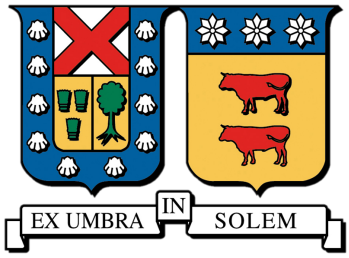
UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA

DEPARTAMENTO DE ELECTRÓNICA
VALPARAÍSO - CHILE



"DISEÑO E IMPLEMENTACIÓN DE INSTALACIÓN INTERACTIVA DE ARTE DIGITAL UTILIZANDO TÉCNICAS DE INTELIGENCIA ARTIFICIAL"

FRANCISCO IGNACIO FREZ ROJAS

MEMORIA DE TITULACIÓN PARA OPTAR AL TÍTULO DE INGENIERO CIVIL ELECTRÓNICO MENCIÓN COMPUTADORES

PROFESOR GUÍA: DR. MARÍA JOSÉ ESCOBAR

PROFESOR CORREFERENTE: DR. MAURICIO ARAYA

JUNIO - 2021

A mi Señor y Salvador Jesucristo que me ha dado
la vida y todo lo que tengo.

Resumen

El continuo avance de la tecnología mantiene en constante cambio la forma en que las personas se comunican, se informan, se relacionan y, posiblemente, la propia percepción de la vida. El arte, por su puesto, también se ha visto sacudido por esta metamorfosis. Los artistas día a día inventan nuevas formas de crear, expresar y vivir el arte y, de la mano de la tecnología, este abanico de posibilidades aumenta cada vez más. En la actualidad, es posible ver exhibiciones artísticas puramente tecnológicas, e indudablemente, la inteligencia artificial, de la mano del machine learning y el deep learning, es una parte importante del motor que permite la creación de este tipo de intervenciones. En este trabajo se propone la creación de una experiencia nueva e innovadora, se plantea introducir la inteligencia artificial como fenómeno artístico y cultural, desarrollando una instalación interactiva de arte digital basada en redes neuronales convolucionales que permitan transferir el estilo artístico de pinturas de Vincent van Gogh a videos breves de no más de 8 segundos. A este proceso se le conoce como transferencia de estilo neuronal, y en la instalación interactiva se utiliza el enfoque basado en optimización de modelos neuronales. Debido al costo computacional que conlleva procesar un video, se proponen distintas optimizaciones al modelo para lograr buenos resultados cualitativos a una mayor velocidad de inferencia.

El desarrollo de este proyecto se lleva a cabo en conjunto con la startup Electroveja Labs, y se sitúa en el contexto de la renovación del Museo Artequin Viña del Mar y la firma de un convenio de colaboración con la Universidad Técnica Federico Santa María de disponer de un espacio en el museo para la exposición y divulgación de la ciencia y la tecnología a niños y jóvenes de la región de Valparaíso. Bajo este contexto, el nivel de desarrollo tecnológico de este proyecto se encuentra a la altura de un producto mínimo viable, es decir, es una versión inicial del producto final capaz de ofrecer el mismo valor al cliente. El desarrollo de este proyecto sigue una metodología iterativa modular para facilitar la implementación de trabajos futuros y la búsqueda de un producto final

validado técnica y comercialmente. Al finalizar, la instalación interactiva de arte digital es montada en las dependencias del Museo Artequin bajo el nombre de “Van Gogh te pinta” quedando en exposición de forma permanente.

Palabras clave: Transferencia de estilo neuronal (NST), deep learning, redes neuronales convolucionales, visión por computador, TensorFlow, Unity, arte, cultura, instalación interactiva de arte digital.

Abstract

The continuous advancement of technology keeps in constant change the way in which people communicate, inform, relate, and possibly, their own conception of life. Of course, art has also been shaken by this metamorphosis. Every day artists invent new ways of creating, expressing and living art and, along with technology, this range of possibilities increases more and more. At present, it is possible to see purely technological artistic exhibitions, and undoubtedly, artificial intelligence, along with machine learning and deep learning, is an important part of the engine that allows the creation of this type of interventions. In this work, it's proposed the creation of a new innovative experience, introducing artificial intelligence as an artistic and cultural phenomenon, developing an interactive installation of digital art based on convolutional neural networks that allow to transfer the artistic style of Vincent van Gogh's paintings to short videos of no more than 8 seconds. This process is known as neural style transfer, and the interactive installation uses the approach based on optimization of neural models. Due to the computational cost involved in video processing, different optimizations are proposed to the model to achieve good qualitative results at a higher speed of inference.

The development of this project is carried out in conjunction with the startup Electroveja Labs, and is situated in the context of the renovation of the Museo Artequin Viña del Mar and the signing of a collaboration agreement with the Universidad Técnica Federico Santa María to have a space in the museum for the exhibition and dissemination of science and technology to children and young people in the Valparaíso region. In this context, the level of technological development of this project is at the level of a minimum viable product, i.e. an initial version of the final product capable of offering the same value to the client. The development of this project follows a modular iterative methodology to facilitate the implementation of future work and the search for a technically and commercially validated final product. At the end, the interactive installation of digital art is mounted in the Museo Artequin under the name of "Van Gogh te pinta",

remaining on permanent display.

Keywords: Neural style transfer (NST), deep learning, convolutional neural networks, computer vision, TensorFlow, Unity, art, culture, interactive digital art installation.

Glosario

Backpropagation Es un algoritmo que permite calcular numéricamente el gradiente de la función de pérdida con respecto a cada uno de los pesos de la red. También llamado retropropagación.

Campo Receptivo En un contexto de aprendizaje profundo, se define como el tamaño de la región en la entrada que produce una característica. Básicamente, es una medida de asociación entre una característica de salida (de cualquier capa) con la respectiva región de entrada (parche).

Desarrollo Humano Se define como el proceso de ampliar la riqueza de la vida humana, en lugar de simplemente la riqueza de la economía en la que viven los seres humanos. Es un enfoque que se centra en la creación de oportunidades y opciones justas para todas las personas.

Downsampling Es la reducción de la resolución espacial mientras se mantiene la representación 2D de una imagen.

Electroveja Labs startup tecnológica nacida en la UTFSM cuya misión es "ser una empresa reconocida a nivel mundial por incentivar e impulsar el interés por la creatividad y el conocimiento mediante vivencias sensoriales inmersivas donde la tecnología y el arte estén unificados y al servicio de la sociedad".

Gradient Descent Es un algoritmo de optimización iterativo de primer orden que permite encontrar un mínimo local de una función diferenciable. Este algoritmo permite ajustar los pesos de una red neuronal para minimizar su función de pérdida. También llamado descenso del gradiente.

Instalación de Arte Digital El arte de instalación es un tipo de arte contemporáneo en el cual el artista utiliza, como parte de la composición, el propio medio (como paredes, piso, luces e instalaciones) además de objetos diversos. En muchas ocasiones, los materiales escogidos llenan más o menos el espacio y el espectador es invitado a moverse alrededor de la obra o interactuar con la pieza. En esta memoria, de acuerdo al contexto, también se utiliza exposición o exposición tecnológica, para referirse al mismo término.

Mapa de Características También denominado mapa de activación, son las activaciones de salida para un filtro dado. En una red neuronal convolucional, son los mapas generados en cada capa de convolución luego de la función de activación.

One-hot La codificación *one-hot* es una forma común de preprocesar características categóricas para modelos de aprendizaje automático. Este tipo de codificación crea una nueva característica binaria para cada categoría posible y asigna un valor de 1 a la característica de cada muestra que corresponde a su categoría original.

Pooling Una capa de pooling toma grupos de pixeles, del mapa de características de entrada, y los agrupa en un solo pixel a la salida. Esta operación de agrupación puede tomar el valor máximo de los pixeles (*max pooling*) o el promedio de ellos (*average pooling*).

Prefab En Unity, los prefabs son objetos reutilizables, y creados con una serie de características dentro de la vista proyecto, que serán instanciados en el videojuego cada vez que se estime oportuno y tantas veces como sea necesario. Es decir, permiten al desarrollador crear 'copias' fácilmente de un objeto.

Red Feed-Forward Es una red neuronal artificial donde las conexiones entre las unidades no forman un ciclo. La información solo viaja hacia adelante en la red neuronal, a través de los nodos de entrada, luego a través de las capas ocultas (una o muchas capas) y finalmente a través de los nodos de salida.

Red VGG Es la red convolucional propuesta por **V**isual **G**eometry **G**roup, un grupo de visión por computador de la Universidad de Oxford.

Stride Es el número de pixeles que se desplaza el filtro de una capa convolucional sobre el tensor de entrada.

Upsampling Es el aumento de la resolución espacial mientras se mantiene la representación 2D de una imagen.

Wrapper Este término hace referencia a programas o códigos que rodean otros componentes de programa. Los wrappers pueden utilizarse por diversos motivos: a menudo se usan para mejorar la compatibilidad o interoperabilidad entre diferentes estructuras de software. Si dentro de un programa hay que usar funciones o bloques de programas en otro lenguaje de programación, estos elementos pueden "envolverse" con un wrapper.

Siglas

AdaIN Adaptive Instance Normalization.

API Application Programming Interface.

ASPM-MOB-NST Arbitrary-Style-Per-Model MOB-NST.

BN Batch Normalization.

Caffe Convolutional Architecture for Fast Feature Embedding.

CIN Conditional instance normalization.

CNN Convolutional Neural Network.

CNTK Microsoft Cognitive Toolkit.

CPU Central Processing Unit.

CUDA Compute Unified Device Architecture.

DL Deep Learning.

DNN Deep Neural Networks.

ENPC Encuesta Nacional de Participación Cultural.

GAN Generative Adversarial Network.

GPU Graphics Processing Unit.

IA Inteligencia Artificial.

IN Instance Normalization.

IOB-NST Image-Optimization-Based Online Neural Methods.

ML Machine Learning.

MOB-NST Model-Optimization-Based Offline Neural Methods.

MRF Markov Random Fields.

MSPM-MOB-NST Multiple-Style-Per-Model MOB-NST.

NPR Non-Photorealistic Nendering.

NST Neural Style Transfer.

PMV Producto Mínimo Viable.

PSPM-MOB-NST Per-Style-Per-Model MOB-NST.

RF Requisito Funcional.

RNF Requisito No Funcional.

SDK Software Development Kit.

TF TensorFlow.

UI User Interface.

UTFSM Universidad Técnica Federico Santa María.

UX User Experience.

WCT Whitening and Coloring Tansforms.

Índice general

Resumen iii

Abstract v

Glosario vii

Siglas xi

1. Introducción 1

1.1. Motivación y Contexto . . . 1

1.1.1. Definición de Inteligencia Artificial . . . 3

1.1.2. Arte e Inteligencia Artificial . . . 5

1.2. Planteamiento del Problema . . . 7

1.2.1. Importancia del Arte . . . 7

1.2.2. Descripción del Problema . . . 9

1.3. Objetivos . . . 12

1.3.1. Objetivo General . . . 13

1.3.2. Objetivos Específicos . . . 13

1.4. Alcances y Consideraciones . . . 15

1.5. Estructura . . . 16

2. Estado del Arte 17

2.1. Transferencia de Estilo Neuronal . . . 17

2.2. Estado Actual de Avance en el Campo de NST . . . 19

2.2.1. Métodos NST Basados en Optimización de Imágenes (IOB-NST) 20

2.2.1.1. IOB-NST Paramétrico . . . 20

2.2.1.2. IOB-NST No-Paramétrico . . . 24

2.2.2. Métodos NST Basados en Optimización de Modelos (MOB-NST) . . . 25
2.2.2.1. MOB-NST de Estilo Único (PSPM-MOB-NST) . . . 25
2.2.2.2. MOB-NST de Múltiples Estilos (MSPM-MOB-NST) 26
2.2.2.3. MOB-NST de Estilos Arbitrarios (ASPM-MOB-NST) 30
2.2.3. Mejoras y Extensiones . . . 33
2.2.3.1. Control de los Factores de Percepción en NST . . . 33
2.2.3.2. *Semantic Style Transfer* . . . 36
2.2.3.3. *Instance Style Transfer* . . . 37
2.2.3.4. *Doodle Style Transfer* . . . 37
2.2.3.5. *Stereoscopic Style Transfer* . . . 37
2.2.3.6. *Video Style Transfer (VST)* . . . 38
2.2.3.7. *Character Style Transfer* . . . 39
2.2.3.8. *Audio Style Transfer (AST)* . . . 39

