Universidad Nacional de Córdoba

Facultad de Matemática, Astronomía, Física y Computación

Transferencia de Estilo en Fotografias mediante Redes Neuronales Convolucionales

Trabajo Final de Licenciatura en Ciencias de la Computación

Autor: Wolfmann, Ariel Mauricio
Director: Doctor Sanchez, Jorge

Julio de 2017

Universidad Nacional de Córdoba

Facultad de Matemática, Astronomía, Física y Computación

Resumen

Dada la facilidad de tomar fotografias a partir de los dispositivos móviles y el aumento de la capacidad de cómputo de dichos dispositivos se han desarrollado muchas aplicaciones con el objetivo de generar nuevas imágenes a partir de las fotografías. Empleando técnicas de aprendizaje automático es posible aplicar metodós de transferencia de estilo a las fotografías, obteniendo imágenes artísticas. La idea básica consiste en seleccionar una obra de arte y una fotografía a la cual se desea aplicar el estilo de la obra. A partir de estas imágenes, se obtiene una nueva imágen compuesta por el contenido de la fotografía y el estilo de la obra. Para lograr dicha transferencia de estilo es necesario aplicar conocimientos relacionados con Visión por Computadoras y Aprendizaje Profundo.

Abstract

Nowadays is really easy to take pictures from smartphones, which have increased notoriously its computing performance, so there are many pictures oriented applications. Using Machine Learning techniques, is possible to apply style transfer methods to the pictures getting new artistic images. The idea is based on selecting a master piece of art and a picture to apply the master piece style. Using both images, you can get a new image composed by the content of the picture and the style of the master piece. To achieve that style transfer, it is necessary to apply some concepts and techniques from Computer Vision and Deep Learning.

Índice general

Capítulo 1

Introducción

1.1. Contexto

En los últimos años las fotografias han pasado a ser un objeto digital en lugar de físico, de la mano del gran aumento en el uso de los dispositivos móviles. Cualquier persona puede tomar miles de fotografías en un instante de tiempo, y compartirlas en las redes sociales, desde su propio dispositivo. A partir de esto, se han desarrollado muchas aplicaciones entorno a las fotografías; ya sea desde redes sociales másivamente utilizadas hasta aplicaciones que utilizando efectos o filtros, transforman la fotografía en un retrato en blanco y negro o sepia, por nombrar algunos.

Recientemente, se comenzó a desarrollar aplicaciones que logran transferir el estilo de una obra de arte a una fotografía. El proceso de transferencia consiste en detectar el estilo de la obra de arte y el contenido visual de la fotografía, generando una nueva imagen que combina ambas características. En la Figura 1.2 se exhibe un ejemplo de dicho proceso. La ejecución de procedimientos como este, son posibles gracias al incremento del poder de cómputo y al descenso del precio de los nuevos dispositivos. Al disponer de dispositivos de bajo costo con gran potencia de cómputo, las aplicaciones pueden emplear técnicas computacionales de mayor complejidad.

La transferencia de estilo requiere de la interacción de dos de las principales áreas de las ciencias de la computación, de gran auge en la actualidad: visión por computadoras y aprendizaje automático, más específicamente aprendizaje profundo (estas tecnologías aparecen como *Computer Vision*, *Machine Learning* y *Deep Learning* en la literatura en inglés). Inicialmente estas áreas evolucionaban en paralelo independientemente una de la otra; hoy en día se han logrado resultados completamente disruptivos, al aplicar aprendizaje profundo para algunas de las principales tareas de visión por computadoras. Clasificación de imágenes, detección y reconocimiento de objetos, son solo algunas de las tareas en las que se emplean estas técnicas.

Un ejemplo de amplia investigación y desarrollo en la actualidad, es el de los vehículos autónomos, los cuales emplean estas técnicas de aprendizaje profundo dentro de sus procesos para la detección de objetos en tiempo real. En la Figura 1.1 se puede observar gráficamente cómo se integran.

El aprendizaje automático es un área de las ciencias de la computación que explora el estudio y la construcción de algoritmos que pueden aprender y hacer predicciones sobre los datos de entrada. Tales algoritmos logran aprender a inferir características que poseen los datos, a través de la construcción de un modelo de predicción, en la etapa de entrenamiento del modelo. Cuando finaliza esta etapa, el modelo debe ser capaz de predecir características de una muestra nueva, nunca antes vista, en base a lo aprendido en el entrenamiento.

El aprendizaje automático esta ideado para ser empleado en una serie de tareas informáticas

Figura 1.1: Vehículo conducido automáticamente, detectando objetos en tiempo real mediante redes neuronales convolucionales.

en las que el diseño y la programación de algoritmos explícitos son inviables, como por ejemplo para la detección de spam, el reconocimiento óptico de caracteres y los motores de búsqueda [1].

El principal objetivo de un sistema de aprendizaje es generalizar desde su experiencia o conocimiento previo. El proceso de aprendizaje consiste observar ejemplos de un conjunto de entrenamiento y construir un modelo que le permita realizar predicciones lo suficientemente precisas sobre nuevos ejemplos. El modelo aprende una función de predicción F paramétrica, a partir del conjunto de instancias de entrenamiento, cada una de ellas, representada por un vector de R^n de características. Durante el entrenamiento, F *aprende a predecir*, lo que equivale a ajustar sus parámetros a partir de los vectores que representan a los datos. Generalizar en este contexto es la habilidad del sistema de realizar predicciones con precisión sobre un ejemplo nuevo no visto antes, luego de aprender sobre el conjunto de entrenamiento.

Dentro del aprendizaje automático, existe un subdominio llamado aprendizaje profundo, que ataca problemas complejos en los cuales representar una instancia no es una tarea sencilla. Por ejemplo para representar una imagen se suelen utilizar sus píxeles o pequeñas subregiones de la imagen. Sin embargo, esta representación no logra capturar características de alto nivel, como es el caso de detectar si existe o no un objeto dada una subregion de píxeles de la imagen.

Dado a que en este tipo de problemas no hay representación intuitiva clara de los datos, los algoritmos de aprendizaje profundo aprenden no solo a realizar predicciones, sino que también aprenden a representar los datos de entrada. Esta representación captura características de alto nivel, de forma tal que la predicción en base a estas representaciones sea lo más precisa posible [2].

Para el caso de los problemas relacionados con imágenes, la técnica de aprendizaje profundo mayormente utilizada es la de Redes Neuronales Convolucionales (CNN por su denominación en inglés, Convolutional Neural Networks). Estas redes, son un tipo específico de Redes Neuronales Artificiales, que asumen explícitamente que los datos de entrada serán imágenes, lo cual permite implementar ciertas optimizaciones dentro del algoritmo. Todos estos conceptos serán posteriormente explicados en detalle a lo largo del siguiente capítulo.

El área de visión por computadoras es el área de las ciencias de la computación encargado de lograr que las computadoras puedan generar una comprensión de alto nivel de imágenes digitales o vídeos. Desde la perspectiva de la ingeniería, busca automatizar tareas que el sistema visual

humano puede hacer. Para lograr este nivel de comprensión, en primer lugar emplea métodos de bajo nivel relacionados con el procesamiento de imágenes para adquirir, procesar, analizar y comprender las imágenes del mundo real. Este procesamiento se realiza con la finalidad de producir información numérica o simbólica para que pueda ser tratada por una computadora. Generalmente la salida de estos métodos suele ser una imagen preprocesada o un conjunto de parámetros calculados que sirven como entrada para que los métodos de visión por computadoras obtengan información de alto nivel a partir de estos. El objetivo principal de esta área es reducir la distancia entre lo que un humano interpreta a partir de una imagen y la forma en la que las computadoras representan esa misma imagen [3].

La transferencia de estilo en fotografias es considerado un problema complejo dentro del área de visión por computadoras. Para introducir al lector en el tema en la Figura 1.2 se puede observar un ejemplo del resultado de aplicar la transferencia de estilo: como imagen de estilo se eligió una imagen abstracta (1.2a) y como imagen de contenido una fotografía de un paisaje (1.2b), obteniendo como resultado de la transferencia de estilo una imagen en la cual se mantienen las estructuras de la imagen de contenido pero con los colores y trazos de la imagen de estilo (1.2c).

1.2. Motivación

Una imagen, digitalmente, es representada por un arreglo de 2 dimensiones donde cada valor representa la intensidad captada por un sensor de un determinado punto espacial, representados como píxeles. La representación de los píxeles es muy sensible a cambios en la iluminación, ángulo, contraste y tamaño que pueda existir.

En cuanto a la transferencia de estilos artísticos en fotografías mediante aprendizaje profundo, existe una publicación *fundacional* [4] en la cual se define un procedimiento para lograrlo. En este proceso, al efectuar el aprendizaje por parte del modelo, se emplean técnicas estocásticas para lograr el objetivo deseado. Ademas, dicho algoritmo requiere una gran cantidad de hiper-parámetros predefinidos empíricamente, es decir parámetros que deben ser ajustados previo a la ejecución del algoritmo, que influyen en gran manera sobre el resultado.

El principal objetivo de este trabajo es poder realizar una optimización de los hiper-parámetros automáticamente. Principalmente el número de iteraciones, el cual determina el tiempo en que se debe ejecutar el algoritmo y la calidad del resultado obtenido. La idea de optimizar este hiper parámetro es obtener un resultado deseable, en el menor tiempo posible.

Determinar la aceptabilidad de un resultado requiere definir una métrica cuantitativa que evalue la calidad del resultado. Sin embargo, las obras de arte suelen ser calificadas con valoraciones cualitativas. Por lo tanto es necesario transformar esta valoración de carácter cualitativo a una métrica cuantitativa. La métrica cuantitativa que se decidió utilizar en este trabajo es el nivel de pertenencia que tiene el resultado con respecto al estilo que se le transfirió. Para medir esta métrica se entreno una red neuronal convolucional que reconoce estilos artísticos y el nivel de pertenencia esta dado por el nivel de confianza que obtiene la imagen resultante al ser evaluada en esta red. En base al puntaje obtenido se determina si el número de iteraciones es suficiente o si es necesario seguir ejecutando el algoritmo.

1.3. Estructura del trabajo

A lo largo de este trabajo se hará un recorrido por los principales conceptos para comprender tanto el problema como la solución y las técnicas empleadas para la transferencia de estilos artísticos en fotografías.

El capítulo 2 desarrolla el marco teórico y cuestiones formales requeridas, principalmente orientado al aprendizaje automático y a las redes neuronales convolucionales. En el capítulo 3 se hace un recorrido por los principales artículos de investigación y los algoritmos empleados en las técnicas de transferencia estilos artísticos en fotografías. Además se presenta la solución propuesta. En el capítulo 4 contiene los experimentos realizados y los resultados obtenidos, junto con un análisis y evaluación empírica de los mismos. Finalmente en el capitulo 5 se presentan las conclusiones obtenidas acerca del trabajo realizado, junto con las perspectivas y posibles tareas a futuro.

(a) Imagen de Estilo

(b) Imagen de Contenido

(c) Resultado obtenido transfiriendo el estilo de la obra de arte a la imagen de contenido

Figura 1.2: Ejemplo de transferencia de estilo

Capítulo 2

Marco teórico

2.1. Aprendizaje Automático

2.1.1. Introducción

El aprendizaje automático (*Machine Learning*, por su denominación en inglés) es un campo que se encuentra en la intersección de las ciencias de la computación y el aprendizaje estadístico. Tiene por objetivo dotar a las computadoras con la habilidad de aprender o inferir reglas que no fueron explícitamente programadas, a partir de los datos que le sean provistos.

Tom M. Mitchell [5] elaboró una definición más formal de este concepto de aprendizaje: "se dice que un programa de computadora aprende de una experiencia E con respecto a una clase de tarea T y medición de desempeño P, si su desempeño en la tarea T, medido por P, mejora con la experiencia E". Por ejemplo, en el problema de clasificación, una posible experiencia E podría ser observar muchos ejemplos de elementos asociados con su respectiva etiqueta. La tarea T sería clasificar a los elementos por etiquetas y P el grado de precisión con la que el programa clasifica a los elementos con su etiqueta correctamente. De esta forma, el programa aprende a clasificar elementos por etiqueta si el grado de precisión mejora a partir de observar ejemplos previamente.

2.1.2. Clasificación de algoritmos de aprendizaje automático

Dentro del aprendizaje automático, los algoritmos se pueden clasificar en algoritmos de aprendizaje supervisados, no supervisados o semi supervisados, según el grado de control humano que requieren de los datos provistos para lograr aprender a inferir patrones. En todos los casos, los algoritmos requieren de una representación de los datos, que codifique la información correspondiente. Estas representaciones serán discutidas en la sección 2.1.3. A continuación se detallan cada uno de los tipos de algoritmos.

Aprendizaje supervisado

Los algoritmos de aprendizaje supervisado emplean como conjunto de entrenamiento ejemplos que consisten de entradas x y etiquetas y, es decir aprenden una función de predicción, a partir de datos de entrenamiento etiquetados. Cada ejemplo del conjunto de entrenamiento suele ser un par compuesto de un objeto de entrenamiento y una etiqueta, el cual es el valor de salida deseado. A partir de los objetos y etiquetas con las que se entrenó, aprende a predecir la etiqueta de un nuevo objeto nunca antes visto. Un ejemplo de esta clase de algoritmos es el de los clasificadores lineales.

Los clasificadores lineales, son un tipo de clasificador que dividen el espacio en regiones a través de hiperplanos. Estos clasificadores se pueden utilizar para asociar objetos con su respectiva etiqueta, como se ilustra en la Figura 2.1. Como se menciona anteriormente, esta división en distintas regiones se realiza en base a los datos observados durante el aprendizaje.

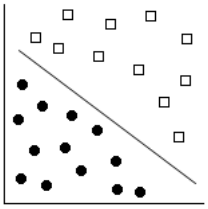


Figura 2.1: Datos clasificados mediante un clasificador lineal

Aprendizaje no supervisado

Los algoritmos de aprendizaje no supervisado se caracterizan por aplicarse a conjuntos de datos a los cuales, durante la etapa de entrenamiento no se les conoce su etiqueta de salida. A diferencia de los algoritmos de aprendizaje supervisado, estos algoritmos solo disponen de los objetos para el entrenamiento, en lugar de disponer de objetos y etiquetas. Dichas técnicas se emplean para detectar patrones o similaridades entre los datos que eran desconocidas.

Un ejemplo de esta clase de algoritmos es el algoritmo *K-Means* empleado para el problema de agrupamiento (clustering por su denominación en inglés) de datos en distintas clases. Este problema consiste en reunir objetos en K grupos, en el que cada observación es asignada al grupo para el cual su valor medio es más cercano, ilustrado en la Figura 2.2. De esta forma el concepto de distancia entre objetos cobra mucha importancia en este tipo de algoritmos, ya que determina la forma de agrupar los objetos.

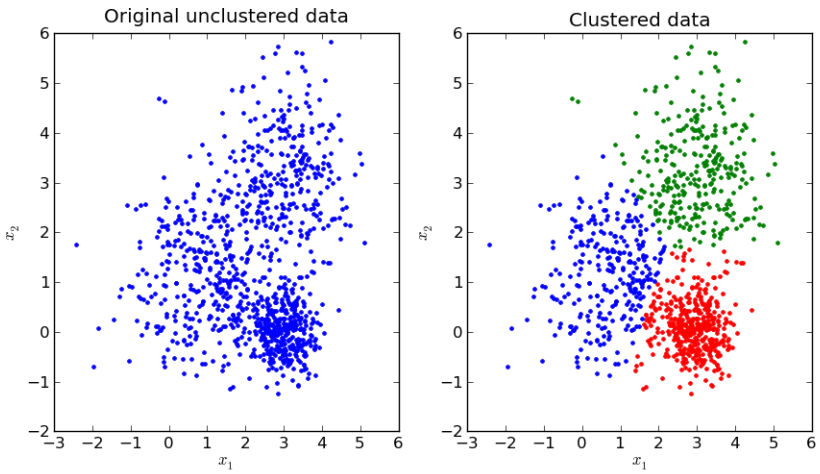


Figura 2.2: Datos no etiquetados agrupados mediante K-means

Aprendizaje semi supervisado

Estos algoritmos se caracterizan por utilizar una pequeña cantidad de datos etiquetados y otro gran conjunto de datos no etiquetados. Un ejemplo de esta clase de algoritmos es el filtrado colaborativo, utilizado para recomendar elementos a usuarios.

El problema de recomendación consiste en tratar de predecir la preferencia que un usuario podría hacer por un artículo. El enfoque mayormente utilizado para realizar una recomendación es el de filtrado colaborativo, que se basa en la recolección y el análisis de una gran cantidad de información sobre los usuarios, los comportamientos, actividades o preferencias. La predicción de un posible articulo de interés para un usuario, se realiza basada en la similitud de este usuario con otros, de los cuáles se les conoce mas información. Por ejemplo para recomendar una película a un usuario, utilizando información de las películas ya observadas, se puede establecer una similitud de intereses con otros usuarios, que además vieron otras películas, las cuales son recomendadas a este usuario.

2.1.3. Representación de la información en aprendizaje automático

Para que un modelo pueda aprender, es necesario lograr una representación adecuada de los objetos sobre los cuales se desea asimilar información. Dependiendo de la información que se desee aprender, estas representaciones pueden ir variando. Si se desea entender sobre el riesgo crediticio de un usuario, la información que represente a este usuario será diferente a la representación de ese mismo usuario si se desea aprender acerca de sus intereses gastronómicos.

Definir la representación de un objeto de forma precisa, tal que le permita al modelo capturar toda la información requerida puede llegar a ser la tarea más importante de todo el proceso de aprendizaje. Estas representaciones suelen estructurarse como un vector de características (*features* en inglés) que el modelo deberá obtener a partir de las instancias de entrenamiento. Una *feature* puede ser cualquier dato asociado a la instancia que podría ser útil al modelo para realizar una predicción más precisa.

Por ejemplo para representar un conjunto de datos conformado por documentos de texto, un método comúnmente utilizado es la bolsa de palabras (*Bag of Words* en la literatura en inglés) [6]. Este método emplea una estructura de datos auxiliar conocida como "vocabulario" o "diccionario", que contiene todas las palabras utilizadas en el conjunto de datos. Cada documento se representa como un histograma de ocurrencias respecto de las palabras del vocabulario. La dimensionalidad de la representación es igual al número de palabras que componen dicha estructura.

