



**UNIVERSIDAD
DE GRANADA**

Sistemas Inteligentes para la Gestión en la
Empresa

Transferencia de estilo en imágenes

Máster en Ingeniería Informática

Curso 2019-2020

Fernando Roldán Zafra

Índice de contenidos

Introducción	3
Marco teórico	4
Aprendizaje automático	4
Aprendizaje supervisado	4
Aprendizaje no supervisado	4
Aprendizaje por refuerzo	5
Visión por computador	5
Redes neuronales convolucionales	5
Transferencia de estilo en imágenes	6
Función de pérdida	8
Representación del contenido	8
Representación del estilo	9
Alternativas	10
Ejemplo práctico	10
Recursos utilizados	10
Fast neural style transfer	12
Modelo a partir de VGG19	13
Conclusión	14
Bibliografía	15

Introducción

Hoy en día, las fotografías o imágenes han pasado de ser un objeto físico a un objeto virtual. Esto se debe a que prácticamente todo el mundo puede realizar una fotografía con su teléfono móvil, objeto indispensable para cualquier persona. Esto, unido al hecho del reducido coste de los equipos en cuanto a potencia computacional, ha propiciado el desarrollo de diversas aplicaciones o programas los cuales permiten modificar las imágenes de diferente forma.

Estas aplicaciones abundan en las redes sociales, ejemplo de ellos son los filtros de [“Instagram”](#). Estos filtros nos permiten aplicar un color sepia a una imagen, convertirla en blanco y negro, etc. Sin embargo, no es solo este tipo de filtros o modificaciones lo que encontramos en las redes sociales y es que con el desarrollo del Deep Learning, la inteligencia artificial y la evolución de la tecnología ha surgido lo que se conoce como *transferencia de estilo en imágenes*. Esto se debe a que debido al descenso del precio por poder de cómputo los desarrolladores pueden realizar operaciones y algoritmos más complejos.

Esta técnica de la transferencia de estilo entre imágenes a grandes rasgos consiste en dadas dos imágenes A y B, siendo A la imagen de entrada y B la imagen de estilo, transferir el estilo de B a A. Esta técnica se conoce también como transferencia de estilo neuronal o NST por sus siglas en inglés [1]. Esto es algo que lleva investigándose desde comienzos de siglo y es que debido al gran desarrollo que tuvieron las redes neuronales y el gran avance que han experimentado, se han pasado por varias fases o etapas siendo la más reciente el uso de redes neuronales convolucionadas para el desempeño de esta tarea. Esto se debe a que las redes neuronales convolucionadas funcionan especialmente bien para problemas de visión por computador.

Refiriéndonos a nuestro problema o al tema a tratar hay que destacar que este tipo de algoritmos hacen uso de técnicas tanto de Deep learning o aprendizaje profundo (como son las redes neuronales) como de técnicas de visión por computador. Tradicionalmente estas dos ramas de la informática han evolucionado de forma paralela, sin involucrarse demasiado la una en la otra pero debido al auge del deep learning, esto ha cambiado y ahora se utilizan ambas ramas para resolver problemas como el que trataremos en el presente trabajo.

Marco teórico

Como se ha comentado, la transferencia de estilo en imágenes consiste en un método por el cual mediante aprendizaje automático se transfiere el estilo de una imagen a otra. Pero ¿qué es el aprendizaje automático? El aprendizaje automático o *machine learning* es un campo de la inteligencia artificial en el cual se pretende que las máquinas consigan la capacidad de aprender a hacer una tarea sin haber sido programadas explícitamente para ello. Esto es algo tremendamente útil y es que hacemos que una máquina sea capaz de generalizar o de aprender a resolver problemas que hasta hace no mucho tiempo se creía que solo eran resolubles por humanos. En esta sección se hablarán de

dichos algoritmos, de la visión por computador y de las redes neuronales convolucionadas.

Aprendizaje automático

El proceso de aprendizaje consiste en que el sistema indague en los datos de entrenamiento para encontrar patrones y que una vez encontrados, en lugar de exponerlos para que sean comprensibles para los seres humanos, modifique su comportamiento para adaptarse a esos patrones [2]. Los diferentes algoritmos de aprendizaje automático se suelen dividir en tres clasificaciones. A continuación se detallan dichas clasificaciones [3].

Aprendizaje supervisado

Esta rama del aprendizaje automático se encarga de los problemas que traen los datos con etiquetas. Este tipo de algoritmos se encargarán de dadas unas variables de entrada asignarles una etiqueta de salida adecuada. Estos algoritmos se denominan "supervisados", ya que trabajan con un conjunto de datos etiquetados sobre los que aprenden a asignar etiquetas a datos de entrada sin etiquetas.