3. Alternativas de Solución **41**
3.1. Métodos de Transferencia de Estilo Neuronal . . . 41
3.1.1. NST Basado en Optimización de Imágenes (IOB-NST) . . . 42
3.1.2. MOB-NST de Estilo Único (PSPM-MOB-NST) . . . 43
3.1.3. MOB-NST de Múltiples Estilos (MSPM-MOB-NST) . . . 44
3.1.4. MOB-NST de Estilos Arbitrarios (ASPM-MOB-NST) . . . 46
3.2. Herramientas de Trabajo para Deep Learning . . . 47
3.2.1. High-Level API, Herramientas de Gestión de Trabajo y Visualización . . . 48
3.2.1.1. Keras . . . 48
3.2.1.2. DIGITS . . . 48
3.2.1.3. Torchnet . . . 49
3.2.1.4. TNT . . . 49
3.2.2. Frameworks . . . 50
3.2.2.1. Caffe . . . 50
3.2.2.2. TensorFlow . . . 50
3.2.2.3. Torch . . . 51
3.2.2.4. CNTK . . . 52
3.2.2.5. PyTorch . . . 52
3.2.3. Plataformas . . . 53

3.3. Herramientas de Trabajo para el Desarrollo de Interfaz de Usuario e Interactividad . . . 54
3.3.1. Herramientas para el Desarrollo de Interfaz de Usuario . . . 54
3.3.2. Herramientas para la Interactividad . . . 54
3.4. Alternativa Seleccionada . . . 55
3.4.1. Selección del Método de Transferencia de Estilo Neuronal . . . 56
3.4.2. Selección de Framework para Deep Learning . . . 57
3.4.3. Selección de Herramientas de Trabajo para el Desarrollo de Interfaz de Usuario e Interactividad . . . 59

4. Diseño de la Instalación Interactiva **61**
4.1. Solución Propuesta . . . 61
4.2. Alcance de la Propuesta de Solución . . . 62
4.3. Requisitos del Sistema . . . 63
4.3.1. Requisitos Funcionales . . . 63
4.3.2. Requisitos No Funcionales . . . 65
4.4. Diagrama de Contexto . . . 65
4.5. Arquitectura de Componentes . . . 66
4.6. Arquitectura de Componentes y Software . . . 68
4.7. Descripción de Módulos . . . 69
4.7.1. Módulo UI . . . 69
4.7.2. Módulo Socket Cliente-Servidor APK . . . 70
4.7.3. Módulo Controlador Central . . . 70
4.7.4. Módulo API Cámara . . . 71
4.7.5. Módulo Socket Cliente-Servidor NST . . . 71
4.7.6. Módulo NST . . . 71
4.7.7. Módulo Visualizador . . . 72
4.8. Matriz de Requisitos Funcionales y Módulos . . . 73
4.9. Flujo de Datos Entre Módulos . . . 73

5. Desarrollo de la Instalación Interactiva **77**
5.1. Consideraciones para el Desarrollo de la Solución . . . 77
5.2. Componentes del Sistema . . . 78
5.2.1. Cámara . . . 78
5.2.2. Tablet . . . 79

5.2.3. Router . . . 79
5.2.4. Computador . . . 80
5.2.5. Proyector . . . 80
5.3. Consideraciones para el Desarrollo de los Módulos de Software . . . 81
5.4. Módulo NST . . . 81
5.4.1. Método . . . 82
5.4.2. Red de Transformación de Imágenes . . . 83
5.4.3. Funciones de Pérdida . . . 87
5.4.4. Red de Pérdidas . . . 91
5.4.4.1. Implementación de las Funciones de Pérdida . . . 93
5.4.5. Entrenamiento . . . 96
5.4.6. Resultados Cuantitativos . . . 99
5.4.7. Resultados Cualitativos . . . 99
5.4.8. Implementación en Video . . . 102
5.5. Módulo Controlador Central . . . 103
5.6. Módulo API Cámara . . . 104
5.7. Módulo UI . . . 105
5.8. Módulo Visualizador . . . 106
5.9. Integración y Comunicación entre Módulos . . . 108

6. Puesta en Marcha y Resultados **111**
6.1. Espacio de instalación . . . 111
6.2. Diseño de Estructuras de Instalación . . . 112
6.2.1. Mueble Contenedor de Hardware . . . 113
6.2.2. Caja Protectora de Cámara . . . 113
6.3. Diseño de Elementos Gráficos . . . 114
6.4. Montaje . . . 115
6.5. Puesta en Marcha en Museo Artequin . . . 116
6.6. Resultados . . . 119

7. Conclusiones y Trabajos Futuros **123**
7.1. Trabajos Futuros . . . 125

A. Detalles de las Calificaciones de las Alternativas de Solución **127**
A.1. Métodos de Transferencia de Estilo Neuronal . . . 127

A.1.1. Velocidad . . . 127
A.1.2. Flexibilidad . . . 128
A.2. Frameworks para Deep Learning . . . 129
A.2.1. Bibliotecas de modelos pre-entrenados . . . 129
A.2.2. Popularidad . . . 130
A.2.3. Actividad en Stack Overflow . . . 130
A.2.4. Documentación . . . 131
A.2.5. Soporte de Técnicas de Paralelismo . . . 131
A.2.6. Herramientas de Visualización . . . 132

B. Optimización Red de Transformación de Imágenes **135**
B.1. Modificación de la Arquitectura de Red . . . 135
B.1.1. Análisis Cuantitativo . . . 137
B.1.2. Análisis Cualitativo . . . 138
B.1.3. Resolución de la Arquitectura de Red . . . 142
B.2. Optimización de la Operación de Convolución . . . 142
B.2.1. Convoluciones Separables en Profundidad . . . 143
B.2.2. Entrenamiento . . . 145
B.2.3. Análisis Cuantitativo . . . 146
B.2.4. Análisis Cualitativos . . . 147
B.2.5. Resolución Optimización de Convoluciones . . . 149

Referencias **151**

Índice de figuras

1.1. Instalación Interactiva Digital: *Universe of Water Particles on a Rock where People Gather*, *teamLab*, 2018. 3

1.2. Relación entre inteligencia artificial, machine learning y deep learning. . 4

1.3. Retrato de Edmond de Belamy - Obvious [9] 5

1.4. *Memories of Passersby I*, Mario Klingemann. 6

1.5. Asistencia a nueve actividades culturales a lo largo de la vida (en porcentajes), ENPC 2017. 10

1.6. Asistencia a museos, centros culturales y bibliotecas a lo largo de la vida (en porcentajes), ENPC 2017. 10

1.7. Comparación de participación cultural entre Chile y otros países del Europa y Latinoamérica. Fuente: Encuestas de participación cultural de los diferentes países [23, 25, 26, 27, 28]. 11

2.1. Diagrama general del algoritmo NST. 18

2.2. Estado del Arte en NST. 20

2.3. Normalización de instancia condicional. El mapa de activación de entrada x ($\mathcal{F}$ en la notación utilizada en la ecuación (2.7)) se normaliza en ambas dimensiones espaciales. Posteriormente, se escala y se desplaza utilizando los vectores de parámetros dependientes del estilo, γ_s y β_s, donde s indexa la etiqueta de estilo. 27

2.4. Arquitectura de red de [58]. Consta de tres módulos: codificador de imágenes $\mathcal{E}$, capa *StyleBank* $\mathcal{K}$ y decodificador de imágenes $\mathcal{D}$. 28

2.5. Visión general de la arquitectura propuesta por [60]. MSG-Net: *Multi-style Generative Network*. 29

2.6. Descripción general de la arquitectura propuesta por [55]. Se utilizan las primeras capas de una red VGG-19 fija para codificar el contenido y el estilo. Una capa AdaIN se utiliza para realizar la transferencia de estilo en el espacio de características. Se entrena un decodificador para invertir la salida de la capa AdaIN a los espacios de la imagen. Se utiliza el mismo codificador VGG para calcular la pérdida de contenido y estilo. 32
2.7. Control del tamaño de trazo en NST. (c) es la salida con un tamaño de pincel más pequeño y (d) con un tamaño de pincel más grande. La imagen de estilo es "La noche estrellada"de Vincent van Gogh. 34
2.8. Flujo de trabajo del método de Lu et al. [69]. 37

3.1. Visión general del algoritmo IOB-NST de Gatys et al. [35]. 42
3.2. Visión general del algoritmo PSPM-MOB-NST de Johnson et al. [52]. . 43
3.3. Visión general de MSG-NET: Multi-style Generative Network [60]. . . 45
3.4. Visión general del algoritmo ASPM-MOB-NST de Huang y Belongie [55]. 46
3.5. Flujo de trabajo de herramientas de deep learning. 47

4.1. Dormitorio en Arlés de Vincent van Gogh. (a) es la pintura original que se encuentra en “Van Gogh Museum”, Ámsterdam [111]. (b) es la representación real de la pintura en el Museo Artequin de Viña del Mar. 62
4.2. Diagrama de contexto. 66
4.3. Diagrama de arquitectura de componentes. 66
4.4. Diagrama de arquitectura de componentes y software. 68
4.5. Diagrama de flujo de datos. 74

5.1. Intel® RealSense™ Depth Camera D435. 78
5.2. Tablet Samsung Galaxy Tab E [115]. 79
5.3. Router DIR-608 Wireless N 150 [116]. 80
5.4. Mini PC Zotac ZBOX QK5P1000 [117]. 80
5.5. Proyector láser LG ProBeam HF80LG [118]. 81
5.6. Visión general del algoritmo de transferencia de estilo neuronal implementado 82
5.7. Implementación de la red de transformación de imágenes en TF. 84
5.8. Implementación de la capa de padding en TF. 84
5.9. Implementación de la capa convolucional en TF. 85

5.10. Implementación de la normalización de instancias en TF. 85
5.11. Diseño del bloque residual. 86
5.12. Implementación del bloque residual en TF. 86
5.13. Implementación de la capa de convolución transpuesta en TF. 87
5.14. En [52] se utiliza la optimización para encontrar una imagen $\hat{y}$ que minimiza la pérdida de reconstrucción de características $\ell_{feat}^{\phi,j}(\hat{y}, y)$ para varias capas j de la red de pérdida ϕ. A medida que se reconstruye a partir de capas superiores, el contenido de la imagen y la estructura espacial general se conservan, pero el color, la textura y la forma exacta no. 88
5.15. Representación intuitiva de la matriz de Gram. 89
5.16. En [52] se utiliza la optimización para encontrar una imagen $\hat{y}$ que minimiza la pérdida de reconstrucción de estilo $\ell_{style}^{\phi,j}(\hat{y}, y)$ para varias capas j de la red de pérdida ϕ. Las imágenes $\hat{y}$ conservan características estilísticas pero no la estructura espacial. 90
5.17. Arquitectura de red del modelo VGG-19. 91
5.18. Implementación red VGG-19 en TF. 92
5.19. Método para extraer los mapas de características de las capas de la red VGG-19. 92
5.20. Activaciones de la red de pérdida $\hat{y}$ dado una imagen de contenido objetivo y una imagen de estilo objetivo. 93
5.21. Implementación cálculo matriz de Gram en TF. 94
5.22. Activaciones de la red de pérdida ϕ dada la entrada $\hat{y}$. 94
5.23. Implementación pérdida de reconstrucción de característica en TF. . . . 94
5.24. Implementación pérdida de reconstrucción de estilo en TF. 95
5.25. Implementación regularización de variación total en TF. 95
5.26. Implementación de la pérdida total en TF. 96
5.27. Imágenes de estilo objetivo. 97
5.28. Gráfico de pérdidas en función del número de iteraciones de la red transformación de imágenes para el estilo “La Noche Estrellada”. 97
5.29. Gráfico de pérdidas en función del número de iteraciones de la red transformación de imágenes para el estilo “Autorretrato”. 98
5.30. Gráfico de pérdidas en función del número de iteraciones de la red transformación de imágenes para el estilo “Dormitorio en Arlés”. 98

5.31. Primer conjunto de ejemplo cualitativos para las 3 redes de transformación de imágenes del módulo NST. 100
5.32. Segundo conjunto de ejemplo cualitativos para las 3 redes de transformación de imágenes del módulo NST. 101
5.33. Implementación del procesamiento de video en TF. 102
5.34. Guardado de video estilizado. 102
5.35. Diagrama de estados Van Gogh te pinta. 103
5.36. Vista desde el inspector de Unity de los objetos utilizados del SDK 2.0 de Intel RealSense. 104
5.37. Objeto RecordCamera que permite grabar y guardar los videos de la cámara. 105
5.38. Vistas del módulo UI. 106
5.39. Vistas del módulo Visualizador. 107
5.40. Diagramas de flujo cliente y servidor. 109