En el caso de las imágenes, el método más sencillo es usar directamente el valor de los píxeles. No obstante, para que una representación de una imagen sea lo suficientemente descriptiva respecto al contenido de la misma, debe superar ciertos desafíos, ilustrados gráficamente en la Figura 2.3:

- Variación del punto de vista: una simple instancia de un objeto puede estar orientada de muchas formas frente a la cámara que toma la imagen.
- Variación de escala: las clases visuales suelen exhibir variaciones en su tamaño en el mundo real y no solo en lo referido a la imagen.
- Deformación: muchos objetos de interés no tienen un cuerpo rígido y pueden ser deformados de muchas formas
- Oclusión: los objetos de interés pueden estar ocluidos y solo una pequeña porción del objeto puede ser visible.
- Condiciones de iluminación: los efectos de la iluminación pueden influir de forma drástica a nivel de píxeles.
- Influencia del fondo: los objetos de interés pueden estar inmersos en un ambiente en el cual sean difíciles de identificar.

- Variaciones intra clase: existen muchos instancias completamente distinta de una misma categoría de objetos.

Figura 2.3: Desafíos que debe superar la representación de una imagen

A lo largo del desarrollo de la visión por computadora se distinguen dos enfoques para la generación de representaciones que permitan reconocer características de una imagen: El enfoque basado en técnicas superficiales (o *Shallow* en inglés) y el enfoque de aprendizaje profundo (o *Deep* en inglés).

Enfoque clásico

En el enfoque clásico (o arquitecturas *Shallow*, como se conocen en la literatura en inglés), se busca generar una representación invariante a la posición, iluminación, fondo, etc. que describan regiones locales de la imagen, de forma robusta. Para ello se utilizan principalmente técnicas estadísticas a partir de conocimiento anterior del objeto y del contexto, con lo cual dependen en cierta forma de una decisión manual de quien construye el algoritmo. El conjunto de estas representaciones locales se codifican y agregan en una representación vectorial de dimensionalidad fija, la cual se emplea como entrada a los algoritmos de aprendizaje subsiguientes. Un ejemplo de esta clase de representaciones es el modelo de Bolsas de Palabras visuales el cual, en analogía al modelo para texto, recurre a una estructura auxiliar para construir un histograma de ocurrencias de "palabras visuales". Este concepto se ilustra en la Figura 2.4 en donde se intenta reconocer un rostro, una bicicleta y una guitarra, basándose en las palabras visuales que contiene su "Diccionario".

Enfoque de aprendizaje profundo

Este enfoque tiene como objetivo lograr una representación composicional de la imagen. Consta de un único algoritmo que, a diferencia del enfoque clásico, aprende a realizar la representación por si mismo, en lugar de depender de decisiones externas. El contenido representado dependerá de la información que tenga como objetivo aprender el modelo. Por ejemplo, suponiendo que una misma imagen contenga un gato y un perro, si el modelo debe reconocer perros, la representación contiene información relevante al perro, en cambio si el modelo debe reconocer animales, la representación incluye información de ambos animales.

El problema asociado a este enfoque es que se desconoce cuales serán las *features* que el modelo aprenderá a representar durante el entrenamiento. No obstante, se puede establecer

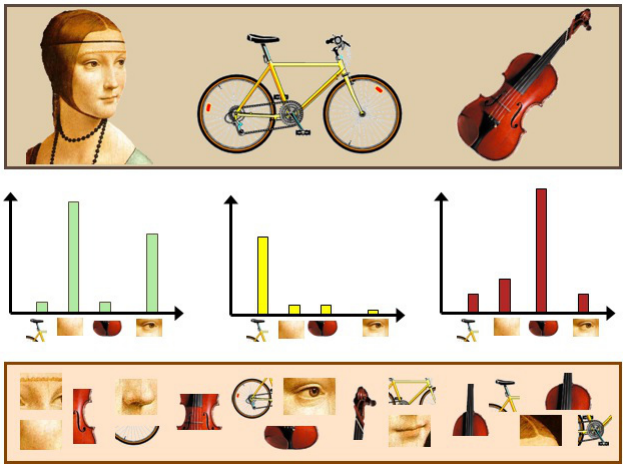


Figura 2.4: Ejemplo de Bolsa de Palabras visuales

que las *features* aprendidas tienen en cierto modo una especie de jerarquía. Las *features* de bajo nivel aprendidas desde los píxeles alimentan el aprendizaje de otras *features* que aprenden a reconocer contornos y bordes, y así sucesivamente hasta llegar a las *features* de más alto nivel que logran reconocer estructuras y objetos. Debido a esto, las estructuras de aprendizaje profundo contemplan esta particularidad organizándose en diferentes capas, donde la salida de una capa es la entrada a la capa siguiente. Las redes neuronales convolucionales soportan el aprendizaje de este tipo *features* y es el modelo empleado en este trabajo.

En la Figura 2.5 se realiza una comparación entre los enfoques clásicos y el enfoque de aprendizaje profundo. En las 3 primeras opciones del diagrama se pueden observar las distintas formas de representar imágenes mediante técnicas de aprendizaje automático tradicional, donde, en todos los casos, las representaciones están predefinidas antes de entrenar el modelo. Para mejorar estas representaciones se van agregando módulos que permiten mejorar la predicción. En cambio para las técnicas de aprendizaje profundo, las representaciones son directamente aprendidas durante el entrenamiento del modelo.

2.1.4. Aprendizaje

Tanto en las representaciones *shallow* como en aquellas basadas en arquitecturas *deep*, el concepto de "aprendizaje a partir de ejemplos" requiere la introducción de dos conceptos fundamentales: el de función de predicción y el de función de pérdida. Mediante la primera se define un modelo que, una vez ajustados sus parámetros, se utiliza para realizar predicciones sobre entradas no vistas durante el entrenamiento. La segunda define el criterio a partir del cual se deben ajustar los parámetros de la anterior.

Para facilitar la exposición, se abordará el problema de clasificación mediante funciones de predicción lineales. En la sección 2.2 se tratará el caso de funciones de predicción más complejas, en particular el caso de redes neuronales artificiales y sus particularidades.

Una función de predicción en clasificación se puede representar en términos generales como un mapeo $f_\theta : \mathcal{X} \rightarrow \{1, \ldots, K\}$ con parámetros θ, del conjunto del espacio de entrada $\mathcal{X}$ (en nuestro caso las imágenes o sus representaciones) a un conjunto discreto de categorías o clases ($\{1, \ldots, K\}$). Al utilizar modelos lineales, la función de predicción toma la forma:

$$f(x) = Wx + b \tag{2.1}$$

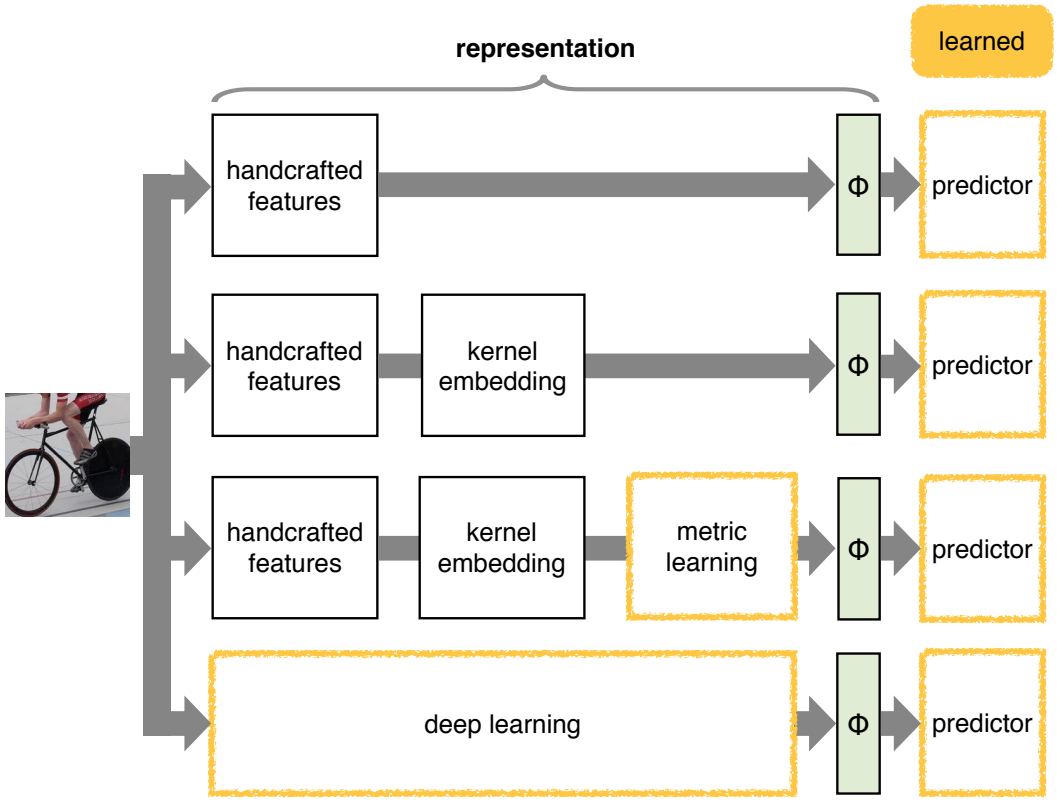


Figura 2.5: Comparación de los distintos enfoques clásicos frente al enfoque de aprendizaje profundo

en donde $\theta = \{W, b\}$, con $W \in \mathbb{R}^{K \times D}$ y $b \in \mathbb{R}^K$.

El objetivo del aprendizaje es estimar θ, a partir de un conjunto de N pares de entrenamiento $\{(x_i, y_i)\}_{i=1}^N$, $y_i \in \mathcal{X}$, $y_i \in \{1, \ldots, K\}$ donde x_i hace referencia a la representación de la i-esima imagen del conjunto e y_i es la respectiva categoría asociada.

A modo de ilustración, en la Figura 2.6 se puede observar gráficamente como funciona un clasificador lineal que distingue aviones, autos y ciervos en imágenes.

El problema de estimar los parámetros de un modelo, se resuelve optimizando una función de costo expresada en la ecuación:

$$L = \frac{1}{N} \sum_{i=1}^{N} L_i(y_i, f(x_i, W, b)) + \lambda \Omega(W) \tag{2.2}$$

Con $L : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_+$, donde $\mathcal{Y}$ en el caso de la clasificación seria $1, .., K$. Esta ecuación contiene 2 términos: uno relacionado a las funciones de pérdida de cada imagen $L_i(y_i, f(x_i, W, b))$ y otro relacionado a un coeficiente de regularización $\lambda\Omega(W)$ que penaliza a los modelos complejos, donde λ es un hiper-parámetro que define cuanto influye la regularización sobre el resultado final y N es la cantidad de datos de entrenamiento.

Una función de pérdida es una función real, no negativa $L(\widehat{y}, y)$, que mide cuán diferente es la predicción $\widehat{y} = f(x, W, b)$ obtenida, con respecto a la salida esperada y, donde $\widehat{y}, y \in 1, .., K$.

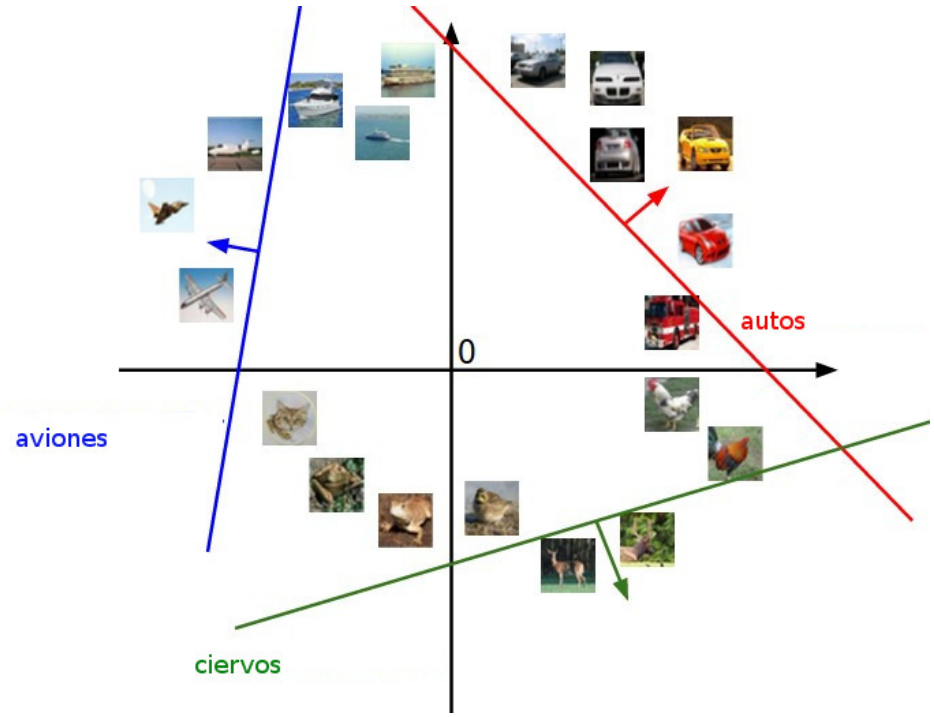


Figura 2.6: Ejemplo de clasificador lineal que utiliza una función de puntuación lineal para clasificar entre autos, aviones y ciervos

Existen diversas funciones de pérdida que se utilizan en distintos contextos. A continuación se mencionan algunas que se emplean usualmente:

- Función de Pérdida 0-1:

$$L(\widehat{y}, y) = \begin{cases} 1 & \text{si } \widehat{y} = y \\ 0 & \text{en caso contrario.} \end{cases}$$

- Función de Pérdida Hinge:

$$L(\widehat{y}, y) = max(0, 1 - \widehat{y}y)$$

  Esta función se utiliza para algunos algoritmos sin embargo no es derivable, característica que necesitaremos más adelante para aplicar métodos de optimización.

- Función de Pérdida Logística:

$$L(\widehat{y}, y) = \log(1 + \exp^{-\widehat{y}y})$$

  Similar a la función Hinge, pero al ser derivable, puede utilizarse para aplicar descenso por el gradiente, un método de optimización explicado en la siguiente sección.

- Función de Pérdida de Entropía Cruzada

$$L(\widehat{y}, y) = -y\log(\widehat{y}) - (1 - y)\log(1 - \widehat{y}) \tag{2.3}$$

  Es una función continua, convexa y derivable que también se adapta a métodos de optimización.

Optimización

Minimizar la función de pérdida (2.2) es un problema de optimización, con lo cual se pueden aplicar las técnicas empleadas en este tipo de problemas, para resolverlo. El objetivo de la optimización es encontrar el vector de pesos W que minimice la función de pérdida.

La estrategia de optimización más utilizada para este tipo de problemas es la de seguir la dirección del gradiente de la función de pérdida. En este caso, la función toma como entrada un vector, se aplican las derivadas parciales y el gradiente es el vector resultante de calcular las derivadas parciales en cada dimensión. La idea de principal de este método es el refinamiento iterativo, se evalúa el resultado del cálculo del gradiente y se actualizan los parámetros repetidamente, toma un hiper parámetro η llamado tasa de aprendizaje que define el tamaño de cada paso en la iteración. Sea F la función de pérdida, w el vector de pesos, N el número de datos de entrenamiento, cada iteración consiste de:

$$w_{n+1} = w_n - \eta \nabla L(w_n) = w_n - \eta \sum_{i=1}^{N} \nabla L_i(w_n) \tag{2.4}$$

Un problema recurrente en el aprendizaje automático es que para una buena generalización, son necesarios grandes conjuntos de entrenamiento. Por consiguiente, el costo computacional de realizar esto en grandes conjuntos es alto. En aplicaciones de gran escala, el conjunto de entrenamiento puede ser del orden de los millones de ejemplos, por lo que computar la función de pérdida completa sobre todo el conjunto para actualizar un solo parámetro, como ocurre en el caso de (2.4), seria algo impracticable.

Un enfoque común que se aplica a este problema es computar el gradiente sobre pequeños lotes del conjunto de entrenamiento, que permite lograr una buena aproximación al objetivo completo con una convergencia mucho más rápida. Al reducir N a un pequeño subconjunto, se reduce el número de cálculos realizados en cada iteración. Este método se lo conoce como **Descenso por el gradiente estocástico**, el cual se exhibe en el diagrama 2.1 en forma de pseudo-código, donde se puede observar como funciona el mismo. El descenso por el gradiente estocástico es el método más utilizado para optimizar funciones de pérdida en las redes neuronales, un algoritmo de aprendizaje automático que se detalla en la sección 2.2.

Algorithm 2.1 Descenso por el gradiente estocástico

Elegir una tasa de aprendizaje η
Elegir un vector de pesos inicial w_0
Elegir un número de iteraciones J
$w \leftarrow w_0$
for $j = 1, ..J$ **do**
  Ordenar aleatoriamente
  for all $i = 1, .., N$ **do** $sum_{gradiente} \leftarrow sum_{gradiente} + \nabla L_i(w_n)$ $w \leftarrow w - \eta sum_{gradiente}$
  end for
end for

2.2. Redes Neuronales Artificiales

2.2.1. Introducción

Un algoritmo de aprendizaje de red neuronal artificial, usualmente llamado "red neuronal", es un algoritmo de aprendizaje que sirve como herramienta de modelado de datos en forma

no lineal. Usualmente se usan para modelar relaciones complejas entre entradas y salidas, para encontrar patrones en los datos. Una red neuronal, básicamente es una versión compleja del clasificador lineal, empleando funciones no lineales, pero que utiliza los mismos conceptos de funciones de pérdida y función de predicción.

Las redes neuronales se pueden modelar como un conjunto de unidades de cómputo (neuronas) conectadas en un grafo acíclico, que se suelen organizar por capas. Las unidades de una capa se conectan con neuronas de sus capas adyacentes pero nunca se conectan 2 unidades de una misma capa, como se puede observar en la Figura 2.7. Las funciones no lineales mencionadas, se ubican entre las conexiones de las distintas capas, *rompiendo* con la linealidad del clasificador, por lo que ésta es la principal diferencia con los clasificadores lineales. Toda red neuronal tiene una capa de entrada, una capa de salida y un número determinado de capas ocultas, que pueden ser de distintos tipos. En la caso de la Figura 2.7 la red neuronal que se ilustra, esta conformada por:

- Una capa de entrada (x_i con $i \in \{0, ..., D\}$, donde x_0 corresponde al sesgo)
- Una capa oculta (z_m con $m \in 0, .., M$ donde z_0 corresponde al sesgo). En el caso de que la red tenga más de una capa oculta, M puede ser distinto para cada una de ellas.
- Una capa de salida (y_j con $j \in \{1, ..., K\}$)
- Conexiones entre unidades de capas adyacentes, las cuales contienen un peso $w_{se}^{(l)}$ donde (l) es el número de capa, e es la unidad de entrada de la conexión y s la unidad de salida.