Un ejemplo de esto lo encontraríamos en los servicios de correo electrónico, y es que este tipo de algoritmos son capaces de diferenciar y etiquetar los correos que se reciben en "Spam" o "no Spam".

Aprendizaje no supervisado

Como caso contrario al modelo anterior, tenemos el aprendizaje no supervisado. En este caso, este tipo de algoritmos surgen cuando no se disponen de datos "etiquetados" para el entrenamiento, sino que solo conocemos los datos de entrada, sin disponer de ninguna salida asociada a dichos datos de entrada. Este tipo de algoritmos tienen un carácter más exploratorio y es que pretender ofrecer una serie de organización o de información que nos permita sacar conclusiones de los datos. Sin embargo, no se garantiza que los resultados que se obtengan ofrezcan información útil o fácilmente interpretable.

Un ejemplo de ello lo encontramos en los algoritmos de clustering, los cuales nos pueden proporcionar una serie de agrupamientos de los datos pero nada nos garantiza que estos agrupamientos tengan algún significado o utilidad.

Aprendizaje por refuerzo

Parece obvio que las dos clasificaciones abarcan todas las opciones pero sin embargo podemos destacar otro tipo de algoritmos. Este tipo de aprendizaje se basa en mejorar la respuesta del modelo usando un proceso de retroalimentación. Esto quiere decir que el modelo aprende en función a sus salidas y es que los datos de entrada serán las respuestas a sus salidas, podríamos decir que se trata de aprender por prueba y error.

Por lo tanto, como podemos ver no se trata ni de un modelo de aprendizaje supervisado por que no se basa estrictamente en un conjunto de datos etiquetados sino que monitoriza la respuesta a las salidas ofrecidas. Tampoco se trata de un modelo de aprendizaje no supervisado por que cuando lo modelamos sabemos de antemano que salida esperamos.

Visión por computador

En este caso ya no hablamos de aprendizaje automático sino de otra rama diferente, la visión por computador. Esta disciplina consiste en el estudio de como procesar, analizar e interpretar imágenes de forma automática [4]. Esta rama está muy ligada al aprendizaje automático y es que una de las aplicaciones más utilizadas en este campo es el etiquetado automático de imágenes. En este tipo de aplicaciones se pretende que dada una imagen el sistema sea capaz de ofrecer como salida una etiqueta de salida que describa lo que hay en la imagen. Otra aplicación será la de diferenciar los diferentes objetos presentes en la imagen y ser capaz de interpretarlos de forma separada. Como se puede ver la relación entre este tipo de sistemas y el aprendizaje supervisado es clara.

Redes neuronales convolucionales

Otro aspecto clave de los algoritmos de transferencia de estilo se encuentra en las redes neuronales convolucionales [5]. Este tipo de redes neuronales consisten en un algoritmo de deep learning que está diseñado para trabajar con imágenes y tomando estas como entrada consiguen diferenciar ciertos elementos en la imagen. Estas contienen varias capas ocultas, especializándose algunas de ellas en reconocer diferentes tipos de formas o atributos, ya sean líneas, curvas, profundidad, etc.

Sin entrar mucho en profundidad en la infraestructura de las redes neuronales por no ser el objetivo de este trabajo, estas imágenes toman como entrada los píxeles de la imagen de forma que la capa de entrada tendrá un tamaño fijo en función al tamaño de la imagen que queramos utilizar. Si la imagen es de 28x28 en escala de grises tendremos 284 neuronas, mientras que si fuese una imagen a color tendríamos 28x28x3 neuronas, es decir, 2352.

Una vez que nos adentramos en la red nos encontramos las diferentes kernel o filtros que aplicaremos a la imagen. Estos filtros nos permitirán extraer características importantes de la misma. Un ejemplo de ello se puede encontrar en la figura 1 y 2



Figura 1: Imagen a la que se aplicará un filtro o kernel [5]

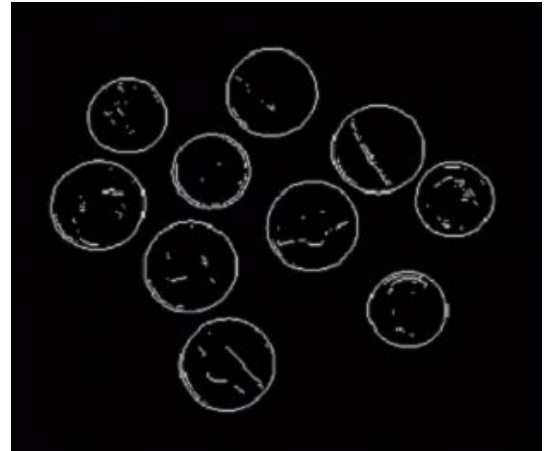


Figura 2: Resultado de aplicar el filtro o kernel [5]

