuesta por 16 capas convolucionales y 3 *fully conected layers* (capas finales de decisión), además de las distintas capas de *pooling* y *relu* intercaladas entre ellas. Esta red, como comentamos, esta entrenada para detección de patrones en imagen, es una red utilizada por tanto, para sistemas de clasificación de imágenes, si bien como explicaremos a continuación, se descubrió que podían ser utilizadas para llevar a cabo modelos de transferencia de textura y estilo entre imágenes.

Un grupo de investigación del Centre for Integrative Neuroscience, Tübingen, Alemania, encabezado por Leon A. Gatys [1] puso de manifiesto la posibilidad de utilizar redes de este tipo para llevar a cabo transferencia de estilo entre imágenes. La propuesta que dan para llevar a cabo dicha transferencia es la siguiente. Se conoce que estas redes neuronales se componen de diferentes capas en las cuales se llevan a cabo una serie de operaciones de procesado de imagen (convolución, *pooling* y *relu*), con el objetivo de detectar diferentes patrones en las mismas para poder establecer unos criterios de clasificación de imagen. Así pues, se pensó que dichas operaciones de filtrado encargadas de la detección de diferentes parámetros en las imágenes podrían ser utilizadas para llevar a cabo un reconocimiento y extracción de los parámetros y componentes intrínsecas de estilo y textura de una imagen y de forma paralela de los parámetros y componentes intrínsecas de información semántica y de contenido de otra y hacer un mezclado de los mismos en una tercera imagen. La mencionada en primer lugar sería nuestra imagen de textura estilo y la otra la de contenido, y la imagen salida obtenida es nuestra imagen estilizada.

Para llevar a cabo este proceso, el grupo de Gatys, propuso un sistema como el que se muestra en la Figura (4.1).

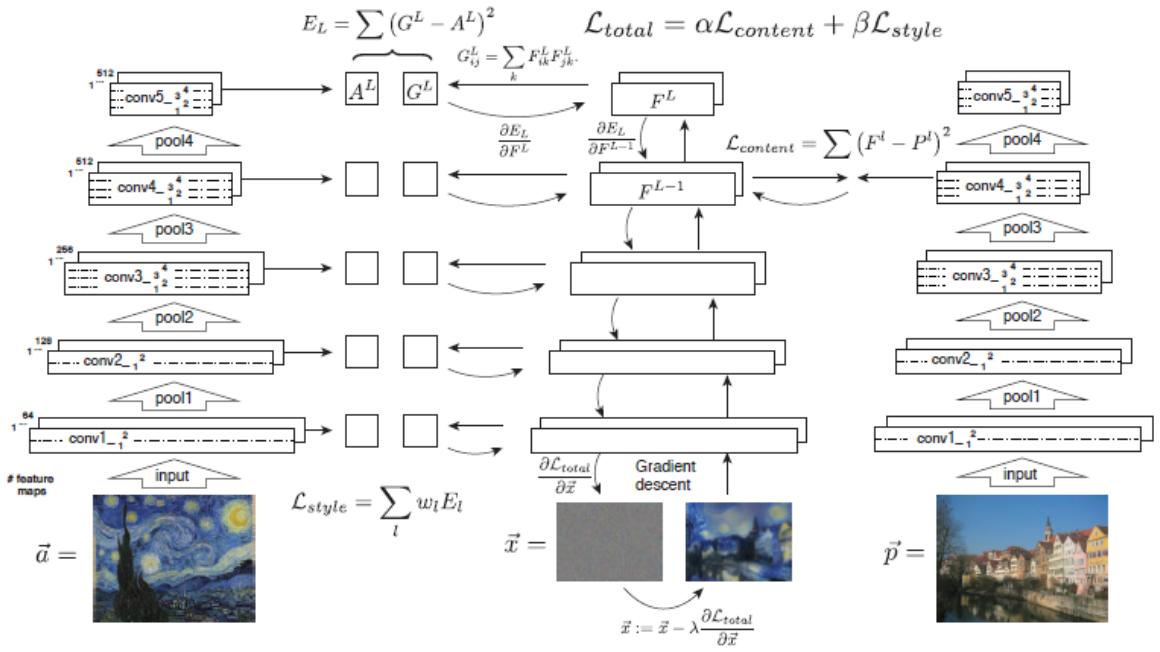


Figura 4.1: Algoritmo de transferencia de estilo (Gatys). A la derecha, se procesa la imagen estilo en la red, donde son extraídos los parámetros de estilo que la caracterizan y que serán transferidos a la imagen *output* final. A la izquierda se procesa en la red la imagen contenido donde se extrae la información de la estructura de la misma, que dará forma a la imagen *output* final. En el centro, se produce el procesado de loq ue inicialmente es la imagen semilla y que va conformando la imagen *output* final en cada iteración sobre la red.

En primer lugar los rasgos de contenido y estilo son extraídos y almacenados. La imagen estilo $\vec{a}$ pasa a través de la red y su representación de estilo A^l es computada en todas las capas incluidas y almacenada. La imagen contenido $\vec{p}$ pasa a través de la red y su representación de contenido P^l es computada en la capa indicada y almacenada. La imagen semilla (ruido blanco gaussiano por defecto) $\vec{x}$, pasa a través de la red y sus rasgos de estilo G^l y de contenido F^l son extraídos. En cada capa incluida en la representación de estilo, se computa la diferencia de cuadrados entre G^l y A^l(ver Ecuación(4.1)) dando lugar a las pérdidas de estilo L_{style} (ver Ecuación(4.2)).

$$E_l = \sum (G^l - A^l)^2 \tag{4.1}$$

$$L_{style} = \sum_l w_l * E_l \tag{4.2}$$

Así mismo la diferencia de cuadrados entre F^l y P^l se computa para dar lugar a las pérdidas de contenido $L_{content}$ (ver Ecuación(4.3)).

$$L_{content} = \sum (F^l - P^l)^2 \tag{4.3}$$

Las pérdidas totales L_{total} se calculan como combinación lineal de las de contenido y las de estilo(ver Ecuación(4.4)).

$$L_{total} = \alpha * L_{content} + \beta * L_{style} \tag{4.4}$$

Su derivada respecto a los valores de píxel puede ser computada usando error *back-propagation*(ver Ecuación(4.5)).

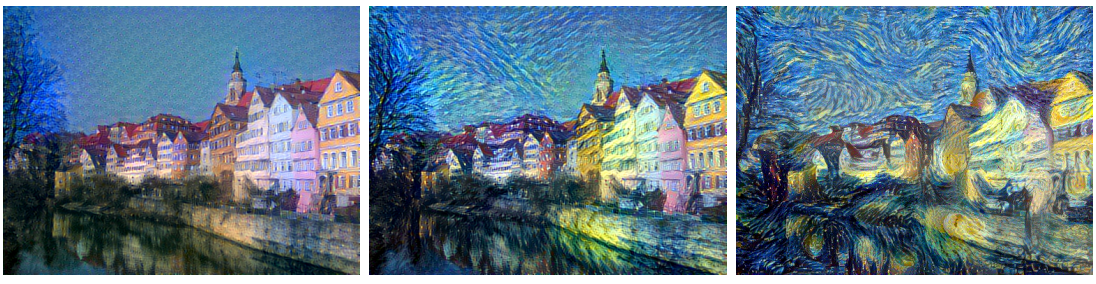
$$\frac{\partial E_l}{\partial F^l} \tag{4.5}$$

Este gradiente es usado para iterativamente actualizar la imagen semilla ($\vec{x}$)(ver Ecuación(4.6)) hasta que de forma simultánea alcanza la correspondencia entre los rasgos de estilo de la imagen $\vec{a}$ y los rasgos de contenido de la imagen $\vec{p}$.

$$\vec{x} := \vec{x} - \alpha * \frac{\partial L_{total}}{\partial \vec{x}} \tag{4.6}$$

El valor de w_l utilizado es igual a 1/5.

El sistema se fundamenta en dos funciones de pérdidas que computan de forma paralela en ambos lados, una diferencia entre patrones de contenido de nuestra imagen contenido y la imagen salida (esta imagen salida se referirá a partir de ahora como imagen *output*) que esta siendo generada y por otro lado se calcula una diferencia entre dos valores que se corresponden a la extracción de patrones de estilo de nuestra imagen que proporciona la transferencia de estilo y nuestra *output* que se esta generando. Es importante señalar el carácter iterativo de este proceso, de forma que se pasa por la red en torno a unas 1000 iteraciones, de forma que la imagen *output* de cada iteración es la imagen entrada (esta imagen salida se referirá a partir de ahora como imagen *input*) de la siguiente, por tanto en función del número de iteraciones que proporcionemos tendremos unos resultados u otros. Como podemos observar en la Figura (4.1) ambas operaciones se suman al final de cada iteración, dando lugar a una función global de pérdidas (ver Ecuación(4.4)), lo importante aquí es que es una suma ponderada, es decir, se le da un peso al inicio del proceso a la imagen que aporta la información de contenido (α) y otro a la que aporta la información de estilo o textura (β) y esto es fundamental a la hora de trabajar con la red, pues serán estos parámetros los que definirán la imagen *output* final, de forma que si el ratio α/β es muy elevado nuestra *output* será una imagen con la información de contenido relativamente bien preservada y con poca presencia del estilo transferido, por otro lado un ratio α/β muy bajo significará todo lo contrario y tendremos entonces una imagen *output* en la que la información de contenido ha quedado totalmente difuminada y hay una gran presencia del estilo transferido. Como criterio de diseño, tras muchas pruebas y observaciones se decidió trabajar dicho ratio con un valor fijo de $\alpha = 10$ y β será el parámetro que iremos variando para obtener las distintas imágenes *output* con distintas intensidades de estilo transferido. A partir de aquí, en los siguientes apartados nos referiremos al peso que se le da a la imagen de estilo como β. En la Figura (4.2) se muestra un ejemplo de transferencia de estilo a una imagen para varios valores de β.



(a) $\beta = 1$ (b) $\beta = 25$ (c) $\beta = 5000$

Figura 4.2: Ejemplos para diferentes valores del parámetro de estilo β, donde se observa como para un valor bajo la transferencia de estilo es mucho inferior que para un valor alto, a su vez, para este último se produce una mayor distorsión de la información de contenido.

4.1. Capas utilizadas

Hemos comentado anteriormente que la red neuronal convolucional utilizada es la VGG-19 [4] y que esta está compuesta por 16 capas convolucionales y 3 *fully conected layers* (capas finales de decisión), además de las distintas capas de *pooling* y *relu* intercaladas entre ellas. Si bien, el sistema de Gatys además de introducir esas funciones de pérdidas a lo largo de la red, lo hace de una forma específica y es la siguiente. Se realiza una pre-selección de un conjunto de capas, una para el procesado de la imagen de contenido y otra para la de estilo o textura. Dicha pre-selección de capas comprende las capas en las cuales se computarán las funciones de pérdidas explicadas anteriormente, es decir, dichas funciones son aplicadas en cada iteración pero no en cada una de las 19 capas de la red, sino que solo se computan en determinados momentos. Así pues, si nos dirigimos al análisis del sistema implementado, las capas seleccionadas son las siguientes:
Contenido: $relu_{4\_2}$
Estilo: $relu_{1\_1}, relu_{2\_1}, relu_{3\_1}, relu_{4\_1}, relu_{5\_1}$
Los subíndices indican el número de capa que corresponde según el modelo utilizado (VGG-19) [4]. Como vemos en dicho sistema se han seleccionado 5 capas para la imagen estilo, por tanto los cálculos de las funciones de pérdidas se realizarían en cada una de estas 5 capas y las pérdidas de estilo totales, L_{style}, serían la suma de las pérdidas obtenidas en cada capa (ver Ecuación 4.2). Por otro lado se ha seleccionado una capa para la imagen contenido, esto significa que será en esta capa donde se computen las pérdidas de contenido $L_{content}$ (ver Ecuación 4.3).

4.2. Inicialización de la imagen *input*

Una opción de diseño del sistema es la selección de la imagen-semilla que es utilizada como primera *input* de la red. Como hemos señalado en cada iteración la entrada de la red es la salida de la iteración anterior, pero tiene que haber una primera entrada a la red para la primera iteración. Normalmente dicha primera entrada del sistema es un ruido blanco gaussiano, pero una opción alternativa es utilizar la imagen-contenido del sistema. Es cierto que tras muchas iteraciones, las funciones de pérdidas entre las imágenes predominarán sobre esta selección y es muy probable en la mayor parte de los casos, encontrar los mismos resultados finales utilizando como semilla una imagen u otra, no obstante nosotros tomamos como decisión de diseño utilizar como semilla la imagen-contenido en cuestión, pues si nuestro objetivo es preservar en la mayor medida de lo posible la información de contenido, el utilizar este método nos permite realizar un mejor análisis sobre los resultados obtenidos, así por ejemplo, resulta muy útil a la hora de analizar las salidas en las primeras iteraciones así como para observar y analizar la evolución de dichas salidas a lo largo de todo el proceso, podemos ver así como en la iteración 1 tenemos como semilla la imagen contenido a la que queremos transferir cierta textura o estilo y podemos ir viendo como en cada iteración que se da en la red, va adquiriendo dicha textura o estilo 4.3. Todo este proceso se explicará en el Sección 5 y se presentarán a su vez, diferentes gráficas e imágenes utilizadas para el estudio.

(a) $n = 1$ (b) $n = 25$ (c) $n = 50$ (d) $n = 100$

(e) $n = 200$ (f) $n = 300$ (g) $n = 400$ (h) $n = 500$

(i) $n = 600$ (j) $n = 700$ (k) $n = 800$ (l) $n = 900$

Figura 4.3: Imágenes correspondientes a la salida del sistema en diferentes iteraciones, donde $n =$ número de iteración.

Capítulo 5

Análisis y estudio de los parámetros del algoritmo

Una de las decisiones que se toman tras realizar el estudio del estado del arte, y el estudio y análisis del sistema y la red neuronal convolucional que van a ser utilizados, es el llevar a cabo un análisis minucioso de las diferentes partes de la red y de las diferentes funciones que da lugar a la transferencia de textura o estilo entre imágenes. Planteamos este proceso analítico como una necesidad previa a la propuesta e implementación de los diferentes métodos y mejoras que se realizaron posteriormente para conseguir el objetivo de confeccionar nuestro sistema de transferencia de texturas y estilos entre imágenes preservando en la mayor medida de lo posible la información de contenido de la imagen que recibe dicha transferencia.

La situación es la siguiente, como se ha comentado anteriormente, el sistema utilizado, es el propuesto por Gatys et al. [1] y dicho sistema hace uso de la red neuronal convolucional de 19 capas del Visual Geometry Group de la Universidad de Oxford [4]. Dicho sistema tal y como explicábamos en la Sección 4 se basa fundamentalmente en unas funciones de pérdidas que se computan entre la imagen semilla y las imágenes contenido y estilo de forma paralela en diferentes puntos de la red. Dichas funciones, dan lugar a una función global de pérdidas que se corresponde con una suma ponderada de ambas, así definíamos los parámetros α y β que marcaban dichas ponderaciones y que nos permitirían ajustar la presencia final de cada una de las imágenes contribuyentes, contenido y estilo o textura en la imagen final.

Nuestro propósito era explorar al máximo las diferentes posibilidades que el sistema aportaba a raíz de las mencionadas funciones de pérdidas, ya que contenían prácticamente toda la información de como se realizaba la transferencia de textura o estilo entre las imágenes y ver así mas detalladamente los problemas y carencias que presentaba, con el fin de proporcionar diferentes conclusiones y tratar de establecer ciertos criterios de predicción de que imagen *output* final nos dará el sistema, en función del tipo de imágenes que utilicemos como *input* y del valor de α y β que seleccionemos.

Uno de los criterios de diseño que se toman inicialmente y que también se explica en la Sección 4 es fijar α a un valor constante e igual a 10 y tomar diferentes valores de β.

Por otro lado, otro de nuestros objetivos principales, que siempre hemos tenido como referencia durante todo el proceso ha sido la optimización del coste temporal de todos los métodos, así pues, cada mejora propuesta sobre el sistema o la red y cada método implementado ha tenido su análisis de coste temporal y se ha trabajado para reducirlo. Por tanto, otra conclusión que sacamos mediante este estudio fue que podíamos recortar el número de iteraciones en la red que el sistema nos daba por defecto, pudiendo así reducir de forma considerable el tiempo de procesado en la red. Explicaremos en la Subsección 5.1, como llegamos a esta conclusión y apoyaremos nuestras explicaciones de diferentes gráficas e imágenes.

Otro problema que resolvimos gracias a estos estudios fue la selección de la imagen semilla. Dicho problema también se ha analizado en la Sección 4, y se basa en la decisión de cual debe ser la imagen semilla, entre un ruido blanco gaussiano y cualquier otra imagen, que coherentemente proponemos que sea la misma imagen que aporta la información de contenido al sistema. Así pues, esta decisión también tiene su estudio analítico el cual queda explicado en la sección 5.2.

Hemos planteado en este apartado hasta este punto, fundamentalmente, el estudio de tres problemas, que podemos sintetizar en, búsqueda de la máxima optimización del coste temporal, selección de la imagen semilla y predicción de la imagen *output* del sistema en función de imagen contenido, imagen estilo y valor de β óptimo.