Una de las principales razones por la cual las redes neuronales están organizadas en capas, es que este tipo de estructura permite evaluar una red, muy simple y eficientemente realizando operaciones matriciales vectoriales. Una red neuronal puede ser pensada como una serie de multiplicaciones de matrices entrelazadas con funciones de activaciones no lineales. Las redes neuronales utilizadas en la actualidad tienen alrededor de 100 millones de parámetros distribuidos entre 10-20 capas, aunque también existen algunas de 150 capas como ResNet 2.4.1.

Cada unidad de procesamiento de una red neuronal toma como parámetros un vector de entrada, un vector de pesos, un valor de sesgo. El cómputo realizado en una unidad puede separarse en 2 etapas.

1. En la primer etapa se realiza la suma de los productos punto a punto de los vectores de entrada, adicionandole finalmente el sesgo. Sea X el vector de entrada, de D dimensiones, W el vector de pesos y w_0 el sesgo, se puede formalizar de la siguiente forma:

$$a = \sum_{i=1}^{D} w_i x_i + w_0 \tag{2.5}$$

   Donde x_i es cada uno de los valores del vector X y w_i los valores del vector W. El valor a se conoce como el valor de *activación.*

2. En la segunda etapa se realiza una transformación no lineal del valor de activación, obtenido en la etapa anterior, dada por la función de activación $h(.)$,es decir:

$$z = h(a) \tag{2.6}$$

La unidad aquí descripta pertenece a la primer capa oculta de la red, ya que en (2.5) se interactúa con los valores del vector de entrada de la red. En caso de que la unidad no pertenezca a la primer capa oculta, se reemplazan los x_i por los correspondientes z_i obtenidos en la capa anterior.

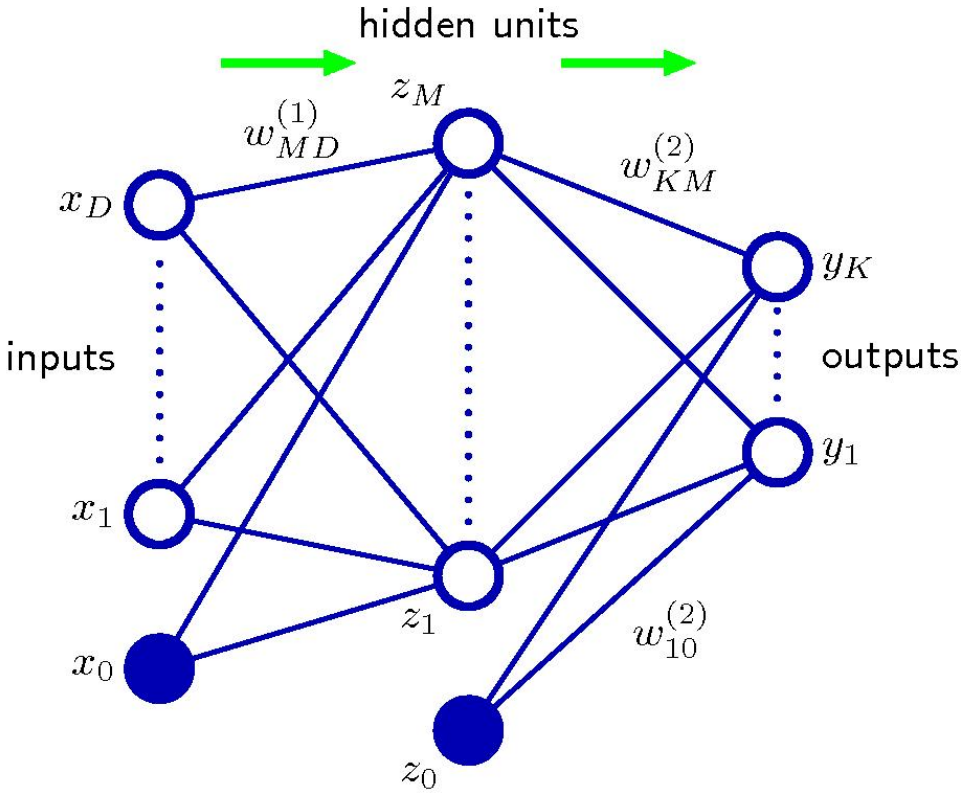


Figura 2.7: Red Neuronal Artificial

Finalmente cuando la unidad pertenece a la capa de salida, el valor obtenido en (2.6) termina siendo el valor de salida de la predicción (y).

Al utilizar una red neuronal como modelo de aprendizaje automático para clasificación, primero es necesario conFigurar la arquitectura de la red para luego comenzar con la etapa de aprendizaje.

ConFiguración y Diseño En la etapa de conFiguración y diseño de una red neuronal es donde se define la arquitectura de la red, es decir cuantas capas tendrá, cuantas neuronas tendrá cada una de las capas y cuales serán las funciones de activación que estas utilizarán. Además en esta etapa se define cual será la función de pérdida que se utilizará para calcular el error durante el entrenamiento. En esta etapa es donde se requiere mayor incidencia manual para este tipo de algoritmos, el resto se aprende automáticamente.

2.2.2. Aprendizaje

Una vez finalizada la etapa de conFiguración y diseño es necesario inicializar los pesos, comúnmente inicializados con valores aleatorios cercanos pero distintos a 0, para luego comenzar con la etapa de entrenamiento de la red. El entrenamiento de una red neuronal consiste en realizar un ciclo de entrenamiento durante un cierto número de iteraciones, utilizando el conjunto de datos de entrenamiento. Durante cada una de estas iteraciones, se reduce el error obtenido por la función de pérdida, al comparar la predicción obtenida con el valor correcto de la etiqueta para esa instancia. El ciclo de entrenamiento se compone de 2 pasos: paso hacia adelante, y paso de retropropagación del error.

1. En el paso hacia adelante se evalúa la red y se obtiene el resultado de salida, con el cual se mide el error, utilizando la función de pérdida elegida.

2. En el paso de retropropagación del error se calcula el gradiente para luego ajustar los pesos internos de la red en base al resultado del gradiente, desde las capas finales de la red hasta las capas iniciales.

2.2.3. Retropropagación del Error

La mayoría de los algoritmos de aprendizaje implican un procedimiento iterativo para la minimización de una función de pérdida o error, en el cual ajustan los pesos mediante una secuencia de pasos. En cada paso, se pueden distinguir dos etapas bien diferenciadas. En la primera etapa, las derivadas de la función de error con respecto a los pesos deben ser evaluados. En la segunda etapa, las derivadas son utilizadas para calcular los ajustes que se les deben efectuar a los pesos, en cada una de las capas de la red. La importante contribución de la retropropagación (propagar el error hacia atrás, a través de la red) es la de proveer un método computacionalmente eficiente para evaluar las derivadas requeridas en la primer etapa. A lo largo de esta sección, iremos desarrollando las ideas fundamentales de dicho método, siguiendo la presentación de Bishop [1].

Como hemos visto, la función de pérdida total, definida en la ecuación (2.2) se calcula midiendo el error en cada una de las N instancias pertenecientes al conjunto de datos. Una forma simplificada de considerar (2.2) es:

$$L(W) = \frac{1}{N}\sum_{n=1}^{N} L_n(Y_n, f(X_n, W)) + \lambda\Omega(W) \tag{2.7}$$

donde el sesgo b se considera parte del vector W. Ademas, como (X_n, Y_n) están predefinidas, podemos considerar que cada L_n solo depende de W. Debido a todo esto, para calcular el gradiente de la función de pérdida total, debemos evaluar $\nabla L_n(W)$ para cada término en la función de error. Esto puede ser usado directamente para la optimización secuencial, o los resultados pueden ser acumulados a lo largo del conjunto de entrenamiento en el caso de los métodos de optimización por lotes.

Consideremos, en primer lugar de un simple modelo lineal en el que las salidas y_k son combinaciones lineales de las variables de entrada x_i, es decir:

$$y_k = \sum_i W_{ki} x_i$$

Por simplicidad se decidió utilizar la función de error cuadrático medio, que aunque no aplica para el caso de clasificación, facilita los cálculos y puede ser reemplazada fácilmente por otras funciones de error como por ejemplo Entropía Cruzada (2.3). La función de error para un determinado valor de entrada n, toma la forma:

$$L_n = \frac{1}{2}\sum_k (y_{nk} - Y_{nk})^2$$

donde $y_{nk} = y_k(X_n, W)$.

La derivada de esta función de error con respecto a un peso w_{ji} está dada por:

$$\frac{\partial L_n}{\partial w_{ji}} = (y_{nj} - Y_{nj}) x_{ni}$$

que puede ser interpretado como un *cómputo local* que involucra el producto entre un *error* $y_{nj} - Y_{nj}$ asociado a la de salida del enlace w_{ji} y la variable x_{ni} asociada a la variable de entrada del enlace. Veremos ahora cómo este simple resultado se extiende hasta el escenario más complejo de redes neuronales artificiales.

En una red neuronal cada unidad de cómputo, calcula una suma pesada de sus valores de entrada, de la forma:

$$a_j = \sum_i w_{ji} z_i \tag{2.8}$$

Donde z_i es el valor de entrada, a_j es el valor que le envía a la unidad j y w_{ji} es el peso asociado a esa conexión. La suma obtenida en (2.8) es luego transformada por una función no lineal $h(.)$ para calcular el valor correspondiente a la unidad j, es decir:

$$z_j = h(a_j) \tag{2.9}$$

Para cada instancia en el conjunto de entrenamiento, vamos a suponer que hemos suministrado el correspondiente vector de entrada a la red y que se calcula la activación de todas unidades en la red mediante aplicaciones sucesivas de (2.8) y (2.9). Este proceso a menudo se denomina propagación hacia adelante porque puede ser considerado como un flujo de avance de la información a través de la red.

Ahora, considere la evaluación de la derivada de L_n con respecto a un peso w_{ji}. Los valores de salidas de las distintas unidades dependerá del vector de entrada n. Sin embargo, con el fin de mantener la notación despejada, vamos a omitir el subíndice n de las variables de red. Primero observamos que L_n depende del peso w_{ji} sólo a través de la suma de entrada a_j a la unidad j. Por lo tanto, podemos aplicar la regla de la cadena para derivadas parciales, con lo cual obtenemos:

$$\frac{\partial L_n}{\partial w_{ji}} = \frac{\partial L_n}{\partial a_j} \frac{\partial a_j}{\partial w_{ji}} \tag{2.10}$$

Se introduce una notación útil:

$$\delta_j \equiv \frac{\partial L_n}{\partial a_j} \tag{2.11}$$

Utilizando (2.8) podemos escribir:

$$\frac{\partial a_j}{\partial w_{ji}} = z_i \tag{2.12}$$

Substituyendo (2.11) y (2.12) en (2.10) obtenemos:

$$\frac{\partial L_n}{\partial w_{ji}} = \delta_j z_i \tag{2.13}$$

La ecuación (2.13) nos dice que la derivada se obtiene simplemente multiplicando el valor de δ para la unidad en el extremo de salida del peso por el valor de z para la unidad en el extremo de entrada del peso. Tenga en cuenta que esto tiene la misma forma que para el simple modelo lineal considerado al inicio de esta sección. Por lo tanto, con el fin de evaluar las derivadas, sólo tenemos que calcular el valor de δ_j para cada unidad oculta y de salida en la red y, a continuación, aplicar (2.13). Dado que para las unidades de salida utilizamos el enlace canónico, tenemos

$$\delta_k = y_k - Y_k \tag{2.14}$$

Para evaluar las δ para las unidades ocultas, volvemos a hacer uso de la regla de la cadena para derivadas parciales:

$$\delta_j \equiv \frac{\partial L_n}{\partial a_j} = \sum_k \frac{\partial L_n}{\partial a_k} \frac{\partial a_k}{\partial a_j} \tag{2.15}$$

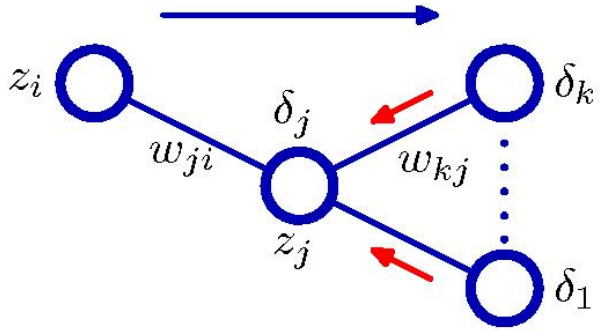


Figura 2.8: Ilustración del cálculo de δ_j de la unidad oculta j por retropropagación de la $\delta's$ a partir de las k unidades para las cuales la unidad j envía conexiones. La flecha azul indica el la dirección del flujo de la información durante la propagación hacia adelante, y las flechas rojas indican la propagación hacia atrás de la información de error.

Donde la suma se ejecuta sobre todas las unidades k donde la unidad j envía conexiones.

La disposición de las unidades y pesos se ilustra en la Figura 2.8. Notar que las unidades etiquetadas k podrían incluir otras unidades ocultas y/o de salida. En la escritura de la ecuación (2.15), estamos haciendo uso del hecho de que las variaciones en un j dan lugar a variaciones en la función de error sólo a través de las variaciones en las variables de k. Si ahora sustituimos la definición de δ dada por (2.11) dentro de (2.15), y hacemos uso de (2.8) y (2.9), obtenemos la siguiente fórmula de retropropagación

$$\delta_j = h'(a_j) \sum_k w_{kj}\delta_k \tag{2.16}$$

Donde h' es la función derivada a partir de h.

La ecuación (2.16) nos dice que el valor de δ para una determinada unidad oculta se puede obtener mediante la propagación de la δ's hacia atrás a partir de unidades de más adelante en la red, como se ilustra en la Figura 2.8. Como ya conocemos los valores de los δ's para las unidades de salida, podemos aplicar de forma recursiva (2.16) para evaluar los δ's para todas las unidades ocultas en una red, independientemente de su arquitectura. Por lo que la retropropagación del error se puede resumir en los siguientes pasos:

1. Aplicar un vector de entrada x_n a la red y propagarlo hacia adelante a través de la red utilizando (2.8) y (2.9) para encontrar las activaciones de todas las unidades ocultas y de salida.

2. Evaluar las δ_k para todas las unidades de salida utilizando (2.14).

3. Propagar hacia atrás los δ's usando (2.16) para obtener el δ_j de cada unidad oculta de la red.

4. Utilizar (2.13) para evaluar las derivadas requeridas.

2.2.4. Funciones de Activación comúnmente utilizadas

Como hemos visto en la sección 2.2, cada unidad de cómputo requiere de una función de activación. Toma un único número como entrada, realiza una operación matemática predefinida, no lineal y devuelve el resultado obtenido. En esta sección se analizan las principales funciones de activación utilizadas.

Sigmoide

La función sigmoide proyecta el dominio al rango (0,1) y su ecuación es:

$$f(x) = \frac{1}{1 + \exp^{-x}}$$

Ha tenido un uso frecuente desde el punto de vista histórico, ya que tiene una interpretación similar a la tasa de activación de una neurona: de no activarse en absoluto (0) a activarse completamente a una frecuencia máxima asumida (1), como puede observarse en la Figura 2.9. En la práctica, la función sigmoide recientemente ha dejado de utilizarse, ya que tiene dos inconvenientes principales:

- Se satura y anula el gradiente. Una propiedad muy indeseable de la sigmoide es que cuando la activación de la neurona se satura en cualquiera de las dos puntas de 0 o 1, el gradiente en estas regiones es casi cero. Esto hace que en la retropropagación, la información deje de retropropagarse, (cuando se topa con un gradiente 0, cualquier valor multiplicado por 0 es 0).

- Las salidas no están centradas en cero. Esto es indeseable ya que las neuronas en capas posteriores de procesamiento en una Red Neuronal estarían recibiendo datos que no están centrados en cero. Esto tiene implicaciones en la dinámica durante el descenso del gradiente, porque si los datos que llegan a una neurona son siempre positivos, entonces el gradiente en los pesos durante la retropropagación o bien serán todos positivos o todos negativos.

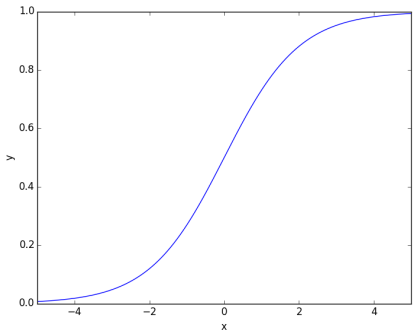


Figura 2.9: Gráfico de la función Sigmoide

Tangente hiperbólica

Proyecta el dominio al rango [-1,1], como se muestra en la Figura 2.10. Al igual que la función sigmoide, sus activaciones saturan, pero a diferencia de ésta, su salida esta centrada en cero. Por lo tanto, en la práctica la tanh siempre es preferida sobre sigmoide.

$$f(x) = \frac{\exp^{x} - \exp^{-x}}{\exp^{x} + \exp^{-x}}$$

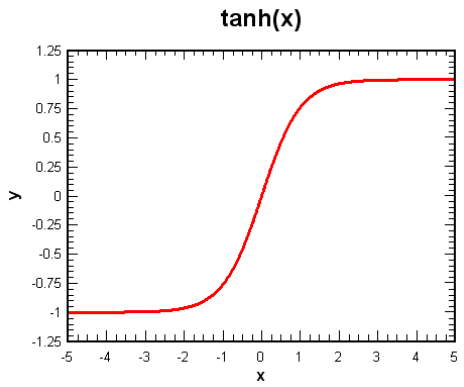


Figura 2.10: Gráfico de la función Tangente hiperbólica

ReLU

La Unidad Rectificada Lineal (Rectified Linear Unit en la literatura en inglés) se ha vuelto muy popular en los últimos años. Toma el máximo entre el valor y cero, es decir simplemente le agrega un umbral igual a 0, su gráfico se puede ver en la Figura 2.11.

$$f(x) = max(0, x) \tag{2.17}$$

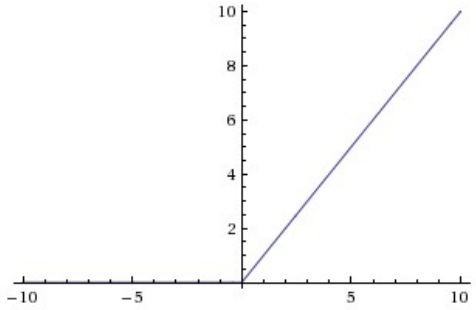


Figura 2.11: Gráfico de la función ReLU

Las unidades lineales rectificadas son fáciles de optimizar porque son muy similares a las unidades lineales. La única diferencia entre una unidad lineal y una unidad lineal rectificada es que una unidad lineal rectificada da salida a cero a través de la mitad de su dominio. Esto hace que las derivadas a través de una unidad lineal rectificada permanezcan grandes siempre que la unidad esté activa. Los gradientes no sólo son grandes sino también consistentes.