Como se puede ver en las Figuras anteriores, aplicando diferentes kernel podemos obtener diferentes características. Sin embargo, por si solo no conseguimos nada solo con esto, tendremos que utilizar convoluciones para que nuestra red neuronal consiga su objetivo. La convolución es el proceso característico de esta arquitectura y consiste en coger un conjunto de píxeles de la imagen de entrada e ir realizando un conjunto de operaciones matemáticas sobre dicho conjunto de píxeles. De esta forma se obtiene un conjunto simplificado de píxeles que estará formado por el resultado de dichas operaciones sobre todos los conjuntos de píxeles de la imagen.

A grandes rasgos se podría explicar el funcionamiento de este tipo de redes de esta forma sin embargo estas son mucho más complejas y tienen una serie de operaciones que no aparecen descritas aquí. Esto se debe a las limitaciones de tamaño del trabajo y a que no es el objetivo del mismo. Pasamos por lo tanto a continuación a hablar sobre la transferencia de estilo en imágenes, y como utilizando este tipo de redes se consigue transmitir el estilo de una imagen a otra.

Transferencia de estilo en imágenes

Como ya se ha comentado anteriormente, las aplicaciones que modifican una imagen y le aplican algún tipo de filtro o transformación abundan en la red. Ejemplos de estas transformaciones se pueden encontrar en GoArt, Prima o Vinci. Sin embargo algunas de estas transformaciones consisten en una transferencia de estilo, es decir en cambiar el estilo de una imagen al estilo de otra. A este tipo de transformaciones se le conoce como transferencia de estilo.

A grandes rasgos podríamos decir que este tipo de algoritmos toma 2 imágenes diferentes como entrada: una con el contenido y otra con el estilo. El objetivo será crear una nueva imagen que represente el contenido original en el estilo de la otra imagen. Este problema ha sido tratado de diversas formas, sin embargo una de las que ha tenido más éxito ha sido la que ha tratado el problema de forma que no es necesario diseñar una nueva arquitectura o una nueva técnica y es que como se ve en [6] basta

con rediseñar una arquitectura existente, más concretamente en usar una red neuronal convolucional pre entrenada.

Gatys y su equipo eligieron la red VGG19 [7], la cual fue entrenada para el reconocimiento y localización de objetos en imágenes. En la figura 3 se puede observar la arquitectura de su versión de 16 capas, VGG16 red neuronal.

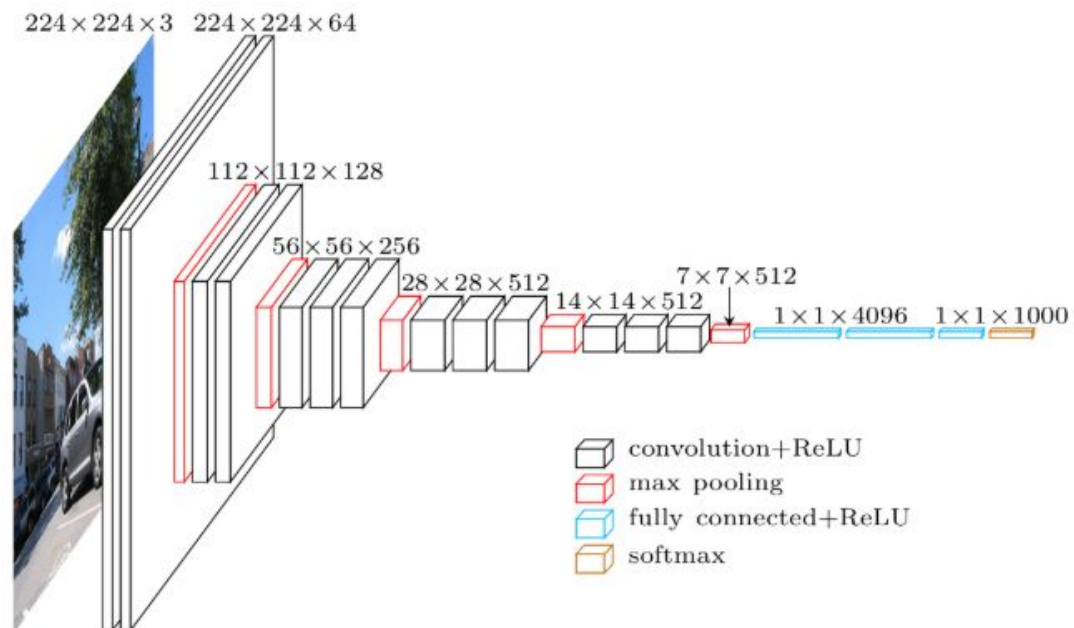


Figura 3: Arquitectura de la red Neuronal Convolucional VGG16 [8]