5.1. Optimización del coste temporal

Como se puede advertir en esta memoria, a lo largo de todo el proyecto siempre se ha puesto el foco en el coste temporal o coste computacional que tiene cada sistema añadido o implementado. Esto es debido al alto coste computacional que presenta el procesado de imagen. Explicamos en cada apartado de cada sistema utilizado, añadido o implementado, cual es su coste computacional y como, si ha sido posible, lo hemos conseguido reducir. Por tanto en este apartado nos centraremos en explicar uno de los ahorros fundamentales en el coste temporal de procesado en la red que conseguimos.

Como explicábamos anteriormente, se ha hecho un análisis de las funciones de pérdidas del sistema, así con ciertas representaciones gráficas de las mismas pudimos observar determinados patrones y sacar las consiguientes conclusiones. Decidimos representar en una gráfica la evolución de dichas funciones de pérdidas, tanto la suma de las pérdidas de de estilo (calculadas en diferentes capas de la red), como las de contenido (calculadas solamente en la última capa) y las totales (suma de las descritas previamente), en función de las iteraciones que se completaban a lo largo del proceso, sobre la red neuronal convolucional utilizada. Observábamos entonces, que el sistema por defecto nos ofrecía un valor de 1000 iteraciones a lo largo de la red y que todas las funciones de pérdidas tenían un punto de convergencia entorno a la iteración 400 o 500, significando esto que habíamos adquirido la máxima transferencia de estilo posible para el valor de β que estuviésemos utilizando, esto unido a un análisis de las imágenes que obteníamos a la salida, nos llevó a la conclusión de que podíamos recortar a la mitad el número de iteraciones procesadas (dejando el sistema finalmente en 500 iteraciones), reduciendo el tiempo de procesado de 9-10 minutos inicialmente, a 4-5 minutos. En la Figura (5.1) mostramos algunos ejemplos de las gráficas de evolución de pérdidas.

Presentamos en la Figura (5.2) algunas de las imágenes correspondientes a la evolución descrita por las gráficas anteriores, para ello hemos seleccionado un ejemplo que se generó mediante el uso de la imagen contenido *BradPitt* y la imagen estilo *Picasso*.

5.2. Selección de la imagen semilla

Otro de los problemas que pudimos solucionar gracias a los estudios explicados en este punto, es la selección de la imagen semilla que utilizaríamos. Recordemos que el sistema nos permitía seleccionar cualquier imagen como tal. Por defecto la imagen semilla era un ruido blanco gaussiano y nos planteamos la posibilidad de utilizar en su lugar la imagen contenido que proporcionábamos al sistema, bajo el razonamiento de que si era esta la que nos iba a proporcionar la información de contenido a la imagen *output* final, por qué no partir de ella desde un principio en lugar del ruido blanco gaussiano.

Decidimos afrontar este análisis mediante una representación de las funciones de pérdidas de igual manera que hicimos en la sección anterior, de esta forma lo que observábamos era

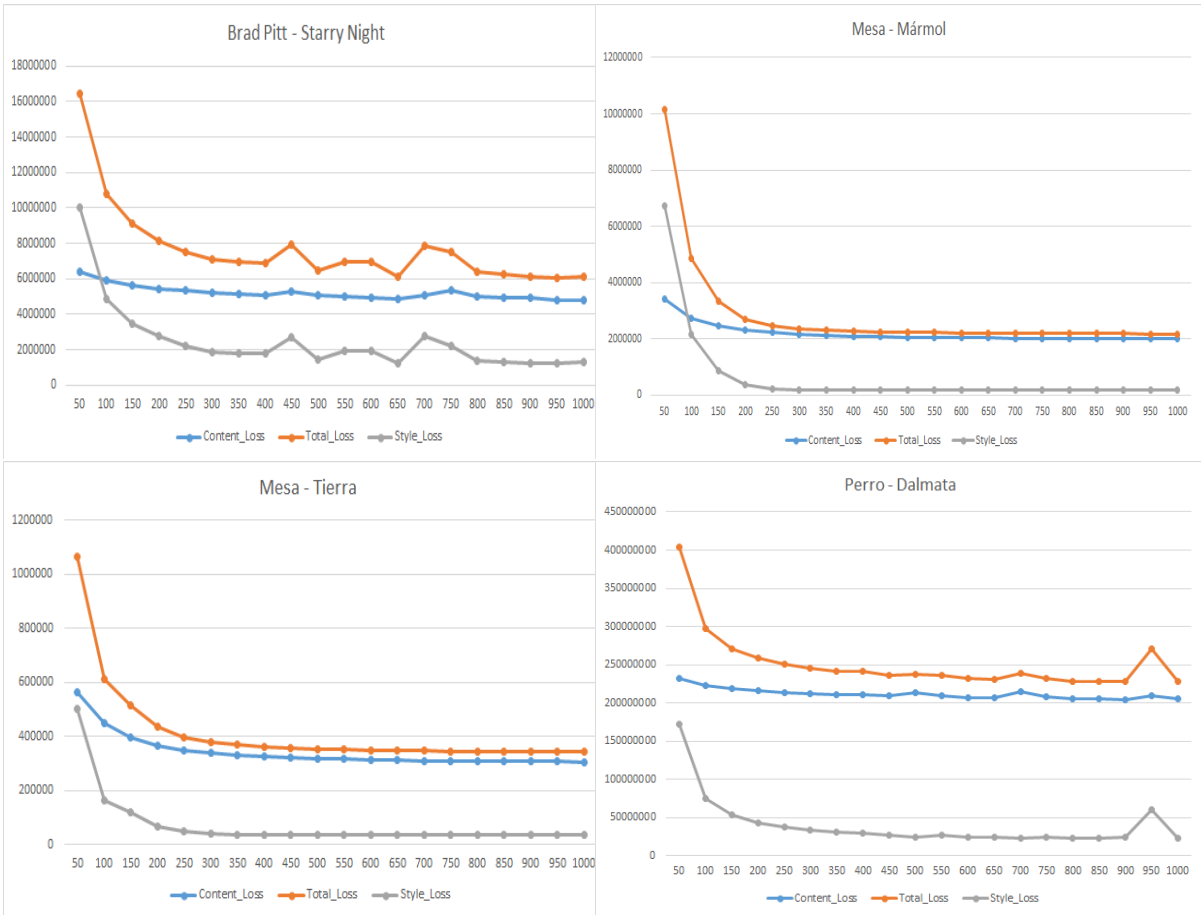


Figura 5.1: Representación de las pérdidas de contenido, estilo y totales para 4 ejemplos de transferencia de estilo con diferentes imágenes contenido y estilo. En ellas podemos observar la convergencia en torno a la iteración 400 o 500. También podemos apreciar ciertos puntos aislados como es el caso de (a) en la iteración 450 o 700, estos puntos corresponden a ciertas anomalías en la transferencia y se manifiestan en determinadas ocasiones en alguna iteración de forma puntual y aislada. La imagen (a) corresponde al procesado de la imagen contenido *BradPitt* con la imagen estilo o textura *StarryNight*, la imagen (b) es a su vez combinación de *Mesa* con *Marmol*, (c) corresponde a *Mesa* con *Tierra* y (d) a *Perro* con *Dalmata*. Todas estas imágenes pueden encontrarse en la Sección 9.2

la misma convergencia en torno a la iteración 400-500, así pues las tres funciones de pérdidas convergían alrededor de dichas iteraciones pero lo hacían de forma diferente y también existía una dependencia de la imagen semilla utilizada, tal y como cabía esperar. Explicamos a continuación las distintas diferencias que encontramos.

Las funciones de pérdidas de estilo o textura se comportaban de forma muy similar independientemente de cual fuese la imagen semilla, dicho comportamiento presentaba, como podemos ver en la Figura (5.1), fuertes oscilaciones inicialmente hasta que llegaba a ese punto de convergencia en torno a la iteración 400-500 en el cual estas pérdidas prácticamente no variaban, este resultado era coherente con lo esperado pues la imagen de estilo utilizada tiene tantas diferencia o similitudes con un ruido blanco gaussiano que con cualquier otra imagen que estuviésemos utilizando como portadora del contenido. El hecho del comportamiento oscilatorio y no lineal previo a la convergencia lo atribuimos a la aleatoriedad del proceso de transferencia de textura a lo largo de las dimensiones de la imagen *output*.

En cambio, las funciones de pérdidas referentes a la imagen contenido, no presentaban

(a) $n = 1$ (b) $n = 25$ (c) $n = 50$ (d) $n = 100$

(e) $n = 200$ (f) $n = 300$ (g) $n = 400$ (h) $n = 500$

(i) $n = 600$ (j) $n = 700$ (k) $n = 800$ (l) $n = 900$

Figura 5.2: Imágenes correspondientes a la salida del sistema en diferentes iteraciones, donde $n =$ número de iteración.

esa aleatoriedad previa al punto de convergencia, y tal y como esperábamos en este caso si había una clara dependencia con la imagen semilla. El comportamiento era el siguiente, cuando utilizábamos la imagen contenido como semilla, las pérdidas en la iteración cero eran nulas y seguían una progresión creciente consecuencia de ese distanciamiento de la imagen *output* con respecto a la imagen contenido debida a la progresiva transferencia de estilo o textura que iba experimentando en cada iteración, así era hasta llegar al mencionado anteriormente, punto de máxima transferencia de textura o estilo y a partir de ahí se produce un leve descenso de las pérdidas con respecto a la información de contenido y entra en la convergencia previamente analizada. En cambio si utilizábamos como semilla el ruido blanco gaussiano lo que observábamos en las pérdidas de contenido era lo mismo que observábamos anteriormente cuando analizábamos las pérdidas de estilo y es un proceso de oscilación previo a la convergencia, esto se debe a que el ruido blanco gaussiano no tiene ninguna similitud inicial con la

imagen contenido, de hecho tiene la misma que tenía con la imagen estilo, así cualquier cambio producido en las primeras iteraciones sobre la imagen *output*, hasta alcanzar la convergencia, se traduce como una diferencia aleatoria entre ambas imágenes tal y como pasaba en el caso de las pérdidas de estilo.

Con esto unido a un estudio y análisis de las imágenes que obteníamos como *output* a lo largo de las distintas iteraciones, observamos que tras la mencionada convergencia los resultados utilizando una u otra imagen semilla eran muy similares, pero el uso de la imagen contenido como semilla, producía menos distorsión en los detalles y bordes de la imagen *output* final, como cabía esperar, así pues decidimos cambiar la imagen semilla y dejar de utilizar el ruido blanco gaussiano, para introducir en su lugar, la misma imagen contenido que utilizásemos en cada caso.

En la Figura (5.3) se muestra la evolución de las pérdidas de contenido $L_{content}$ y las de estilo L_{style} en las 60 primeras iteraciones, por un lado para semilla igual a la imagen de contenido y por otro lado para semilla igual al ruido blanco gaussiano. Lo que observamos es lo comentado anteriormente, esto es, cuando usamos como semilla la imagen contenido las pérdidas correspondientes a la información de contenido en las primeras iteraciones son nulas y su valor se va incrementando con el incremento de las mismas, en cambio cuando utilizamos como semilla el ruido blanco gaussiano, dichas pérdidas comienzan desde la primera iteración siendo muy altas, normalmente y como cabe esperar es en estas primeras iteraciones toman los valores mas altos y con el incremento del número de las mismas estas van disminuyendo, pues la imagen semilla se va adaptando y va tomando las correspondencias de información de contenido que aporta la imagen contenido. Por toro lado si analizamos el comportamiento de las gráficas correspondientes a las pérdidas de estilo L_{style}, el comportamiento es independiente de la imagen semilla utilizada, tal y como cabe esperar, pues en un caso se buscan correspondencias entre una imagen contenido y una imagen estilo que carecen totalmente de ninguna correspondencia inicial tal y como ocurre cuando dichas correspondencias se calculan entre la mencionada imagen estilo y un ruido blanco gaussiano.

De igual modo que en la sección anterior acompañaremos estas gráficas con algunas imágenes correspondientes a la salida del sistema en las iteraciones analizadas en las gráficas, esto es, en las primeras iteraciones del sistema (ver Figura (5.4) y la Figura(5.5)). Vamos a seleccionar un intervalo coherente, pues no vemos necesario el representar las 60 imágenes correspondientes a cada una de las iteraciones que aparecen en las gráficas. Utilizaremos el mismo ejemplo que en el caso anterior, es decir, *BradPitt* como imagen contenido y *Picasso* como imagen estilo. Presentaremos para dicho ejemplo, las dos posibilidades analizadas, por un lado cuando se utiliza como semilla la imagen contenido y por otro lado cuando usamos como semilla el ruido blanco gaussiano.

5.3. Selección del parámetro de estilo, β

La parte mas compleja era la de tratar de predecir como sería la salida del sistema en función de los parámetros de entrada, pues resulta muy complejo obtener información cuantitativa y estadística de una imagen, así pues, tuvimos que proponer una método y una forma de enfrentarnos a este problema.

La decisión que se tomó fue centrar el estudio en las imágenes que transferían el estilo o textura, asumiendo que el comportamiento en función de la imagen contenido no era tan significativo. Se observaba que para una misma imagen contenido y distintas imágenes estilo o textura, las salidas tomaban formas muy diferentes y los valores óptimos de β variaban de forma considerable para cada caso, en cambio, si seleccionábamos una imagen estilo y la aplicábamos a diferentes imágenes contenido, dichos valores óptimos de β solían ser muy similares. Entonces las preguntas que nos planteamos fue ¿Qué relación tienen dichas imágenes estilo o textura?,

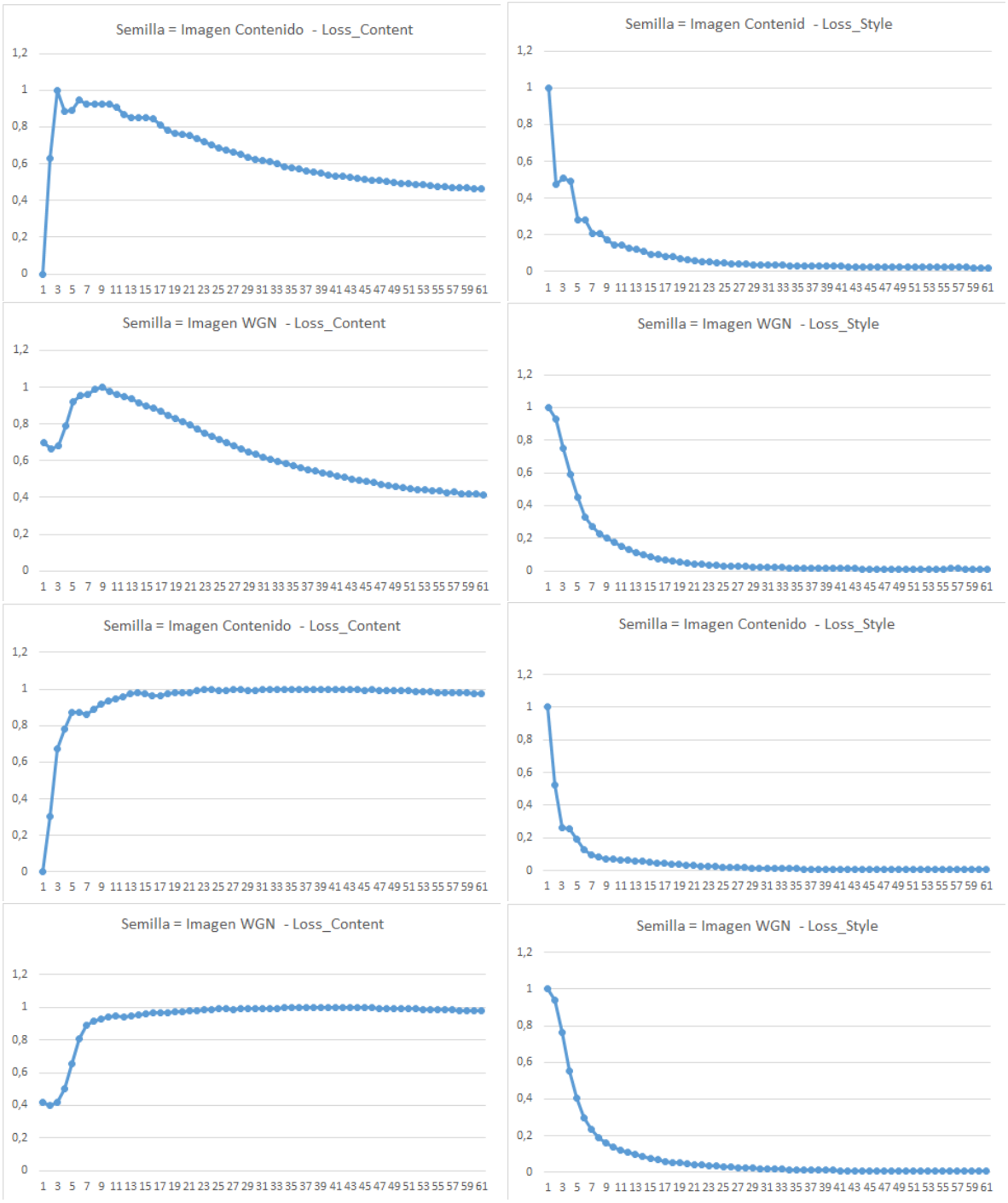
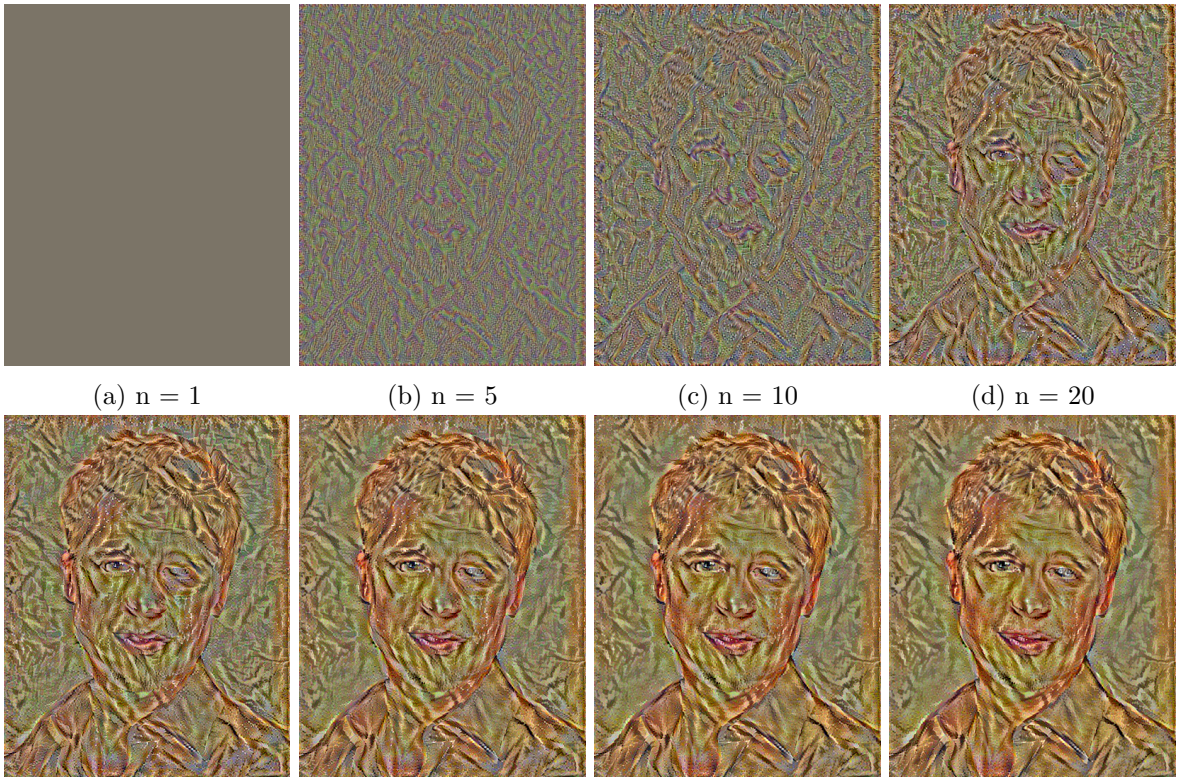