6.1. Espacio de instalación museo Artequin Viña del Mar. 112
6.2. Modelo 3D del mueble contenedor de hardware y sus medidas. 113
6.3. Modelo 3D de la caja protectora de la RealSense y sus medidas. 113
6.4. Gráfica para mueble. 114
6.5. Gráficas para situar en la pared y mostrar la posición de la instalación interactiva. 114
6.6. Montaje del mueble contenedor de hardware y caja protectora de cámara en el espacio de instalación junto con las gráficas para mostrar la posición de la instalación interactiva. 115
6.7. Vista panorámica del espacio de instalación de Van Gogh te pinta. 116
6.8. Instalación interactiva Van Gogh te pinta. 117
6.9. Dimensión efectiva del video en la proyección. 118
6.10. Instalación interactiva Van Gogh te pinta siendo utilizado por visitantes del museo Artequin. 119
6.11. Resultados de la interacción de los visitantes con la instalación interactiva Van Gogh te pinta. 120

A.1. Cantidad de stars en GitHub de cada framework. 130
A.2. Cantidad de contribuyentes en GitHub de cada framework. 131

A.3. Cantidad de preguntas en Stack Overflow que tienen como etiqueta cada framework. 132

B.1. Estilo utilizado para comparar cualitativamente el efecto de modificar la cantidad de bloques residuales en la arquitectura de red. 138
B.2. Resultados cualitativos para las distintas redes de transferencia de estilo. 139
B.3. Resultados cualitativos para distintas redes de transferencia de estilo con imágenes de entrada de personas. 140
B.4. Resultados cualitativos para distintas redes de transferencia de estilo con imágenes de entrada de personas. 141
B.5. Convolución 2D estándar con 1 filtro. 143
B.6. Convolución 2D estándar con 128 filtros. 143
B.7. Primera etapa de la convolución separable en profundidad. 144
B.8. Segunda etapa de la convolución separable en profundidad. 144
B.9. Gráficos de pérdida en función del número de iteraciones de la red transformación de imágenes con convolución separable en profundidad. 146
B.10. Resultados cualitativos que comparan las convoluciones estándar con las convoluciones separables en profundidad. 148
B.11. Resultado cualitativo para una arquitectura de red con 5 bloques residuales y capas de convolución separable en profundidad. 149

Índice de tablas

3.1. Calificaciones de los métodos de transferencia de estilo neuronal. 56
3.2. Calificaciones de los frameworks para deep learning. 59

4.1. Requisitos funcionales. 64
4.2. Requisitos no funcionales. 65
4.3. Módulos de software de la arquitectura del sistema. 69
4.4. Matriz de requisitos funcionales y módulos. 73

5.1. Velocidad de fotogramas en función de la resolución de la imagen para el sensor RGB de la cámara RealSense [114]. 78
5.2. Arquitectura de red utilizada para las redes de transferencia de estilo. . . 83

A.1. Soporte de técnicas de paralelismo en frameworks para deep learning. . 132

B.1. Arquitectura de red utilizada por [52] para transferencia de estilo. . . . 136
B.2. Comparación de tiempos para redes de transformación con distinto número de bloques residuales. 137
B.3. Arquitectura de red utilizada para comprobar la eficiencia de las convoluciones separables en profundidad. 145
B.4. Comparación de tiempos para red de transformación de imágenes con convoluciones estándar y convoluciones separables por profundidad. Ambas redes tienen una capa residual y procesan 420 imágenes de 640×640. 146

Capítulo 1

Introducción

1.1. Motivación y Contexto

Pocas cosas pueden parecer tan lejanas como el sentimiento y la expresividad de un cuadro de Picasso o una sinfonía de Mozart, frente a la frialdad de un algoritmo matemático que procesa grandes cantidades de datos dentro de un superordenador. Sin embargo, a lo largo de toda la historia, el arte y la tecnología han estado muy relacionados. Los avances tecnológicos permiten a los artistas explorar diferentes formas de expresar y crear arte, mientras que al público en general, un mayor acceso a las distintas creaciones artísticas.

Un caso interesante de la influencia que ha tenido la tecnología en el arte es la aparición de la fotografía en 1839, que pese a ser una realización puramente técnica, pronto se vislumbró la artisticidad de este nuevo medio, pues la obra resultante podía ser considerada artística en cuanto suponía la intervención de la creatividad de la persona que capta la imagen. Más allá de la lucha por la aceptación de las imágenes fotográficas como arte, se estaba produciendo un desarrollo de consecuencias mucho mayores. No hay duda de que, desde su aparición, las imágenes de las cámaras han sido el proveedor más importante de piezas visuales para el mayor número de personas, revolucionando el acceso público al patrimonio de las artes visuales del mundo. Hoy en día, hay pocas dudas de que la fotografía, además de su enorme variedad de usos, se considera legítimamente una disciplina de bellas artes. Casi cualquier museo conocido y respetado tiene secciones dedicadas exclusivamente al arte fotográfico; y hay varios museos y galerías dedicados específicamente a la fotografía [1].

Del mismo modo, al estudiar la historia de la música, también se percibe la estrecha

relación que ha existido entre el arte y la tecnología. Los avances tecnológicos en la música han acelerado los procesos de producción musical, facilitando y dando amplias posibilidades en el momento de la grabación. Mediante nuevos equipos o formas de trabajo, se puede conseguir una calidad extrema, un tratamiento al mínimo detalle que da como resultado un sonido pulido y cuidado al máximo. Este amplio abanico de posibilidades impulsa nuevos procesos creativos, nuevas formas de crear o interpretar la música, nuevos sonidos y géneros musicales. La forma de vivir la música también ha cambiado, la difusión digital ha hecho que la música esté al alcance de todos, sea accesible y se sienta de manera distinta.

Ejemplos como los anteriores hay muchos, y muestran que el arte y la tecnología tienen una relación simbiótica que ha existido desde siempre. Sin embargo, actualmente se hace más evidente este vínculo debido a aquellas obras que expresan esta relación como parte de su producto final, haciendo uso de desarrollos tecnológicos cada vez más sofisticados. Un ejemplo de lo anterior es *Mori Building Digital Art Museum*, en Japón, el primer museo de arte digital del mundo, inaugurado en junio de 2018 por el colectivo de artistas *teamLab*. Este espacio, enteramente dedicado al arte digital, incluye 50 obras de *teamLab* que conviven y se diluyen unas con otras, las cuales son producidas enteramente por 520 computadores y 470 proyectores distribuidos por todo el lugar. El museo en sí es una experiencia inmersiva.

Una de las obras más populares del museo es la cascada con flores bajo el nombre de *Universe of Water Particles on a Rock where People Gather* (ver Figura 1.1). Esta instalación simula que el agua cae sobre una roca, donde la gente se reúne, y el flujo del agua dibuja la forma de la cascada. Cuando una persona se para en la roca o toca la cascada, también se convierte en una roca que cambia el flujo de agua. El flujo de agua continúa transformándose en tiempo real debido a la interacción de las personas provocando estados visuales nuevos en cada momento [2].

La tecnología ha revolucionado los paradigmas del arte en cada época. Ocurrió con la fotografía, que se pensaba que haría desaparecer la pintura, pero resultó ser lo contrario, se potenciaron. El debate de esa época provocó que se crearan nuevos estilos de pintura para diferenciarse de la fotografía, e incluso algunos promovían la conjunción de ambos (pintura y fotografía). De la misma forma ocurrió con la aparición del cine o la televisión, que todos pensaban que harían desaparecer el teatro, pero tampoco fue así. Los avances de la tecnología impulsan el progreso del arte otorgando a los artistas nuevas formas de expresión, y al público en general, un mayor acceso a ella.



Figura 1.1: Instalación Interactiva Digital: *Universe of Water Particles on a Rock where People Gather*, *teamLab*, 2018.

En la actualidad se vive lo que muchos han llamado la cuarta revolución industrial o industria 4.0, y la inteligencia artificial se ha convertido, indudablemente, en uno de los motores que desatarán todo su potencial. Así como la tecnología revolucionó el arte con la fotografía, el cine, el internet y las comunicaciones, hoy lo está haciendo con la inteligencia artificial.

1.1.1. Definición de Inteligencia Artificial

La Inteligencia Artificial (IA) ha adquirido progresivamente un rol protagónico en los últimos años, sin embargo, hasta el día de hoy no existe consenso sobre su definición debido a los cambios que ha sufrido en sus más de 60 años de historia y los que, probablemente, siga experimentando. John McCarthy, padre de la inteligencia artificial, conocido por sus importantes contribuciones en el campo de la IA, fue el responsable de introducir el término "inteligencia artificial" en la Conferencia de Dartmouth en 1956, y la definió como "la ciencia e ingenio de hacer máquinas inteligentes, especialmente programas informáticos inteligentes" [3]. En la actualidad, la IA se ha convertido en un curso interdisciplinario que involucra varios campos. En el sentido más amplio, la IA se refiere a la ciencia que busca que las máquinas puedan aprender, razonar y actuar por sí mismas; que pueden tomar sus propias decisiones cuando se enfrentan a nuevas

situaciones, de la misma forma que los humanos y los animales [4].

En su forma actual, la gran mayoría de los avances y aplicaciones de IA de los que se oye hablar se refieren a una categoría de algoritmos conocida como Aprendizaje Automático o *Machine Learning* (ML), el cual se considera un subcampo de la inteligencia artificial. El machine learning se puede definir como el estudio de algoritmos informáticos que mejoran automáticamente a través de la experiencia [5]. Estos algoritmos utilizan estadísticas para encontrar patrones o modelos, generalmente, en cantidades masivas de datos. Y los datos, aquí, abarcan muchas cosas: números, palabras, imágenes, clicks, etc., con el fin de hacer predicciones o decisiones sin estar programados explícitamente para hacerlo [6].

Dentro del campo del ML existe un conjunto de algoritmos inspirados ligeramente en la comprensión biológica del cerebro humano, específicamente en la interconexión de neuronas. Esta familia de algoritmos se conoce como Aprendizaje Profundo o *Deep Learning* (DL) e intenta modelar abstracciones de alto nivel en datos usando arquitecturas computacionales que admiten transformaciones no lineales múltiples e iterativas de datos expresados en forma matricial o tensorial [7]. Los modelos de deep learning se basan en Redes Neuronales Artificiales que consisten en un conjunto de unidades básica, llamadas neuronas, conectadas entre sí para transmitir señales, como la sinapsis en el cerebro biológico. La información de entrada atraviesa la red neuronal, que está conformada por muchas capas, donde se somete a diversas operaciones produciendo valores de salida que son interpretados y utilizados dependiendo de la aplicación [8].

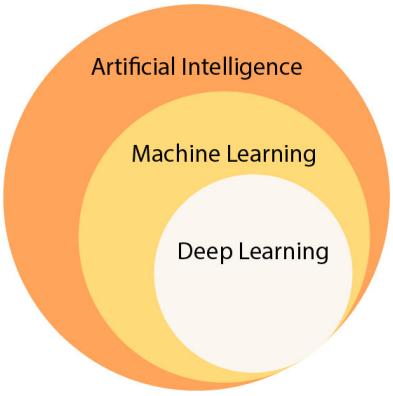


Figura 1.2: Relación entre inteligencia artificial, machine learning y deep learning.

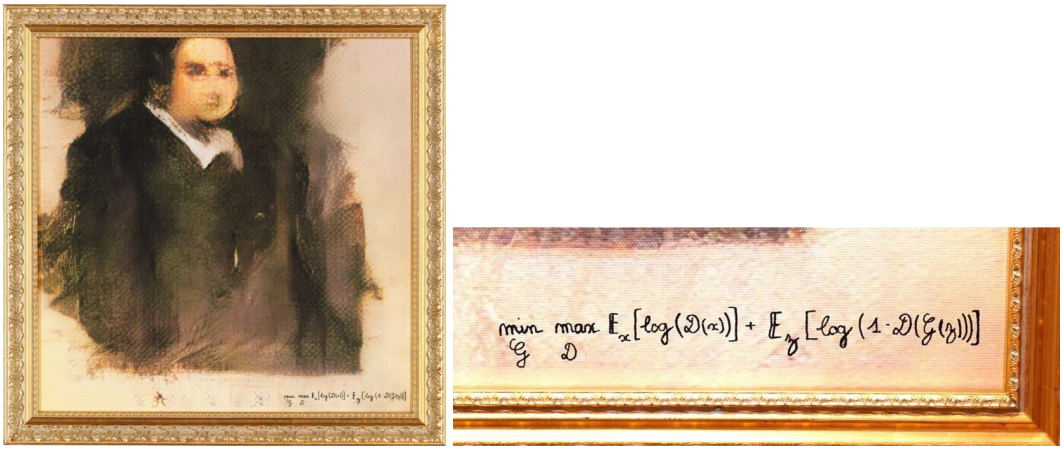
El deep learning ha permitido muchas aplicaciones prácticas de machine learning y, por extensión, del campo general de la IA. Los automóviles sin conductor, una mejor atención médica preventiva, incluso mejores recomendaciones de películas, están todos

aquí hoy o en el horizonte. Con la ayuda del machine learning y el deep learning, la inteligencia artificial puede llegar a ese estado de ciencia ficción que tanto tiempo se ha imaginado.