Como se comentó anteriormente, las redes neuronales convolucionales, intentan extraer características o atributos de los objetos que aparecen en las imágenes de forma que sean capaces de reconocer dichos patrones para poder clasificar las diferentes imágenes. Esto se realiza de forma que cada una de las capas se especializa en reconocer un atributo o característica concreta. Por ejemplo, es posible que en la primera capa de convolución se detecten algunas características simples, como líneas rectas o algún patrón simple. Por otro lado, en las capas más profundas se detectarán patrones más complejos como por ejemplo círculos como los de las ruedas de un coche.

Al proceso anterior de reconocer ciertos patrones se le conoce como aprendizaje de representaciones de características [8]. Esta representación o codificación de las diferentes características de una imagen es la clave para la transferencia de estilo y es que es usada para calcular la función de pérdida entre la imagen generada y las imágenes de contenido y estilo. Pero ¿cómo podemos estar seguros de que la imagen de contenido y la imagen generada son similares en cuanto a contenido y no en cuanto a estilo? Y de forma análoga ¿cómo podemos estar seguros de que la imagen de estilo y la imagen generada son similares en cuanto a estilo y no en cuanto a contenido?

Función de pérdida

Una forma de solventar el problema anterior consiste en dividir la función de pérdida en dos partes, una para la función de pérdida en cuanto a contenido y otra para la función de pérdida en cuanto a estilo. La expresión matemática de esto se puede ver a continuación [6].

$$\mathcal{L}_{total} = \alpha \mathcal{L}_{content} + \beta \mathcal{L}_{style}$$

Como se puede ver, la formula anterior además de lo comentado anteriormente muestra dos expresiones extra: Alpha y Beta. Estas dos expresiones representan los pesos que queremos asignarle a cada parte, es decir, cuanto queramos conservar del estilo con respecto al contenido o viceversa.

Sin embargo ¿Cuáles son los valores de entrada de estas funciones de pérdida? Dado que no sabemos cual será la salida no deberíamos de poder introducir nada, sin embargo esto no es así.

Primero y como se puede leer en [6] y en [8] introducimos en cada iteración la imagen de estilo, de contenido y la generada, que al principio será inicializada con una imagen con ruido blanco, es decir, con valores aleatorios para cada píxel y con la forma de la imagen de contenido. El valor de las activaciones de las neuronas de las capas ocultas representan las características de cada imagen y estos valores de las activaciones serán los que se utilicen como entrada para la función de pérdida descrita anteriormente. Por cada iteración, el modelo realizará cambios en la imagen generada para minimizar la función de pérdida, obteniendo al final el resultado deseado. A continuación describiremos como se calculan estas funciones de pérdida para el contenido y el estilo.

Representación del contenido

De forma general, cada capa de la red representa un “filtro” no lineal en el cual se recogen unas características concretas. La complejidad de estos filtros aumenta con respecto a la profundidad a la que se encuentre en la red. De esta forma obtenemos que por cada filtro de la red tenemos un mapa de características de una imagen en concreto y es que cada capa reacciona de manera diferente a cada imagen. De esta forma podemos almacenar las características de una imagen en una matriz que almacene las activaciones por cada neurona de cada capa.

Una vez tenemos almacenados todos estos datos, podemos aplicar descenso del gradiente a una imagen de forma que obtengamos una imagen que minimice las diferencias entre su mapa de características y la de la imagen inicial. De esta forma, si tenemos una imagen de contenido x y una imagen blanca p , siendo P^L y F^L sus representaciones de características en la capa L , podemos definir la diferencia en el error cuadrático en su función de pérdida como se puede ver a continuación.

$$\mathcal{L}_{\text{content}}(\vec{p}, \vec{x}, l) = \frac{1}{2} \sum_{i,j} (F_{ij}^l - P_{ij}^l)^2 .$$