Figura 5.3: Representación de algunos ejemplo de pérdidas de contenido y estilo.

¿Podemos emplear algún sistema que las caracterice?, ¿Qué valores de β óptimos se asocian a cada tipo de imagen estilo o textura?.

El objetivo era ahora conseguir establecer una relación entre los diferentes valores de β posibles para cada imagen estilo. Para conseguir establecer dicha relación haríamos pasar un conjunto de imágenes por la red para distintos valores de β y finalmente analizaríamos las imágenes obtenidas a la salida de la red para cada caso, y veríamos si podíamos extraer alguna conclusión.

Necesitábamos entonces un método para analizar las imágenes a la salida del sistema, así



(a) n = 1 (b) n = 5 (c) n = 10 (d) n = 20

(e) n = 30 (f) n = 40 (g) n = 50 (h) n = 60

Figura 5.4: Imágenes correspondientes a la salida del sistema en diferentes iteraciones para el caso de utilizar como semilla un ruido blanco gaussiano. El valor de n indica el número de iteración correspondiente a la imagen.

pues nos planteamos medir la similitud de dichas imágenes *output* con respecto a la imagen original que aportaba la información de contenido, pues recordamos que nuestro objetivo es preservar en la mayor medida de lo posible dicha información de contenido. Así sometemos en la red distintas imágenes estilo con diferentes valores de β para una misma imagen contenido, seguidamente realizamos un análisis de similitud de dichas salidas obtenidas con la imagen contenido original, para ello utilizamos una función de correlación, y el objetivo es observar la variación de dicha correlación en función de los diferentes valores de β seleccionados. Este proceso se repite tomando el mismo set de distintas imágenes estilo para varias imágenes contenido. Por tanto si conseguíamos observar algún patrón en dichas variaciones de correlación en función del valor de β utilizado para una misma imagen estilo o textura aplicada a distintas imágenes contenido, podríamos establecer algún tipo de relación directa entre un valor de β y una imagen estilo.

Lo que observamos fue, como se muestra en la Figura (5.6) que imágenes generadas con ciertas imágenes estilo se distanciaban mucho más y con mayor celeridad de la imagen contenido original (lo que significa una mayor pérdida de la información de contenido) en cambio para otras imágenes estilo ocurría lo contrario. Necesitábamos entonces establecer algún tipo de relación entre las imágenes estilo y propusimos como hipótesis analizar las distribuciones frecuenciales de las imágenes estilo.

De esta forma, tal y como se muestra en la Figura (5.7), nos encontramos con que aquellas imágenes estilo que habían generado imágenes que perdían mayor información de contenido presentaban unas distribuciones frecuenciales con componentes de alta frecuencia y, en cambio, aquellas que generaban imágenes con menores pérdidas de contenido presentaban componentes

(a) n = 1 (b) n = 5 (c) n = 10 (d) n = 20

(e) n = 30 (f) n = 40 (g) n = 50 (h) n = 60

Figura 5.5: Imágenes correspondientes a la salida del sistema en diferentes iteraciones para el caso de utilizar como semilla la imagen contenido. El valor de n indica el número de iteración correspondiente a la imagen.

frecuenciales de menor frecuencia. En el apartado 5.3.1 se explica en detalle el procedimiento llevado a cabo para obtener y medir las distribuciones frecuenciales de las imágenes estilo y como han sido generadas las gráficas de la Figura (5.7).

Dados estos resultados, la decisión que tomamos fue la de basar la selección del valor del parámetro de estilo β en función de la distribución frecuencial de la imagen estilo, de forma que se utilizarán valores altos del mismo para imágenes cuyas componentes frecuenciales sean de baja frecuencia y valores mas bajos para aquellas imágenes estilo que presenten distribuciones con componentes de mas alta frecuencia.

5.3.1. Metodología utilizada para el análisis de las distribuciones frecuenciales

El método que utilizamos para definir que imágenes estilo eran de alta frecuencia y cuales de baja frecuencia es el que explicaremos a continuación. En primer lugar realizábamos la representación del módulo de las distribuciones frecuenciales de las imágenes en 2-Dimensiones, de forma que en el centro de la imagen quedan representadas las componentes de baja frecuencia y en los extremos las de alta frecuencia. En dichas representaciones, como bien es sabido, cuanta mayor magnitud presenta es una región, mayor presencia de las componentes que se encuentran en dicha región tendrá la imagen, así por ejemplo, una representación con alta magnitud en el centro y los baja en los extremos se corresponderá con una imagen con componentes de baja frecuencia.

En segundo lugar, transformábamos la representación anterior en una gráfica como las mostradas en la Figura (5.7), ya que el quitar una dimensión a las representaciones iniciales

facilitaba en gran medida el tratamiento de los datos. Para ello, llevamos a cabo una operación que consistía en establecer cuadrantes concéntricos sobre la representación frecuencial de la imagen y realizar una media de los valores que encerraba cada cuadrante (todas ponderadas al valor máximo, es decir, divididas por la media del cuadrante mayor que engloba toda la imagen). Este proceso es conocido como el cálculo de los ratios de densidad de energía para diferentes rangos frecuenciales. De esta manera, conseguíamos unas gráficas con unas curvas que daban una representación de las distribuciones frecuenciales de las imágenes estilo, de forma que si dichas curvas tenían fuertes pendientes, significaba que en el centro teníamos valores de magnitud mucho mayores que en los extremos , y por tanto tendríamos en ese caso una imagen con componentes de baja frecuencia y en caso contrario, si la curva era mas plana, significaba que los valores se mantenían constantes a lo largo de la representación frecuencial de la imagen tendríamos una imagen con componentes de alta frecuencia.

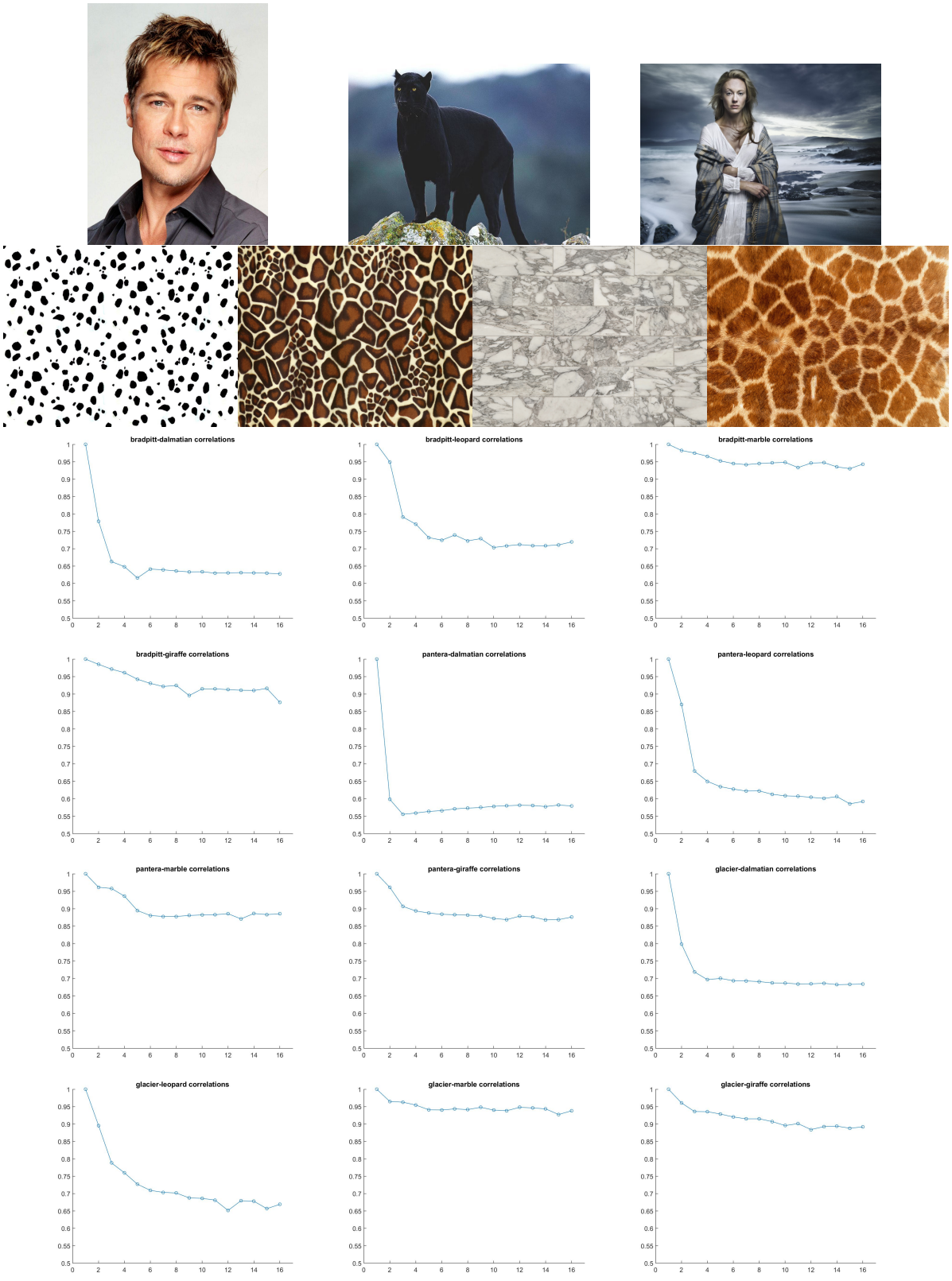
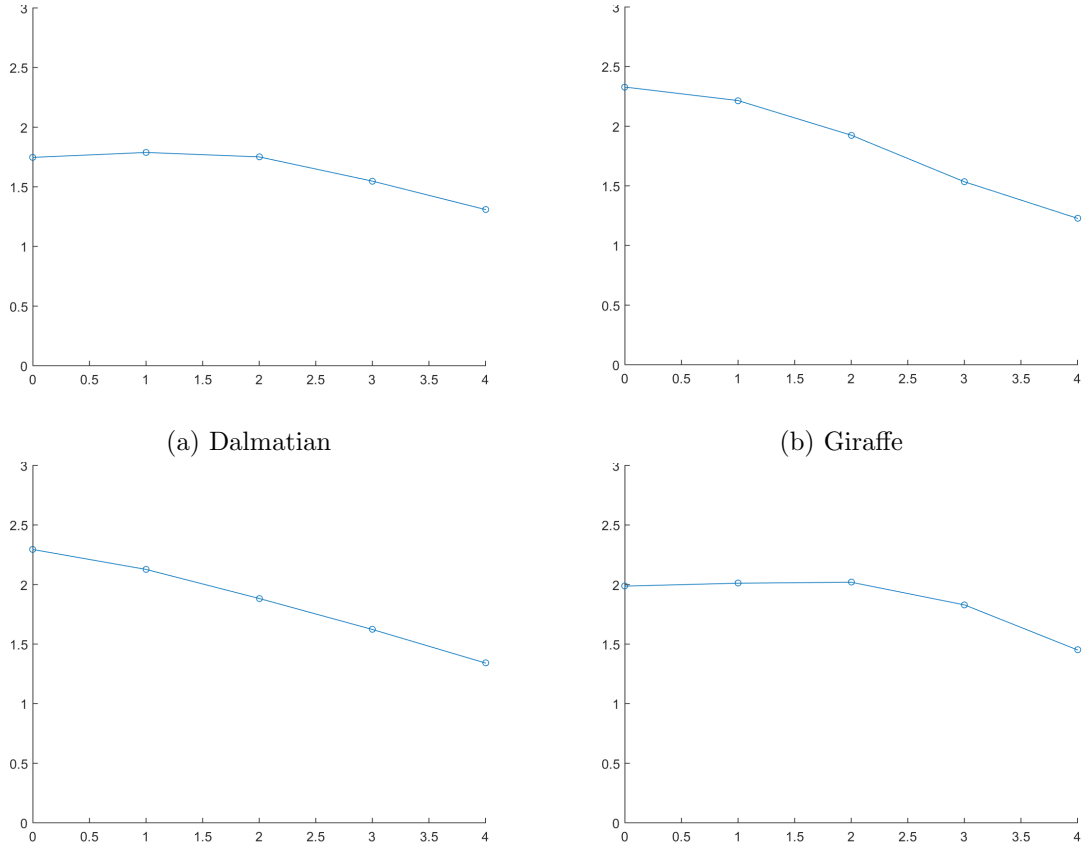


Figura 5.6: Las imágenes de la parte superior son las imágenes contenido utilizadas para representar este análisis, son aquellas que han recibido la transferencia de textura o estilo. La segunda fila de imágenes corresponde a las imágenes estilo que hemos seleccionado para este ejemplo y son las que se han transferido a las imágenes contenido de la fila superior. Las gráficas representan la correlación calculada entre las imágenes estilizadas y las imágenes contenido originales utilizando distintos valores de β (cada punto corresponde a un valor, por el siguiente orden $\beta = 0, 1, 5, 10, 25, 50, 75, 100, 200, 400, 600, 800, 1000, 2000, 3000, 5000$).



(a) Dalmatian

(b) Giraffe

(c) Marble

(d) Leopard

Figura 5.7: Representación de los ratios de densidad de energía para diferentes rangos frecuenciales para cuatro ejemplos de imágenes estilo. Podemos observar que *Dalmatian* (a) y *Leopard* (d) presentan una pendiente mucho menor que *Giraffe* (b) y *Marble* (c). Una pendiente mas pronunciada significa una diferencia en magnitud alta entre el centro de una representación del módulo de la distribución frecuencial de una imagen con respecto a los bordes, lo que se traduce en una distribución frecuencial de componentes de baja frecuencia. Una pendiente menos pronunciada, representaría una diferencia en magnitud menor entre centro y bordes y por tanto una distribución frecuencial de componentes de alta frecuencia.

Capítulo 6

Transferencia selectiva de textura y estilo

6.1. Introducción

El principal objetivo del proyecto es, como se ha explicado anteriormente, conseguir una transferencia de textura o estilo entre dos imágenes, intentando transferir la mayor cantidad del estilo o textura de la imagen que lo aporta, y a su vez preservando en la mayor medida de lo posible los detalles mas pequeños y bordes de la imagen que recibe dicho estilo o textura, y que es la que aporta la información de contenido a la imagen *output* final.