1.1.2. Arte e Inteligencia Artificial

La sociedad actual ya se ha acostumbrado a la presencia de la inteligencia artificial en muchas esferas de la vida: el uso de *smartphones*, los asistentes de voz, las recomendaciones de productos en la web o el buscador de Google son algunas de las tantas aplicaciones en las que está presente la IA. Del mismo modo, áreas como la creatividad y el arte, que han sido consideradas durante mucho tiempo como el territorio de la creatividad humana, también han sido invadidas por la inteligencia artificial.

A fines de octubre de 2018, en Nueva York, Christie's fue la primera casa de subastas en subastar una obra realizada por una IA. La pieza es un retrato, una impresión en lienzo ligeramente borrosa, de un hombre austero vestido de negro: el epónimo Edmond de Belamy.



(a) Edomnd de Belamy (b) Firma del retrato

Figura 1.3: Retrato de Edmond de Belamy - Obvious [9]

Aunque las estimaciones iniciales para el precio de la pintura eran menos de USD 10.000, el retrato finalmente se vendió a un postor anónimo por la asombrosa cantidad de USD 432.000, un precio inaudito para las ilustraciones de IA [10]. El marketing desempeñó un papel importante en la venta de la obra; la forma en que se utilizó la IA para crear el cuadro no fue especialmente innovadora, utilizaron algoritmos de redes generativas existentes, pero la presentación del rol de la IA en el cuadro si lo fue. El

cuadro no está firmado con un nombre humano, sino con parte del algoritmo que lo produjo: $\min_{\mathscr{G}} \max_{\mathscr{D}} \mathbb{E}_x \left[\log\left(\mathscr{D}(x)\right)\right] + \mathbb{E}_z \left[\log\left(1 - \mathscr{D}(\mathscr{G}(z))\right)\right]$ (ver Figura1.3). Christie's describió la obra como "no producto de una mente humana", y empaquetó la venta como "la primera pieza de arte de inteligencia artificial que sale a subasta". Fue un movimiento curioso por parte de Christie's, ya que decidió desviar la atención de las impresionantes mentes humanas que han creado una complicada tecnología capaz de producir obras de arte y, en su lugar, dirigió la narrativa hacia un imaginario vacío humano [11].

Posterior a Edmond de Belamy, el 6 de marzo de 2019 ocurre nuevamente el mismo hecho, pero en otra casa de subastas. Sotheby's subasta por primera vez en su historia una obra creada por inteligencia artificial: *Memories of Passersby I* de Mario Klingemann. En la subasta de arte contemporáneo en Londres, Sotheby's vendió la pieza por 40.000 libras (USD 53.000) y se convierte en la segunda pieza de "arte de inteligencia artificial" en ser vendida en subasta por un alto precio [12].



Figura 1.4: *Memories of Passersby I*, Mario Klingemann.

La obra es totalmente autónoma, utiliza un complejo sistema de redes neuronales para generar un flujo interminable de retratos, rostros imaginarios masculinos y femeninos. La obra de arte se presenta como una pieza de instalación: la máquina de IA está alojada en un gabinete de madera de castaño hecho a medida, conectado a dos pantallas enmarcadas.

A diferencia de las instalaciones de arte generativo anteriores, *Memories of Passersby I* no contiene una base de datos, es una IA desarrollada y entrenada por Mario Klingemann, que crea nuevos retratos, píxel a píxel, en tiempo real. Los resultados que se muestran en la pantalla no son combinaciones aleatorias o programadas de imágenes existentes, sino obras de arte únicas generadas por IA. El flujo de imágenes presentadas no sigue una coreografía predefinida, sino que es el resultado de la IA que interpreta su propia salida; la naturaleza compleja de este circuito de retroalimentación significa que nunca se repetirá ninguna imagen. *Memories of Passersby I* contiene todos los algoritmos y GAN (redes generativas adversarias) necesarios para producir una sucesión interminable de nuevas imágenes mientras se esté ejecutando [13].

Casos como los anteriores muestran que la tecnología va camino a revolucionar por completo el arte. La explosión tecnológica ha cambiado la forma en que las personas se comunican, se informan, se relacionan y, posiblemente, la propia concepción de la vida. El arte, por supuesto, también se ha visto sacudido por esta metamorfosis, la revolución digital y la inteligencia artificial abren posibilidades infinitas de expresión y exhibición. Los artistas han cambiado su forma de crear, a la vez que buscan respuestas a las incógnitas que plantea una realidad hipertecnológica y acelerada.

1.2. Planteamiento del Problema

1.2.1. Importancia del Arte

No existe una definición universal de arte, aunque existe un consenso general de que el arte es la expresión o aplicación de la imaginación y la habilidad creativa humana, que produce obras que se aprecian principalmente por su estética o poder emocional [14, 15, 16]. Dicho concepto agrupa ámbitos diferentes, como la escultura, la pintura, la danza, la poesía, la cocina, el cine, los grabados, el teatro, las historietas, la fotografía, el arte numérico y mucho otros, los cuales han ido evolucionando a lo largo de la historia de la humanidad.

El arte es un elemento intrínseco del ser humano. Las diferentes formas de hacer arte corresponden a la necesidad o, más bien, a la característica fundamental de expresarse que poseen los seres humanos. No solo los artistas utilizan el arte como medio de expresión, cualquier persona puede hacer provecho de él, por ejemplo, los niños y jóvenes, que muchas veces se ven dificultados para transmitir lo que les ocurre, hallan

en el arte el escape necesario para dejar fluir sus emociones y el medio para comunicar lo que hay en su interior.

Pablo Picasso dijo: "Todo niño es un artista. El problema es cómo seguir siendo artistas al crecer"[17]. Para entenderlo, basta con fijarse en los niños que son capaces de tomar un lápiz y garabatear en una hoja de papel mucho antes de poder hablar: desde este punto de vista, el arte es sinónimo de creatividad, por consiguiente, es importante tanto para el desarrollo de habilidades y conocimientos, así como para implementar el aprendizaje y la experiencia. Las complejidades internas del ser humano son reflejadas al mundo exterior a través del arte.

El arte desempeña un papel mediador y motor de la comunicación, ya que el artista a través de su creación transmite no solo emociones, sino también mensajes, y nos hace reflexionar sobre nuestra existencia, los problemas sociales o la vida en general. Desde esta perspectiva, se convierte en una herramienta que puede cambiar o educar a una sociedad. En este mismo sentido, el arte se puede utilizar para crear conciencia sobre una gran variedad de causas. Diversas actividades artísticas han tenido como objetivo crear conciencia sobre el autismo [18], derechos humanos [19], contaminación [20], conservación de los océanos [21] y una variedad de otros temas.

Una característica importante del arte, que la vuelve imprescindible para el ser humano, es que puede brindar paz, felicidad, amor, esperanza y muchas otras emociones: por ejemplo, en situaciones donde las personas necesitan escuchar música para curar su tristeza, ver comedias teatrales para reír, ver películas para soñar o simplemente para divertirse, aprender a conocer el pasado o contemplar obras artísticas para apreciar su belleza.

El arte, por su forma y contenido, constituye un medio idóneo para reflejar la cultura humana. Evoluciona constantemente, poco tienen en común las obras de arte de la prehistoria, con las de la edad media o moderna. Por medio del arte se puede analizar el contexto histórico y cultural en el que fueron producidas las obras artísticas y, por lo mismo, sirve para conservar el patrimonio cultural de un pueblo y transmitirlo de generación en generación. Además, el arte es subjetivo, se expresa en un lenguaje universal y comprensible para cualquier ser humano, ya que apela a los sentidos, emociones y facultad de pensar. La educación, hoy en día, se sigue basando en obras artísticas del pasado, porque estas, en sus diferentes manifestaciones, nunca han perdido su importancia para la sociedad[1].

[1]Esta sección está basada en el artículo de [22].

1.2.2. Descripción del Problema

A pesar de lo importante que es el arte para una sociedad, en Chile, las múltiples expresiones del arte y la cultura todavía no escapan de los efectos de la desigualdad, y los excluidos se encuentran tristemente marginados de participar en la construcción simbólica de la sociedad.

Las barreras de acceso a la cultura son múltiples, y en el campo del arte estas barreras se ven dramáticamente reflejadas. Desde el acceso a la infraestructura, hasta la falta de formación artística, impiden que parte de la ciudadanía se aproxime de manera comprensiva a una obra de arte, no pudiendo acceder al goce estético y a la expresión artística (Consejo Nacional de la Cultura y las Artes, 2016).

Desde el año 2004, Chile cuenta con la Encuesta Nacional de Participación Cultural (ENPC), instrumento que ha sido aplicado en cuatro oportunidades y que ha posibilitado la observación simultánea de la realidad global del país y de las regiones en lo que a artes y cultura respecta, dando cuenta de la variedad de escenarios que se enfrentan y facilitando la generación de diagnósticos acuciosos. La última encuesta, ENPC 2017 [23], reveló que el 84.6 % de los encuestados nunca ha asistido a la ópera, un 74.7 % a un espectáculo de música clásica, 56.8 % a un centro cultural y un 53 % a una exposición de arte.

Las actividades culturales con mayor asistencia según la ENPC 2017 fueron el cine (43.4 %), la compra de artesanía (34 %), los conciertos de música actual/popular (30 %), parques nacionales (28.7 %), danza (20.9 %) y museo (20.5 %); mientras que las con menor asistencia se cuenta la ópera (1.9 %), conciertos de música clásica (6.1 %), circo (13.6 %) y teatro (14.2 %). Las Figuras 1.5 y 1.6 presentan gráficos detallados de la participación en actividades culturales.

El análisis histórico de participación cultural muestra una tendencia a la baja en la asistencia al teatro (de 20.1 % a 14.2 %) y una baja abrupta, el 2017, en la asistencia a exposiciones de artes visuales (de 24.9 % a 16 %) y lectura de libros (de 47 % a 38.9 %). Las demás actividades artístico-culturales, danza, conciertos de música actual, cine, museos y bibliotecas, han mantenido sus porcentajes de asistencia en los últimos años y ninguna actividad presenta una gran alza en participación.

Los datos de la encuesta reafirman la importancia del grupo socioeconómico, la escolaridad y la edad como determinante de la participación cultural. Asimismo, resultan especialmente lamentables los resultados que arroja la pregunta referida a **los motivos por los cuales las personas no participan en actividades artístico-culturales**, entre

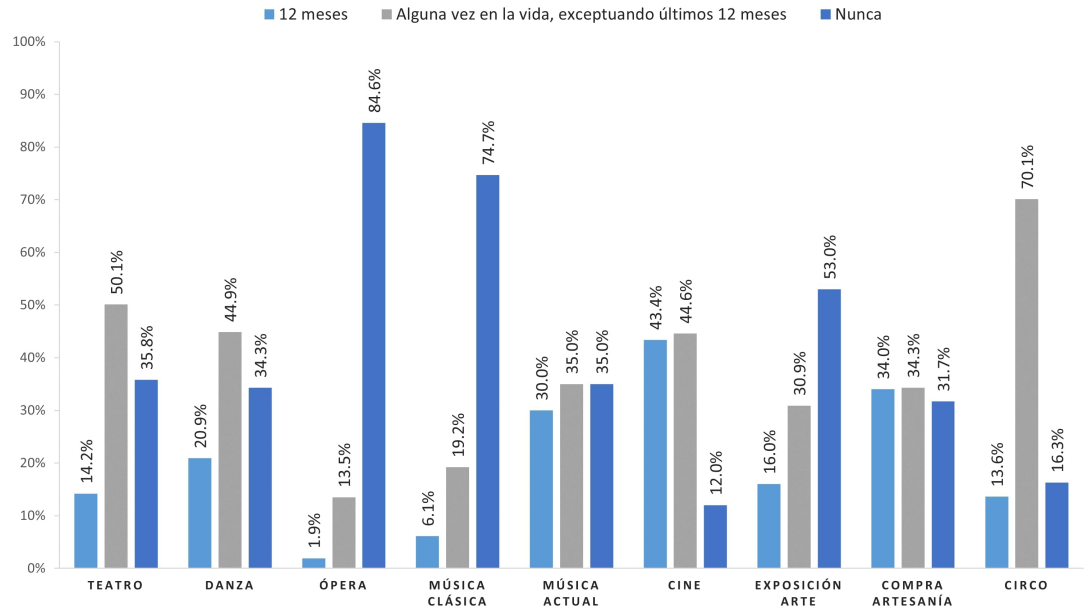


Figura 1.5: Asistencia a nueve actividades culturales a lo largo de la vida (en porcentajes), ENPC 2017.