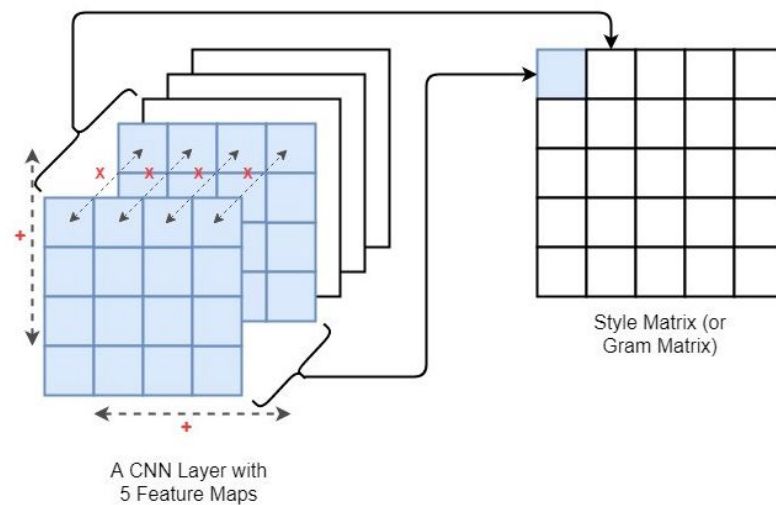
A partir de esta expresión podremos calcular mediante técnicas de backpropagation la imagen generada que obtenga la misma respuesta en una determinada capa de la red que la imagen de contenido inicial. De esta podremos asemejar las matrices de características de ambas imágenes.

Por otro lado, cuando una red está expresamente entrenada para reconocer objetos en ella, desarrolla representaciones de las imágenes que estarán relacionadas con los objetos que en ellas aparecen. Estas representaciones están cada vez más explícitamente relacionadas con los objetos cuanto más profunda es la capa en la red. De forma que una imagen que se introduzca en una red se verá transformada en representaciones que serán cada vez más sensitivas al contenido de la imagen y que serán, por contra, menos sensitivas a la apariencia de la imagen en si. Por lo tanto, podemos decir que las capas más profundas de la red son las capas que representan las características del contenido de la imagen.

Representación del estilo

En este caso, de forma similar a como se realizó anteriormente para representar el contenido de las imágenes, utilizaremos para representar el estilo las matrices de características de varias capas del modelo. Sin embargo, en este caso lo realizaremos computando la matriz de Gram. Esta matriz no será otra cosa que una matriz de estilo que calcularemos para la imagen generada y para la imagen de estilo. Una vez calculadas calcularemos la diferencia entre una y otra tal y como se hizo en el apartado anterior.

Esto se realiza, ya que en este caso tenemos por cada capa de la red varios canales, estos canales harán referencia a diferentes mapas de características y mediante una matriz de Gram o matriz de estilo conseguimos unificar todos estos mapas. A continuación se muestra una imagen que ejemplifica lo que acabamos de comentar [9].



En esta imagen se puede ver una capa de la red neuronal con sus diferentes mapas de características a la izquierda y la matriz de estilo o de Gram a la derecha. Dicha matriz se calcula de forma que el elemento (i,j) representa la multiplicación *element-wise* de los mapas de características i y j sumando los elementos a lo largo y a lo ancho del mapa de características. En la imagen las cruces rojas, denotan multiplicación de tipo *element-wise* y los símbolos “+” denotan suma.

Una vez calculada dicha matriz bastará computar la pérdida de estilo de la misma forma que se computó la pérdida de contenido.

Alternativas

Como se comentó anteriormente, no hay una única aproximación a este problema y aunque el modelo presentado por Gatys et al. [6] ofrezca unos resultados impresionantes presenta un problema: tiene un coste computacional elevado. Esto se debe al lento proceso iterativo de optimización de la función de pérdida [11].

Una alternativa a esto viene de la mano de Ulyanov et al. [12] el cual consiguió aumentar la velocidad de este proceso a través del entrenamiento de una red en la cual con una sola interacción se conseguía estilizar una imagen. A la solución que desarrollaron se la conoce como ***Fast neural style transfer***. Cabe destacar que esta solución no ofrece los resultados con tanta calidad como la propuesta por Gatys et al. y es que, esta solución está limitada a los estilos con los cuales ha sido entrenada, sin embargo, los resultados no son nada despreciables en algunos casos.

Ejemplo práctico

Para finalizar este trabajo, vamos a hacer un ejemplo del funcionamiento de estas redes, para ello seleccionaremos varias imágenes, algunas de contenido y otras de estilo e intentaremos combinarlas como se ha descrito en este trabajo.

Recursos utilizados

Primero decir que el código que se ha utilizado para este ejemplo es uno de los ejemplos que ofrece tensorflow para este tipo de algoritmos y puede encontrarse en [10]. Este código ha sido importado a google colab y modificado para incorporar las imágenes que se presentan a continuación. Para las imágenes, se ha utilizado como fotos de contenido una imagen de mi mismo y otra imagen descargada de una ciudad. En la parte del estilo se han utilizado varias obras de arte para comparar los resultados. A continuación se presentan las diferentes imágenes que se han utilizado.