Algunas ventajas de utilizar ReLUs:

- Se encontró que aceleran en gran medida la convergencia del descenso del gradiente en comparación con las funciones sigmoides/tanh, esto se debe a su forma lineal, sin saturarse.
- En comparación con las funciones tanh/sigmoide que implican operaciones caras, la ReLU puede implementarse simplemente marcando una matriz de activaciones a cero.

Existen varias generalizaciones de unidades lineales rectificadas. La mayoría de estas generalizaciones se comportan de forma comparable a las unidades lineales rectificadas y, en ocasiones,

tienen un mejor desempeño. Un inconveniente para las unidades lineales rectificadas es que no pueden aprender a través de métodos basados en gradiente en ejemplos para los cuales su activación es cero. Una variedad de generalizaciones de unidades lineales rectificadas garantizan que reciben gradiente en todas partes. Existen generalizaciones sobre esta, que agregan un parámetro de regularización:

- Leaky ReLU: Agrega un parámetro de fuga α, fijándolo en un valor pequeño como por ejemplo 0.01.
$$f(x) = 1(x < 0)(\alpha x) + 1(x >= 0)(x)$$
- ReLU Paramétrica: Trata el parámetro de fuga α como un parámetro aprendible.
$$f(x) = \begin{cases} x & \text{si } x > 0 \\ \alpha x & \text{en caso contrario.} \end{cases}$$

2.3. Redes Neuronales Convolucionales

Muchos de los enfoques modernos en Visión por Computadoras explotan la propiedad de que píxeles cercanos están más correlacionados entre ellos. Esto aprovecha mediante la extracción de características locales que dependen sólo de pequeñas subregiones de la imagen. La información de tales características pueden entonces ser combinada en etapas posteriores de procesamiento con el fin de detectar de características de alto nivel hasta finalmente proveer información de la imagen completa. Además, las características locales que son útiles en una región de la imagen, probablemente sean útiles en otras regiones de la imagen. Estas nociones son incorporadas a las CNN mediante tres mecanismos: (i) campos receptivos locales, (ii) compartir pesos y (iii) realizar submuestreos o agrupaciones.

Las CNN son muy similares a las redes neuronales antes vistas en la sección 2.2. La principal diferencia es que la arquitectura de una red neuronal convolucional asume explícitamente que el conjunto de entrada son imágenes. Esto le permite codificar ciertas propiedades de las imágenes dentro de la arquitectura, obteniendo una mejor representación de las mismas. Están inspiradas en las redes neuronales tradicionales, incorporando operaciones no-lineales como ReLU y otras específicas de manipulación de imágenes como la convolución.

En el área del procesamiento de imágenes, la convolución es una técnica que modifica una imagen, utilizando el valor de sus píxeles, mediante una pequeña matriz comúnmente llamada filtro (por lo general de 3×3 o 5×5), que puede ser empleada por ejemplo, para aplicar un efecto a la imagen. Al realizar la convolución sobre una imagen, se aplica un filtro, obteniendo como resultado otra imagen nueva. Cada uno de los píxeles de esta nueva imagen se calcula de la siguiente forma: El filtro actúa determinando el valor del píxel central donde se esta aplicando, sumando los valores de los píxeles vecinos pesados por el valor correspondiente de la matriz para cada píxel. En la Figura 2.12, se puede observar la aplicación de un filtro de 3×3. El cálculo realizado es (40*0)+(42*1)+(46*0) + (46*0)+(50*0)+(55*0) + (52*0)+(56*0)+(58*0) = 42, por lo que el valor obtenido para ese píxel es 42. Este filtro se va desplazando sobre todos los píxeles de la imagen de entrada hasta lograr calcular el valor todos los píxeles de la imagen de salida. Para el caso de los píxeles correspondientes a los bordes de la imagen, que no disponen de píxeles vecinos en alguna de las posiciones donde se aplica el filtro, existen diversas formas de completarlo. Una forma de completar el cálculo es considerando que donde hay píxeles ausentes su valor es cero, otra es considerar que tienen el mismo valor que el píxel más cercano.

La estructura de una CNN se ilustra en la Figura 2.14. En una capa convolucional (capa compuesta por unidades que realizan convoluciones), las unidades se organizan en tres dimensiones, donde los distintos planos generados, utilizando la dimensiones alto y ancho se llaman mapa de características.

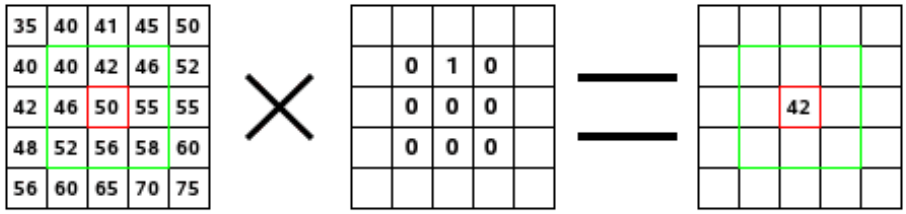


Figura 2.12: Convolución

Las unidades en un mapa de características toman como entradas solo una pequeña subregión de la imagen (campos receptivos locales) lo que le permite mantener la localidad espacial obteniendo invarianza translacional (*shif invariance* en inglés). Por otro lado todas las unidades de un mismo mapa de características se ven obligadas a compartir los mismos valores de peso. Por ejemplo, una mapa de características puede consistir de 100 unidades dispuestas en una de cuadrícula 10x10, donde cada unidad toma de entradas una porción de 5x5 píxeles de la imagen. El mapa de características completo tiene entonces veinticinco parámetros de pesos ajustables, más uno de sesgo. Los valores de entrada de cada porción de imagen son linealmente combinados utilizando los pesos y el sesgo, y el resultado es transformado por una función no lineal como la función sigmoide.

Si consideramos a las unidades como detectores de características, entonces, todas las unidades en un mismo mapa de características detectan el mismo patrón, pero en diferentes ubicaciones en la imagen de entrada. Debido a que comparten los pesos, la evaluación de las activaciones de estas unidades es equivalente a realizar una operación de convolución sobre la intensidad de los píxeles de la imagen con un filtro compuesto por los parámetros de peso.

Además de las capas convolucionales, existen otros dos tipos de capas comúnmente utilizadas en las CNN: Capas de submuestreo o agrupación (subsampling o pooling en inglés) y capas completamente conectadas. Las capas de submuestreo tienen como función reducir progresivamente el tamaño espacial de la representación. De esta forma, se reduce la cantidad de parámetros y por ende de cálculos en la red, como así también permite obtener una mejor generalización a la hora de predecir. Estas capas funciona independientemente en cada segmento de profundidad de la entrada y la redimensiona espacialmente, utilizando la operación de máximo como se observa en la Figura 2.13. En algunos casos se suele utilizar la operación de promedio o la norma $L2$ en lugar del máximo.

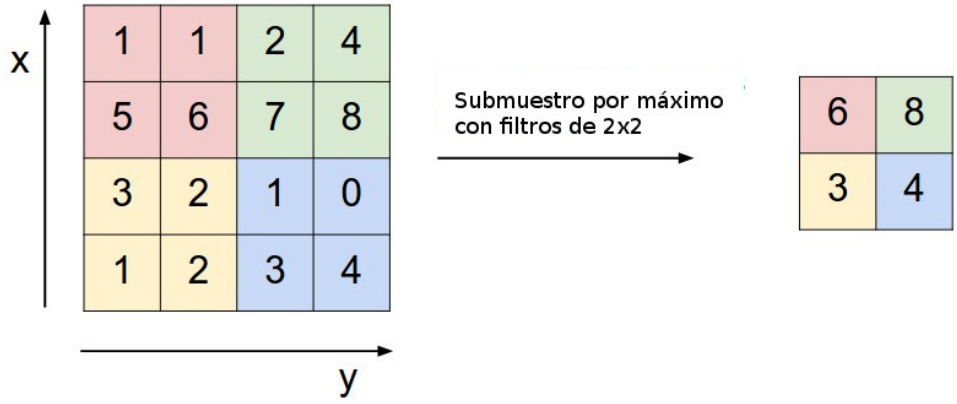


Figura 2.13: La capa agrupación reduce el volumen espacialmente, de forma independiente en cada una profundidad de la entrada de volumen. A la izquierda: En este ejemplo, el volumen de entrada de tamaño [224x224x64] es agrupado con un filtro de tamaño 2 y salto 2 en un volumen de salida de tamaño [112x112x64]. Observe que la profundidad del volumen se conserva. A la derecha: la forma más común de reducción de tamaño de la operación de máximo, dando lugar agrupación por máximo, que aquí se muestra, con un salto de 2. Es decir, cada operación se toma con 4 números (pequeño cuadrado de 2x2). [7]

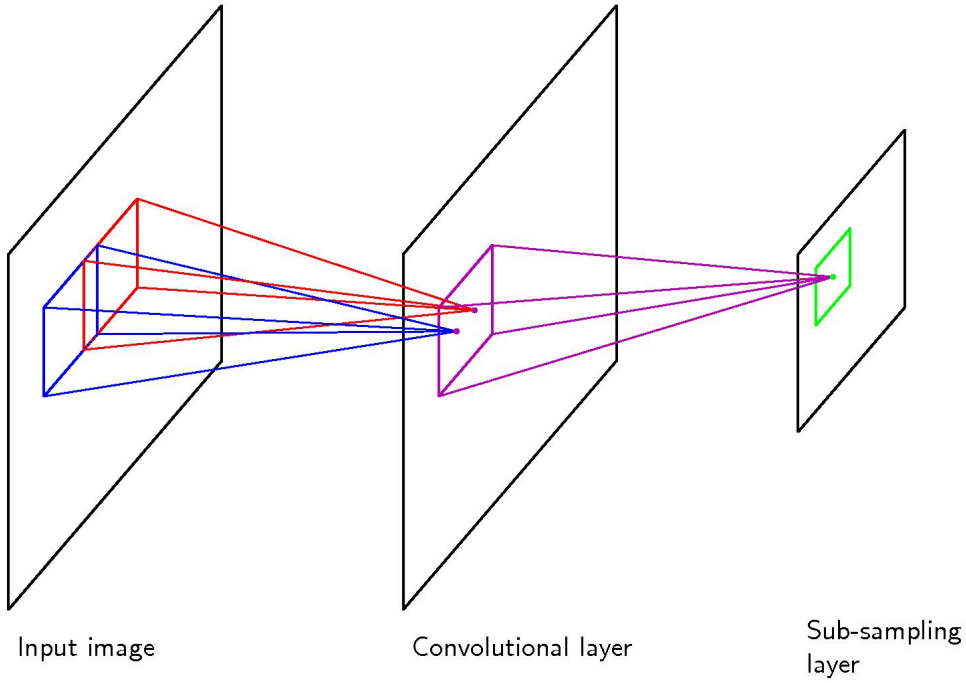


Figura 2.14: Diagrama que ilustra parte de una CNN, mostrando una capa convolucional, seguido por una capa de submuestreo.

En cuanto a las capas completamente conectadas, éstas toman todas las neuronas de la capa anterior y las conectan a cada una de las neuronas que posee, es decir, mediante una capa completamente conectada, las neuronas entre dos capas adyacentes están completamente conectadas de a pares. De esta forma, se pierde la localidad espacial, por lo que este tipo de capas suelen estar al final de la red.

Lograr diseñar una red convolucional que funcione correctamente, no es una tarea simple. Ademas, una red neuronal convolucional moderna requiere entre dos y tres semanas para entrenarse, utilizando múltiples GPUs para el conjunto de datos ImageNet [8]. Debido a estos motivos, en la actualidad un practica común es adaptar redes famosas ya preentrenadas, realizando sobre estas ajuste fino ("finetuning" en la literatura en ingles), técnica que consiste en adaptar el clasificador de una red preentrenada a las nuevas etiquetas, proveyendo un pequeño conjunto de datos que permita *reentrenar* la misma, ajustando los pesos a los datos actuales. Dicha técnica se detalla en la siguiente sección.

2.4. Ajuste fino

El ajuste fino (Finetuning en inglés), es una técnica que toma un modelo ya entrenado, con sus respectivos pesos y lo ajusta a las nuevas etiquetas de clasificación. Dado un modelo inicial, preentrenado y un pequeño nuevo conjunto de datos junto con las respectivas etiquetas en las cuales se desea clasificar, el ajuste fino permite adaptar el modelo inicial a los nuevos datos y etiquetas. Por lo general, consiste en reentrenar la arquitectura del modelo inicial, reemplazando la última capa, que se encarga de la clasificación, por una nueva capa adaptada a las nuevas etiquetas. Para reentrenar el modelo, se utilizan los pesos obtenidos del modelo preentrenado como inicialización, y luego se van ajustando en base al nuevo conjunto de entrenamiento. Ésta técnica suele ser muy exitosa en casos donde la clasificación inicial de la red no difiera en gran manera de la clasificación a la cual se desea adaptar.

Por ejemplo una red que fue entrenada para reconocer 100 razas de perros en imágenes, ahora se desea clasificar entre 50 razas de gatos. En lugar de entrenar una red desde un estado aleatorio inicial, se puede utilizar la red preentrenada para perros, reemplazando la ultima capa completamente conectada (de 100 salidas) por otra que tenga la cantidad de salidas igual a la cantidad de razas de gato que se desean reconocer (en este caso 50). Para el entrenamiento de esta red adaptada se le brinda como entrada un conjunto de imágenes, de gatos de las distintas razas. De este modo los pesos se van ajustando para lograr reconocer este nuevo tipo de conceptos en las imágenes.

2.4.1. Redes Convolucionales Famosas y Modelos preentrenados

Como mencionamos anteriormente, una práctica usual es reentrenar modelos preentrenados, adaptándolos a las necesidades especificas. Existen modelos de redes convolucionales que fueron reconocidos por haber ganado competencias orientadas a la clasificación de imágenes, los cuales son de disponibilidad publica. Estos modelos suelen ser empleados como base preentrenada. A continuación se mencionan algunos de los más famosos:

LeNet5

El primer éxito en las aplicaciones de redes neuronales convolucionales, fue desarrollado por Yann LeCun [9] en la década de 1990. La aplicación más conocida que utiliza la arquitectura LeNet es para leer los códigos postales, números, etc. Esta arquitectura es de las más rudimentarias y solo contiene cinco capas de las cuales dos son capas convolucionales como se ilustra en la Figura 2.15.

AlexNet

La primera obra que popularizó las Redes Neuronales Convolucionales en Visión por Computadoras fue el AlexNet [10], desarrollada por Alex Krizhevsky, Ilya Sutskever y Geoff Hinton.

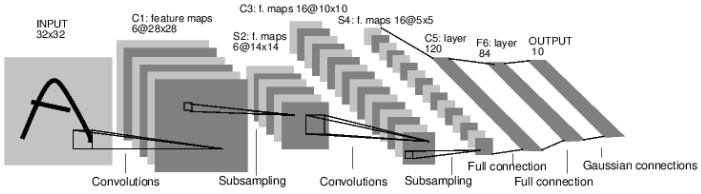


Figura 2.15: Arquitectura de la red LeNet

La red AlexNet fue presentada al desafío ImageNet ILSVRC en 2012 y obtuvo el primer puesto superando significativamente al segundo (top 5 de error de 16 % en comparación con el subcampeón con el 26 % de error). La red tenía una muy arquitectura similar a LeNet, pero era más profunda, con más capas convolucionales apiladas una encima de la otra (anteriormente era común tener solo una capa convolucional siempre seguida inmediatamente por una capa de agrupación). La arquitectura de esta red se puede observar en la Figura 2.16.

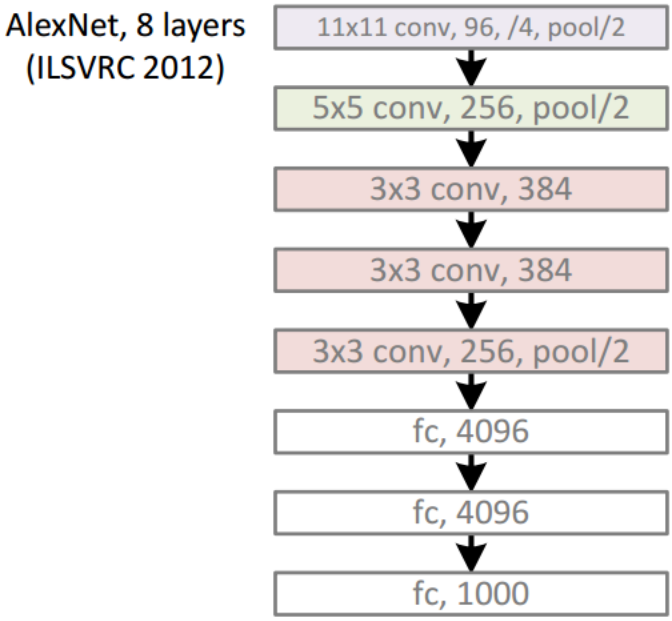


Figura 2.16: Arquitectura de la red AlexNet

GoogLeNet

El ganador de ILSVRC 2014 fue la Red Convolucional de Szegedy [11], proveniente de Google. Su contribución principal fue el desarrollo de un módulo de "Inception" que reduce drásticamente el número de parámetros de la red (4M, en comparación con AlexNet con 60M). Este módulo básicamente actúa como múltiples filtros de convolución procesados sobre el mismo dato de entrada. Todos los resultados obtenidos son concatenados, lo cual le permite al modelo extraer características de múltiples niveles sobre cada entrada. Por ejemplo puede extraer características generales (filtros de 5x5) y locales (1x1) en el mismo momento. En la Figura 2.17 se ilustra la arquitectura de este módulo gráficamente.

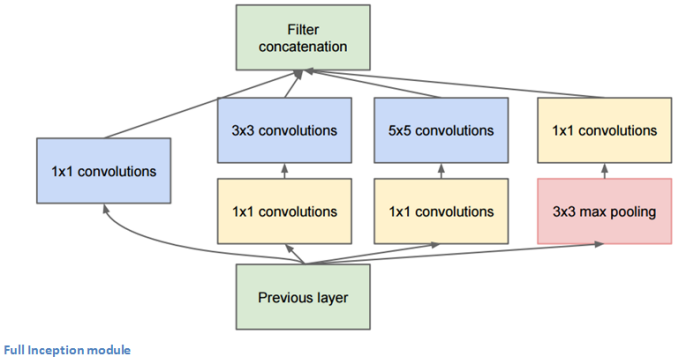


Figura 2.17: Bloque “Inception” de la red GoogLeNet

VGG

El segundo puesto en ILSVRC 2014 fue la red de Karen Simonyan y Andrew Zisserman que llegó a ser conocido como VGG [12]. Su contribución principal fue demostrar que la profundidad de la red es un componente crítico para el buen desempeño. Su versión final contiene 19 capas de las cuales 16 son convolucionales. Esta red ofrece una arquitectura muy homogénea, sólo realiza convoluciones de 3x3 y agrupaciónes de 2x2 desde el principio hasta el final, ilustrado en la Figura 2.18. Una desventaja de la VGG es que es muy cara para evaluar y consume mucha memoria y requiere de muchos parámetros (140 millones). La mayoría de estos parámetros se encuentran en la primera capa totalmente conectada, y desde entonces se ha encontrado que estas capas pueden ser removidos con ninguna disminución de rendimiento, reduciendo significativamente el número de parámetros necesarios.