Como se ha explicado en el apartado anterior, el sistema utilizado para llevar a cabo la transferencia de estilo o textura entre imágenes, ha sido un sistema propuesto por Leon A. Gatys et al. [1] y que hace uso de redes neuronales convolucionales.

En el apartado anterior analizábamos las funciones algebraicas que resuelven el sistema mencionado y describíamos como dichas funciones permiten realizar diferentes transferencias de estilo para una misma imagen,es decir, para un mismo par imagen estilo - imagen contenido podemos conseguir diferentes resultados en función de los valores de los parámetros de la red. En la Figura (6.1) se presenta un ejemplo con distintas intensidades de transferencia de estilo.

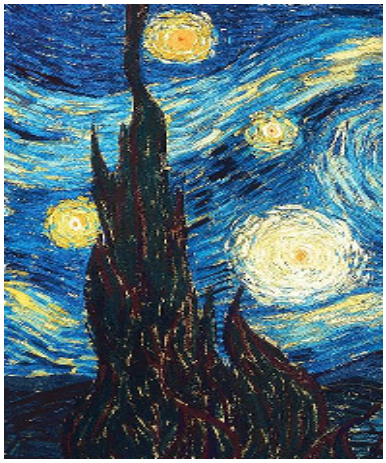
De esta manera, nace la idea de implementar un sistema de transferencia selectiva de estilo o textura de forma que a las partes con detalles mas pequeños y bordes, les correspondiese una transferencia con un valor de β menor y a las zonas mas redundantes y con menos detalles les correspondiese una transferencia equivalente con un peso de β mayor.

6.2. Segmentación de la imagen en regiones, método *Split and merge*

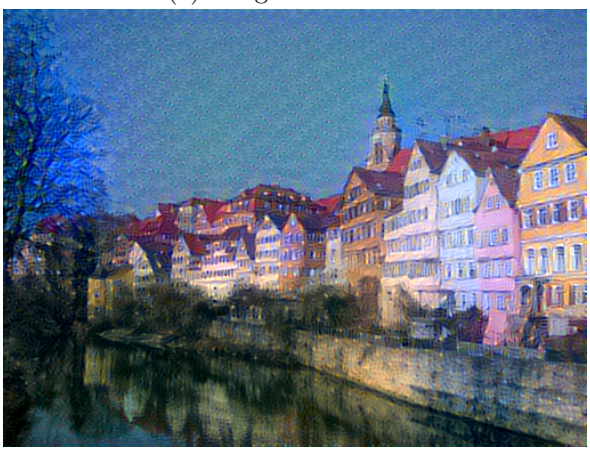
El primer problema que teníamos que afrontar era la detección de las zonas descritas anteriormente. Necesitamos separar las zonas sin detalles y homogéneas de aquellas con muchos detalles y bordes. Para ello decidimos utilizar un método de segmentación de imagen orientado a regiones. En concreto, basamos nuestra implementación en una técnica de árbol cuaternario (mas comúnmente conocido su término en inglés, *quadtree*) cuyo fundamento teórico se basa en lo siguiente. La imagen de máxima resolución se divide en cuatro rectángulos iguales. Cada región es posteriormente dividida en otras cuatro subregiones. Estas divisiones se van haciendo recursivamente hasta alcanzar a nivel de píxel. Por tanto, por cada nodo o región padre colgarán otras cuatro subregiones. Dando motivo a denominarlo como árbol cuaternario. En la Figura (6.2) se muestra un ejemplo ilustrativo a grandes rasgos del método.

En la Subsección 6.2.1 expondremos una explicación con mas detalle de la implementación del método.

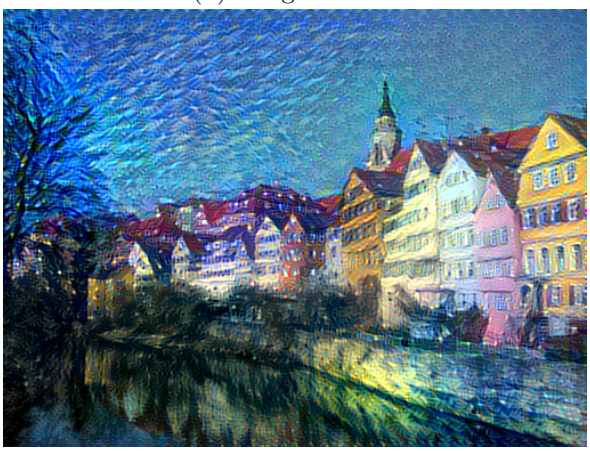
(a) Imagen contenido.



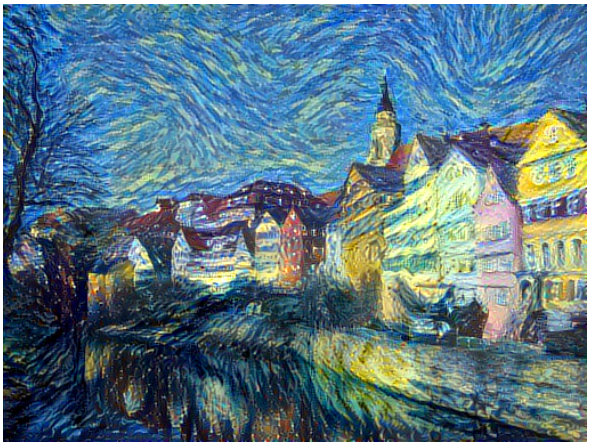
(b) Imagen estilo.



(c) $\beta = 1$.



(d) $\beta = 10$.



(e) $\beta = 400$.

(f) $\beta = 5000$.

Figura 6.1: Imágenes correspondientes a la salida del sistema para diferentes valores de β que dan como resultado distintas transferencias de estilo.

6.2.1. Breve explicación del método de subdivisión

En primer lugar, el programa consiste en la división de la imagen de forma iterativa en parches en escala log2 en función de los valores máximo y mínimo de cada parche y un *threshold*. El programa divide en primer lugar la imagen en 2 parches del mismo tamaño, después toma en cada uno de ellos el valor de intensidad máximo y mínimo de píxel en escala de grises,

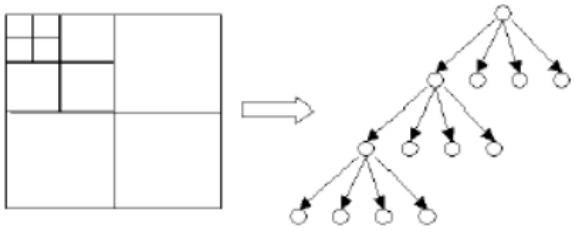


Figura 6.2: Ejemplo básico del funcionamiento de la técnica de segmentación basada en árbol cuaternario.

y calcula la diferencia entre ambos, después la compara con un *threshold* y si la diferencia calculada supera ese valor de *threshold* el parche en cuestión vuelve a dividirse en otros 2 del mismo tamaño, así pues una zona muy homogénea no quedará dividida y se completará con un bajo número de parches de gran tamaño, en caso contrario una zona muy heterogénea y con muchos detalles quedará dividida en muchos parches de pequeño tamaño. En la Figura (6.3) podemos ver un ejemplo de la división de una imagen utilizando este método.

De esta manera, debido al tamaño de las imágenes utilizadas, y que los parches siguen una escala log2, tenemos parches de los siguientes tamaños: 1,2,4,8,16,32,64,128 píxeles. Así pues, a pesar de que la imagen queda dividida en un gran número de parches, podemos tratar el análisis de la misma en función de los tamaños de parche. Entonces, la propuesta inicial era asignar un valor de β a cada tamaño de parche, así a parches muy pequeños les corresponderían valores de β muy bajos (normalmente a los de tamaño 1 píxel, $\beta = 0$ = imagen original) y a parches de gran tamaño les corresponderían valores de β altos. De este modo transferimos la textura o estilo en cuestión, con mayor intensidad a los parches mas grandes y con menor intensidad a los mas pequeños. Estos últimos que contienen las zonas con mas detalle, líneas y bordes, conservarán con menor distorsión la información de contenido.

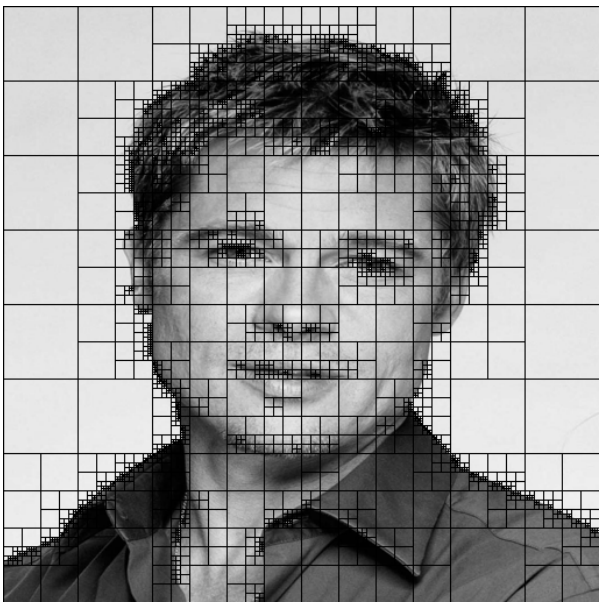


Figura 6.3: Descomposición de una imagen mediante segmentación en árbol cuaternario.

6.2.2. Asignación del parámetro de estilo β a cada parche

Una vez dividida la imagen en parches, teníamos que afrontar la selección de un valor de β que le correspondiese a cada parche y el objetivo era plantear un análisis estadístico de imagen a la imagen de estilo que nos permitiese automatizar este proceso de selección. Dicha propuesta se explica en la Sección 5.

Tras asignar a cada tamaño de parche su correspondiente valor de β, teníamos que realizar tantos procesados en la red neuronal convolucional como distintos valores de β tuviésemos y estos en principio eran iguales al número de parches de distinto tamaño que hubiésemos obtenido. Posteriormente decidimos en según que casos utilizar el mismo valor de β para distintos tamaños de parche dado que en muchas ocasiones dos tamaños distintos no definían zonas muy diferentes bajo nuestro punto de interés, por ejemplo, un cielo en una imagen podría haber sido dividido en dos parches de 128 píxeles mas otros cuatro de 64, en ese caso podríamos ahorrarnos un procesado en la red y repetir el mismo valor de β para ambos tamaños.

Seguidamente al procesado en la red, la tarea era asignar a cada región o parche su correspondiente imagen salida de la red con su correspondiente β y unir cada parche para conseguir la imagen final con la transferencia de textura o estilo selectiva que hemos planteado. Dicha unión como cabía esperar genera diferentes líneas o ejes en las distintas fronteras entre parches, esto es debido a las diferentes intensidades de transferencia de textura que se dan en cada parche. También planteamos una solución para este problema cuyo objetivo fuese disimular o atenuar la presencia de dichos ejes. El método implementado para ello queda explicado con detalle en la Sección 6.3.

6.3. Suavizado de los bordes, método *Blending*

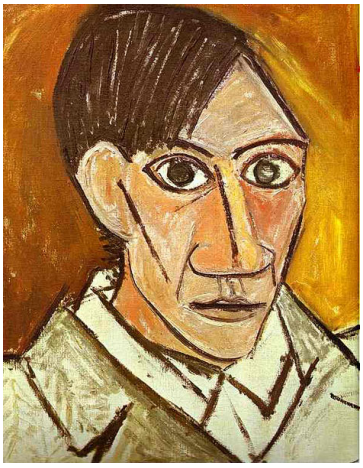
Como explicábamos en la Sección 6, tras realizar el último paso de unión de los distintos parches para formar la imagen final, se observaban ejes o bordes entre los mencionados parches. Así pues, se propone buscar una solución que mejore el resultado final del sistema, la intención es hacer que esas transiciones entre parches que dan lugar a los ejes no deseados y que resultan incómodas para la experiencia visual, desaparezcan o queden atenuadas de forma que sean prácticamente inapreciables.

Tras el estudio de varios métodos empleados para resolver este tipo de problemas, decidimos implementar un método que consiste en el siguiente proceso: En primer lugar se aumenta el tamaño de cada parche en los ejes inferior y derecho, de forma que todos los parches quedan solapados en sus cuatro aristas por el mismo número de píxeles. Este solapamiento significa que en las zonas donde antes había una arista de transición entre dos o más parches ahora estos comparten una zona común cuyo tamaño hemos elegido por defecto 1 píxel pero que puede ser cualquiera que seleccionemos. Por tanto tenemos zonas donde convergen parches creados con diferentes valores de β, o intensidad de transferencia de estilo o textura, esto se traduce en diferentes valores de los canales RGB, por tanto en las zonas de solapamiento, en cada posición de píxel, tenemos diferentes píxeles correspondientes a diferentes parches, con distinto valor. Como el objetivo es hacer una transición coherente con los parches convergentes en dicha zona, la solución propuesta es una media aritmética de los valores de píxel que convergen en cada punto, así si en la posición (x_i, y_i) de la imagen, convergen N parches diferentes, el valor final que adquirirá dicho píxel será la suma de cada píxel convergente dividido por N y para cada canal de forma individual.

De esta manera, resolvemos el problema de los ejes indeseados, podemos observar la aportación de nuestro sistema en las imágenes de la Figura (6.5).

En la Figura (6.4) se muestran algunos resultados conseguidos tras la aplicación de ambos métodos, *Split and merge* y *Blending*, en ellos podemos observar como mejora la conservación

de la información de contenido de la imagen original, y a su vez se consigue una abundante transferencia de estilo.



(a) Imagen estilo.

(b) Imagen a la salida de la red.

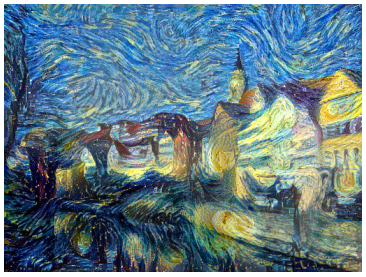
(c) Nuestro resultado.

(d) Imagen estilo.

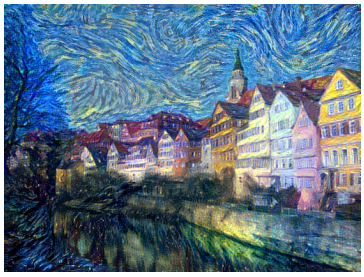
(e) Imagen a la salida de la red.

(f) Nuestro resultado.

(g) Imagen estilo.

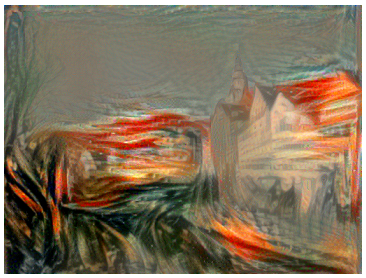


(h) Imagen a la salida de la red.



(i) Nuestro resultado.

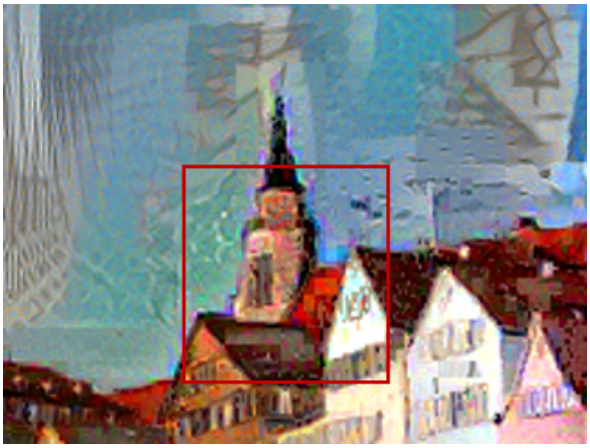
(j) Imagen estilo.



(k) Imagen a la salida de la red.

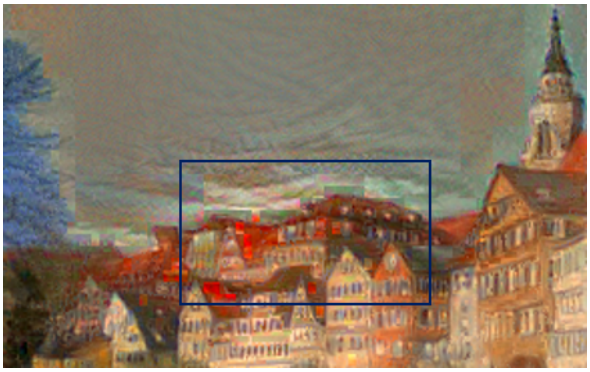
(l) Nuestro resultado.

Figura 6.4: A la izquierda se muestran iv ejemplos de imagen estilo. En el centro las correspondientes imágenes *output* de la red para un valor de β óptimo. A la derecha los resultados tras aplicar los métodos *Split and merge* y *Blending*.



(a) Antes de aplicar el método de *blending*.

(b) Después de aplicar el método de *blending*.