Imagen de contenido



Segunda imagen de contenido



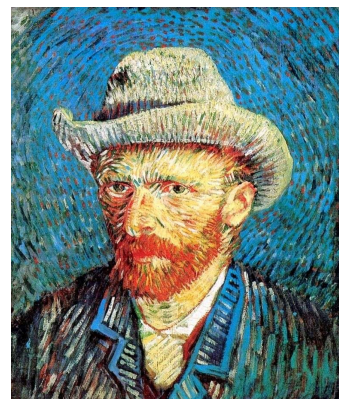
La Gioconda, Leonardo da vinci



Pintura abstracta al óleo,
Andreas Hoffman







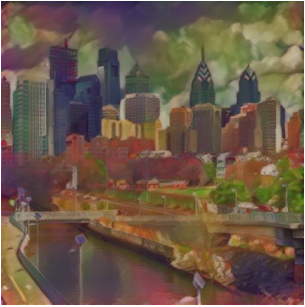
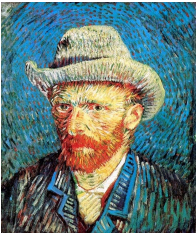

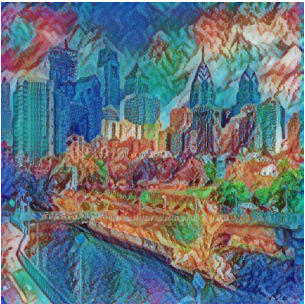


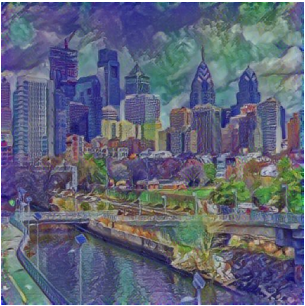
Noche estrellada, Vincent Van Gogh