ResNet

Las Redes Residuales (ResNet) fueron desarrolladas por Kaiming He [13], que fue el ganador de ILSVRC 2015. La característica particular de este tipo de redes es que utilizan bloques de conexiones salteadas ilustrados en la Figura 2.19. Esta arquitectura también solo tiene una capa completamente conectada en el extremo final de la red, en las capas intermedias hacen uso intensivo de la normalización por lotes. La normalización por lotes consiste en normalizar el conjunto de datos pertenecientes a un mismo lote en el proceso de descenso por el gradiente en lotes, lo que permite incrementar la tasa de aprendizaje, como se explica en [14]. En cuanto al número de capas de la red, existen diversas versiones de las ResNet, algunas de las cuales llegan a tener más de 150 capas.

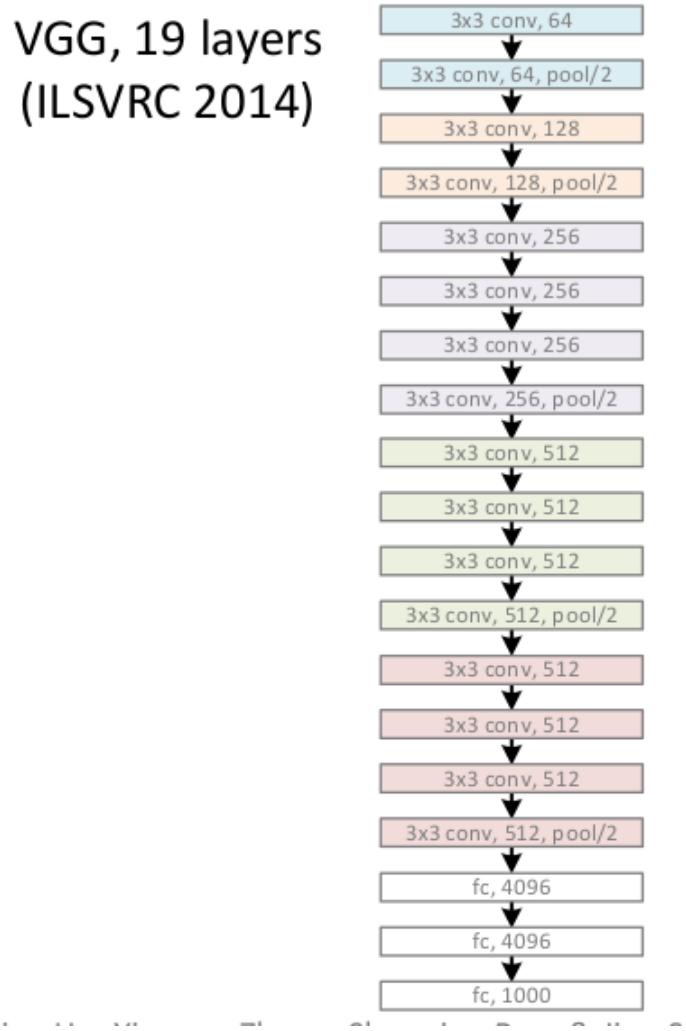


Figura 2.18: Arquitectura de la red VGG

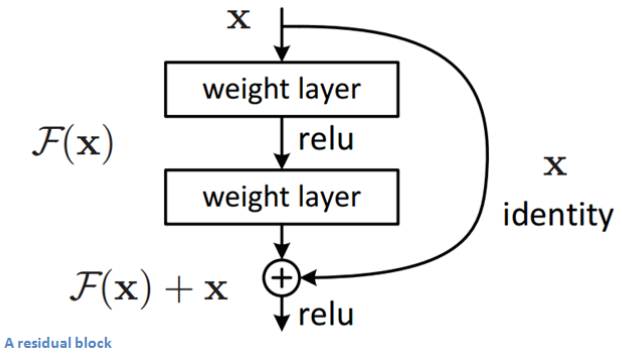


Figura 2.19: Bloque residual de ResNet

Capítulo 3

Algoritmo de Transferencia de estilo

3.1. Introducción

En este capítulo se desarrollan los contenidos específicos relativos al Algoritmo de Transferencia de Estilos artísticos a fotografías. El algoritmo elegido para tal fin es el que presenta Gatys et al. [4].

En lo que al arte respecta, los humanos han desarrollado la capacidad de crear experiencias visuales únicas componiendo una compleja interacción entre el contenido y el estilo de una imagen. Sin embargo, las bases algorítmicas de este proceso se desconocen y no existen sistemas artificiales que con capacidades similares. Sin embargo, en otras áreas fundamentales de la percepción visual como el reconocimiento de objetos y rostros recientemente se ha alcanzado precisión cercana a la de un humano, utilizando modelos de redes neuronales profundas. Aquí se presenta un sistema artificial basado en una Red Neural Convolucional que crea imágenes artísticas de alta calidad perceptiva. El sistema utiliza representaciones neuronales para separar y recombinar el contenido y el estilo de imágenes arbitrarias, proporcionando un algoritmo para la creación de imágenes artísticas.

La publicación de este artículo, ha incentivado a muchos investigadores del área a continuar desarrollando estas ideas, a tal punto que hasta el día de hoy se siguen publicando avances el tema a gran velocidad, por ejemplo [15, 16, 17] entre otros.

3.2. Contexto

Inicialmente, Gatys et al. publica un artículo para sintetizar texturas naturales utilizando Redes Neuronales Convolucionales entrenadas para el reconocimiento de objetos [18]. En el contexto del procesamiento de imágenes, el concepto de textura hace referencia a imágenes digitales que se componen por elementos repetitivos a lo largo de la misma. La síntesis de texturas es el proceso de construir algorítmicamente una imagen digital de grandes dimensiones basándose en una muestra pequeña, respetando la estructura de su contenido. La misma puede ser utilizada para rellenar agujeros en imágenes (como en los algoritmos de restauración) y la expansión de pequeñas imágenes. Los algoritmos de síntesis de texturas tienen como objetivo principal que la imagen generada sea lo más parecida posible a la imagen de muestra que están sintetizando.

En el artículo de Gatys et al., una textura es definida como la correlación entre los distintos mapas de características de cada una de las capas de la red. Aplicando este enfoque logra obtener resultados muy interesantes como se muestran en la Figura 3.1. En la parte inferior de la Figura se observa la imagen de muestra y en la parte superior se ilustra la reconstrucción de la imagen a partir de los mapas de características de una de las ultimas capas de la CNN. Ademas, en la

última columna se exhibe la representación de una imagen que no corresponde a una textura, para dar una mejor intuición de cómo se representa la información de la imagen.

Figura 3.1: Ejemplos de síntesis de texturas utilizando Redes Neuronales Convolucionales

Transferir el estilo de una imagen a otra puede ser considerado un problema de transferencia de texturas. En la transferencia de texturas el objetivo es sintetizar la textura de la imagen de muestra de forma restringida. Estas restricciones permiten preservar el contenido semántico de la imagen a la cual se le desea transferir la textura. Teniendo en cuenta todo esto, Gatys et al. adapta su idea propuesta para sintetizar texturas a la Transferencia de Estilos. Dicho problema, normalmente se encaraba en una rama de la visión de computadoras llamada representación no fotorrealista [19].

Existen otros algoritmos que utilizan este enfoque, por ejemplo Hertzmann [20], el cual utiliza analogías de imágenes para transferir la textura de una imagen de estilo a otra imagen. Ashkimin [21] se centra en la transferencia de la información de alta frecuencia de la textura, preservando la información a gruesa escala de la imagen de destino. Efros y Freeman [22] introducen un mapa de correspondencias que incluye las características de la imagen de destino, tales como la intensidad de la imagen para restringir el proceso de sintetización de la textura. Lee et al. [23] mejoran este algoritmo agregando a la transferencia de textura, información adicional acerca de los contornos y bordes.

Aunque estos algoritmos logran resultados notables, todos ellos sufren de la misma limitación fundamental: usan sólo características de bajo nivel de la imagen de destino para informar a la transferencia de textura. Sin embargo, un algoritmo de transferencia de estilo debe ser capaz de extraer contenido semántico de la imagen de destino (por ejemplo, los objetos y el paisaje general). Luego debe informar al procedimiento de transferencia de textura para que represente lo extraído junto con el estilo de la imagen de origen. Por lo tanto, un requisito fundamental es encontrar representaciones de la imagen que modelen independientemente las variaciones de contenido semántico de la imagen y el estilo en el que se les presentan. Dicho requisito se puede lograr utilizando los mapas de características de las Redes Neuronales Convolucionales 2.3 entrenadas en el reconocimiento de objetos. Se ha comprobado que éstas logran extraer información semántica de alto nivel del contenido de las imágenes [10].

3.3. Síntesis

El algoritmo desarrollado por Gatys et al. es, conceptualmente un algoritmo de transferencia de texturas que utiliza mapas de características de redes neuronales convolucionales. Cuando una Red Neuronal Convolucional (CNN por su denominación en inglés) es entrenada para el reconocimiento de objetos, desarrolla una representación interna de la imagen. Dicha representación composicional, hace que la información del objeto sea cada vez más concreta a lo largo de la jerarquía de las capas de la red. Por lo tanto, a lo largo de las distintas capas de la red, la imagen de entrada se transforma en representaciones que cada vez más se preocupan por el contenido específico de la imagen en lugar del valor de sus píxeles.

Las capas más altas de la red capturan el contenido de alto nivel en términos de objetos y su ordenamiento en la imagen. En cambio, las reconstrucciones de las capas inferiores simplemente pretenden reproducir los valores exactos de píxeles de la imagen original y algunas formas básicas como líneas o curvas. Debido a esto, para representar el contenido de una imagen se utilizaran los mapas de características de las capas superiores de la red. Para obtener la representación del estilo de la imagen, se utiliza un espacio de características que fue originalmente diseñado para capturar información de texturas. Estas representaciones se obtienen calculando la correlación entre los mapas de características de cada capa de la red.

Incluyendo las correlaciones de múltiples capas, se obtiene una representación que captura información de la textura pero no del ordenamiento global de la imagen. Al reconstruir las imágenes a partir de las representaciones obtenidas, se puede observar que producen una versión texturizada de la imagen. Estas representaciones, que capturan su apariencia general en términos de colores y estructuras localizadas, serán llamadas representaciones de estilo.

El principal descubrimiento de Gatys et al. es que la representación del estilo y del contenido de una imagen pueden ser separadas con una Red Neuronal Convolucional entrenada para el reconocimiento de objetos. De esta forma, al manipularse independientemente se pueden generar una nueva imagen desde dos imágenes de entradas distintas, que simultáneamente se corresponda con la representación del contenido de una imagen y la representación del estilo de la otra.

En la Figura 3.2 se puede observar el resultado de reconstruir el estilo y el contenido utilizando los mapas de características que se obtienen al realizar una pasada hacia adelante en la red VGG [12]. Dada una imagen de entrada, en cada capa de la red, ésta se representa como el mapa de característicasen respuesta a los filtros que pertenecen a esa capa. Mientras que el número de filtros va aumentando a lo largo de la jerarquía de la red, el tamaño de las representaciones de la imágenes va siendo reducido por algunos mecanismos disminución de resolución(por ejemplo, agrupación) que conduce a una disminución en el número total de unidades por cada capa de la red.

En la parte inferior de la Figura 3.2 podemos visualizar la información en diferentes etapas de procesamiento en la CNN por la reconstrucción de la imagen de entrada solamente conociendo las respuestas de los filtros en una capa en particular. Al reconstruir la imagen de entrada a partir de las capas 'conv1_1'(a), 'conv2_1' (b), 'conv3_1' (c), 'conv4_1' (d) y 'conv5_1' (e) de la red VGG, se puede observar que la reconstrucción de las capas inferiores es casi perfecta (a,b,c). En cambio, en las capas más altas de la red, información detallada de los píxeles se pierde, mientras que el contenido de alto nivel de la la imagen se conserva (d,e).

En la parte superior de la Figura 3.2 construimos un nuevo espacio de características que captura el estilo de una imagen de entrada. La representación de estilo se obtiene calculando las correlaciones entre los mapas de características de diferentes capas de la CNN. Se reconstruyó el estilo de la imagen de estilo utilizando diferentes subconjuntos de capas de la red ('conv1_1' (a), çonv1_1' y 'conv2_1' (b), 'conv1_1', 'conv2_1' y 'conv3_1' (c),'conv1_1', 'conv2_1', 'conv3_1 ' y 'conv4_1' (d), 'conv1_1', 'conv2_1', 'conv3_1', 'conv4_1' y 'conv5_1' (e)). Esto crea imágenes que

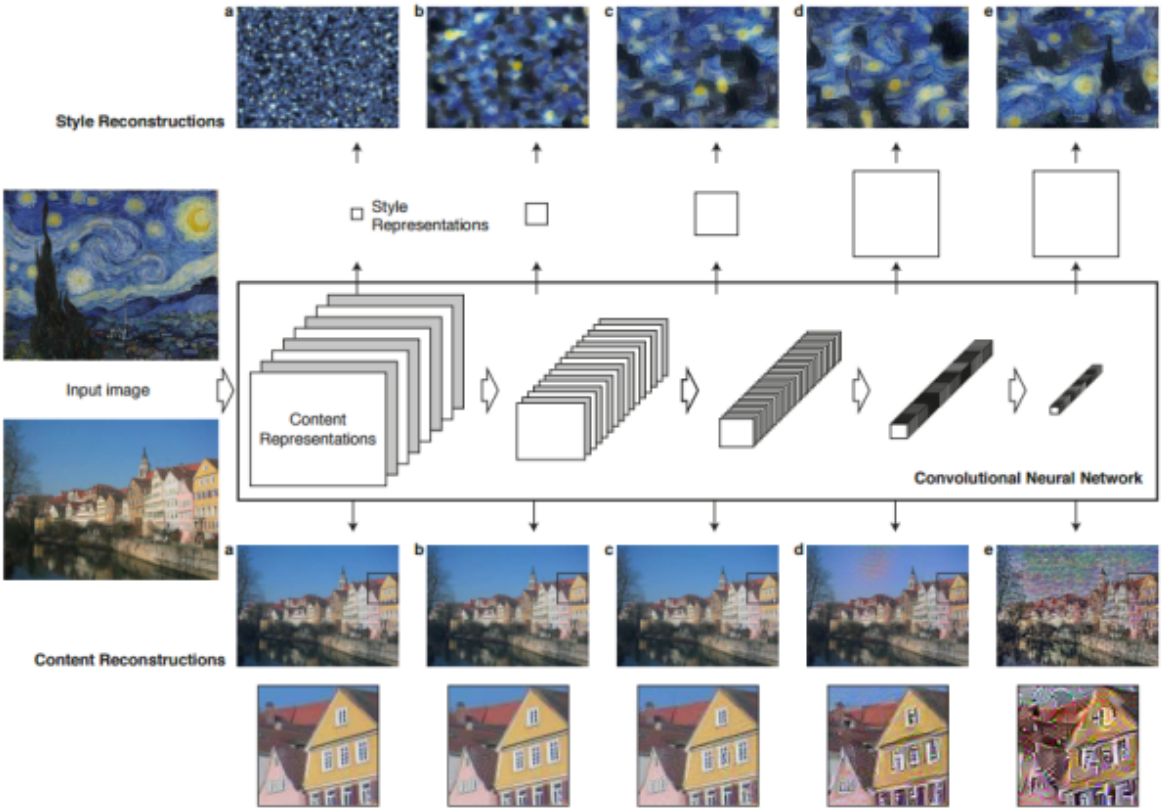


Figura 3.2: Reconstrucciones del Estilo y Contenido utilizando una CNN

coinciden con el estilo de la imagen dada, descartando la información del orden espacial de la escena.

El algoritmo presentado se reduce a un problema de optimización, en el cual se define una función de pérdida que mide el error entre las representaciones de estilo y contenido de la imagen generada, con las respectivas representaciones de las imágenes de entrada. Esta función de pérdida es una combinación lineal de dos funciones de pérdida distintas, una relacionada al contenido y otra al estilo, las cuales se desarrollan en las secciones 3.4 y 3.5 respectivamente. Comenzando desde una imagen de ruido blanco, se va minimizando el valor obtenido por la función de pérdida, iterativamente, mediante el método de descenso por el gradiente, hasta llegar al resultado deseado. De esta forma, tanto el estilo como el contenido de la imagen generada se va aproximando al estilo y al contenido de las respectivas imágenes de entrada.

Al ser un algoritmo iterativo, el número de iteraciones determina el grado de similitud entre la imagen generada con las imágenes elegidas para capturar el contenido y el estilo. La gran influencia de dicho hiper-parámetro radica en que el mismo determina durante cuántos ciclos se ejecutará la minimización de la función de pérdida mediante el método de descenso por el gradiente. Usualmente, a medida que la cantidad de iteraciones es mayor, el valor obtenido por la función de error se irá reduciendo. Ademas, como hemos visto en la sección 2.1.4, dicho método requiere de otro hiper-parámetro llamado tasa de aprendizaje que define el tamaño de cada paso en la iteración. Los hiper-parámetros requeridos por el método de optimización aquí mencionado, son solo algunos de los hiper-parámetros exigidos por el algoritmo. En la siguiente sección se mencionan los demas.

3.4. Representación del Contenido

Generalmente, cada capa de la red define un banco de filtros no lineal cuya complejidad aumenta dependiendo de la jerarquía de la capa en la red. Por lo tanto, una imagen de entrada $\overrightarrow{x}$, se codifica en cada una de las capas de la red convolucional como un mapa de características. Estos mapas se obtienen a partir de las respuestas de los filtros pertenecientes a dicha capa con respecto a la imagen de entrada. Una capa con N_l filtros distintos, tiene N_l mapas de características, cada uno de tamaño M_l, donde M_l es el ancho por el largo del mapa de características. Por consiguiente las respuestas de una capa l pueden ser alojadas en una matriz $F^l \in R^{N_l \times M_l}$, donde $F^l_{i,j}$ es la respuesta del i-esimo filtro en la posición j.