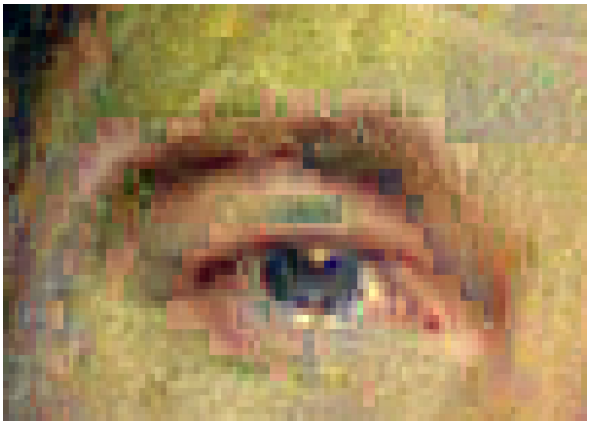


(c) Antes de aplicar el método de *blending*.

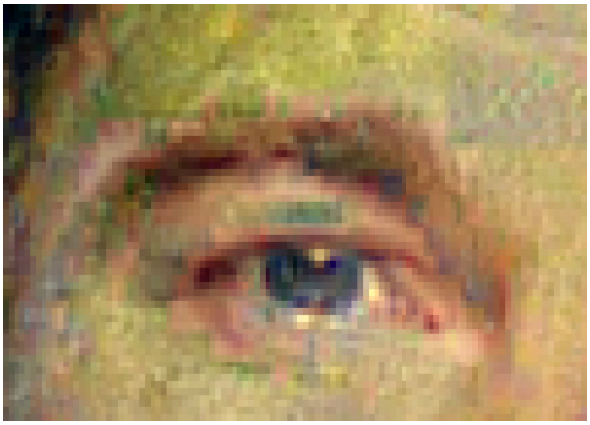
(d) Después de aplicar el método de *blending*.

(e) Antes de aplicar el método de *blending*.

(f) Después de aplicar el método de *blending*.



(g) Antes de aplicar el método de *blending*.



(h) Después de aplicar el método de *blending*.

Figura 6.5: Algunos ejemplos de como mejora la imagen resultado tras aplicar el método de *blending* que suaviza los bordes.

Capítulo 7

Extensión a vídeo

Tras el modelado del sistema de transferencia de estilo y textura entre imágenes mediante el uso de redes neuronales convolucionales y la implementación de los métodos de post-procesado a la red para imágenes estáticas nos planteamos dar un paso más e intentar adaptar el sistema para conseguir la misma transferencia de estilo y textura en vídeo.

En primer lugar, vamos a exponer los primeros pasos básicos, elementales, que llevamos a cabo para transferir estilo a los frames del vídeo. A continuación describiremos los problemas y limitaciones que esto supuso y finalmente expondremos las soluciones que planteamos.
Método naive:
1. Selección del vídeo que queremos procesar y del estilo o textura que queremos transferirle.
2. Extracción de los fotogramas (a partir de aquí se referirá a ellos con el nombre de *frames*) que componen dicho vídeo.
3. Transferencia de la textura o estilo seleccionado a cada *frame* del vídeo, se hacen pasar todos ellos por la red neuronal convolucional.
4. Aplicación a cada *frame* de los métodos de post-procesado implementados para preservación de la información de contenido.
5. Combinación de los *frames* estilizados para formar el vídeo final.

El primer paso entonces era extraer los *frames* del vídeo. Una vez teníamos dichos *frames*, nos enfrentábamos a la parte más compleja del proceso, y ésta puede dividirse en dos problemas principales.

El primero de ellos se trata de la transferencia de estilo o textura a cada *frame*, pues recordamos, que cuando trabajábamos con imágenes individuales, se invertía cierto tiempo en probar diferentes valores de β (pesos de intensidad de transferencia de estilo). Cuando tratábamos de transferir textura o estilo a dicha imagen para ver el comportamiento de las mismas según el valor de dicho β, realizábamos una búsqueda de los valores óptimos de β que aplicaríamos dependiendo del análisis de la imagen. Recordemos que utilizábamos valores de β altos para partes redundantes y con poco detalle y bajos para partes con mas detalles. Así pues, el trabajo de transferencia de estilo para una imagen tenía un coste de tiempo considerable, y esto debía reducirse a la hora de afrontar transferencia de estilo o textura a vídeo. Si por ejemplo, tenemos un vídeo 12 segundos de duración a 30 fps[1], tenemos unos 360 *frames* y no podemos invertir tanto tiempo estudiando y analizando qué valores de β son óptimos para cada *frame*.

El segundo problema es el del tiempo de procesado en la red neuronal convolucional. Si anteriormente para llevar a cabo la transferencia de textura o estilo a una única imagen uti-

[1] fps = *frames* por segundo

lizábamos una media de unos 5 valores de β distintos, que equivalen a 5 procesados distintos en la red, y cada uno de ellos tiene un coste en tiempo de unos 5 minutos, en el ejemplo del vídeo anterior supondría un coste total de 360 *frames* * 5 procesados/*frame* * 5 minutos/procesado = 9000 minutos = 150 horas = 6.25 días.

Por tanto, había que plantear soluciones a estos problemas, que pasaban por aceptar un compromiso entre calidad y coste temporal. El primer paso que se dio fue intentar conseguir los mejores resultados posibles haciendo uso del menor número de valores de β posible, y fue a lo largo del proceso de resolución de este problema junto con los análisis explicados en la Sección 5 cuando observamos que se podían conseguir resultados muy cercanos a los óptimos conseguidos hasta el momento, empleando tan solo dos valores de β. Uno de ellos será siempre $\beta = 0$ (imagen original, no hay transferencia de estilo), el otro será determinado para cada caso. De este modo sólo se ha procesar la imagen en la red para un único valor de β, pudiendo así reducir de 5 a 1 el número de procesados en la red. La solución al otro problema, el del estudio y análisis de los *frames* lo dividimos por escenas, es decir, en lugar de analizar cada uno de los 360 *frames*, si el vídeo se dividía en 5 escenas diferentes, decidimos analizar únicamente un *frame* por escena y aplicar las conclusiones del análisis al resto de *frames* de la misma escena, reduciendo así el arduo trabajo de analizar 360 *frames* a tan solo analizar 5.

Tras realizar el proceso anterior, el siguiente paso es adaptar el sistema de post procesado (*split and merge, blending*) para vídeo. No se introdujeron muchos cambios significativos: fundamentalmente el proceso consiste en aplicar a cada *frame* el procesado que aplicábamos a las imágenes estáticas.

Una vez aplicado el proceso anterior solo quedaría volver a reagrupar todos los *frames* estilizados en el orden correspondiente aplicando una función inversa a la extracción de *frames*. Podemos seleccionar el valor de *frames* por segundo que queremos que tenga el vídeo (generalmente el mismo valor que el vídeo original).

Así pues, conseguimos una transferencia de estilo y textura en vídeo competitiva con el estado del arte en esta materia. Resulta necesario indicar, que tal y como se explica en la Sección 2, si ya encontrábamos pocas referencias de este tipo de transferencia de textura y estilo con redes neuronales convolucionales en el caso de imagen, todavía encontramos menos en el caso de vídeo.

Seguidamente, una vez conseguidos unos resultados competentes en transferencia de estilo y textura a vídeo, nos centramos en el problema de la optimización temporal, pues además de lo ya abarcado anteriormente, el tiempo seguía siendo un problema con un amplio margen de mejora. Recordemos que un vídeo de unos 12 segundos a 30 fps nos llevaba finalmente: 360 *frames* * 1 procesado/*frame* * 5 min/procesado = 1800 minutos = 30 horas = 1.25 días. Entonces nos planteamos la realización de un sistema que contemplase el procesado de una fracción de los *frames* totales mientras que el resto fuesen predichos de alguna manera, dicho sistema se explica detalladamente en la Sección 7.1.

7.1. Estimación de Movimiento

Tras conseguir la transferencia de estilo a vídeo, uno de los problemas que debíamos enfrentar era el tiempo de procesado, pues si hasta ahora nos costaba la transferencia de estilo con un único β a una única imagen o *frame* 4 minutos, teniendo alrededor de 400 *frames*, el coste sería de unas 27 horas de procesado (el tiempo de procesado del resto de programas en Matlab es despreciable frente a esta cantidad de horas producto del procesado de la red neuronal convolucional).

Surge entonces el planteamiento de intentar reducir, de alguna forma ese tiempo. La solución propuesta fue la siguiente: el problema estaba en el gran numero de *frames* que había que procesar, por tanto, el objetivo tenía que ser reducir de alguna manera, ese número de

iteraciones de procesado. Así se pensó en poder procesar una imagen de cada dos y predecir la imagen que falta. Para ello existen distintos métodos de interpolación de imagen. Nosotros elegimos reconstruir la imagen ausente a partir de un método de estimación de movimiento de píxel [16]. Dicho método parte del supuesto de que se dispone de todos los *frames* que componen el vídeo original y un subconjunto de dichos *frames* procesados y estilizados que son aquellos que hemos decidido estilizar (antes de la introducción de este método eran todos los que componen el vídeo, ahora solo hemos tomado uno de cada dos *frames* por tanto la suma global de los *frames* estilizados es igual a la mitad de los *frames* totales que componen el vídeo). Por tanto, si queremos reconstruir el *frame* estilizado f_{i+1}^{pred}, hacemos la estimación de movimiento de píxeles entre los *frames* $f_i^{original}$ y $f_{i+1}^{original}$ del vídeo original, es decir, tomamos dichos *frames* y calculamos los vectores de movimiento que hay entre ambos. Los vectores de movimiento representan una estimación del desplazamiento horizontal y vertical de los píxeles de cada región de una imagen entre *frames* consecutivos. Esto puede expresarse matemáticamente de la siguiente forma: Sean $f(x, y)$ y $g(x, y)$ dos *frames* consecutivos:

$$g(x, y) = f(x + \Delta x, y + \Delta y) \approx f(x, y) + \Delta x * \frac{\partial}{\partial x} * f(x, y) + \Delta y * \frac{\partial}{\partial y} * f(x, y) \tag{7.1}$$

Esta última aproximación se corresponde con la serie de Taylor de primer orden. Podemos consultar la explicación completa del método en el artículo de Stanley H. Chan [16]. No obstante a continuación exponemos una explicación de lo más fundamental del método.

El método propuesto de estimación de movimiento se basa en una combinación de un algoritmo de búsqueda de correspondencias de bloque y un algoritmo de *optical flow* simplificado (ver Figura 7.1). En el primer paso, se utiliza un un algoritmo de búsqueda de correspondencias de bloque para determinar los desplazamientos de píxel $\overline{\Delta x}$ y $\overline{\Delta y}$. Si el desplazamiento (Δx, Δy) es el verdadero, entonces ($\overline{\Delta x}$, $\overline{\Delta y}$) determinado mediante el algoritmo de búsqueda de correspondencias de bloque, se corresponde con una correcta estimación de (Δx, Δy). Cuando ($\overline{\Delta x}$, $\overline{\Delta y}$) es determinado, desplazamos el bloque de la imagen $\overline{\Delta x}$ píxeles en la dirección x y a su vez $\overline{\Delta y}$ píxeles en la dirección y. El segundo paso es utilizar el algoritmo de la aproximación por la serie de Taylor para redefinir la búsqueda. Como la imagen desplazada $f(x+\delta x, y+\delta y)$ difiere de la imagen original solamente (δx, δy), donde $|\delta x| < 1$ y $|\delta y| < 1$, se considera válida la aproximación por series de Taylor. El desplazamiento total queda determinado por

$$\Delta x = \overline{\Delta x} + \delta x \tag{7.2}$$

$$\Delta y = \overline{\Delta y} + \delta y \tag{7.3}$$

Es importante señalar que el primer paso puede ser implementado a partir de cualquier algoritmo de búsqueda de correspondencias de bloque, por ejemplo: búsqueda completa, búsqueda de tres pasos, correlación plana de fase o estimación de movimiento bilateral.

Como observamos en la Figura 7.1, la salida del sistema de estimación de movimiento ($\overline{\Delta x}$, $\overline{\Delta y}$) conformaría los vectores de movimiento que buscábamos inicialmente.

Un algoritmo de búsqueda de correspondencias de bloque, es un método utilizado para encontrar el bloque, dentro del espacio de búsqueda, que mejor se ajusta a una búsqueda determinada. Como se muestra en la Figura 7.2, si queremos encontrar el bloque que mejor se ajusta en el *frame* f_i, con un determinado bloque del *frame* f_{i+1}, uno de los métodos mas directos es buscar a lo largo de todas las posibles posiciones en un rango de búsqueda definido. Este método de búsqueda es conocido como método de búsqueda completa [17].

Para reducir el número de puntos de búsqueda, se pueden utilizar métodos mas avanzados, por ejemplo, el método de búsqueda en tres pasos [18] o el método de búsqueda bilateral [19]. En la Figura 7.3, se muestra el concepto de búsqueda en tres pasos. En el primer paso, nueve candidatos centrados en $(0, 0)$ con un tamaño de paso inicial determinado, son analizados con

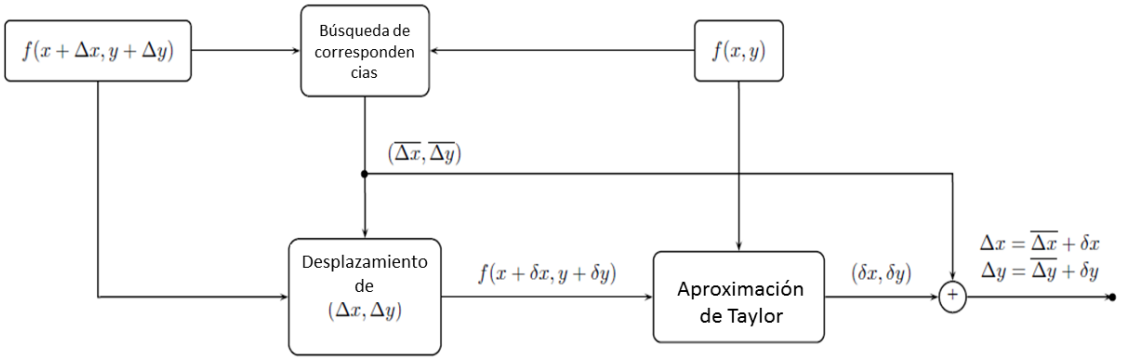


Figura 7.1: Ilustración del proceso de estimación de movimiento a partir de la combinación de un algoritmo de búsqueda de correspondencias de bloque y un algoritmo de *optical flow* simplificado.

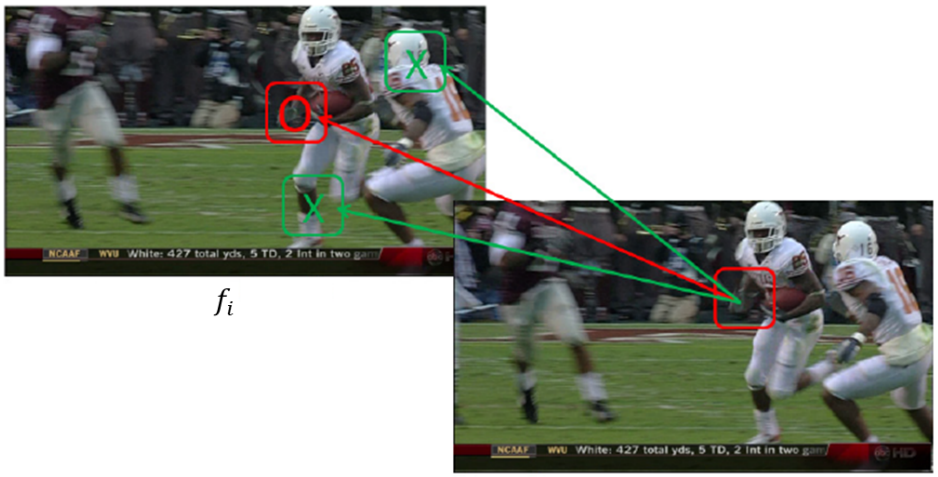


Figura 7.2: Método de búsqueda completa.

el objetivo de encontrar el mejor candidato con el menor error posible. En el segundo paso, el centro se mueve a la posición del mejor candidato encontrado y otros ocho candidatos se seleccionan con la mitad del tamaño de paso utilizado anteriormente. El tercer paso consiste en repetir los pasos anteriores escogiendo diferentes grupos de candidatos hasta que el tamaño de paso alcanza la precisión requerida en la estimación de movimiento.

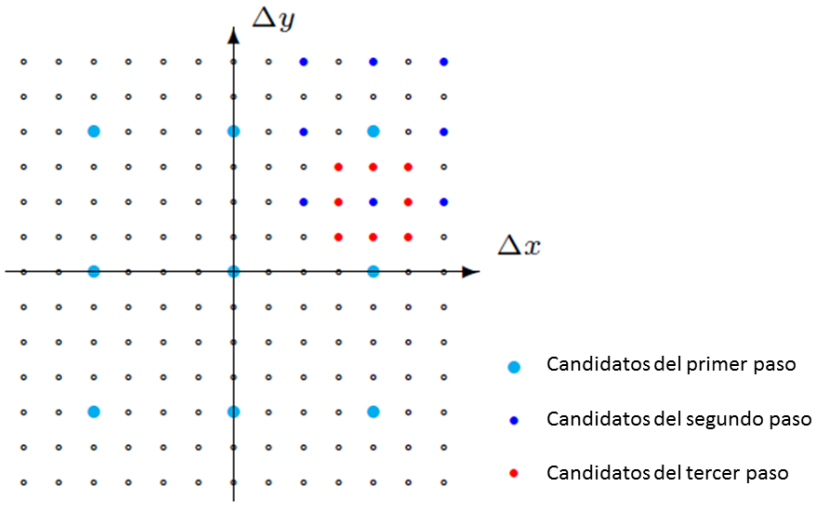


Figura 7.3: Método de búsqueda en tres pasos.