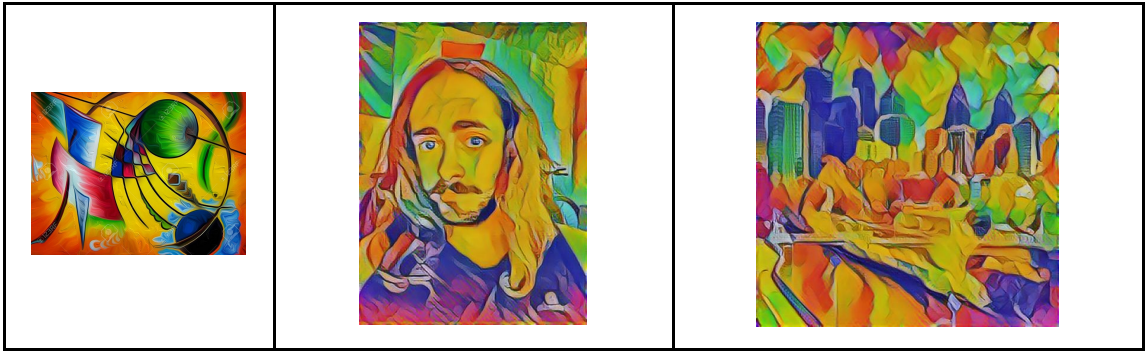


Autoretrato, Vincent Van Gogh

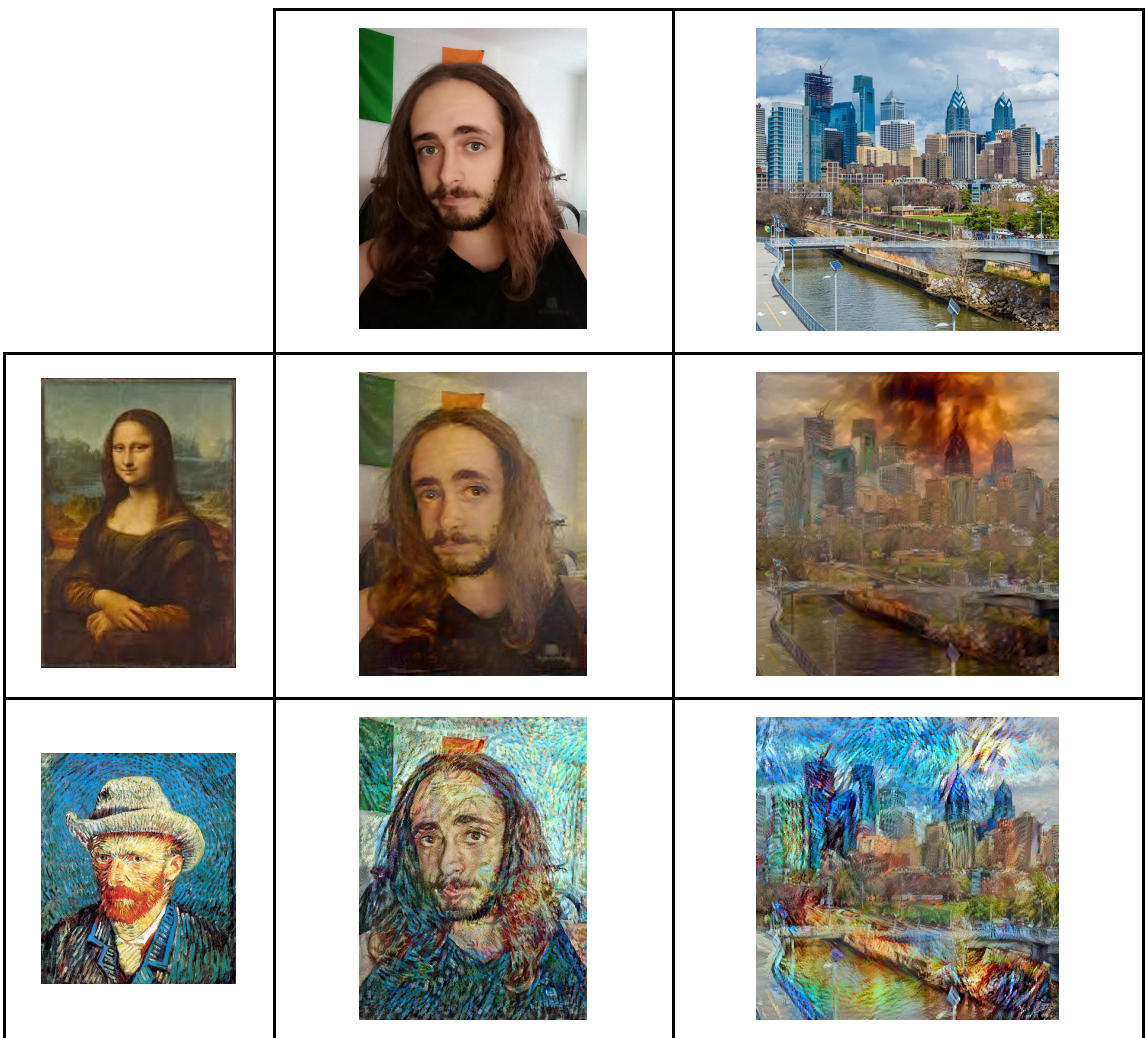
En el proceso, se han obtenido varias imágenes y es que el código utilizado incorporaba además de una implementación similar a la descrita por Gatys et al. [6], una implementación del modelo **Fast Neural style transfer**. A continuación se muestran algunos de los resultados obtenidos por ambas soluciones.