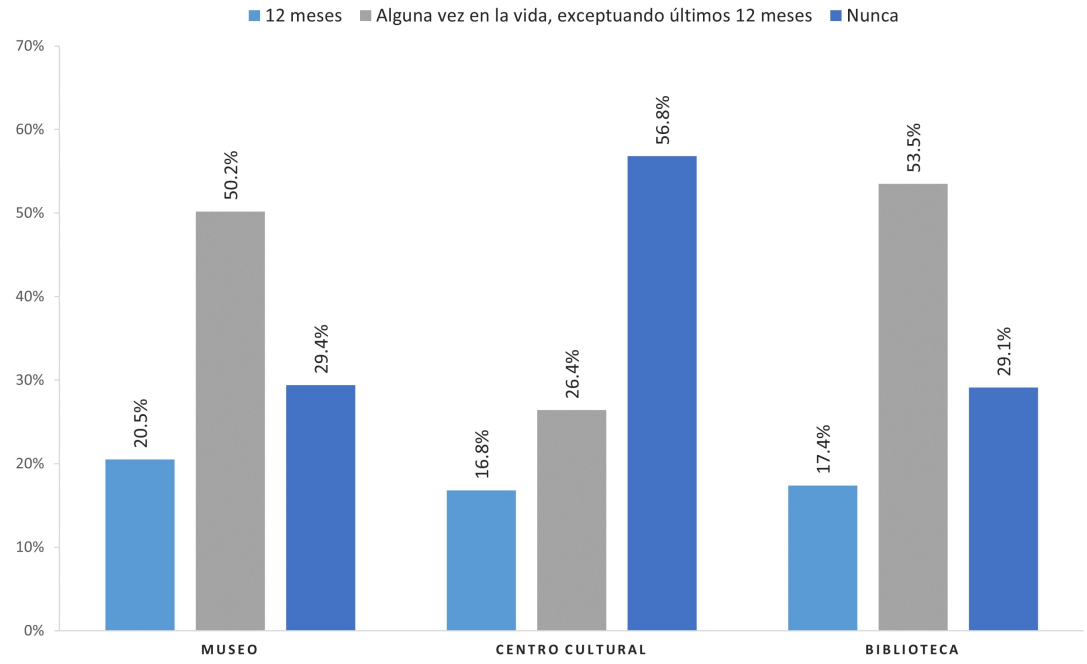


Figura 1.6: Asistencia a museos, centros culturales y bibliotecas a lo largo de la vida (en porcentajes), ENPC 2017.

ellas, el 46 % lleva la delantera con “porque me aburren”, mientras que le sigue el 15 % con “porque no las entiendo”.

Para evaluar correctamente los porcentajes de participación cultural en Chile, se deben comparar los datos expuestos anteriormente con los de otros países, esto permite valorizar correctamente las cifras considerando la situación de otras naciones. Para ello, se han recopilados datos de dos países latinoamericanos, por su cercanía geográfica, y dos de Europa, por su alto desarrollo humano[2] [24]. Es necesario enfatizar que esta comparación no pretende ser de ninguna manera exhaustiva, sino que busca dar una mirada superficial de la participación cultural de Chile tomando como referencia países circundantes, y otros más desarrollados, donde el acceso al arte y la cultura tiene un alcance mayor que en Chile.

Se comparan los datos de participación cultural de Chile con los de Argentina, Colombia, España e Inglaterra. Los datos son extraídos de las encuestas de participación cultural de los países mencionados y se presentan en el gráfico comparativo a continuación. Notar que hay que actividades culturales sin cifras debido a que las fuentes de datos no los presentaban. El gráfico de comparación se muestra en la Figura 1.7.

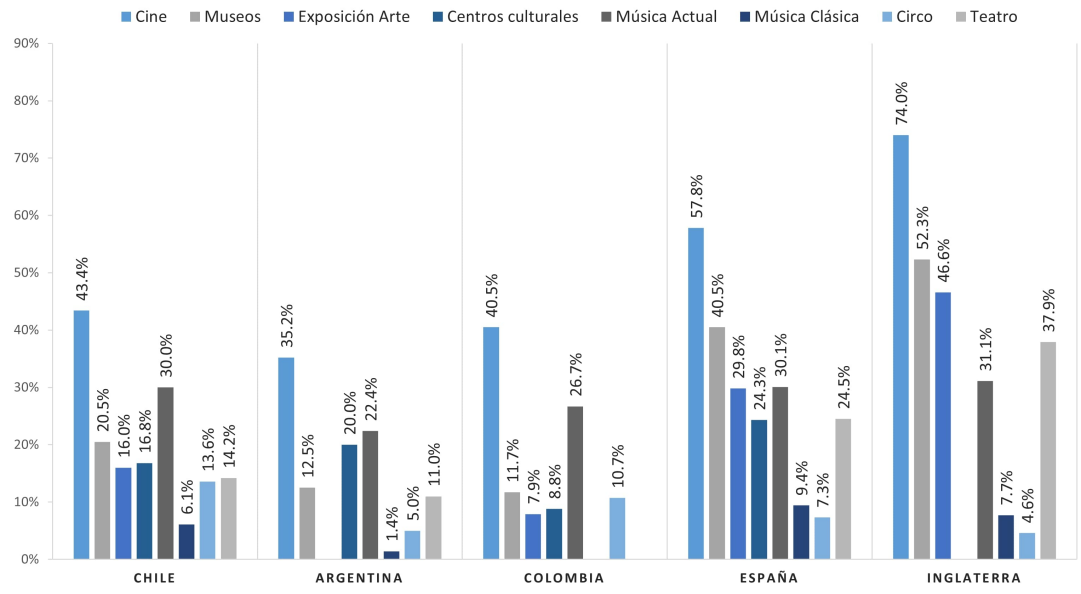


Figura 1.7: Comparación de participación cultural entre Chile y otros países del Europa y Latinoamérica. Fuente: Encuestas de participación cultural de los diferentes países [23, 25, 26, 27, 28].

[2]Desarrollo Humano: Término definido en el glosario.

Las cifras de participación en Chile están a un buen nivel en el contexto regional. Los niveles de asistencia en Chile están por encima de los porcentajes que se dan en Argentina y Colombia a excepción de la asistencia a centros culturales en Argentina. Por otro lado, al comparar con los países europeos como Inglaterra y España, Chile se encuentra por debajo de ellos. Lo anterior muestra que los esfuerzos del Ministerio de las Culturas, las Artes y el Patrimonio (antes el Consejo Nacional de la Cultura y las Artes) por democratizar el acceso a la cultura han tenido buenos resultados respecto al panorama regional, pero aún hay trabajo que hacer siguiendo el ejemplo de los países más desarrollados.

Los datos de la ENPC 2017 y los análisis realizados confirman que en Chile hace faltar captar y fomentar el interés de la ciudadanía por las actividades culturales. Sumado a esto, las barreras de acceso a la cultura como el grupo socioeconómico, la escolaridad, el territorio, etc., impiden que la mayor parte de la ciudadanía disfrute y participe de las distintas actividades artístico-culturales y se vea beneficiada de los efectos positivos que el arte y la cultura ofrecen en la cohesión social, la libertad, el sentimiento de pertenencia, el bienestar e incluso la salud de los ciudadanos.

1.3. Objetivos

En este escenario, donde las barreras de acceso a la cultura y el arte son múltiples y el interés de la ciudadanía por las actividades culturales se debe fomentar, para tener éxito y ser sostenibles, las organizaciones y entidades culturales necesitan crear nuevas formas de pensamiento, así como desarrollar mecanismos para implementar ideas y transformarlas en oportunidades que respondan a un entorno desfavorable y que cambia vertiginosamente, siendo un elemento fundamental en las estrategias exitosas, la innovación tecnológica.

Los estudios realizados hasta ahora han demostrado que las innovaciones tecnológicas ejercen un papel importante en el buen desempeño de las organizaciones culturales [29, 30]. Gracias a ellas, los museos, centros culturales, galerías de arte, etc., pueden mejorar la experiencia de los visitantes en sus dependencias, ofreciendo espacios mucho más interactivos que permitan a los visitantes convertirse en participantes activos, en vez de recibir información de manera pasiva. Además, dan la posibilidad de incorporar más información de contenido audiovisual a las exposiciones, lo que permite ofrecer un contenido más claro, entendible y entretenido.

1.3.1. Objetivo General

En virtud de lo antes expuesto, el presente proyecto tiene como objetivo general introducir la inteligencia artificial como fenómeno artístico y cultural, desarrollando una instalación interactiva[3], tecnológica y artística, donde niños y adultos se encanten con esta nueva experiencia que pretende promover la creatividad, la educación y la imaginación. Este arte interactivo busca aportar valor a las entidades culturales y sus visitantes, ofreciendo una experiencia innovadora, donde el espectador se vuelva un actor principal en la obra de arte, provocando que el resultado final sea dinámico, es decir, cada espectador verá un resultado artístico distinto que los otros.

1.3.2. Objetivos Específicos

- **Diseñar e implementar un algoritmo de transferencia de estilo basado en redes neuronales**

  Los algoritmos con redes neuronales que manipulan imágenes digitales o videos para adoptar la apariencia o el estilo visual de otra imagen se conocen como Transferencia de Estilo Neuronal o Neural Style Transfer (NST) [31]. Esta clase de algoritmos son una de las aplicaciones más llamativas del aprendizaje profundo en el área de visión por computador y, en el último tiempo, se han utilizado para desarrollar experiencias interactivas en muchos museos de arte digital. La instalación interactiva de arte digital que se desarrolla en esta memoria se basa en el algoritmo NST, por este motivo, los módulos de este sistema estan restringidos por la rapidez, las entradas y las salidas de este algoritmo.

- **Diseñar e implementar un módulo de interfaz de usuario e interactividad**

  La interactividad es una de las características esenciales de este proyecto porque es un factor de diferenciación respecto al resto de las exposiciones/actividades que pueden existir en los espacios culturales. En este sentido, el algoritmo NST puede ser considerado como un componente que ofrece interactividad al resultado, pues permite que los usuarios sean parte del resultado final de la instalación. Por otro lado, se debe diseñar una interfaz de usuario (UI) que permita la interacción del usuario con el sistema. Es importante que la UI sea sencilla y fácil para usuarios de todos los rangos etarios.

[3]Revisar glosario para una definición detallada del concepto instalación interactiva.

- **Diseñar e implementar un módulo de visualización**

  Se debe elaborar un módulo de visualización que permita mostrar el resultado del algoritmo NST. Además, al ser arte de instalación, existirán momentos de reposo (no uso), por lo tanto, es necesario crear escenas que inviten a los participantes a interactuar con la instalación.

- **Desarrollar una capa de contenido audiovisual que entretenga y estimule la curiosidad de los usuarios**

  Esta propuesta busca mitigar los motivos principales por los cuales las personas no participan en actividades artístico-culturales, los cuales son: 1) que los espectadores se **aburren** y 2) **no entienden** este tipo de actividades (ENPC 2017). El uso de la IA como concepto central de la experiencia abre un abanico de posibilidades y, sumado con la tecnología para dar interacción, permite desarrollar una obra de arte digital dinámica e innovadora, que entretenga y capte la atención de todos los visitantes. Además, el desarrollo de una experiencia digital concede la flexibilidad de adicionar capas de contenido audiovisual que permitan entregar información adicional a los espectadores con la finalidad de que todos sean capaces de entender, disfrutar y cuestionar lo que ocurre en la exposición.

- **Desarrollar un sistema de interconexión de módulos**

  La instalación poseerá componentes físicos que deben ser conectados entre si para poder funcionar correctamente. Pueden ser conexiones alámbricas o inalámbricas y se deben implementar los protocolos de comunicación correspondientes en caso de ser necesario. Por otro lado, pueden existir conexiones entre programas de software y también será necesario el mismo trabajo.