El siguiente paso consiste en tomar el *frame* $f_i^{stylized}$ que tenemos procesado y estilizado y los vectores de movimiento, $MV_{i,i+1}$, que hemos calculado y con estos dos datos realizamos el cálculo que nos dará la predicción del *frame* f_{i+1}^{pred}. Este proceso se repite hasta completar de nuevo el número total de *frames*. En la Figura (7.4) mostramos un sencillo ejemplo ilustrativo.

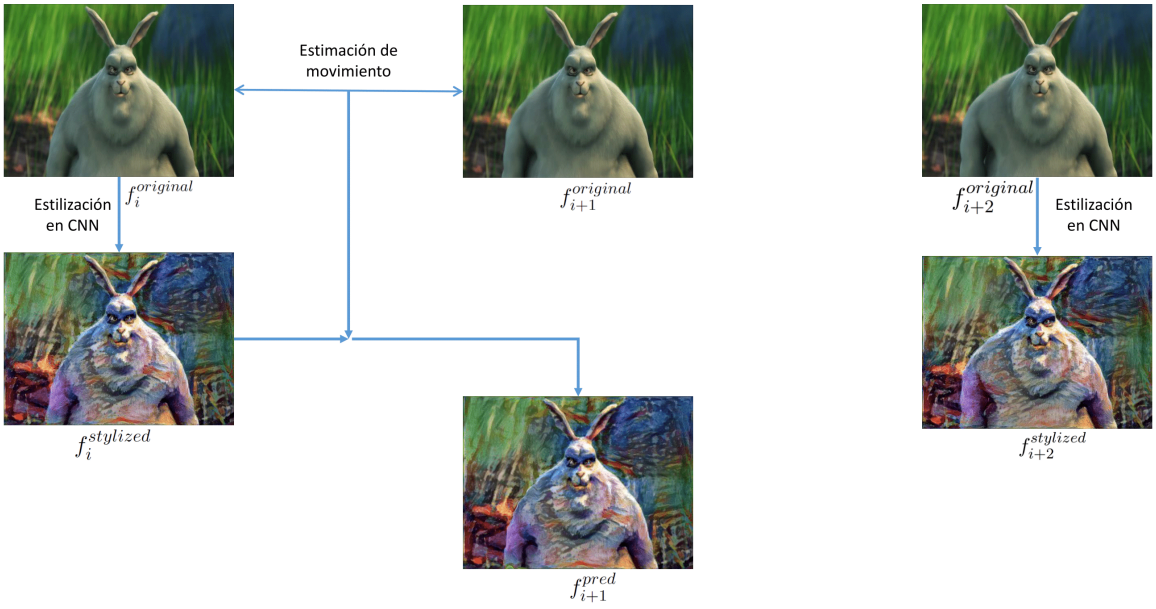


Figura 7.4: Ilustración del proceso de estimación de un *frame* a partir del *frame* anterior y vectores de movimiento.

Como estamos trabajando con vídeos de 30 fps, todos los *frames* corresponden a capturas o imágenes muy próximas entre sí en el tiempo. Por tanto, al predecir uno de cada dos *frames*,

el cálculo es muy preciso y el error es mínimo, por lo que los resultados son prácticamente iguales a los conseguidos cuando procesábamos todos los *frames*. Por tanto, tenemos unos resultados que podemos considerar de igual calidad que antes de introducir el método y nos hemos ahorrado la mitad del tiempo de procesado (pasamos, por ejemplo, de 25 horas que cuesta procesar 350 *frames*, a 12 horas ahora que solo procesamos 175).

Una consecuencia de lo anterior es que se puede reducir aún más el tiempo de procesado prediciendo más *frames* y procesando menos. Se ha probado también a predecir 2 de cada 3 *frames*. En este caso el proceso a seguir sería el siguiente. Tendríamos por un lado solamente procesado y estilizado el *frame* $f_i^{stylized}$ y tendríamos que predecir los *frames* f_{i+1}^{pred} y f_{i+2}^{pred}. Entonces, computaríamos la estimación de movimiento de píxel mediante el cálculo de los vectores de movimiento entre los *frames* originales $f_i^{original}$ y $f_{i+1}^{original}$ y por otro lado $f_{i+1}^{original}$ y $f_{i+2}^{original}$. Así tenemos dos grupos de vectores de movimiento: $MV_{i,i+1}$ y $MV_{i+1,i+2}$. Después procederíamos a la predicción de los *frames* $f_{i+1}^{stylized}$ y $f_{i+2}^{stylized}$. Primero utilizando el *frame* $f_i^{stylized}$ y $MV_{i,i+1}$ obtendríamos la predicción del *frame* f_{i+1}^{pred}, después utilizando dicho *frame* y $MV_{i+1,i+2}$ obtendríamos la predicción del *frame* f_{i+2}^{pred}. Así pues, con esta segunda solución perdemos un poco más de calidad pero ganamos mucho en tiempo de procesado, reduciéndolo ahora a una tercera parte del empleado inicialmente (si eran 25 horas los 350 *frames*, ahora sólo serían alrededor de 8 horas). Se trata por tanto de un compromiso entre calidad y tiempo de procesado: cuanta mayor calidad deseemos mayor será el tiempo que necesitemos invertir. No obstante las dos soluciones propuestas se sitúan en el equilibrio que proporciona buena calidad (comparando con el vídeo donde se utilizan todos los *frames* procesados, que asumimos es el caso de mejor calidad posible), y un tiempo de procesado reducido en gran medida (a la mitad y a una tercera parte en cada caso). Si pretendemos utilizar mayor predicción perderemos calidad, así nosotros proponemos nuestro segundo ejemplo como límite, es decir, predicción de 2 de cada 3 *frames*.

En la Figura (7.5) mostramos un esquema que recoge el proceso de estimación de movimiento.

En la Figura (7.6) mostramos una comparativa entre el *frame* f_{i+1}^{pred} del ejemplo anterior predicho y el que obtendríamos si hiciésemos pasar el correspondiente *frame* original $f_{i+1}^{original}$ por la red. En algún caso, debido a la aleatoriedad comentada anteriormente que introduce la red neuronal convolucional en la transferencia de estilo o textura, hemos observado que se generan ciertas anomalías en el proceso de transferencia y este es el caso del *frame* seleccionado (como podemos observar en la imagen de la derecha). Este defecto no es apreciable en el vídeo, debido a que ocurre en un número de ocasiones despreciable en relación al número total de *frames* por segundo que contiene el vídeo, no obstante resulta muy interesante apreciar que dichos errores se corregirían utilizando el método de estimación de movimiento, ya que el *frame* f_{i+1}^{pred} se genera con los vectores de movimiento calculados sobre los *frames* originales ($f_i^{original}$ y $f_{i+1}^{original}$) y el *frame* estilizado previo $f_i^{stylized}$, y este último no presentaba las mencionadas anomalías, por tanto el predicho f_{i+1}^{pred} tampoco las presenta. Ilustramos éste fenómeno en la Figura (7.6)

Resulta importante señalar que hemos basado la explicación en análisis cualitativos en lugar de utilizar cifras de error cuantitativas dado el carácter subjetivo de las soluciones. No existe una forma de medir si la estimación de un *frame* es correcta o no, es necesaria la supervisión de un maestro para llevar a cabo dicha tarea.

7.1.1. Cambios de escena

Uno de los problemas que hemos tenido que resolver consecuencia del sistema de estimación de movimiento de píxel, ha sido el cambio de escena. La estimación de movimiento de píxel

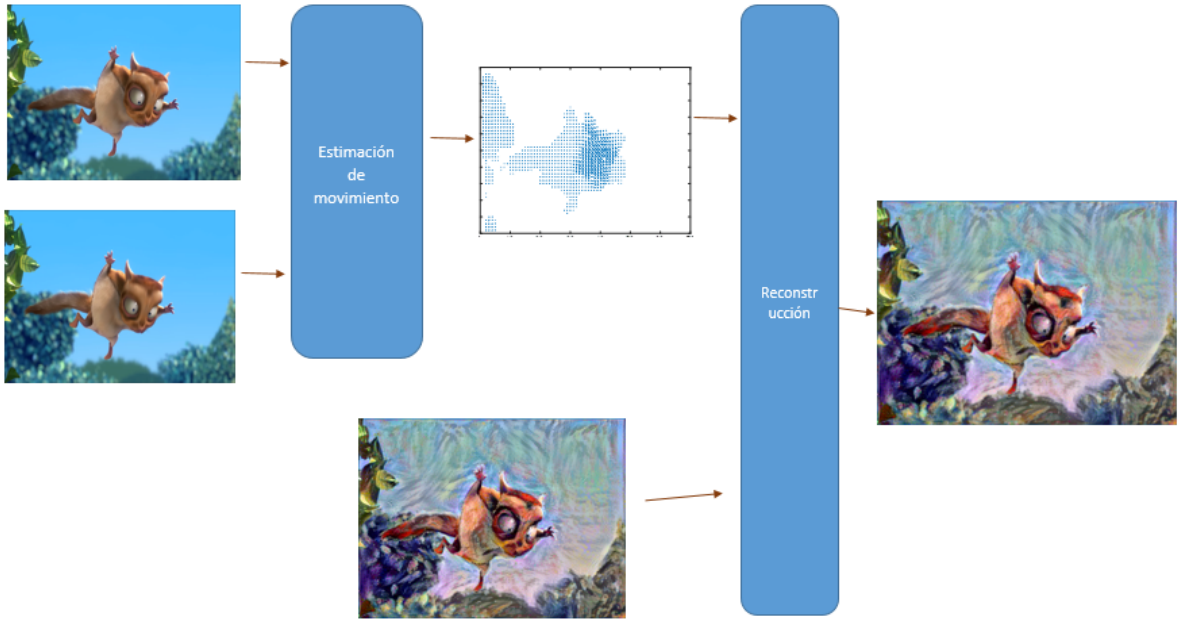


Figura 7.5: A la izquierda, los *frames* originales $f_i^{original}$ y $f_{i+1}^{original}$ pasan por el estimador de movimiento, que calcula los vectores de movimiento entre ambos *frames*. Después el reconstructor recibe como entrada el *frame* estilizado $f_i^{stylized}$ y la salida del estimador, con ello realiza la predicción del *frame* f_{i+1}^{pred}.

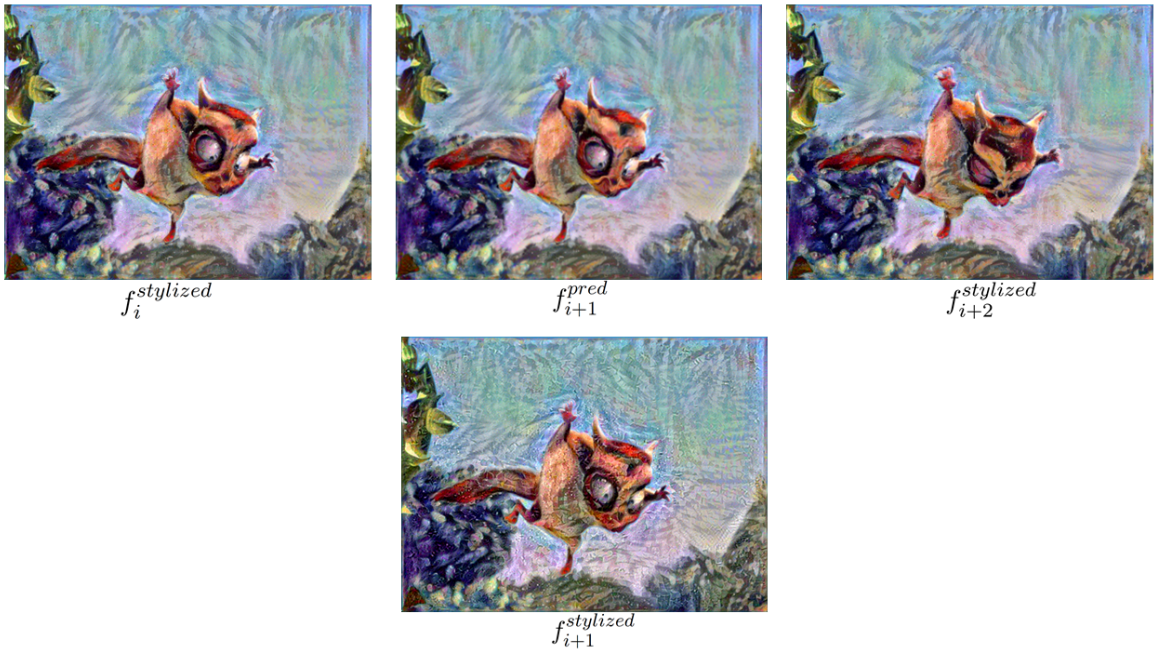
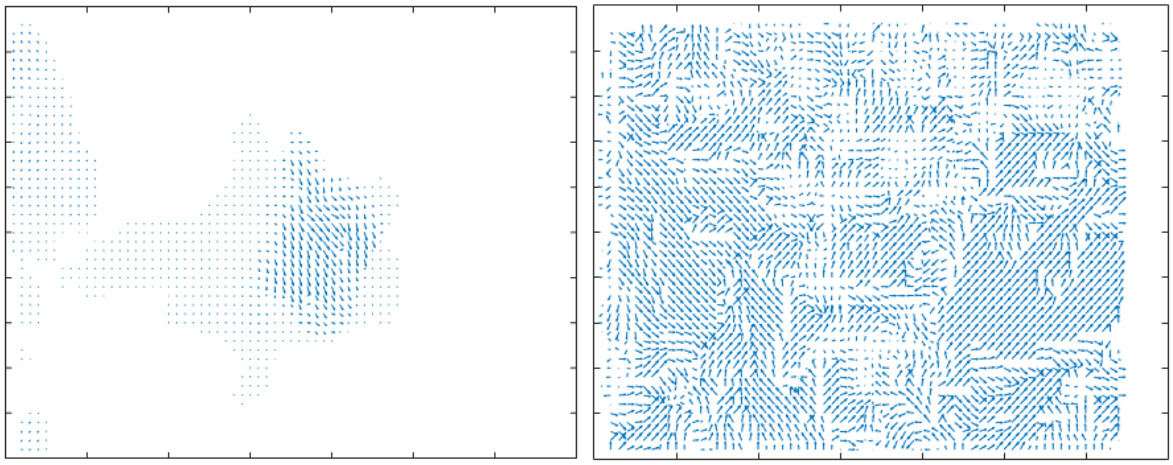


Figura 7.6: Como podemos observar el *frame* predicho f_{i+1}^{pred} sigue la línea de estilo del *frame* anterior $f_i^{stylized}$ y del siguiente $f_{i+2}^{stylized}$, de forma que no muestra el ruido que sí aparecía en el *frame* $f_{i+1}^{stylized}$, que sería el que aparecería en ese lugar si no hubiésemos utilizado estimación de movimiento.

coherente se lleva a cabo entre 2 *frames* cuyo contenido es prácticamente el mismo pero hay

ligeros cambios, es decir, hay un movimiento en la escena y se calcula cuánto se ha movido un píxel determinado en una imagen con respecto a la anterior. Esto obviamente, no se puede llevar a cabo cuándo hay un cambio de escena, ya que toda la información de contenido en la imagen es completamente distinta. Teníamos que detectar de forma automática cuando un *frame* pertenecía a una escena distinta que su *frame* anterior. Esto se ha logrado haciendo un análisis del módulo de los vectores de movimiento obtenidos, pues como hemos comentado anteriormente, al trabajar con vídeos de unos 30 fps, 2 *frames* corresponden a capturas muy próximas entre sí en el tiempo. Por tanto, el movimiento entre dos *frames* de una misma escena nunca será muy grande. Esto se traduce en unos vectores de movimiento con unos valores de módulo normalmente menores a un valor umbral determinado ($|\vec{MV}| < k$, donde k = umbral). Cuando se produce un cambio de escena se están procesando los vectores de movimiento de 2 *frames* con información de contenido totalmente distinta y por tanto dichos vectores, además de no tener ningún significado, tienen unos valores en módulo extremadamente grandes y están presentes en toda la imagen, pues ningún píxel del *frame* en cuestión tiene correspondencia en el anterior (en cambio cuando son *frames* de una misma escena, normalmente se centran en una parte de la imagen por ejemplo el movimiento de cabeza de una persona en la misma, mientras que el fondeo, invariable no produce vectores de movimiento). De esta manera podemos detectar un cambio de escena. En la Figura (7.7) se muestra un comparación de una predicción normal (basada en dos *frames* consecutivos de una misma escena) con una predicción falsa correspondiente a dos *frames* consecutivos de escenas distintas. Ahora queda por decidir que hacer cuando se produce dicho cambio y la opción por la que se ha optado es eliminar el *frame* que correspondería a esa predicción falsa del último *frame* de la escena.