Para visualizar la información codificada en distintas capas de la red, según su jerarquía, se puede ejecutar descenso por el gradiente a partir de una imagen de ruido blanco para encontrar otra imagen que coincida con la respuesta de los filtros de la imagen original. Esto se puede ver en la parte inferior de la Figura 3.2. Sean $\overrightarrow{p}$ y $\overrightarrow{x}$ la imagen original y la imagen generada, y sean P^l y F^l las respectivas representaciones en la capa l. La función pérdida del contenido se define como el error cuadrático medio entre sus mapas de características:

$$L_{contenido}(\overrightarrow{p}, \overrightarrow{x}, l) = \frac{1}{2} \sum_{i,j} (F^l_{i,j} - P^l_{i,j})^2 \tag{3.1}$$

El gradiente con respecto a la imagen $\overrightarrow{x}$ puede ser fácilmente calculado utilizando la retropropagación del error, detallada en 2.2.3 (Figura 3.3 en la parte derecha). Por consecuente, se puede ir adaptando la imagen $\overrightarrow{x}$ hasta que genere la misma respuesta que $\overrightarrow{p}$ en una determinada capa de la red.

3.5. Representación del Estilo

Para obtener la representación del estilo de una imagen, se construye un espacio características sobre las respuestas de los filtros de cualquier capa de la red. Sobre las respuestas de la CNN en cada capa de la red construimos una representación de estilo que calcula las correlaciones entre las diferentes respuestas de los filtros. Estas correlaciones de las características están dadas por la matriz $G^l \in R^{N_l \times N_l}$, en la cual $G^l_{i,j}$ se calcula como el producto punto entre los vectores de los mapas de características i y j en la capa l:

$$G^l_{i,j} = \sum_k F^l_{i,k} F^l_{j,k} \tag{3.2}$$

Nuevamente, se puede visualizar la información capturada por estas características construidas, se puede ejecutar descenso por el gradiente a partir de una imagen de ruido blanco para encontrar otra imagen que coincida con la representación de estilo de la imagen original. Esto se hace minimizando la distancia cuadrática media entre las entradas de la matriz de la imagen original y la matriz de la imagen a generar. En la parte superior de la Figura 3.2 se puede observar la reconstrucción de estilo.

Sean $\overrightarrow{a}$ y $\overrightarrow{x}$ la imagen original y la imagen generada, y sean A^l y G^l las respectivas representaciones de estilo en la capa l, la contribución de esa capa a la función de pérdida total del estilo es:

$$E_l = \frac{1}{4N_l^2 M_l^2} \sum_{i,j} (G^l_{i,j} - A^l_{i,j})^2 \tag{3.3}$$

De esta forma, la función de pérdida del estilo total queda definida como una combinación lineal de las funciones de pérdida de estilo de cada capa:

$$L_{estilo}(\overrightarrow{a}, \overrightarrow{x}) = \sum_{l=0}^{L} w_l E_l \tag{3.4}$$

donde w_l son los factores de peso que tiene cada capa sobre el resultado final, estos factores son introducidos como hiper-parámetros del algoritmo. Los gradientes de E_l con respecto a las activaciones de las capas de la red puede ser fácilmente calculados utilizando la técnica retropropagación del error, explicada en la sección 2.2.3, (Figura 3.3 en la parte izquierda).

3.6. Transferencia de estilo

Para transferir el estilo de una obra de arte $\overrightarrow{a}$ a una fotografía $\overrightarrow{p}$, se genera una nueva imagen la cual simultáneamente deba coincidir con la representación de contenido de $\overrightarrow{p}$ y la representación de estilo de $\overrightarrow{a}$ (Figura 3.3). Es decir, se minimizan en conjunto la distancia del mapa de características de una imagen de ruido con la representación del contenido de la fotografía en una capa y la representación del estilo de la obra de arte. Por lo tanto la función de pérdida total que se minimiza es:

$$L_{total}(\overrightarrow{p}, \overrightarrow{a}, \overrightarrow{x}) = \alpha L_{contenido}(\overrightarrow{p}, \overrightarrow{x}) + \beta L_{estilo}(\overrightarrow{a}, \overrightarrow{x}) \tag{3.5}$$

Esta función combina linealmente las funciones de pérdida del contenido y el estilo, por lo que se conforma por dos términos principales, uno correspondiente a la función de pérdida del contenido y otro correspondiente a la función de pérdida del estilo, donde α y β son los factores de peso para el contenido y el estilo respectivamente, a la hora de combinarlos, los cuales se consideran hiper-parámetros.

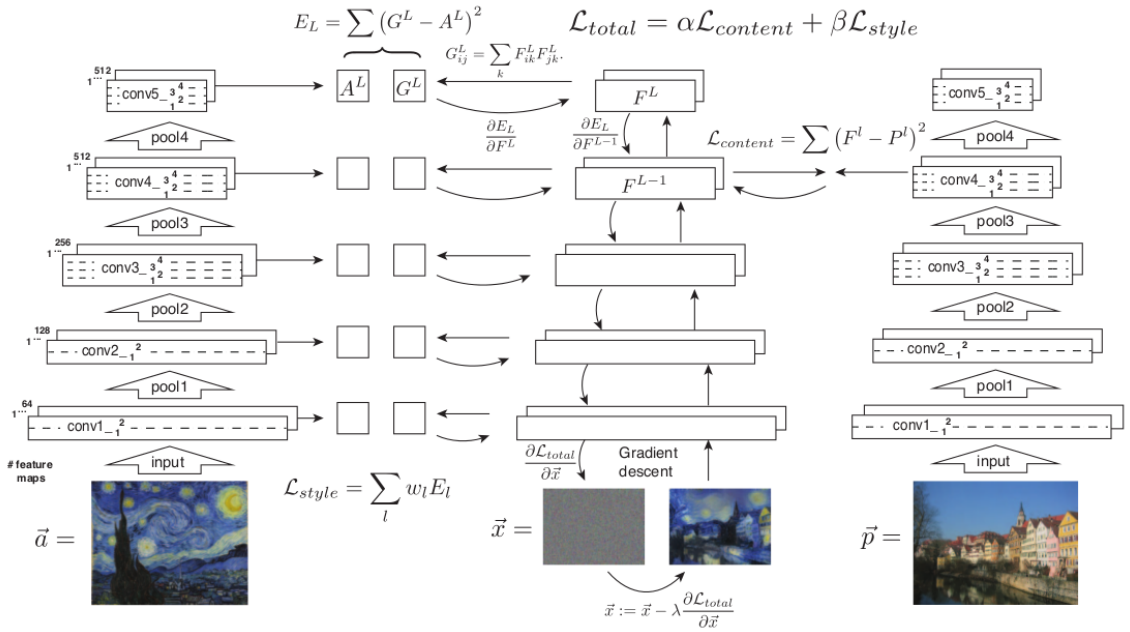


Figura 3.3: Algoritmo de transferencia de estilo

A continuación se realiza una explicación detallada de lo que se muestra en la Figura 3.3.

En primer lugar, las características de contenido y estilo se extraen y almacenan. La imagen de estilo $\overrightarrow{a}$ se pasa a través de la red y sus representaciones de estilo A^l en todas las capas incluidas, son calculadas y almacenadas (a la izquierda). La imagen de contenido $\overrightarrow{p}$ pasa a través de la red y la representación del contenido P^l en una capa se almacena (a la derecha). Luego a partir de una imagen de ruido blanco $\overrightarrow{x}$ pasa a través de la red y sus características de estilo de G^l y sus características de contenido F^l son calculadas.

En cada capa incluída en la representación de estilo, se calcula la diferencia cuadrática media punto a punto entre G^l y A^l para obtener la función de pérdida del estilo L_{estilo} (a la izquierda). También se calcula la diferencia cuadrática media entre F^l y P^l para obtener la función de pérdida del contenido $L_{contenido}$ (a la derecha). La función pérdida total L_{total} es entonces una combinación lineal entre la función de pérdida del contenido y la función de pérdida del estilo. La derivada de L_{total} con respecto a los valores de los píxeles pueden ser calculadas con el método de retropropagación del error (en el medio). Este gradiente se utiliza de forma iterativa actualizando la imagen $\overrightarrow{x}$ hasta que simultáneamente coincida con las características de estilo de la imagen de estilo $\overrightarrow{a}$ y con las características de contenido de la imagen de contenido $\overrightarrow{p}$ (en la parte inferior central).

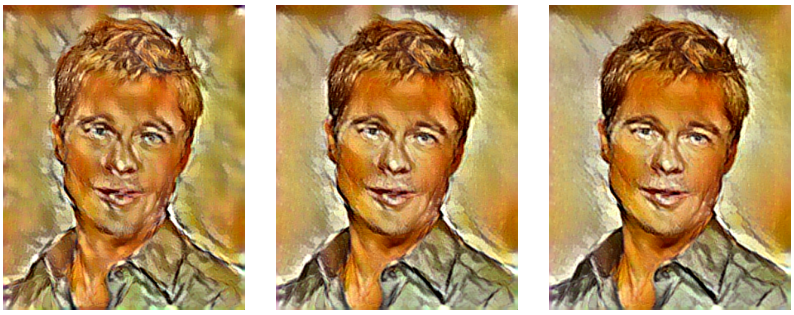
Finalmente, la salida del algoritmo, a partir de lo enunciado en la ecuación (3.5), sera el argumento $\overrightarrow{x}$ que minimice la función de pérdida total, como se puede observar en la ecuación (3.6).

$$\overrightarrow{x} = \arg\min_{\overrightarrow{x}} L_{total}(\overrightarrow{p}, \overrightarrow{a}, \overrightarrow{x}) \tag{3.6}$$

3.7. Hiper-parámetros del algoritmo

A continuación se enumera la lista de hiper-parámetros que requiere el algoritmo para ejecutarse y se discute acerca de su influencia en el mismo:

- Número de iteraciones: Determina cuantas veces se ejecuta el método de descenso por el gradiente para minimizar la función de pérdida. A medida que éste número aumenta, mayores son las posibilidades de obtener un resultado aceptado, como se ilustra en la Figura 3.4.



(a) 100 Iteraciones (b) 200 Iteraciones (c) 300 Iteraciones

Figura 3.4: Ejemplos de obras de arte correspondientes a los estilos seleccionados

- Tasa de aprendizaje: Define el tamaño del paso en cada iteración del descenso por el gradiente. Si esta tasa es muy pequeña, se requieren mayor número de iteraciones, sin embargo si es muy grande, se puede caer en un ciclo donde nunca se termine minimizando la función de pérdida.

- Red Convolucional: Dependiendo de la red convolucional que se utilice, cambia el tamaño de los mapas de características, las capas, etc. Como se menciona en la sección 2.4.1, existen varias CNNs entrenadas para el reconocimiento de objetos, que tienen arquitecturas muy distintas, al igual que el procesamiento requerido para utilizarlas.

- Capas de la red para la representación de estilo y sus respectivos pesos ($w_l, l \in \{1 \dots L\}$): Como hemos vistos anteriormente en la Figura 3.2, la ubicación de las capas en la jerarquía de la red, definen distintas experiencias visuales. Como para la representación de estilo se calcula la correlación entre las capas elegidas, elegir capas intercaladas dentro de toda la jerarquía de la red otorgan una experiencia visual suave y continua. Con respecto a los pesos, por lo general a todas las capas elegidas se les asignada el mismo valor, y a las que no fueron elegidas se les asigna 0.

- Capa de la red para la representación del contenido: En la Figura 3.5 se analiza la elección de la capa *conv*2_2 y la capa *conv*4_2. Al elegir la capa inferior, gran parte de la información detallada de los píxeles en la fotografía y en la imagen generada coinciden (Figura 3.5 al medio). En contraste, al elegir la capa superior, la estructura fina de la imagen se altera, en concordancia con el estilo (Figura 3.5 abajo).

- Factores de peso para las funciones de pérdida de estilo y contenido en la función de pérdida total (α, β): Esto determina el grado de influencia de cada función de pérdida. Un fuerte énfasis sobre el estilo resultara en una imagen que coincida con la apariencia de la obra de arte pero difícilmente se logre observar el contenido de la fotografía ($\alpha/\beta = 1 \times 10^{-4}$ Figura 3.6 arriba a la izquierda). Al enfatizar sobre el contenido, se puede visualizar claramente la fotografía, pero el estilo no coincide completamente con la obra de arte ($\alpha/\beta = 1 \times 10^{-1}$ Figura 3.6 abajo a la derecha).

- Inicialización: A pesar de que en las secciones anteriores, en todo momento se comenzó desde una imagen de ruido, el algoritmo podría partir desde la imagen de contenido o la de estilo. En las primeras iteraciones se puede observar una inclinación hacia la imagen de la cual se comienza (comenzar desde ruido o desde la imagen de estilo generan resultado similares) como muestra la Figura 3.7, pero en el resultado final no termina mostrando una gran influencia.

3.8. Elección Automática de hiper-parámetros para Transferencia de estilo

3.8.1. Introducción

Basado en el algoritmo de transferencia de estilo se pueden obtener resultados muy interesantes. Sin embargo, como se menciona en la sección 3.7 es necesario definir una serie de parámetros previa a la ejecución del mismo. Entre ellos, el número de iteraciones se ha identificado como el hiper-parámetro de mayor importancia en cuanto a la calidad visual de resultado. Es por ello que ha sido seleccionado como el principal hiper-parámetro a automatizarse en este trabajo.

En lo que respecta a las obras de arte, al ser un campo donde la subjetividad tiene gran poder, las métricas suelen ser cualitativas generalmente. Por lo tanto, para poder definir el criterio de aceptación de un resultado generado, es necesario lograr definir una métrica cuantitativa que plasme en algún sentido este tipo de métricas cualitativas. De esta manera, el algoritmo puede definir si aceptar un resultado generado, basándose en el valor obtenido por dicha métrica.

El criterio de elección de este hiper-parámetro termina siendo muy influyente en el resultado final. A medida que el número de iteraciones aumenta, el resultado generado se va aproximando

cada vez más a las representaciones de estilo y contenido de las respectivas imágenes de entrada. No obstante, este número influye notablemente en el tiempo de ejecución requerido por el algoritmo. El objetivo principal de este trabajo realizar un análisis acerca de cómo se comporta el resultado del algoritmo frente a variaciones en el número de iteraciones. De modo que, en caso de detectar una regularidad en el comportamiento del mismo, se podría lograr obtener un resultado aceptado en la menor cantidad de iteraciones posible, reduciendo así el tiempo de ejecución del algoritmo.

3.8.2. Solución propuesta

En este trabajo se plantea definir un criterio de aceptación, de carácter cuantitativo, como criterio de parada del algoritmo iterativo. Si la imagen generada cumple con este criterio, entonces el algoritmo detiene la ejecución y retorna como salida esta misma imagen. En caso contrario, el algoritmo continua iterando hasta obtener una imagen cuyo valor se aproxime al esperado. De esta forma, el algoritmo evita requerir como hiper-parámetro el número de iteraciones que debe ejecutarse, liberando al usuario de esta responsabilidad. La idea consiste en generar una imagen mediante el algoritmo iterativo anteriormente presentado en 3.3, clasificar el estilo de la imagen generada y en base al puntaje obtenido en la clasificación, se decide si continuar o no. La solución consta de 3 módulos principales: un módulo encargado de la generación de la imagen (a partir de una imagen de contenido y otra de estilo), otro módulo encargado de evaluar la imagen generada. Finalmente, un tercer módulo que determina en base al puntaje de clasificación obtenido, si es necesario continuar el proceso de generación, como se puede observar en la Figura 3.8.

El módulo de generación de imágenes implementa el algoritmo propuesto por Gatys et al. [4], para la transferencia de estilo, descripto a lo largo de este capítulo. Dicho algoritmo requiere como entrada una imagen de estilo, una imagen de contenido y una serie de hiper-parámetros mencionados en la sección 3.7. Para el caso de este módulo, se realizará un análisis acerca de la variación del hiper parámetro correspondiente al número de iteraciones, comenzando ya sea desde una imagen de ruido blanco o desde la imagen de contenido. El resto de los hiper parámetros serán fijos, para poder analizar el comportamiento en detalle del mismo. La salida del módulo es una imagen generada donde el contenido se acerca a la imagen de contenido y el estilo se acerca a la imagen de estilo. A medida que aumenta el número de iteraciones, el valor obtenido por la función de error definida en (3.5), entre las imágenes de entrada y la imagen generada se irá reduciendo.

En lo que respecta al módulo de evaluación, como se mencionó en Sec. 3.8.1, una de las limitaciones del proceso de transferencia de estilo es que la evaluación de la imagen resultante es de carácter cualitativo. A fin de establecer una medida cuantitativa y así controlar el resultado de la generación de manera más objetiva, se propone definir un clasificador. Este clasificador se encarga de determinar a cuál estilo artístico pertenece una imagen, brindando un puntaje de clasificación para cada estilo. De esta forma la imagen se evalúa cuantitativamente según el puntaje obtenido en el estilo objetivo. La entrada de este módulo es la imagen generada y la salida es un vector de puntajes de clasificación obtenidos por la imagen para cada uno de los estilos.

Con la idea de definir un clasificador de estilos, se decidió basarse en el algoritmo presentado en [24], el cual involucra técnicas de aprendizaje profundo. Este articulo plantea que los vectores de características obtenidos de CNNs entrenadas para el reconocimiento de objetos, son muy útiles a la hora de reconocer estilos. Aunque muchos estilos parecieran ser principalmente definidos por las elecciones de colores, los vectores de características de las CNNs, proveen mucha más información que los vectores obtenidos a partir de histogramas de colores por ejemplo. Basado en esta idea, decidimos implementar una versión simplificada que consiste en realizar

ajuste fino sobre una CNN preentrenada para el reconocimiento de objetos. De esta forma se van realizando evaluaciones de la imagen de salida obtenido por el algoritmo generador contra la predicción que realiza el modelo reentrenado para el reconocimiento de estilos.

Finalmente, el módulo de selección, es el encargado de determinar si se acepta la imagen obtenida en la iteración actual o es necesario seguir iterando. La entrada de este módulo es el vector de puntajes de clasificación calculados por el módulo generador, y la salida es un valor binario indicando si se acepta o no la imagen.

El criterio de selección que se propone para elegir la imagen óptima es el siguiente: Basado en la hipótesis de que el resultado generado por el módulo generador obtenga un puntaje similar que la imagen de estilo en el módulo evaluador, se define una tolerancia del $\pm 20\,\%$ con respecto al puntaje obtenido por la imagen de estilo, en caso de que el resultado obtenga un puntaje que pertenezca a este rango, la imagen es aceptada como resultado final, en caso contrario el generador debe seguir iterando hasta obtener un resultado que cumpla con este requisito.

Figura 3.5: Comparación de distintas capas de la red para la representación del contenido. La capa *conv*2_2 es una de las capas inferiores y la capa *conv*4_2 es una de las capas superiores.