- **Diseñar y construir las estructuras físicas que habiliten el montaje de la instalación interactiva**

  Se deben elaborar ciertas estructuras para montar la exposición en el contexto real. Por ejemplo, en el caso de utilizar un computador, debe existir un mueble o *case* que contenga y proteja el computador en la interacción del usuario con la instalación. Debe ser diseñado correctamente para que sea atractivo y lo suficientemente seguro para evitar robos o fallas por golpe.

- **Validar PMV en entorno real**

  El producto desarrollado en esta memoria debe cumplir con las funcionalidades mínimas para ser considerado producto mínimo viable (PMV), y debe ser probado en un entorno real que valide sus características técnicas.

1.4. Alcances y Consideraciones

El desarrollo de este proyecto se lleva a cabo en conjunto con la startup Electroveja Labs,[4] y se sitúa en el contexto de la renovación del Museo Artequin Viña del Mar y la firma de un convenio de colaboración con la Universidad Técnica Federico Santa María (UTFSM) de disponer de un espacio en el museo para la exposición y divulgación de la ciencia y la tecnología a niños y jóvenes de la región de Valparaíso. Tomando en consideración la naturaleza interactiva del Museo Artequin, en donde todas sus experiencias son multisensoriales, es que la UTFSM propuso a Electroveja Labs encargarse del diseño e implementación de dicho espacio interactivo.

El espacio se encuentra actualmente contiguo a una representación real de la pintura "Dormitorio en Arlés" de Vincent van Gogh, por ello, con la finalidad de tener una coherencia escenográfica se determina que el algoritmo NST extraiga los estilos artísticos de las pinturas de Vincent van Gogh para modificar los videos capturados por la instalación interactiva y permita a los visitantes mezclarse en las pinturas de este gran pintor.

La instalación interactiva de arte digital propuesta será la exposición con la que se inaugurará el espacio del convenio entre el Museo Artequin y la UTFSM, y se mantendrá en exposición de forma permanente. Por tal motivo, este proyecto debe alcanzar, como mínimo, un nivel de desarrollo de PMV. Debido a lo anterior, y el trabajo que conlleva crear un PMV, el desarrollo de la instalación interactiva se lleva a cabo con la participación de un equipo multidisciplinario de desarrolladores de software y diseñadores de Electroveja Labs, donde el principal trabajo del alumno memorista es la investigación, diseño, implementación y optimización del algoritmo NST. En consecuencia, el presente escrito se enfoca en la investigación y desarrollo de los algoritmos de deep learning aplicados en el proyecto más que en los otros módulos implementados para alcanzar el PMV.

[4] Se presenta un descripción de la empresa en el glosario.

1.5. Estructura

Este documento se estructura de la siguiente forma: En el Capítulo 2 se hace una investigación detallada del estado actual de avance en algoritmos de transferencia de estilo neuronal. Luego, en el Capítulo 3, se realiza un análisis de las distintas alternativas de solución de los elementos más relevantes de la construcción de la instalación interactiva de arte digital. Se escoge el algoritmo NST a utilizar en la instalación junto con el framework para su implementación. Además, se escogen las herramientas de trabajo para el desarrollo de la interfaz de usuario e interactividad de la instalación. A continuación, en el Capítulo 4, se presenta el diseño de la instalación interactiva. Se definen los requisitos del sistema, la arquitectura del sistema y la interacción entre los módulos. Después, en el Capítulo 5, se detalla el desarrollo de cada uno de los módulos del sistema y la selección del hardware utilizado para la instalación interactiva. Luego, en el Capítulo 6, se muestra el proceso de montaje y puesta en marcha de la instalación interactiva. Además, se muestran algunos resultados de la interacción de los visitantes con la instalación. Por último, en el Capítulo 7, se entregan las conclusiones del trabajo desarrollado y las futura mejoras que se pueden llegar a implementar en la instalación interactiva.

Capítulo 2

Estado del Arte

Antes de abordar el diseño y la implementación de la solución, es necesario realizar un estudio del estado actual de avance en el campo de la transferencia de estilo neuronal. En este capítulo se presentan los trabajos relacionados con la transferencia de estilo neuronal y sus extensiones más importantes.

2.1. Transferencia de Estilo Neuronal

Desde mediados de la década de 1990, las teorías artísticas detrás de las atractivas obras de arte han atraído la atención no sólo de los artistas, sino de muchos investigadores de informática. Existen muchos estudios y técnicas que exploran cómo convertir automáticamente las imágenes en obras de arte sintéticas. Entre estos estudios, los avances en el renderizado no-fotorrealista o *non-photorealistic rendering* (NPR) [32, 33, 34] son inspiradores, y en la actualidad, es un campo firmemente establecido en la comunidad de los gráficos por computador. Sin embargo, la mayoría de estos algoritmos de estilización de NPR están diseñados para estilos artísticos particulares [34, 35] y no se pueden extender fácilmente a otros estilos. En la comunidad de visión por computador, la transferencia de estilo suele estudiarse como un problema generalizado de síntesis de texturas, que consiste en extraer y transferir la textura de la fuente al destino [36, 37, 38, 39]. Sin embargo, la limitación común de estos métodos es que sólo utilizan características de imagen de bajo nivel y a menudo no logran capturar estructuras de imagen de manera efectiva.

El año 2015, inspirados por el poder de las Redes Neuronales Convolucionales o *Convolutional Neural Networks* (CNN), Gatys et al. [40] estudiaron por primera vez

cómo usar una CNN para reproducir estilos de pintura famosos sobre imágenes naturales. Propusieron modelar el contenido de una foto como los mapas de características de una CNN pre-entrenada, y también modelar el estilo de una obra de arte como las estadísticas de resumen. Sus resultados experimentales demostraron que una CNN es capaz de extraer información de contenido de una fotografía arbitraria e información de estilo de una obra de arte bien conocida. Sobre la base de este hallazgo, Gatys et al. [40] propusieron por primera vez utilizar los mapas de activaciones de una CNN para recombinar el contenido de una foto dada y el estilo de obras de arte famosas. La idea clave detrás de su algoritmo es optimizar iterativamente una imagen con el objetivo de hacer coincidir las distribuciones de características deseadas, lo que implica tanto la información de contenido de la foto como la información de estilo de la obra de arte. Su algoritmo propuesto produce con éxito imágenes estilizadas con la apariencia de una determinada obra de arte. La Figura 2.1 muestra un ejemplo de transferir el estilo de la pintura *"La Muse"* de Pablo Piccaso a una foto de Chicago.

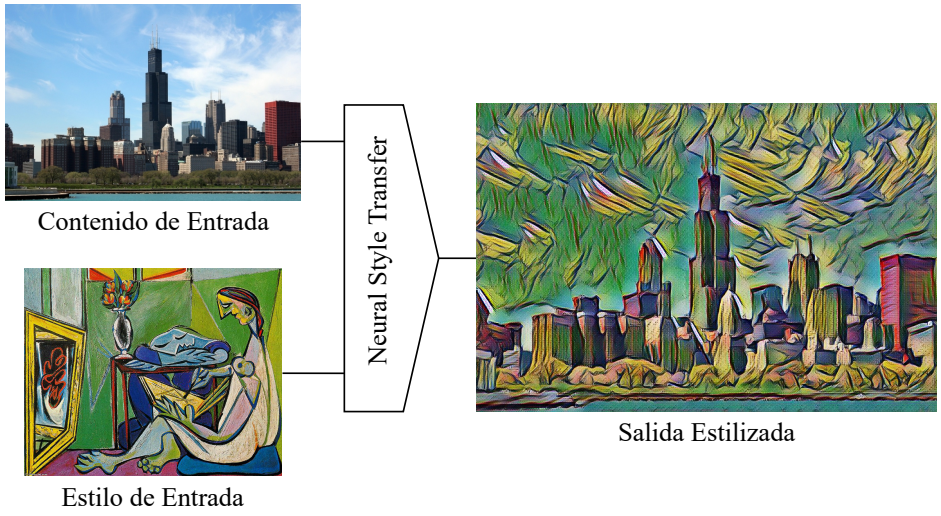


Figura 2.1: Diagrama general del algoritmo NST.

Dado que el algoritmo de Gatys et al. [35] no tiene ninguna restricción explícita sobre el tipo de imágenes de estilo y tampoco necesita datos de resultados para el entrenamiento, rompe las limitaciones de los enfoques anteriores. El trabajo de Gatys et al. abrió un nuevo campo llamado Transferencia de Estilo Neuronal o Neural Style Transfer (NST), que es el proceso de usar redes neuronales convolucionales para renderizar una imagen de contenido en diferentes estilos.

El trabajo de Gatys et al. ha atraído una amplia atención tanto de la academia como

de la industria. En el ámbito académico, se realizaron muchos estudios de seguimiento para mejorar o ampliar este algoritmo NST. Las investigaciones conexas de NST también han dado lugar a muchas aplicaciones industriales exitosas (por ejemplo, Prisma [41], Ostagram [42], Deep Forger [43]).

2.2. Estado Actual de Avance en el Campo de NST

Con el fin de dar a conocer el estado del arte en el campo de la transferencia de estilo neuronal, se presentan los trabajos de investigación científica más importantes desarrollados entorno a esta temática. Dado que este campo de investigación es demasiado extenso para ser resumido completamente en este trabajo, se abordarán los desarrollos fundamentales de esta área (de los cuales derivan todos los demás estudios) y sus extensiones más importantes. Se sigue la misma estructura de Ying et al. [31], que presentan un estudio exhaustivo de los algoritmos NST comparándolos cualitativa y cuantitativamente.

Para una mejor comprensión de los avances en NST, es necesario entender que estos algoritmos se pueden generalizar en dos grandes etapas: 1) Una primera parte de modelado de estilo visual, que resuelve el problema de cómo modelar y extraer el estilo de una imagen (dado que el estilo está muy relacionado con la textura[1], una manera directa es relacionar el modelado de estilo visual con métodos de modelado de textura visual, los cuales, han sido profundamente estudiados). 2) Después de obtener la representación de estilo, sigue una segunda etapa de reconstrucción de imagen, que resuelve la tarea de reconstruir la imagen con la información de estilo deseada mientras se preserva su contenido.

De acuerdo a la taxonomía de [31], los métodos actuales de NST se dividen en dos categorías: **Métodos NST basados en optimización de imágenes** o *Image-Optimization-Based Online Neural Methods* (IOB-NST) y **Métodos NST basados en optimización de modelos** o *Model-Optimization-Based Offline Neural Methods* (MOB-NST). La primera categoría, transfiere el estilo optimizando iterativamente una imagen; en cambio, la segunda categoría, optimiza un modelo generativo y produce la imagen estilizada con una sola inferencia del modelo. En la Figura 2.2 se presenta un diagrama general de los métodos NST presentados en este capítulo.

[1]Se aclara que el estilo está muy relacionado con la textura, pero no limitado a la textura. El estilo también implica un gran grado de simplificación y efectos de abstracción de forma, que se remonta a la composición o alineación de las características de textura.

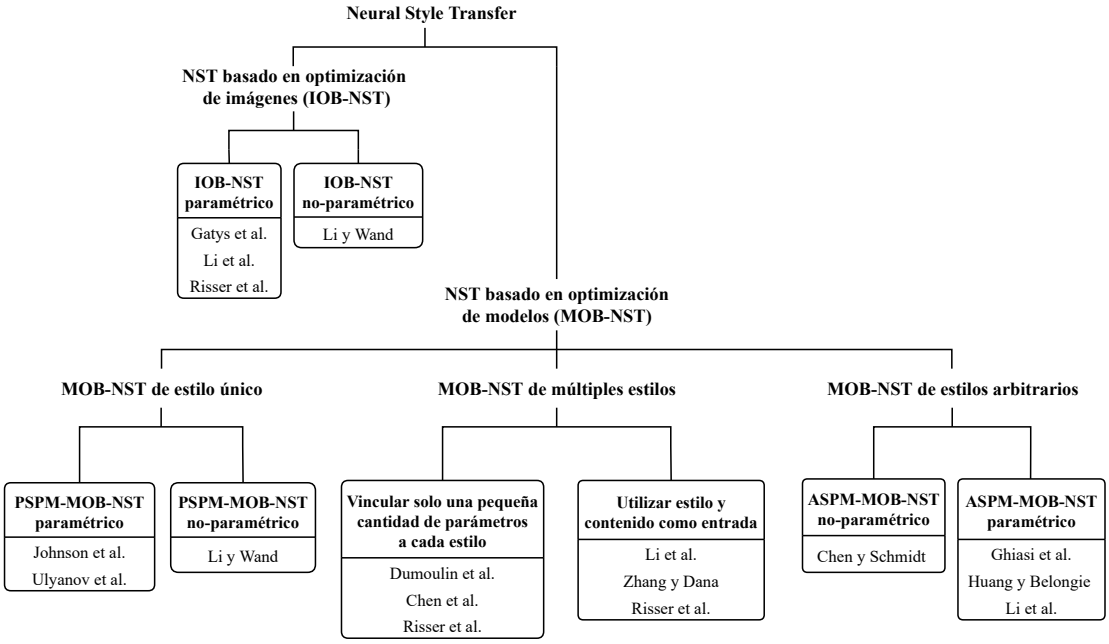


Figura 2.2: Estado del Arte en NST.