(a) Predicción normal, entre *frames* consecutivos de una misma escena

(b) Predicción falsa, entre *frames* consecutivos de escenas distintas

Figura 7.7: Comparación de los campos de vectores de movimiento obtenidos entre *frames* de de una misma escena(a) y entre *frames* de escenas distintas(b).

Capítulo 8

Resultados

La estructuración y desarrollo de esta memoria ha sido llevada a cabo de tal forma que se han ido exponiendo y explicando los resultados que obteníamos en cada apartado correspondiente a los diferentes estudios, análisis y mejoras implementadas. Por tanto con el objetivo de no repetir lo ya contado, se ha decidido dedicar este apartado a establecer un resumen de carácter generalista de los resultados obtenidos, así referenciaremos cuando se precise a los apartados referentes donde se ha entrado y explicado en profundidad los resultados obtenidos en ese determinado caso.

Además recopilamos en esta sección las diferentes imágenes que mostramos en la Sección 6 y añadimos otros resultados para complementar la colección de resultados obtenidos tras la implementación global del sistema aplicado a imagen estática.

Así dividimos este apartado en función del impacto que tiene cada mejora, implementación o análisis conseguidos.

8.1. Sistema de transferencia de textura y estilo mediante el uso de CNNs y optimización del coste temporal

En primer lugar, hablaremos de la importancia que ha tenido el profundo y minucioso análisis que realizamos sobre el sistema y la red neuronal convolucional utilizada. Como explicamos en la Sección 5, analizamos y estudiamos el sistema de transferencia utilizado, el propuesto por Gatys et al en su artículo *Image Style Transfer Using Convolutional Neural Networks* [1], así tras llevar a cabo diferentes pruebas y modificaciones del mismo, nos propusimos analizar las diferentes funciones encargadas de llevar a cabo la transferencia de textura o estilo entre las imágenes seleccionadas. Tras contemplar la evolución de dichas funciones en su paso por la red y así mismo en las diferentes iteraciones que sobre esta se sometían a las imágenes, decidimos establecer un criterio de selección de los parámetros que dirigían dicha transferencia, las mencionadas variables α y β. Así tras realizar el exhaustivo análisis de las distribuciones frecuenciales de las imágenes estilo, tal y como explicamos en la Sección 5, observamos un patrón en el comportamiento de estas cuando eran sometidas a la red, así concluíamos que observando dichas distribuciones frecuenciales, si presentaban estas componentes de alta frecuencia, producirían una mayor distorsión que si lo eran de baja frecuencia y que además llegarían al punto de convergencia de máxima transferencia de estilo o textura en menos iteraciones.

8.1.1. Optimización del coste temporal

Uno de los descubrimientos fundamentales con respecto a reducción del coste temporal del procesado en la red, fue el que deducimos cuando analizamos las funciones de pérdidas

que permitían generar la transferencia de textura y estilo entre imágenes, observamos en una representación de las mismas en función del número de iteraciones que se procesaban en la red que tanto las referentes a al contenido, como las referentes al estilo y por tanto las globales, alcanzaban un punto de convergencia a partir del cual, en las siguientes iteraciones las variaciones de dichas funciones eran despreciables. Con esto, unido a un estudio y análisis de las imágenes salida de la red en cada una de las mencionadas iteraciones, nos permitió concluir que no eran necesarias las 1000 iteraciones que el sistema de Gatys et al. [1] proponía por defecto. El punto de convergencia se alcanzaba antes de la mitad de dicho valor en la mayoría de los casos, lo que nos permitió reducir el número de iteraciones de 1000 a 500, reduciendo también a la mitad el tiempo de procesado en la red, inicialmente de unos 8 o 9 minutos y ahora inferior a 5.

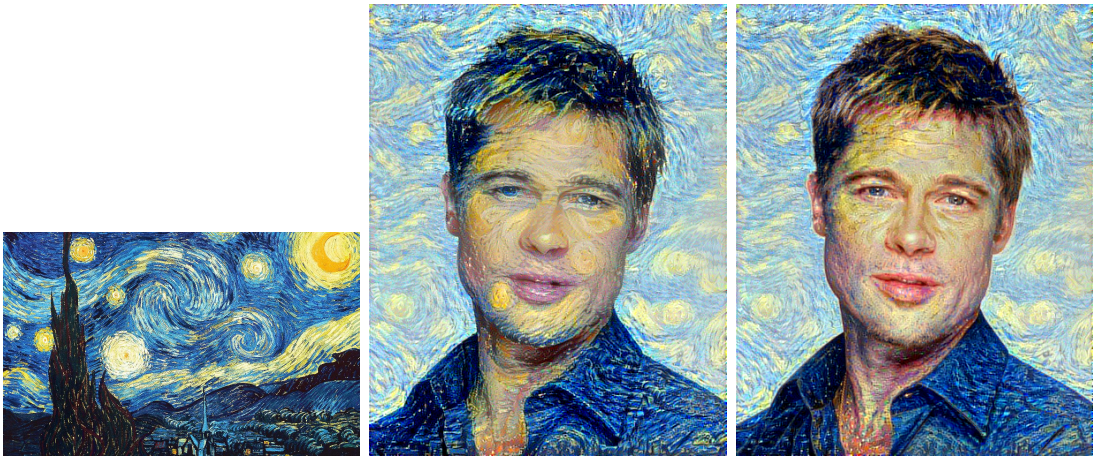
8.2. Transferencia selectiva de textura o estilo

Como explicamos en la Sección 6, gracias a la implementación de este método conseguimos llevar a cabo una transferencia selectiva de textura o estilo sobre la imagen objetivo. Conseguíamos de este modo, unas imágenes que preservaban mucho mejor los pequeños detalles, líneas y bordes de la imagen original portadora de la información de contenido que recibía la transferencia de la textura o estilo en cuestión.

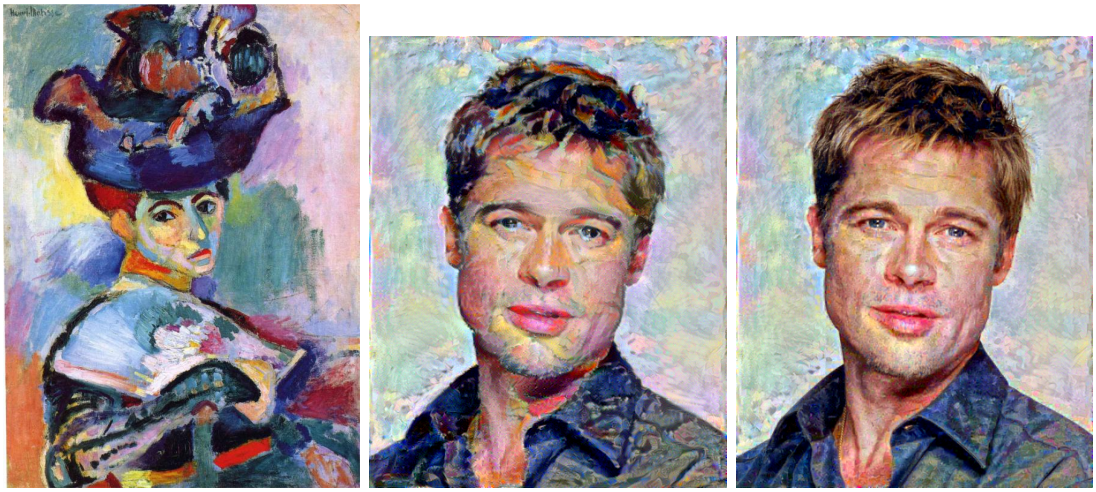
También añadimos la implementación del método de *blending* que corregía como bien explicamos en el apartado que le dedicamos al mismo sistema, los problemas que presentaba el paso final del *Split and Merge*, estos problemas eran la presencia de transiciones bruscas entre parches tras la unión de los mismos que daba lugar a líneas o betas en la frontera de dichas transferencias, así pues con esta solución conseguíamos suavizar dichas transiciones haciendo prácticamente inapreciables las mencionas betas o bordes de frontera.

(a) Imagen estilo. (b) Imagen a la salida de la red. (c) Nuestro resultado.

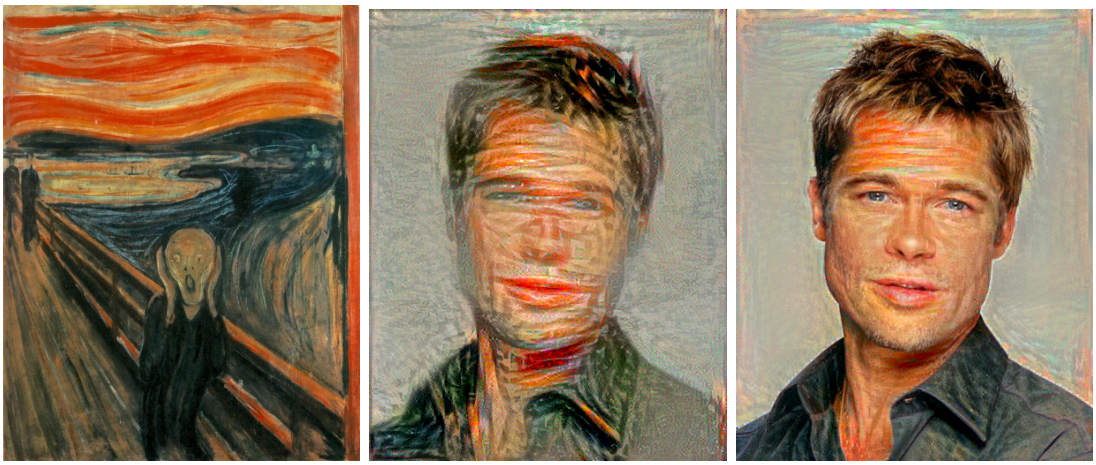
Figura 8.1: Resultados para la imagen de contenido Brad Pitt con diferentes estilos (I). Se observa como nuestro resultado preserva mejor la información de contenido que la imagen que se obtiene a la salida de la red.



(a) Imagen estilo. (b) Imagen a la salida de la red. (c) Nuestro resultado.

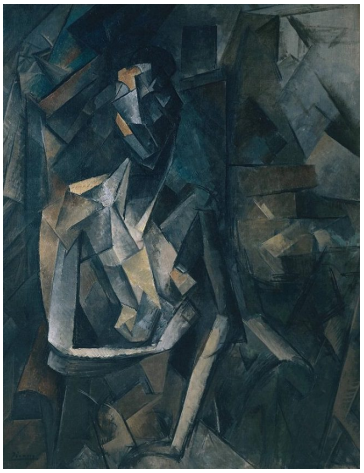


(a) Imagen estilo. (b) Imagen a la salida de la red. (c) Nuestro resultado.



(a) Imagen estilo. (b) Imagen a la salida de la red. (c) Nuestro resultado.

Figura 8.4: Resultados para la imagen de contenido Brad Pitt con diferentes estilos (II). Se observa como nuestro resultado preserva mejor la información de contenido que la imagen se obtiene a la salida de la red.



(a) Imagen estilo. (b) Imagen a la salida de la red. (c) Nuestro resultado.

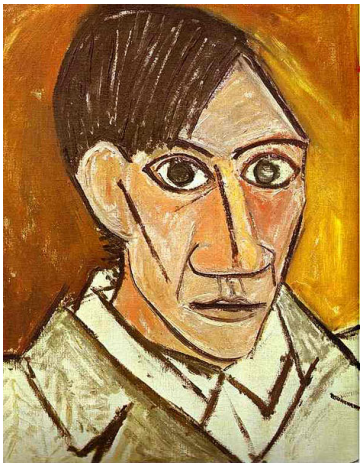


(a) Imagen estilo. (b) Imagen a la salida de la red. (c) Nuestro resultado.



(a) Imagen estilo. (b) Imagen a la salida de la red. (c) Nuestro resultado.

Figura 8.7: Resultados para la imagen de contenido Brad Pitt con diferentes estilos (III). Se observa como nuestro resultado preserva mejor la información de contenido que la imagen se obtiene a la salida de la red.



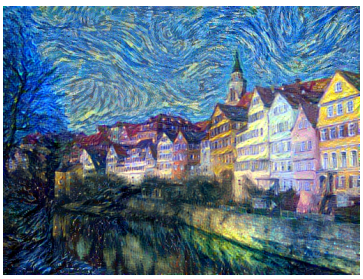
(a) Imagen estilo.

(b) Imagen a la salida de la red.

(c) Nuestro resultado.

(a) Imagen estilo.

(b) Imagen a la salida de la red.



(c) Nuestro resultado.

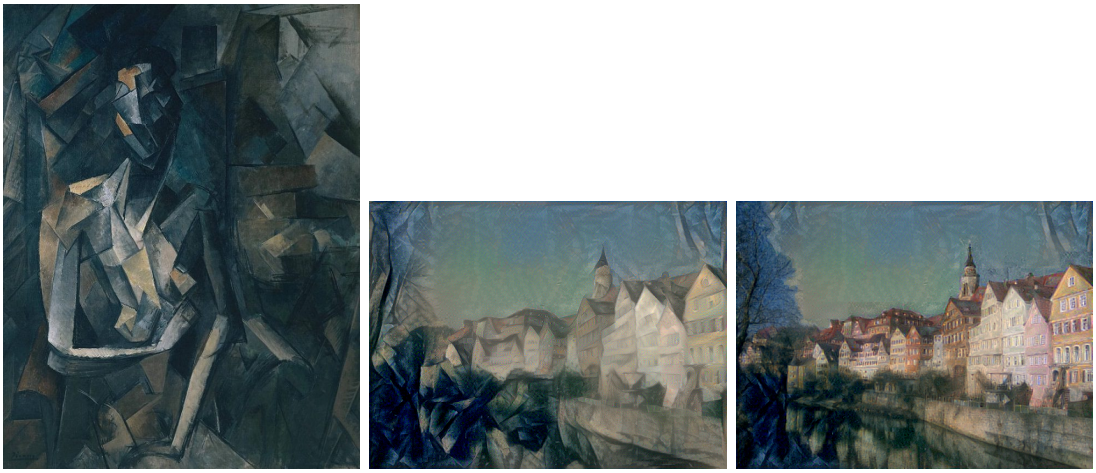
(a) Imagen estilo.

(b) Imagen a la salida de la red.

(c) Nuestro resultado.

Figura 8.10: Resultados para la imagen de contenido Tubingen con diferentes estilos (I). Se observa como nuestro resultado preserva mejor la información de contenido que la imagen se obtiene a la salida de la red.

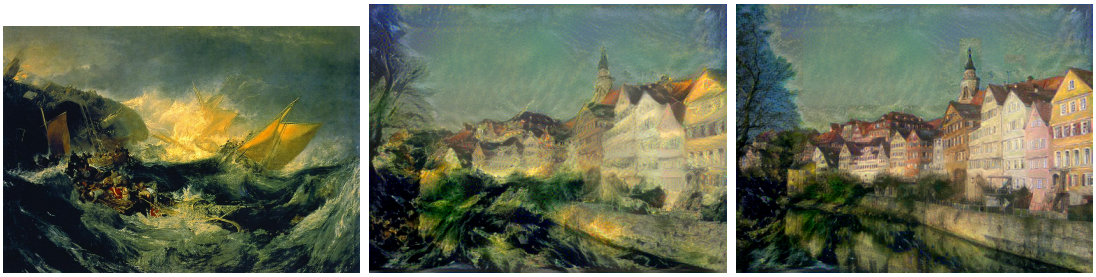
(a) Imagen estilo. (b) Imagen a la salida de la red. (c) Nuestro resultado.



(a) Imagen estilo. (b) Imagen a la salida de la red. (c) Nuestro resultado.

(a) Imagen estilo. (b) Imagen a la salida de la red. (c) Nuestro resultado.

Figura 8.13: Resultados para la imagen de contenido Tubingen con diferentes estilos (II). Se observa como nuestro resultado preserva mejor la información de contenido que la imagen se obtiene a la salida de la red.



(a) Imagen estilo. (b) Imagen a la salida de la red. (c) Nuestro resultado.

Figura 8.14: Resultados para la imagen de contenido Tubingen con diferentes estilos (III). Se observa como nuestro resultado preserva mejor la información de contenido que la imagen se obtiene a la salida de la red.

8.3. Extensión del sistema a vídeo

En la Sección 7 explicamos todo el proceso que llevamos a cabo para conseguir adaptar nuestro sistema de transferencia de textura y estilo entre imágenes a vídeo. Así pues, explicamos como extraíamos los *frames* del vídeo que queríamos estilizar, después procesábamos cada *frame* en la red neuronal convolucional bajo el sistema de transferencia de textura y estilo utilizado, seguidamente aplicábamos nuestros métodos de *Split and Merge y Blending* a cada uno de los mencionados *frames* y finalmente creábamos de nuevo un vídeo con los *frames* ahora estilizados.

8.3.1. Estimación de movimiento y optimización del tiempo de procesado

Tal y como explicábamos en la Sección 7, uno de los problemas que presentaba la realización de transferencia de estilo y textura a vídeo era el gran coste temporal que suponía, pues procesar un solo *frame* nos llevaba unos 4 o 5 minutos, así por ejemplo, procesar un vídeo de unos 20 segundos a 30 fps, suponía unos 2400 minutos de procesado que son unas 40 horas. Así decidimos introducir un sistema de estimación de movimiento de píxel, con el que conseguíamos reducir el procesado en 2 o 3 veces el anterior, pues como explicamos en el correspondiente apartado, podíamos procesar la mitad o la tercera parte de los *frames* del vídeo y el resto serían predichos después haciendo uso del *frame* anterior y del cálculo de unos vectores de movimiento, obtenidos a partir de los *frames* originales correspondientes a dicho *frame* anterior y el que estamos prediciendo.