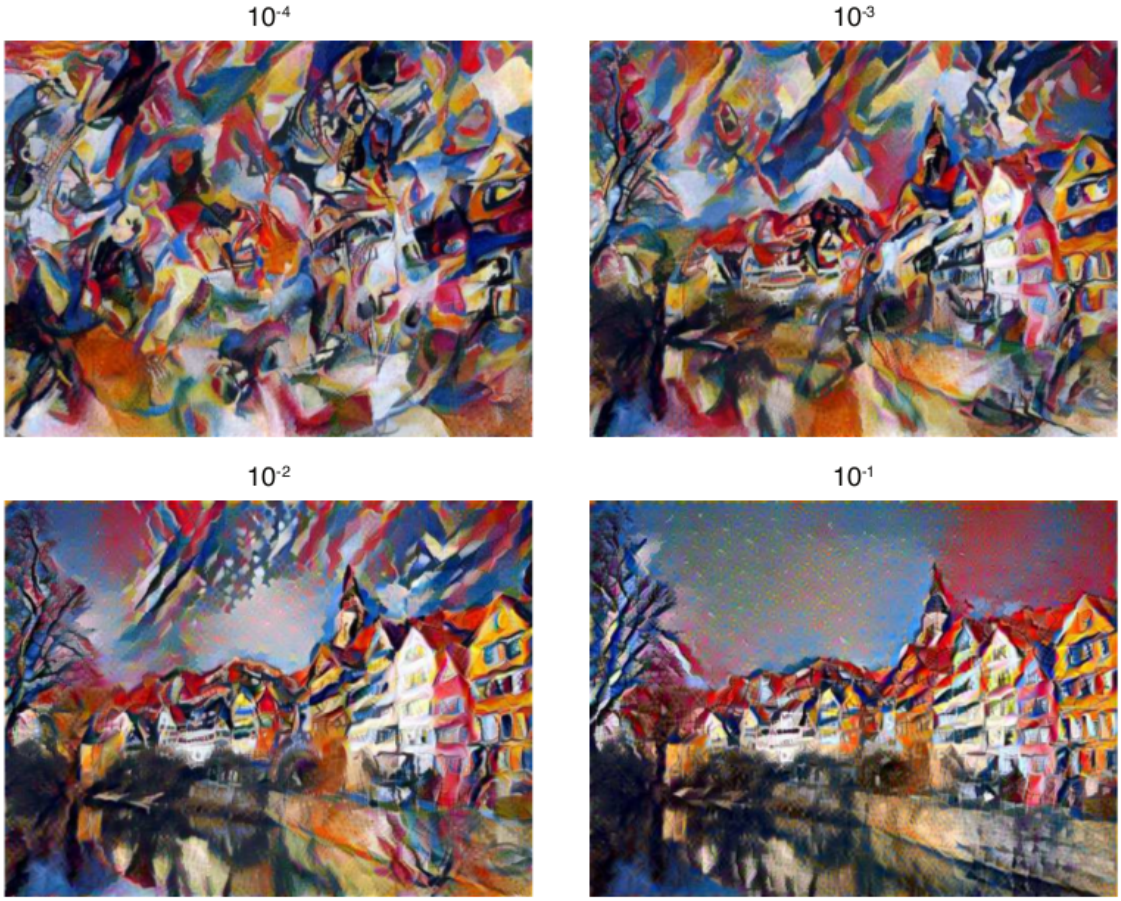
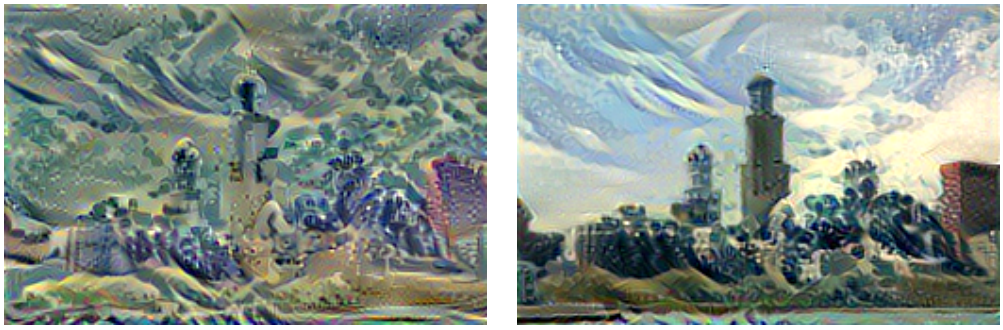


Figura 3.6: Comparación de los distintos ratios entre el peso del estilo y el contenido en la función de pérdida total



(a) Imagen generada a partir de ruido luego de 100 iteraciones

(b) Imagen generada a partir de la fotografía luego de 100 iteraciones

Figura 3.7: Inicialización del algoritmo

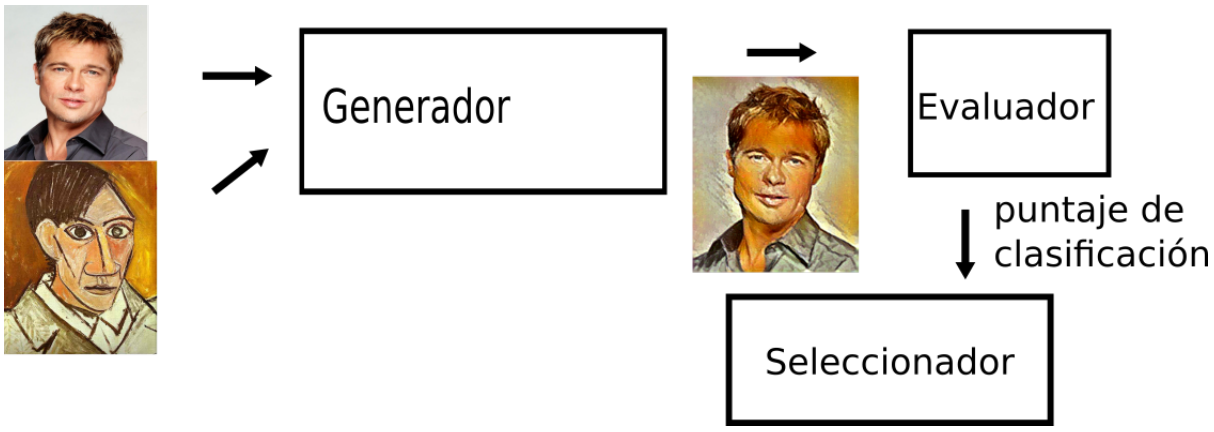


Figura 3.8: Diagrama de interacción entre los distintos módulos propuestos

Capítulo 4

Experimentos

En este capítulo se exponen los experimentos realizados basados en la solución propuesta en la sección 3.8.2. Para ello se realizaron dos experimentos principales: El primero consiste en crear un módulo de reconocimiento de estilos artísticos realizando ajuste fino sobre una CNN preentrenada para el reconocimiento de objetos. En el segundo experimento se realiza el análisis empírico para la elección del número de iteraciones mediante el criterio de selección propuesto.

4.1. Reconocimiento de estilo

Para implementar el clasificador de estilos se decidió utilizar una CNN, debido a su buen rendimiento en la clasificación de imágenes. Al no existir de forma pública una red entrenada para reconocer estilos artísticos, fue necesario realizar ajuste fino sobre una CNN preentrenada para el reconocimiento de objetos. Para realizar el ajuste fino, es necesario definir la red sobre la cual se realizará, además del conjunto de datos que le será provisto a la misma. A lo largo de esta sección se detallan las características de la implementación de este modulo.

4.1.1. Conjunto de datos

El conjunto de datos utilizado para en este trabajo, se construye como un subconjunto a partir de WikiArt [1]. Este conjunto proviene de un proyecto de disponibilidad pública que tiene como objetivo principal hacer accesible el mundo del arte a cualquier persona y en cualquier lugar. WikiArt, actualmente cuenta con unas 150.000 obras de arte, de 2.500 artistas aproximadamente. La obras contenidas en este conjunto se encuentran en museos, universidades y otros edificios civiles de más de 70 países. Aunque la mayoría de estas obras de arte no está disponible a la vista pública.

Para construir el conjunto, se utiliza un subconjunto de los 10 estilos que mayor cantidad de obras de arte disponen. Luego, se ejecuta una etapa de recolección de datos. Inicialmente es necesario definir los 10 principales estilos, para lo cual se eligen aquellos que dispongan de mayor cantidad de obras. Posteriormente se debe adquirir localmente los archivos de los estilos elegidos, verificando que estén en condiciones de ser utilizados. Al finalizar este proceso, el conjunto de datos generado queda conformado por 68807 imágenes.

A continuación se enumeran los estilos contemplados:

1. Impresionismo (14433 imágenes disponibles)
2. Realismo (13595 imágenes disponibles)

[1] `https://www.wikiart.org/`

3. Romanticismo (10563 imágenes disponibles)
4. Expresionismo (9747 imágenes disponibles)
5. Post Impresionismo (7226 imágenes disponibles)
6. Surrealismo (6636 imágenes disponibles)
7. Modernismo (5969 imágenes disponibles)
8. Barroco (5156 imágenes disponibles)
9. Simbolismo (4689 imágenes disponibles)
10. Neo Clasisimo (3306 imágenes disponibles)

A modo de ejemplo, en la Figura 4.1 se muestra una imagen correspondiente a cada uno de los estilos elegidos.

4.1.2. Red neuronal y detalles de implementación

Para realizar el ajuste fino, sobre este conjunto de datos generado, se utilizo la arquitectura de la red AlexNet [10], reemplazando la ultima capa por una capa que aprenda a reconocer entre los posibles estilos. Los valores con los que se inicializo la red, eran los valores obtenidos al entrenar AlexNet para el reconocimiento de objetos de ImageNet. Para reentrenar la red ajustando los valores para el reconocimiento de estilo, se le proveyó el conjunto de datos generado, y se definió realizar 50000 iteraciones para el ajuste fino. Para realizar el ajuste fino se utilizó las librerías Caffe y OpenCV en Python, y para el reentrenamiento fue requerido el uso de una GPU GTX 680.

4.2. Elección del número de iteraciones empleando el módulo de reconocimiento

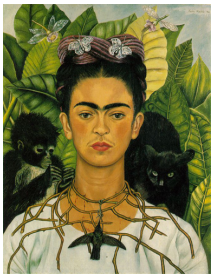
Este experimento consiste en realizar un análisis empírico de la evolución de los puntajes de clasificación de estilo a medida que aumenta el número de iteraciones que se ejecuta el módulo de generación de imágenes.

4.2.1. Conjunto de datos

Para ejecutar el procedimiento, es necesario disponer de dos conjuntos de imágenes, uno correspondiente a las imágenes de contenido y otro correspondiente a las imágenes de estilo. Para el conjunto de imágenes de contenido fueron elegidas cinco imágenes, de las cuales dps son retratos de personas y las tres restantes son fotografías de paisajes. Con respecto al conjunto de imágenes de estilo, para cada uno de los estilos fueron elegidas tres imágenes de forma aleatoria.

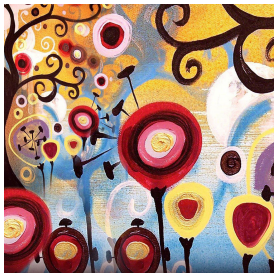
4.2.2. Detalles de implementación

El proceso consistió en aplicar el algoritmo de transferencia de estilo combinando cada una de las fotos del conjunto de contenido con cada una de las fotos del conjunto de estilo, y guardando el resultado cada 10 iteraciones, hasta llegar a las 1000 iteraciones para cada uno de los procesos particulares. Además para cada combinación de imagen de estilo e imagen de contenido el algoritmo se ejecuto dos veces, una comenzando desde la imagen de contenido y



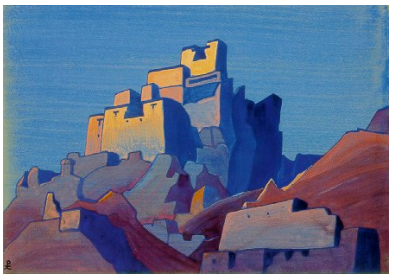
(a) Modernismo

(b) Impresionismo



(c) Surrealismo

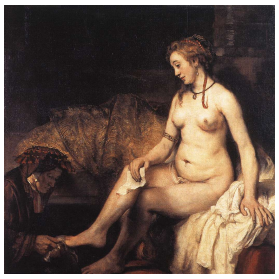
(d) Post Impresionismo



(e) Simbolismo

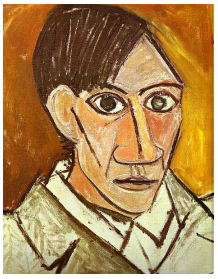
(f) Neo Clasisimo

(g) Romanticismo



(h) Barroco

(i) Realismo



(j) Expresionismo

Figura 4.1: Ejemplos de obras de arte correspondientes a los estilos seleccionados

otra comenzando desde una imagen de ruido. Finalmente todo el conjunto de imágenes obtenido es evaluado por el módulo de reconocimiento de estilo, que otorga un puntaje de la imagen para cada uno de los estilos considerados.

Algoritmo de Transferencia de estilo

Existen una gran variedad de implementaciones de código abierto que implementan el algoritmo de transferencia de estilo definido por Gatys. Se decidió basarse en la implementación de Justin Johnson [25], la cual posee el mayor grado de aceptación en la industria. La principal tecnología utilizada en esta implementación es el entorno llamado Torch, ampliamente utilizado para cuestiones relacionadas con aprendizaje profundo. Esta implementación permite además controlar algunas otras variables mediante hiper-parámetros extras a lo requerido por el algoritmo originalmente, como por ejemplo el método de optimización a utilizar y tamaño de la imagen generada.

4.2.3. Análisis del comportamiento frente a variaciones en el número de iteraciones

Luego de generar distintas imágenes para cada uno de los estilos, variando entre imágenes de retratos y paisajes como imágenes de contenido, se realizó un análisis de la evolución de los puntajes obtenidos por estas imágenes, a medida que el número de iteraciones va aumentando. En esta sección se presentan algunos casos de ejemplo que permiten ilustrar lo sucedido en la mayoría de los casos del conjunto de datos generado.

El primer caso presentado en la Figura 4.2 es el más simple, como imagen de estilo se utiliza The Shipwreck of the Minotaur, Joseph Mallord William Turner, 1805 (a), el cual pertenece al Romanticismo, y como imagen de contenido se utiliza una fotografía de un paisaje de la ciudad de Tübingen, Alemania (b). En la imagen (c) se puede observar el resultado obtenido partiendo desde una imagen de ruido, luego de 620 iteraciones y en la imagen (d) se puede observar el resultado partiendo desde la imagen de contenido, luego de 600 iteraciones. Estas cantidades de iteraciones fueron elegidas con el criterio al término de esta sección, basado en la experiencia y los resultados obtenidos a partir de todos los experimentos realizados. La diferencia entre ambos resultados en este caso es sutíl, pero al comenzar desde la imagen de contenido, los bordes y contornos de esta, son más notorios en el resultado que al comenzar desde ruido.

(a) Imagen de Estilo: Shipwreck, Turner, 1805

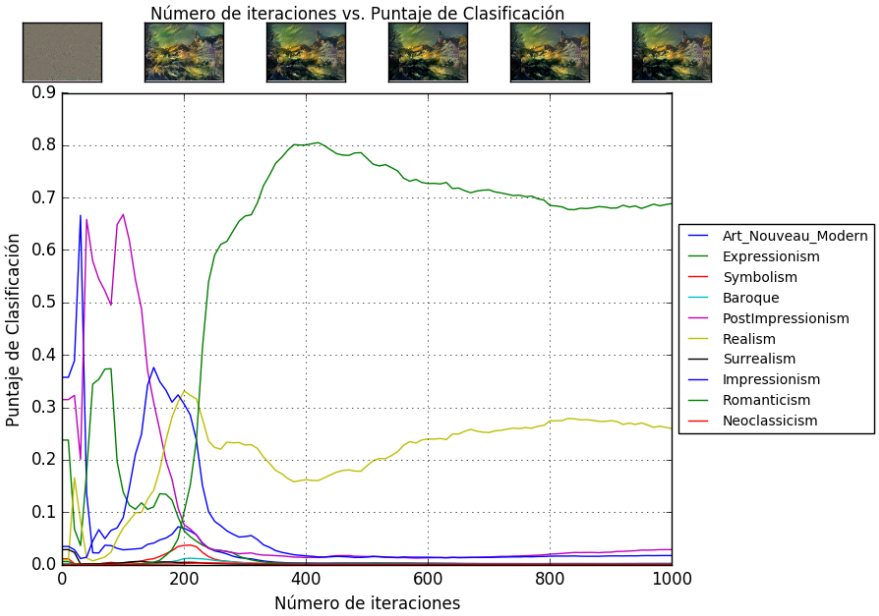
(b) Imagen de Contenido: Fotografía de Tübingen

(c) Resultado comenzando desde ruido

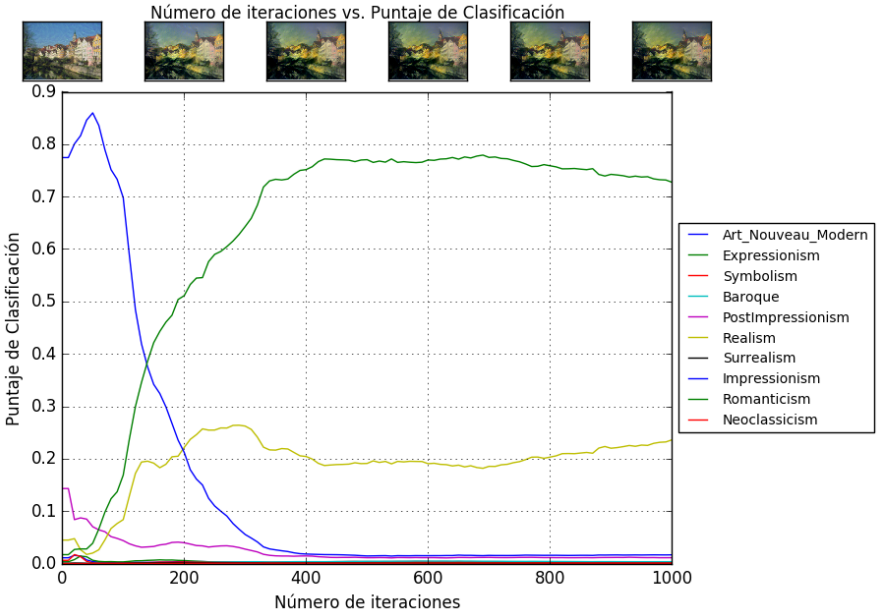
(d) Resultado comenzando desde el contenido

Figura 4.2: Ejemplos de obras de arte correspondientes a los estilos seleccionados

En cuanto a la evolución de los puntajes de clasificación, como se puede observar en la Figura 4.3(a), en este caso el estilo predicho para el resultado coincide con el estilo de la imagen de estilo (Romanticismo). Al partir desde una imagen de ruido, en las primeras 200 iteraciones el clasificador no logra reconocer un claro estilo, pero luego se va normalizando hasta obtener una predicción confiable, ya que la diferencia con los otros estilos es notoria. Lo mismo ocurre comenzando desde la imagen de contenido, ilustrado en 4.3(b), en ambos los gráficos generados son similares.