2.2.1. Métodos NST Basados en Optimización de Imágenes (IOB-NST)

La idea básica de esta categoría de algoritmos es modelar y extraer información de estilo y contenido de las imágenes de estilo y contenido correspondientes, recombinarlas como la representación objetivo y luego, reconstruir iterativamente un resultado estilizado que coincida con la representación objetivo. En general, los diferentes algoritmos IOB-NST comparten la misma técnica de *reconstrucción de imagen* [44, 45], pero difieren en la forma en que *modelan el estilo visual*. La limitación común de los algoritmos IOB-NST es que son computacionalmente costosos, debido al procedimiento iterativo de optimización de imágenes.

2.2.1.1. IOB-NST Paramétrico

El primer subconjunto de métodos IOB-NST se basa en el modelado de textura paramétrico con estadísticas descriptiva, donde el estilo se caracteriza por ser un conjunto de estadísticas de resumen. Gatys et al. [46] son los primeros en medir estadísticas en el dominio de una red neuronal convolucional para la síntesis de texturas. Ellos diseñan una representación basada en la matriz de Gram para modelar texturas, que son las

correlaciones entre los mapas de características en diferentes capas de una red de clasificación previamente entrenada (red VGG [47]). Específicamente, la representación basada en matriz de Gram codifica las estadísticas de segundo orden del conjunto de respuestas de los filtros de una CNN. Esta representación de textura basada en la matriz de Gram es eficaz para modelar una amplia variedad de texturas.

Posteriormente, Gatys et al. [35, 40] introducen el primer algoritmo de NST, utilizando la misma idea de modelado de texturas basado en redes neuronales convolucionales [46] para la transferencia de estilo. Al reconstruir representaciones de capas intermedias de la red VGG-19, observan que una red neuronal convolucional profunda es capaz de extraer información de contenido de una imagen arbitraria, y cierta información de la apariencia (estilo) de una obra de arte conocida.

Para transferir el estilo de una obra de arte a una imagen arbitraria, se sintetiza una nueva imagen que coincida simultáneamente con la representación del contenido de la imagen arbitraria, y la representación del estilo de la obra de arte. Si bien se conserva la disposición global de la imagen original, la obra de arte proporciona los colores y las estructuras locales que componen el escenario global. Efectivamente, esto convierte la imagen original en el estilo de la obra de arte, de manera que la apariencia de la imagen sintetizada se asemeja a la obra de arte, aunque muestre el mismo contenido que la imagen original.

Dada una imagen de contenido I_c y una imagen de estilo I_s, el algoritmo intenta buscar una nueva imagen estilizada I que minimice la siguiente función objetivo:

$$I^* = \arg\min_I \mathcal{L}_{total}(I_c, I_s, I) = \arg\min_I \ \alpha\mathcal{L}_c(I_c, I) + \beta\mathcal{L}_s(I_s, I) \tag{2.1}$$

Donde $\mathcal{L}_c$ compara la representación de contenido de una imagen de contenido dada con la de la imagen estilizada, y $\mathcal{L}_s$ compara la representación de estilo derivada de una imagen de estilo con la de la imagen estilizada. Además, α y β se utilizan para equilibrar la componente de contenido y la componente estilo en el resultado estilizado.

La pérdida de contenido $\mathcal{L}_c$ se define por la distancia euclidiana al cuadrado entre las representaciones de características $\mathcal{F}^l(I_c)' \in \mathbb{R}^{C_l \times (H_l W_l)}$ de la imagen de contenido I_c en la capa l, y la representación de la imagen estilizada I que se inicializa con una imagen de ruido:

$$\mathcal{L}_c = \sum_{l \in \{l_c\}} \left\| \mathcal{F}^l(I_c)' - \mathcal{F}^l(I)' \right\|^2 \tag{2.2}$$

Donde $\{l_c\}$ denota el conjunto de capas de la red VGG para calcular la pérdida de contenido, C_l representa el número de filtros en la capa l, y H_l y W_l representan el alto y el ancho del mapa de características en la capa l. Note que $\mathcal{F}^l(I_c)' \in \mathbb{R}^{C_l \times (H_l W_l)}$ es una versión redimensionada del mapa de características $\mathcal{F}^l(I_c) \in \mathbb{R}^{H_l \times W_l \times C_l}$ de la imagen de contenido I_c en la capa l.

Para obtener una representación del estilo de una imagen de entrada, se usa un espacio de características diseñado para capturar información de textura [46]. Este espacio de características se puede construir sobre las respuestas de los filtros en cualquier capa de la red. Consiste en las correlaciones entre las diferentes respuestas de los filtros sobre la extensión espacial de los mapas de características. Estas correlaciones de características vienen dadas por la matriz de Gram $\mathcal{G}\left(\mathcal{F}^l(I_s)'\right) \in \mathbb{R}^{C_l \times N_l}$ sobre la representación de características $\mathcal{F}^l(I_s) \in \mathbb{R}^{C_l \times (H_l W_l)}$ que se calcula como:

$$\mathcal{G}\left(\mathcal{F}^l(I_s)'\right) = \left[\mathcal{F}^l(I_s)'\right]\left[\mathcal{F}^l(I_s)'\right]^T \tag{2.3}$$

Al incluir las correlaciones de características de múltiples capas, se obtiene una representación estacionaria de múltiples escalas de la imagen de entrada, que captura su información de textura, pero no la disposición global.

Finalmente, la pérdida de estilo define por la distancia euclidiana al cuadrado entre las representaciones de estilo basadas en la matriz de Gram de I_s e I:

$$\mathcal{L}_s = \sum_{l \in \{l_s\}} \left\| \mathcal{G}\left(\mathcal{F}^l\left(I_s\right)'\right) - \mathcal{G}\left(\mathcal{F}^l(I)'\right) \right\|^2 \tag{2.4}$$

Donde $\{l_s\}$ representa el conjunto de capas de la red VGG para calcular la pérdida de estilo.

La elección de las capas de contenido y estilo es un factor importante en el proceso de transferencia de estilo. Diferentes posiciones y números de capas pueden resultar en experiencias visuales muy diferentes. Dada la red pre-entrenada VGG-19 [47] como la red de pérdidas, la elección de $\{l_s\}$ y $\{l_c\}$ de Gatys et al. [35] es: $\{l_s\} = \{conv1\_1, conv2\_1, conv3\_1, conv4\_1, conv5\_1\}$ y $\{l_c\} = \{conv4\_2\}$. Notar que cuando se refiere a la capa $convX\_1$ en la red VGG, hace referencia a la salida luego de la función ReLU en la capa convolucional.

Para $\{l_s\}$, la idea de combinar múltiples capas (hasta capas superiores) es fundamental para el éxito del algoritmo NST de Gatys et al [35]. Hacer coincidir las representaciones de estilo con las capas superiores de la red conserva las estructuras de las

imágenes locales a una escala cada vez mayor, lo que conduce a una experiencia visual más fluida y continua. Esto implica que las imágenes visualmente más atractivas generalmente se crean haciendo coincidir la representación del estilo con capas altas en la red.

Para $\{l_c\}$, conviene coincidir el contenido en solo una capa superior de la red, esto se debe que las redes neuronales convolucionales que se entrenan en el reconocimiento de objetos, desarrollan una representación de la imagen que hace que la información del objeto sea cada vez más explícita a lo largo de la jerarquía de procesamiento [46]. Por lo tanto, a lo largo de la jerarquía de procesamiento de la red, la imagen de entrada se transforma en representaciones que son cada vez más sensibles al contenido real de la imagen, pero se vuelven relativamente invariantes a su apariencia precisa. Así, las capas más altas de la red capturan el contenido de alto nivel en términos de objetos y su disposición en la imagen de entrada, pero no restringen mucho los valores exactos de pixel de la reconstrucción. Por el contrario, las reconstrucciones de las capas inferiores simplemente reproducen los valores exactos de pixel de la imagen original. Por lo tanto, Gatys et al. [35] se refiere a las respuestas de características en capas superiores de la red como la representación de contenido.

Dicho de otro modo, al hacer coincidir el contenido en una capa inferior de la red, el algoritmo hace coincidir gran parte de la información detallada de los pixeles en la imagen de contenido, y la imagen generada aparece como si la textura de la obra de arte simplemente se mezclara sobre la imagen de contenido. Por el contrario, al coincidir las características del contenido en una capa superior de la red, la información detallada de los pixeles de la imagen de contenido no es tan restrictiva, y la textura de la obra de arte con el contenido de la imagen de contenido se fusionan correctamente. Es decir, las estructuras finas de la imagen, por ejemplo, los bordes y el mapa de color, se alteran de tal manera que coincidan con el estilo de la obra de arte mientras muestra el contenido de la imagen de contenido.

En la ecuación (2.1), tanto $\mathcal{L}_c$ como $\mathcal{L}_s$ son diferenciables. Por lo tanto, con ruido aleatorio como I inicial, la ecuación (2.1) se puede minimizar mediante el uso del descenso del gradiente en el espacio de la imagen con backpropagation. Además, en la práctica se suele añadir un término de eliminación de ruido de variación total para fomentar el suavizado en el resultado estilizado.

En la categoría de métodos IOB-NST paramétricos existen investigaciones posteriores que reutilizan el algoritmo de Gatys et al. [35] realizando pequeñas modificaciones

para solucionar alguna limitación del algoritmo. Li et al. [48] realizan transferencia de estilo con otras representaciones de estilo derivadas de la representación basada en la matriz de Gram. Por otro lado, Risser et al. [49] corrigen algunas inestabilidades en el algoritmo agregando un nuevo término de error, una pérdida de histograma adicional. Por último, Li et al. [50] mejoran el resultado visual de la transferencia de estilo incorporando restricciones adicionales sobre las características de bajo nivel en el espacio de pixeles con una pérdida laplaciana adicional.

2.2.1.2. IOB-NST No-Paramétrico

IOB-NST no paramétrico se construye sobre la base del modelado de texturas no paramétrico con MRF (*Markov Random Fields*). Esta categoría considera NST a nivel local, es decir, opera en parches de la imagen para que coincida con el estilo.

Li y Wand [51] son los primeros en proponer un algoritmo NST basado en MRF. Encuentran que el método NST paramétrico con estadísticas de resumen solo captura las correlaciones de las características por pixel y no restringe el diseño espacial, lo que conduce a un resultado menos plausible visualmente para estilos fotorrealistas. Su solución es modelar el estilo de una manera no paramétrica e introducir una nueva función de pérdida de estilo que incluye MRF basado en parches:

$$\mathcal{L}_s = \sum_{l \in \{l_s\}} \sum_{i=1}^{m} \left\| \Psi_i \left(\mathcal{F}^l(I) \right) - \Psi_{NN(i)} \left(\mathcal{F}^l(I_s) \right) \right\|^2 \tag{2.5}$$

Donde $\Psi \left(\mathcal{F}^l(I) \right)$ es el conjunto de todos los parches locales del mapa de características $\mathcal{F}^l(I)$. Ψ_i denota el i-ésimo parche local y $\Psi_{NN(i)}$ es el parche de estilo más similar con el i-ésimo parche local en la imagen estilizada I. La mejor coincidencia $\Psi_{NN(i)}$ se obtiene calculando la correlación cruzada normalizada $NN(i)$ sobre todos los parches de estilo en la imagen de estilo I_s. m es el número total de parches locales. Dado que su algoritmo coincide con un estilo en el nivel de parche, la estructura fina y la disposición se pueden conservar mucho mejor.