Capítulo 9

Conclusiones y trabajo futuro

Las conclusiones fundamentales que hemos podido sacar a lo largo de la realización del proyecto son las siguientes. En primer lugar, hemos podido comprobar lo poco predecibles que son este tipo de sistemas de transferencia de textura o estilo, es decir, la imposibilidad de predecir con total certeza que salida dará el sistema dadas dos imágenes de entrada. Esto se debe fundamentalmente, a los filtros de convolución de las diferentes capas que componen una red neuronal convolucional entrenados para la extracción y detección de patrones en imágenes.

Así bajo estas condiciones de incertidumbre decidimos, llevar a acabo un estudio y análisis del sistema con el objetivo de conseguir cierta predicción o intuición de como será la imagen salida en función de las imágenes de entrada. Gracias a este análisis, basado en el estudio de las distribuciones frecuenciales de las imágenes que aportaban el estilo o textura, conseguimos establecer un cierto grado de correspondencia entre dichas distribuciones y el comportamiento de la imagen salida en función de los valores de β utilizados. De forma que establecimos el siguiente criterio, si las distribuciones frecuenciales de la imagen estilo de entrada presentaban componentes de baja frecuencia, valores pequeños de β, generarán una transferencia de textura o estilo muy suave, produciendo una distorsión muy leve de los pequeños detalles y bordes de la imagen final, así tendremos que utilizar valores mas altos para tener una mayor presencia de la textura o estilo en la imagen final. En cambio si las componentes son de alta frecuencia, valores pequeños de β rápidamente generarán salidas con mucha presencia de la textura o estilo utilizadas, produciendo así una transferencia fuerte de la misma y con ello altas distorsiones en los detalles pequeños y los bordes de la imagen salida.

Otra conclusión a la que llegamos en los primeros meses de estudio y que nos sirvió para establecer uno de los criterios de diseño que mantendríamos hasta el final, fue el sustituir el ruido blanco gaussiano propuesto por defecto como imagen semilla por la misma imagen que la que elegimos como portadora de la información de contenido, que es la que recibe la transferencia de textura o estilo. Como se explica en la Sección 5, tomamos esta decisión basándonos en el hecho de que resulta mas coherente comenzar el proceso con la imagen a al cual queremos transmitir la textura o estilo que con un ruido blanco gaussiano, dado el carácter de nuestro proyecto, el cual se fundamenta en el intento de preservar con la menor distorsión posible la información de contenido de la imagen que recibe dicha transferencia. De esta manera nuestra imagen salida comienza siendo la misma que la imagen contenido y poco a poco, iteración a iteración va tomando esa transferencia de textura o estilo de la imagen que seleccionamos como portadora de los mismos. En la sección mencionada pueden encontrarse los estudios y análisis que nos llevaron a esta conclusión, así como también pueden encontrase diferentes gráficas e imágenes que apoyan estos estudios y análisis.

En la Subsección 5.1 , explicábamos como conseguimos reducir el coste temporal de procesado en la red neuronal a través de un estudio de las funciones de pérdidas en función del

número de iteraciones a las que se sometían las imágenes sobre la red, junto con una observación de las imágenes salida del sistema en cada iteración. Llegamos a través de dicho análisis a la conclusión de que no era necesario someter las imágenes en la red a las 1000 iteraciones que se proponían por defecto, pues bastaba con la mitad. Vimos en las funciones de pérdidas que estas convergían en ciertos valores en torno a las iteraciones 400 o 500. Observando las imágenes salida de la red correspondientes a las iteraciones señaladas, llegábamos a la misma conclusión. De esta manera, redujimos el tiempo de procesado de cada imagen a la mitad, inicialmente, con las 1000 iteraciones dicho tiempo tenía unos valores en torno a 8-10 minutos, tras la reducción a 500 iteraciones, el coste pasaba a tomar valores de 4-5 minutos.

Tras analizar los diferentes problemas o carencias que presentaba el sistema, intentamos conseguir la transferencia de textura o estilo mas óptima desde el punto de vista de la conservación de la información de contenido, de forma que los detalles, lineas y bordes se distorsionen en la menor medida de los posible, a través de todos los análisis y estudios que se han explicado. No obstante, nos dimos cuenta de la necesidad de implementar o añadir nuevos métodos de procesado de imagen al sistema con el que trabajábamos, pues podíamos mejorar los resultados realizando las modificaciones oportunas en el mismo pero esto no era suficiente. Así, concluimos que necesitábamos añadir un sistema nuevo y diferentes soluciones fueron analizadas y llevadas a la práctica, si bien, la que mejor resultado nos aportó fue la implementación de un sistema de transferencia selectiva de textura o estilo, dicho sistema se explica detalladamente en la Sección 6. Tras su puesta en práctica también tuvimos que enfrentarnos a la mejora de diferentes puntos y a la solución de ciertos problemas o carencias que todavía presentaba, uno de ellos, posiblemente el mas significativo es el que explicamos en la Sección 6.3 y que trataba de solucionar ciertos problemas de transiciones entre parches que aparecían producto de la aplicación del método *Split and Merge*.

Posteriormente, procedimos a la adaptación del sistema que transfería textura y estilo para poder llevar a cabo dicha transferencia a un vídeo. Destacamos la incorporación de un sistema de estimación de movimiento, explicado en la Sección 7.1. La conclusión fundamental que sacamos de aquí es que este sistema puede funcionar correctamente gracias a la conservación de la información de contenido. Esto es así, ya que la extracción de vectores de movimiento que se lleva a cabo, se realiza sobre los *frames* originales del vídeo ($f_i^{oiginal}$ y $f_{i+1}^{original}$) y por tanto, la predicción del *frame* f_{i+1}^{pred} se realiza con los vectores de movimiento $MV_{i,i+1}$ junto con el *frame* estilizado $f_i^{stylized}$. Así gracias a dicha conservación de la información de contenido se guarda una correspondencia entre el mencionado *frame* estilizado $f_i^{stylized}$ y los correspondientes *frames* originales, si no se produjese dicha preservación de la información de contenido, al llevar a cabo el paso de predicción y aplicar los vectores calculados sobre un *frame* que no guarda la información mencionada se produciría una traslación de píxeles incoherente, dando lugar a una imagen aleatoria donde aun quedaría mas distorsionada la información de contenido de la imagen o *frame* que ha recibido la transferencia de textura o estilo. Por tanto la conclusión fundamental, es que el sistema funciona correctamente, gracias al enfoque que le hemos dado a todo el proceso, que es llevar a cabo una transferencia de textura o estilo entre imágenes intentando siempre preservar y conservar la información de contenido de la imagen que recibe dicha transferencia, y gracias a la introducción de este sistema predictivo conseguimos reducir considerablemente el coste temporal de la producción de un vídeo estilizado. Esta reducción queda también explicada en la Sección 7.1 y ahí podemos ver como conseguimos reducir en 2 o 3 veces el coste inicial.

9.1. Conclusión personal

Una conclusión general que podemos hacer al finalizar el proyecto, es la comprobación empírica de que la ingeniería se basa en la búsqueda de la solución no perfecta a uno o varios problemas. Cada solución a un problema lleva consigo la aparición de un problema nuevo y finalmente todo se basa en la optimización de un compromiso aplicado a dicha solución, el cual es un balance entre lo que se mejora y lo que se empeora y en que medida, al aplicar la solución propuesta.

9.2. Trabajo futuro

Se podría seguir profundizando en el análisis predictivo de la red, de forma que sea posible partiendo de un análisis previo de diferentes parámetros de la imagen que aporta la transferencia de textura o estilo, llevar a cabo una predicción de mayor nivel y precisión que la conseguida hasta el momento. Nosotros comenzamos dicho análisis estudiando las componentes frecuenciales de la mencionada imagen, si bien, una imagen dispone de muchas y diversas estadísticas que pueden ser analizados. El problema está en encontrar cual o cuales de ellos son los predictores apropiados. Nuestro intento de llevar a cabo una correspondencia entre las mencionadas distribuciones frecuenciales y los valores de β que nos dan una salida u otra, tiene sus limitaciones, y quizá se pueda llegar a una predicción de la salida en función de β mucho mas precisa si se tiene en cuenta un mayor número de parámetros.

En cuanto a la generación de vídeo estilizado, múltiples opciones se presentan como alternativa al sistema de estimación de movimiento que hemos utilizado, basado en la estimación de los vectores de movimiento entre *frames* consecutivos. También se puede centrar el estudio en otros métodos orientados a la reducción del tiempo de procesado.

Como se ha explicado, el sistema de transferencia de textura o estilo entre imágenes haciendo uso de redes neuronales convolucionales que se ha utilizado es un sistema que fue propuesto y planteado por primera vez por Gatys et al [1], pero cuya implementación en código abierto fue aportada por Justin Johnson de la Universidad de Stanford [3], y dicha implementación es la que nosotros empezamos a utilizar al inicio del proyecto, el cual se inició en junio de 2016. Así, continuamos con esta implementación hasta el final del mismo, no obstante es importante señalar que en octubre de 2016, el mismo Johnson llevo a cabo una mejora del sistema inicialmente propuesto que permitía reducir considerablemente el tiempo de procesado de la red.

Adicionalmente, se esta trabajando en la elaboración de un artículo científico a partir del trabajo realizado durante este proyecto.

Anexos

Anexo: Imágenes contenido

A lo largo de la realización del trabajo se han utilizado numerosas imágenes como portadoras de contenido en las diversas pruebas que han sido llevadas a cabo a durante dicho periodo. No obstante, presentamos a continuación aquellas imágenes-contenido que más hemos utilizado y que son a su vez las que hemos seleccionado para representar los diferentes ejemplos y resultados presentados en esta memoria.

Adjuntamos un enlace a un repositorio de Google Drive donde se encuentran y pueden consultarse las diferentes imágenes contenido utilizadas.

ENLACE

https://drive.google.com/drive/folders/0B4LQLStIJSMxWkh3WFdVMTd4UTQ

Anexo: Imágenes estilo

Del mismo modo que ocurría con las imágenes contenido, múltiples imágenes han sido utilizadas para desarrollar las numerosas pruebas y experimentos que se llevaron a cabo durante el tiempo que duró el proyecto. Sin embargo, hemos decidido hacer una selección de las mas importantes y las mas utilizadas para mostrarlas en este anexo.

Adjuntamos un enlace a un repositorio de Google Drive donde se encuentran y pueden consultarse las diferentes imágenes estilo utilizadas.

ENLACE

https://drive.google.com/drive/folders/0B4LQLStIJSMxQUFhVHVTYXh4S00

Anexo: enlaces a los vídeos

En este anexo incluimos el enlace a la cuenta de Vimeo donde se encuentran los vídeos principales que representan los resultados conseguidos durante la realización de la transferencia de estilo a vídeo en este proyecto, con esto queremos señalar que no hemos cargado en el servidor del enlace que se facilita todos los vídeos que hemos generado, sino que hemos decidido subir una selección de los mismos, representativa de los resultados conseguidos y que incorpora los siguientes vídeos:

1. Dos vídeos originales correspondientes a escenas de:
1.1 Big Buck Bunny (*Cartoons*).
1.2 The lord of the rings (Realista).
2. Dos imágenes correspondientes a las texturas o estilos que hemos transferido a los vídeos anteriores:
2.1 *Woman-Hat*
2.2 *Starry Night* 3. Cuatro vídeos de transferencia de textura correspondientes a:
3.1 *Big Buck Bunny* con *Woman-Hat*
3.2 *Big Buck Bunny* con *Starry Night*
3.3 *The lord of the rings* con *Woman-Hat*
3.4 *The lord of the rings* con *Starry Night*
4. Cuatro vídeos correspondientes a los 4 anteriores pero aplicando los métodos *Split and Merge y Blending.*
5. Cuatro vídeos correspondientes a 4 ejemplos de aplicación de estimación de movimiento:
5.1 *Big Buck Bunny* con *Woman-Hat* (estimación de movimiento de 1 frame de cada 2)
5.2 *Big Buck Bunny* con *Starry Night* (estimación de movimiento de 1 frame de cada 2)
5.3 *The lord of the rings* con *Woman-Hat* (estimación de movimiento de 1 frame de cada 2)
5.4 *The lord of the rings* con *Woman-Hat* (estimación de movimiento de 2 frames de cada 3)

ENLACE

https://vimeo.com/user60374213

Anexo: enlaces a los códigos

En este Anexo, adjuntamos un enlace a un repositorio de Google Drive donde se encuentran y pueden consultarse los códigos mas importantes y fundamentales implementados y utilizados para la realización del proyecto y corresponden a:

1. *neural-style*: carpeta donde se encuentran los códigos para la transferencia de textura o estilo, mediante el uso de la red neuronal indicada. Dichos códigos están implementados en *lua* y en *caffe*.

2. *Motion Estimation*: carpeta que contiene los códigos necesarios para llevar a cabo la estimación de movimiento, como ya se ha indicado anteriormente, el código original es de Stanley Chan [16], y el que se aporta es una adaptación que desarrollamos. Todos ellos están implementados en *Matlab*.

3. *analisis_frecuencial*: código de *Matlab* implementado para el estudio de las componentes frecuenciales de las imágenes estilo o textura utilizadas.

4. *correlations_calc*: código de *Matlab* implementado para la estimación de la correlación que se explica en el apartado *Análisis estadísticos de imagen* en la sección *Análisis de las distribuciones frecuenciales de las imágenes estilo. Uso de correlación como método de análisis de similitud*.

5. *Split_and_Merge_With_Blend*: código de *Matlab* implementado para generar el procesado Split and Merge y el Blending, ambos explicados en los apartados *Split and Merge* y *Blending*.

6. *extraer_frames*: código de *Matlab* implementado para extraer los frames de los vídeos originales que después serán procesados en la red neuronal para transferirles la textura o estilo en cuestión.

7. *introducir_frames*: código de *Matlab* implementado para, una vez han sido procesados y estilizados los frames originales, volver a unirlos y generar así los vídeos estilizados.

8. *S_and_M_W_B_Video_adapted*: código de *Matlab* adaptación del Split and Merge del punto (5) para múltiples frames, que es aplicado en vídeo.

ENLACE

https://drive.google.com/drive/folders/0B4LQLStIJSMxakM3YWhrUEtjejA

Bibliografía

[1] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. A neural algorithm of artistic style. *arXiv preprint arXiv:1508.06576*, 2015.

[2] Andrej Karpathy. Convolutional neural networks for visual recognition. *Unversity Lecture*, 2016.

[3] Justin Johnson. neural-style. `https://github.com/jcjohnson/neural-style`, 2015.

[4] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[5] Rujie Yin. Content aware neural style transfer. *arXiv preprint arXiv:1601.04568*, 2016.

[6] Oriel Frigo, Neus Sabater, Julie Delon, and Pierre Hellier. Split and match: Example-based adaptive patch sampling for unsupervised style transfer. 2016.

[7] Leon Gatys, Alexander S Ecker, and Matthias Bethge. Texture synthesis using convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 262–270, 2015.

[8] Dmitry Ulyanov, Vadim Lebedev, Andrea Vedaldi, and Victor Lempitsky. Texture networks: Feed-forward synthesis of textures and stylized images. *arXiv preprint arXiv:1603.03417*, 2016.

[9] Alexei A Efros and William T Freeman. Image quilting for texture synthesis and transfer. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 341–346. ACM, 2001.

[10] HARDIK MODI. Multi order minimum error boundary cut patch based algorithm: Innovative image inpainting application. *image*, 4:26.

[11] Javier Portilla and Eero P Simoncelli. A parametric texture model based on joint statistics of complex wavelet coefficients. *International journal of computer vision*, 40(1):49–70, 2000.

[12] David J Heeger and James R Bergen. Pyramid-based texture analysis/synthesis. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 229–238. ACM, 1995.

[13] Manuel Ruder, Alexey Dosovitskiy, and Thomas Brox. Artistic style transfer for videos. In *German Conference on Pattern Recognition*, pages 26–36. Springer, 2016.

[14] Claudio Javier Tablada and Germán Ariel Torres. Redes neuronales artificiales. *Revista de Educación Matemática*, 24(3), 2009.

[15] José Manuel Gutiérrez. Introducción a las redes neuronales. *Canabria, España*, 2006.

[16] Stanley H Chan, Truong Q Nguyen, et al. Subpixel motion estimation without interpolation. In *2010 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 722–725. IEEE, 2010.

[17] Video Processing. Communications, yao wang, jorn ostermann, ya-qin zhang, 2002.

[18] T Koga. Motion-compensated interframe coding for video conferencing. In *proc. NTC 81*, pages C9–6, 1981.

[19] Byeong-Doo Choi, Jong-Woo Han, Chang-Su Kim, and Sung-Jea Ko. Motion-compensated frame interpolation using bilateral motion estimation and adaptive overlapped block motion compensation. *IEEE Transactions on Circuits and Systems for Video Technology*, 17(4):407–416, 2007.