(a) Evolución de los puntajes comenzando desde ruido

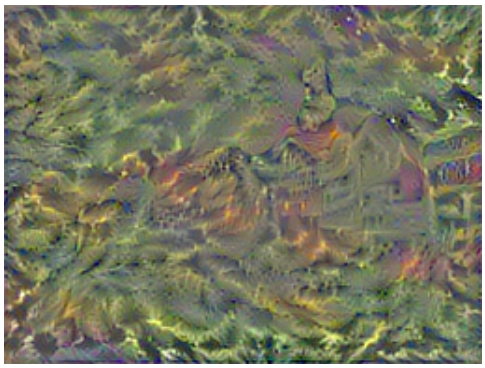


(b) Evolución de los puntajes comenzando desde contenido

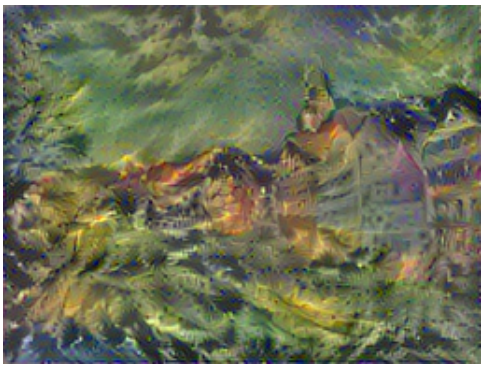
Figura 4.3: Evolución de los puntajes

Con respecto a la calidad de la imagen, en la Figura 4.4, se puede observar cómo va incrementando la calidad de la imagen a medida que aumenta el número de iteraciones en las cuales se ejecuta el algoritmo. Durante las iteraciones iniciales, el cambio es notorio. Al comenzar, en 50 iteraciones (a), todavía parece ser una imagen de ruido. En 100 iteraciones (b) ya se comienzan a visualizar destellos de estilo y contenido. Ya a partir de los 200 (c) ciclos de ejecución, la imagen comienza a ser de una calidad visual aceptable. Finalmente, luego de 600 ciclos (d), la heurística elige esta imagen como óptima. No obstante, las diferencias entre el resultado generado entre las 200 y las 600 iteraciones son de menor impacto visual que en las

primeras iteraciones.



(a) Imagen generada en 50 iteraciones



(b) Imagen generada en 100 iteraciones

(c) Imagen generada en 200 iteraciones

(d) Imagen generada en 600 iteraciones, elegida como óptima

Figura 4.4: Análisis de la calidad de la imagen generada según el número de iteraciones

Como segundo caso de análisis se eligió una situación en la que la predicción de la imagen resultante coincide con el estilo de la imagen de estilo provista, si se comienza el procedimiento desde la imagen de ruido, pero esto no ocurre si se comienza desde la imagen de contenido. Para este caso exhibido en la Figura 4.5, como imagen de estilo se utilizó El grito, de Edward Munch, 1893 correspondiente al Expresionismo (a) y como imagen de contenido una fotografía del Golden Gate, un icono de la ciudad de San Francisco, Estados Unidos (b). En la imagen (c) se muestra el resultado partiendo desde ruido y en la (d) desde la imagen de contenido, ambas luego de 540 iteraciones (mismo criterio de selección que el caso anterior), desde el aspecto visual estos resultados son muy similares, pero para el clasificador son muy distintos como luego de muestra en la Figura 4.6.

(a) Imagen de Estilo: The Scream

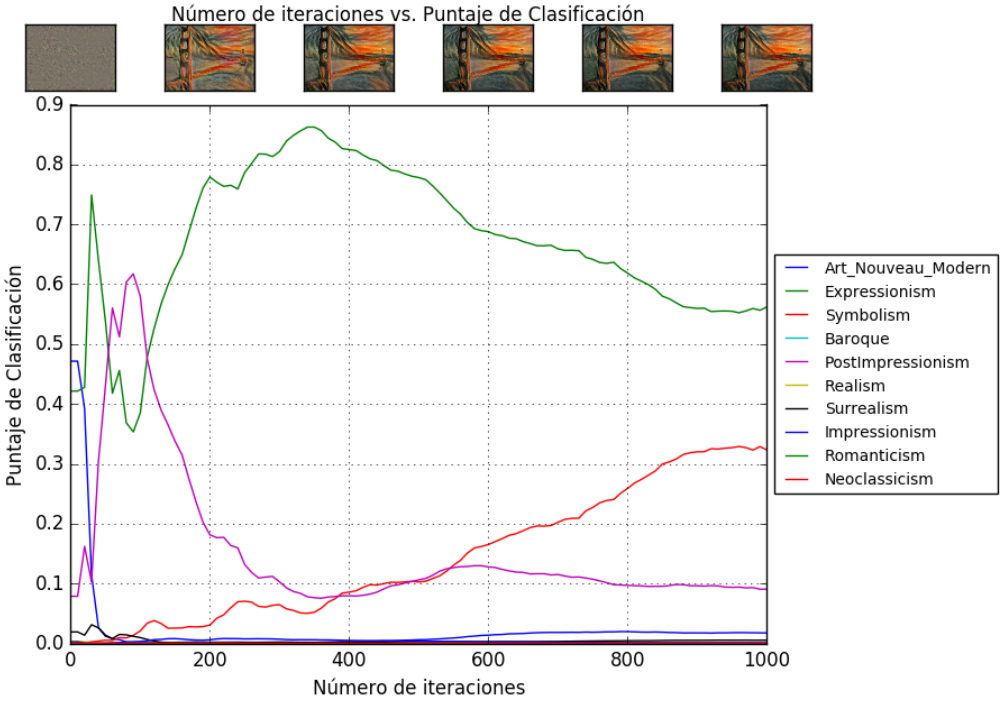
(b) Imagen de Contenido: Fotografía del Golden Gate

(c) Resultado comenzando desde ruido

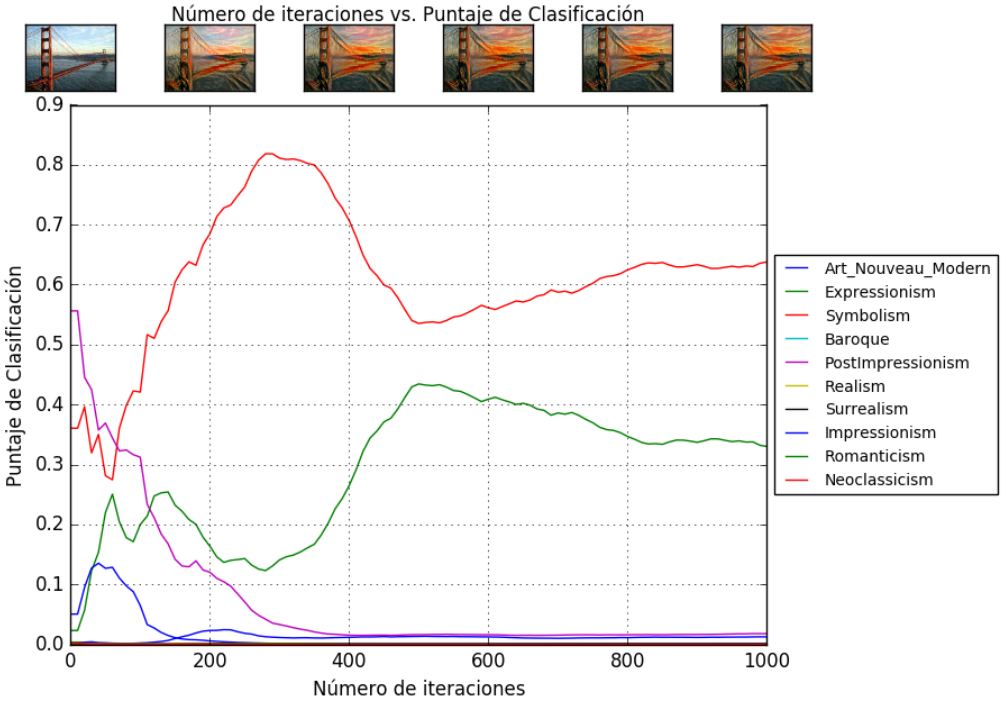
(d) Resultado comenzando desde el contenido

Figura 4.5: Ejemplos de obras de arte correspondientes a los estilos seleccionados

Al analizar la evolución de los puntajes predichos por el clasificador nos encontramos con que el puntaje máximo de clasificación varía dependiendo de la imagen en la cual comienza el proceso de transferencia de estilo, como se puede ver en la Figura 4.6, en el gráfico (a) se muestra la evolución de los puntajes partiendo de la imagen de ruido, que luego de las 200 iteraciones se termina normalizando al Expresionismo como predicción, lo cual coincide con la imagen de estilo provista. Sin embargo, en el gráfico (b) se muestra la evolución de los puntajes de predicción partiendo de la imagen de contenido, en la cual se exhibe al Simbolismo como predicción mayoritaria.



(a) Evolución de los puntajes comenzando desde ruido

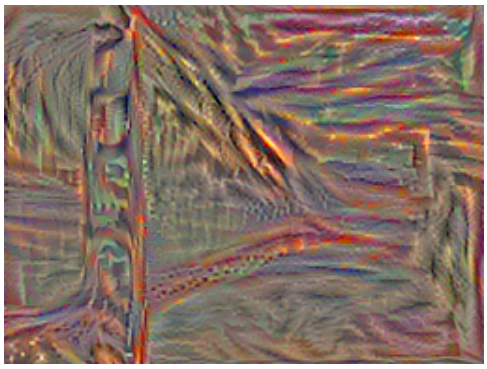


(b) Evolución de los puntajes comenzando desde contenido

Figura 4.6: Evolución de los puntajes

Con respecto a la calidad de la imagen, en el cuadro 4.7 se puede observar que sucede algo similar a lo detallado en la Figura 4.4. Esto mismo ocurre en el resto de los experimentos realizado, generalmente, desde el comienzo hasta las 200 iteraciones aproximadamente es donde las

mejoras son notorias. Luego de esa cantidad de ciclos, se efectúan las mejoras de menor impacto visual. A pesar de que estas mejoras en el resultado generado no se visualizan claramente, en algunos casos asisten al clasificador a realizar una predicción más precisa y confiable.



(a) Imagen generada en 50 iteraciones

(b) Imagen generada en 100 iteraciones

(c) Imagen generada en 200 iteraciones

(d) Imagen generada en 540 iteraciones, elegida como óptima

Figura 4.7: Análisis de la calidad de la imagen generada según el número de iteraciones

Análisis del criterio de selección

Luego se realizar un análisis en varios gráficos como los presentados en la sección anterior se concluye que la hipótesis sobre que el puntaje de clasificación de la imagen de estilo y el puntaje de la imagen generada coinciden es falsa. Esto no ocurre en todos los casos analizados. Dependiendo de la imagen de contenido el estilo que obtiene mayor puntaje de clasificación puede ser distinto al estilo de la imagen de estilo. Dicha situación ocurre dado que la imagen de contenido puede influir en el puntaje de clasificación de estilo de la imagen generada, aunque en mucho menor proporción respecto a la imagen de estilo. Este resultado avala la idea que presenta Gatys acerca de que el estilo y el contenido de una imagen no son absolutamente separables.

Criterio de Selección: debido a lo expuesto en el párrafo anterior, el criterio de selección que se definió para elegir la imagen óptima fue el siguiente: el estilo objetivo, en lugar de ser el estilo correspondiente a la imagen de estilo provista para la transferencia de estilo, se definió como el estilo que tiene asignada una mayor probabilidad de reconocimiento luego de 1000 iteraciones. Una vez definido el estilo objetivo, se definen los intervalos donde el puntaje obtenido por el estilo objetivo supera al puntaje del resto de los estilos. La imagen óptima elegida es la correspondiente al número de iteraciones correspondiente al punto medio del intervalo donde el estilo objetivo es el máximo. En caso de que exista más de un intervalo que cumpla

con este requisito, se devuelve una imagen óptima de cada intervalo como posibles imágenes óptimas para que el usuario elija cual es la óptima según su criterio cualitativo.

Capítulo 5

Conclusiones, Perspectivas y trabajos a futuro

5.1. Conclusiones

En este trabajo se presenta la idea de evaluar los resultados generados por el algoritmo de transferencia presentado por Gatys et al. [4], mediante una red neuronal que clasifica estilos artísticos. En base al puntaje obtenido en la clasificación, se determina la calidad de la imagen generada. Esto permite automatizar la elección del número de iteraciones, uno de los hiper parámetros que requiere el algoritmo de transferencia de estilo. En el presente trabajo se utilizan 2 redes neuronales convolucionales: una para la transferencia de estilo y otra para la clasificación de los estilos. Esta última, inicialmente había sido entrenada para clasificación de objetos y luego mediante ajuste fino se reentrenó para clasificar estilos artísticos. Dado que las obras de arte suelen ser calificadas cualitativamente, era necesario definir una métrica cuantitativa para poder establecer la calidad de la imagen obtenida. La métrica que empleada para definir la calidad de una imagen generada fue el puntaje obtenido por ésta, al ser evaluada en la red de clasificación de estilos. Al establecer una métrica cuantitativa de evaluación, el algoritmo libera al usuario de la responsabilidad de definir el número de iteraciones, uno de los principales factores de influencia en la imagen generada.

Los resultados obtenidos no fueron los esperados, dado que el estilo de la imagen generada no depende solo de la imagen de estilo sino tambien de la imagen de contenido, con lo cual, en algunos casos, el puntaje obtenido no era comparable con el puntaje de la imagen de estilo. No obstante, se observó que evaluar la imagen generada con el clasificador de estilo es un buen indicio de la calidad de la misma. Cuando los puntajes obtenidos por la imagen son bajos, visualmente, no es una imagen de calidad. A medida que los puntajes van aumentando, la imagen suele ser cualitativamente aceptada. En casos donde la gráfica el puntaje obtenido a lo largo de las iteraciones entra en una meseta, es decir no mejora a medida que aumenta el número de iteraciones, tampoco se logran visualizar grandes cambios a nivel cualitativo. Esto indica que en cierta forma, que la evaluación cualitativa logró ser plasmada en la métrica cuantitativa definida.

5.2. Trabajos a futuro

Dado que es un area que ha tenido mucha investigación en el último tiempo, es posible seguir profundizar el trabajo realizado, en distintos aspectos:

Reconocimiento de estilos Basándose en la misma idea de entrenar una red convolucional para reconocer solo 10 estilos, se podría extender esto a todos los estilos y utilizando todas las imágenes disponibles en WikiArt.

Automatización de hiper parámetros Se podría realizar una especie de predefinición del número de iteraciones que requiere cada estilo, ejecutando el algoritmo una gran cantidad de veces y definiendo este valor estadísticamente, aunque esto puede llegar a no funcionar correctamente en todos los casos ya que dependiendo de la imagen de contenido, el estilo objetivo podría variar.

Compresión de CNN A partir del articulo presentado por Gatys, se fueron publicando una serie de artículos que presentaban mejoras u optimizaciones al algoritmo, además de buscar comprimir la red, para que una vez entrenada en un dispositivo de gran capacidad de cómputo, pueda ser insertado y ejecutado para transferir estilo en dispositivo menor como lo son los dispositivos móviles. Por ejemplo se lograron entrenar modelos que transfieran el estilo de una obra de arte en particular, que puede ser ejecutado en un dispositivo móvil [15, 16]. Además ya existen algunos otros avances para lograr ejecutar una red neuronal en dispositivos móviles: [26].

Bibliografía

[1] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[2] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[3] Richard Szeliski. *Computer Vision: Algorithms and Applications*. Springer-Verlag New York, Inc., New York, NY, USA, 1st edition, 2010.

[4] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. A neural algorithm of artistic style. *CoRR*, abs/1508.06576, 2015.

[5] Thomas M. Mitchell. *Machine Learning*. McGraw-Hill, Inc., New York, NY, USA, 1 edition, 1997.

[6] Gerard Salton and Michael J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, Inc., New York, NY, USA, 1986.

[7] Andrej Karpathy. Stanford university CS231n: Convolutional neural networks for visual recognition, 2016.

[8] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[9] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. In *Intelligent Signal Processing*, pages 306–351. IEEE Press, 2001.

[10] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States.*, pages 1106–1114, 2012.

[11] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Computer Vision and Pattern Recognition (CVPR)*, 2015.

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

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[14] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167, 2015.

[15] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European Conference on Computer Vision*, 2016.

[16] Dmitry Ulyanov, Andrea Vedaldi, and Victor S. Lempitsky. Instance normalization: The missing ingredient for fast stylization. *CoRR*, abs/1607.08022, 2016.

[17] Fujun Luan, Sylvain Paris, Eli Shechtman, and Kavita Bala. Deep photo style transfer. *arXiv preprint arXiv:1703.07511*, 2017.

[18] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. Texture synthesis and the controlled generation of natural stimuli using convolutional neural networks. *CoRR*, abs/1505.07376, 2015.

[19] Jan Eric Kyprianidis, John P. Collomosse, Tinghuai Wang, and Tobias Isenberg. State of the .ªrt": A taxonomy of artistic stylization techniques for images and video. *IEEE Trans. Vis. Comput. Graph.*, 19(5):866–885, 2013.

[20] Aaron Hertzmann, Charles E. Jacobs, Nuria Oliver, Brian Curless, and David H. Salesin. Image analogies. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '01, pages 327–340, New York, NY, USA, 2001. ACM.

[21] Michael Ashikhmin. Fast texture transfer. *IEEE Computer Graphics and Applications*, 23(4):38–43, 2003.

[22] 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*, SIGGRAPH '01, pages 341–346, New York, NY, USA, 2001. ACM.

[23] Hochang Lee, Sanghyun Seo, Seungtaek Ryoo, and Kyunghyun Yoon. Directional texture transfer. In *Proceedings of the 8th International Symposium on Non-Photorealistic Animation and Rendering*, NPAR '10, pages 43–48, New York, NY, USA, 2010. ACM.

[24] Sergey Karayev, Aaron Hertzmann, Holger Winnemoeller, Aseem Agarwala, and Trevor Darrell. Recognizing image style. *CoRR*, abs/1311.3715, 2013.

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

[26] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *CoRR*, abs/1704.04861, 2017.