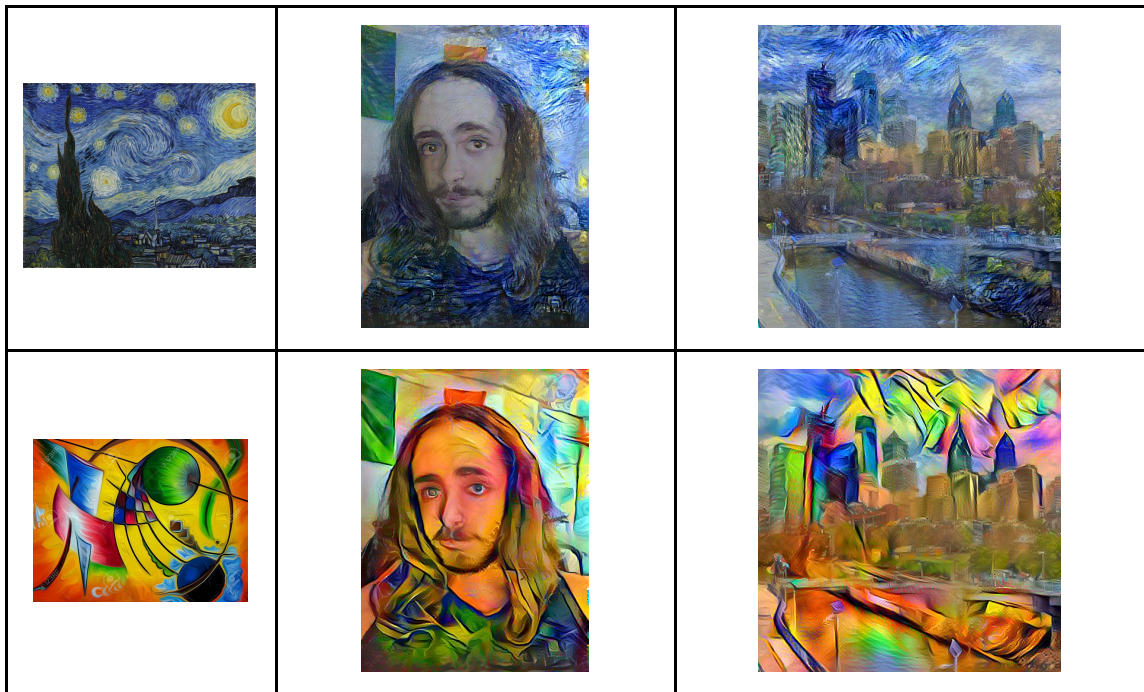
Fast neural style transfer



Modelo a partir de VGG19





Conclusión

Tras la realización del presente trabajo se han alcanzado varias conclusiones. Estas pueden verse en la tabla de resultados del apartado anterior y es que como conclusión podemos comentar que en este tipo de algoritmos, dadas dos imágenes, una de estilo y otro contenido, nunca se conoce el resultado que estas van a ofrecer. Es de esperar, que la imagen generada nos evoque más o menos a la imagen de estilo en función al peso que le demos a esta característica pero nunca podremos predecir cual será la salida.

Por otra parte hemos visto que dependiendo de las arquitecturas que sigamos se obtienen resultados muy diferentes, en los resultados hemos visto que el modelo que utiliza la red VGG19 obtiene unos resultados bastante buenos en la mayor parte de los casos aunque tarda bastante más. Por otro lado el algoritmo Fast neural style transfer tarda bastante menos pero a costa de unos resultados que podrían considerarse peores.

En cualquier caso, la elección del modelo a utilizar dependerá de las necesidades y características del sistema que estemos implementado, no pudiendo aun así estimar si la salida se corresponderá con lo que esperamos obtener.

Bibliografía

- [1] Cornell University, Yongcheng Jing, Yezhou Yang, Zunlei Feng, Jingwen Ye, Yizhou Yu, Mingli Song, Neural Style Transfer: A review, 2018 <https://arxiv.org/abs/1705.04058>
- [2] SearchDataCenter, Margaret Rouse, Aprendizaje automático (machine learning), 2017,
<https://searchdatacenter.techtarget.com/es/definicion/Aprendizaje-automatico-machine-learning>
- [3] Blog ThinkBig, Tipos de aprendizaje en Machine Learning: supervisado y no supervisado, 2017,
<https://empresas.blogthinkbig.com/que-algoritmo-elegir-en-ml-aprendizaje/>
- [4] Universitat Oberta de Catalunya, La visión por computador: Una disciplina en auge, 2012,
<http://informatica.blogs.uoc.edu/2012/04/19/la-vision-por-computador-una-disciplina-en-auge/>
- [5] Medium, Introducción a las redes neuronales convolucionales, 2019,
<https://medium.com/@bootcampai/redes-neuronales-convolucionales-5e0ce960caf8>
- [6] University of Tübingen, Leon A. Gatys, Alexender S. Ecker, Matthias Bethge, Image style transfer using Convolutional Neural Networks,
https://www.cv-foundation.org/openaccess/content_cvpr_2016/papers/Gatys_Image_Style_Transfer_CVPR_2016_paper.pdf
- [7] Towards Data science, Jerry Wei, VGG neural networks: The next step after AlexNet, 2019,
<https://towardsdatascience.com/vgg-neural-networks-the-next-step-after-alexnet-3f91fa9ffe2c>
- [8] Towards Data science, Vamshik Shetty, Neural style Transfer Tutorial, 2018,
<https://towardsdatascience.com/neural-style-transfer-tutorial-part-1-f5cd3315fa7f>
- [9] Towards Data Science, Thushan Ganegedara, intuitive guide to neural style transfer, 2019,
<https://towardsdatascience.com/light-on-math-machine-learning-intuitive-guide-to-neural-style-transfer-ef88e46697ee>
- [10] Tensorflow, Neural style transfer,
https://www.tensorflow.org/tutorials/generative/style_transfer
- [11] Towards Data Science, Connor Shorten, Towards Fast Neural Style Transfer, 2019,
<https://towardsdatascience.com/towards-fast-neural-style-transfer-191012b86284>

[12] Dmitry Ulyanov, Vadim Lebedev, Andrea Vedaldi, Victor Lempitsky, Texture Networks: Feed-forward Synthesis of Textures and Stylized Images, 2016, <https://arxiv.org/abs/1603.03417>