La ventaja del algoritmo de Li y Wand [51] es que funciona especialmente bien para estilos fotorrealistas, o más específicamente, cuando la foto de contenido y el estilo son similares en forma y perspectiva, debido a la pérdida de MRF basada en parches. Sin embargo, generalmente falla cuando el contenido y el estilo de las imágenes tienen fuertes diferencias en perspectiva y estructura, ya que los parches de la imagen no pueden coincidir correctamente. También es limitado en la preservación de los detalles

agudos y la información de profundidad.

2.2.2. Métodos NST Basados en Optimización de Modelos (MOB-NST)

Aunque IOB-NST puede producir imágenes estilizadas impresionantes, todavía existen algunas limitaciones. La limitación más preocupante es la cuestión de la eficiencia. La segunda categoría, MOB-NST, aborda el problema de la velocidad y el costo computacional mediante una red *feed-forward* g, que se optimiza sobre un gran conjunto de imágenes I_c para una o más imágenes de estilo I_s:

$$\theta^* = \arg\min_{\theta} \mathcal{L}_{total}\left(I_c, I_s, g_{\theta^*}(I_c)\right), \quad I^* = g_{\theta^*}(I_c) \tag{2.6}$$

Dependiendo del número de estilos artísticos que puede producir una sola red generadora g, los algoritmos MOB-NST se dividen en: **Métodos MOB-NST de estilo único** o *Per-Style-Per-Model MOB-NST Methods* (PSPM-MOB-NST), **Métodos MOB-NST de múltiples estilos** o *Multiple-Style-Per-Model MOB-NST Methods* (MSPM-MOB-NST) y **Métodos MOB-NST de estilos arbitrarios** o *Arbitrary-Style-Per-Model MOB-NST Methods* (ASPM-MOB-NST).

2.2.2.1. MOB-NST de Estilo Único (PSPM-MOB-NST)

PSPM-MOB-NST Paramétrico

El primer algoritmo MOB-NST es propuesto por Johnson et al. [52]. Este método consiste en entrenar previamente una red *feed-forward* de estilo específico y producir un resultado estilizado con una sola inferencia en la etapa de prueba. La arquitectura de red propuesta por Johnson et al. sigue aproximadamente el diseño de red propuesto por Radford et al. [53] pero con bloques residuales. Además, la red, en lugar de usar capas de *pooling*, utiliza capas convoluciones con *stride* 2 para reducir el tamaño de los mapas de características en la etapa de codificación y capas convolucionales con *stride* $1/2$ para la etapa de decodificación. La función objetivo es similar al algoritmo de Gatys et al. [35] y agrega un término de regularización a la pérdida total para mejorar la suavidad espacial en la imagen de salida.

El algoritmo de Johnson et al. logra una transferencia de estilo en tiempo real. Sin embargo, el diseño de su algoritmo sigue básicamente el algoritmo de Gatys et al., lo

que les hace sufrir de los mismos problemas antes mencionados para el algoritmo de Gatys et al., por ejemplo, una falta de consideración en la coherencia de los detalles y la información de profundidad.

Poco después de [52], Ulyanov et al. [54] descubren que la simple aplicación de la normalización a cada imagen por separado en lugar de un lote de imágenes (precisamente la normalización por lotes o *batch normalization* (BN)), conduce a una mejora significativa en la calidad de la estilización. Esta normalización de imagen única se denomina normalización de instancia o *instance normalization* (IN), que es equivalente a la normalización por lotes cuando el tamaño del lote se establece en 1. Se muestra que la red de transferencia de estilo con IN converge más rápido que con BN, y también logra mejores resultados visuales. Una interpretación es que IN es una forma de normalización de estilo, y puede normalizar directamente el estilo de cada imagen de contenido al estilo deseado [55]. Por lo tanto, el objetivo es más fácil de aprender ya que el resto de la red solo necesita hacerse cargo de la pérdida de contenido.

PSPM-MOB-NST No-Paramétrico

Inspirados en su trabajo previo, el algoritmo NST basado en MRF [51] de la Sección 2.2.1.2, Li y Wand [56] proponen un método MOB-NST no-paramétrico que aborda el problema de la eficiencia mediante la formación de una red *feed-forward* de Markov con entrenamiento adversario. Similar a [51], su algoritmo es un método no-paramétrico basado en parches con MRF. Se ha demostrado que su método supera al algoritmo de Johnson et al. [52] en la preservación de la coherencia de texturas en imágenes complejas, gracias a su diseño basado en parches. Sin embargo, su algoritmo tiene un rendimiento menos satisfactorio con estilos sin textura (por ejemplo, imágenes de caras), ya que su algoritmo carece de consideración semántica. Otras debilidades de su algoritmo incluyen la falta de consideración en la información de profundidad y las variaciones de trazos de pincel, que son factores visuales importantes.

2.2.2.2. MOB-NST de Múltiples Estilos (MSPM-MOB-NST)

Aunque los enfoques de PSPM anteriores pueden producir imágenes estilizadas dos órdenes de magnitud más rápido que los métodos IOB-NST anteriores, se deben entrenar redes generativas separadas para cada imagen de estilo en particular, lo que requiere bastante tiempo y es inflexible. Pero muchas pinturas (por ejemplo, pinturas impresionistas) comparten trazos de pintura similares y solo difieren en sus paletas de

colores. Intuitivamente, es redundante entrenar una red separada para cada uno de ellos. Por lo tanto, se propone MSPM, que mejora la flexibilidad de PSPM al incorporar múltiples estilos en un solo modelo. En general, hay dos caminos para manejar este problema: 1) vincular una pequeña cantidad de parámetros de la red a cada estilo [57, 58] y 2) seguir ocupando solo una red como PSPM, pero utilizando estilo y contenido como entradas [59, 60].

Vincular Solo Una Pequeña Cantidad de Parámetros a Cada Estilo

El trabajo de Dumoulin et al. [57] se basa en la capa IN propuesta en el algoritmo PSPM [54] (Sección 2.2.2.1). Sorprendentemente, encuentran que el uso de los mismos parámetros convolucionales, pero escalando y trasladando los parámetros en las capas IN es suficiente para modelar diferentes estilos. Por lo tanto, proponen un algoritmo para entrenar una red de transferencia de múltiples estilos basada en la normalización de instancia condicional o *conditional instance normalization* (CIN), que se define como:

$$\text{CIN}(\mathcal{F}(I_c), s) = \gamma^s \left(\frac{\mathcal{F}(I_c) - \mu\left(\mathcal{F}(I_c)\right)}{\sigma\left(\mathcal{F}(I_c)\right)} \right) + \beta^s \tag{2.7}$$

Donde $\mathcal{F}$ es el mapa de activación de entrada a la capa, s es el índice del estilo deseado de un conjunto de imágenes de estilo, y μ y σ son la media y la desviación estándar de $\mathcal{F}(I_c)$ a través de los ejes espaciales. Las constantes γ_s y β_s se obtienen seleccionando la fila correspondiente a s en las matrices γ y β como se muestra en la Figura 2.3.

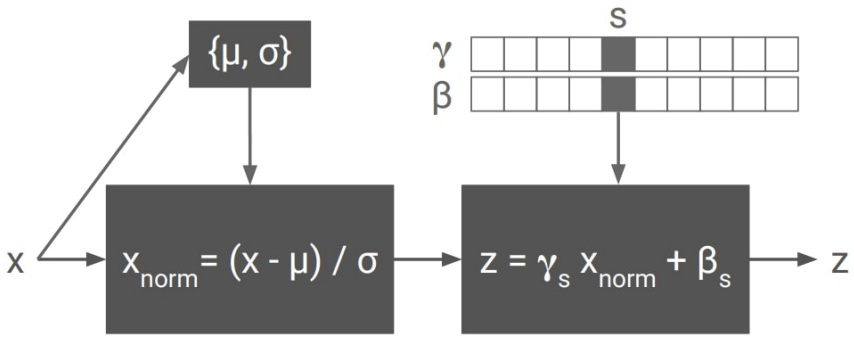


Figura 2.3: Normalización de instancia condicional. El mapa de activación de entrada x ($\mathcal{F}$ en la notación utilizada en la ecuación (2.7)) se normaliza en ambas dimensiones espaciales. Posteriormente, se escala y se desplaza utilizando los vectores de parámetros dependientes del estilo, γ_s y β_s, donde s indexa la etiqueta de estilo.

Como se muestra en la ecuación (2.7), la adaptación de cada estilo I_s se realiza escalando y trasladando con los parámetros γ_s y β_s después de normalizar el mapa de activación $\mathcal{F}(I_c)$, es decir, cada estilo I_s se puede lograr ajustando los parámetros de una transformación afín. La interpretación es similar a la de [54] en la Sección 2.2.2.1, es decir, la normalización de las estadísticas de los mapas de características con diferentes parámetros afines puede normalizar la imagen de contenido de entrada a diferentes estilos. Además, el algoritmo de Dumoulin et al. [57] también se puede ampliar para combinar varios estilos en un único resultado estilizado mediante la combinación de parámetros afines de diferentes estilos.

Chen et al. [58] proponen otro algoritmo que sigue el primer camino del MSPM. Su idea es desacoplar explícitamente el estilo y el contenido, es decir, utilizar componentes de red separados para aprender el contenido y la información de estilo correspondientes. Más específicamente, utilizan filtros convolucionales de nivel medio (llamados capa *StyleBank*) para aprender individualmente diferentes estilos. Cada estilo está vinculado a un conjunto de parámetros en la capa *StyleBank*. El resto de los componentes de la red se utilizan para aprender información de contenido, que es compartida por diferentes estilos. Su algoritmo también admite un entrenamiento incremental flexible, que consiste en ajustar los componentes de contenido en la red y solo entrenar una capa *StyleBank* para un nuevo estilo.

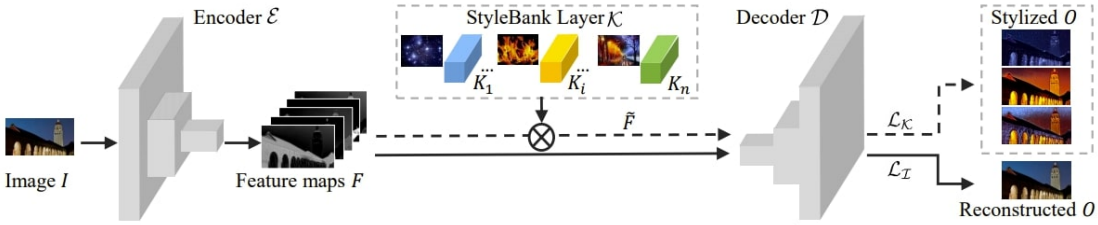


Figura 2.4: Arquitectura de red de [58]. Consta de tres módulos: codificador de imágenes $\mathcal{E}$, capa *StyleBank* $\mathcal{K}$ y decodificador de imágenes $\mathcal{D}$.

En resumen, tanto los algoritmos de Dumoulin et al. [57] y Chen et al. [58] tienen los beneficios de los pequeños esfuerzos necesarios para aprender un nuevo estilo y un control flexible sobre la fusión de estilos. Sin embargo, no abordan las limitaciones comunes de los algoritmos NST, por ejemplo, la falta de detalles, semántica, profundidad y variaciones en las pinceladas.

Utilizar Estilo y Contenido Como Entradas

Una desventaja de la primera categoría es que el tamaño del modelo generalmente se hace más grande con el aumento del número de estilos aprendidos. La segunda ruta de MSPM aborda esta limitación explorando completamente la capacidad de una sola red y combinando tanto contenido como estilo en la red para la identificación de estilo. Diferentes algoritmos MSPM difieren en la forma de incorporar estilo en la red.

En [59], dados N estilos de destino, Li et al. diseñan una unidad de selección para escoger el estilo, que es un vector *one-hot* N-dimensional. Cada bit en la unidad de selección representa un estilo I_s específico en el conjunto de estilos objetivo. Para cada bit en la unidad de selección, Li et al. primero muestrean un mapa de ruido $f(I_s)$ de una distribución uniforme y luego alimentan la sub-red de estilo con $f(I_s)$ para obtener las características codificadas de estilo correspondientes $\mathcal{F}(f(I_s))$. Al introducir la concatenación de las características codificadas de estilo $\mathcal{F}(f(I_s))$ y las características codificadas de contenido $Enc(I_c)$ en la parte del decodificador Dec de la red de transferencia de estilo, se puede obtener el resultado estilizado deseado: $I = Dec\left(\mathcal{F}(f(I_s))^\frown Enc(I_c)\right)$.

Por otro lado, Zhang y Dana [60] proponen una capa *CoMatch* para la transferencia de múltiples estilos (Figura 2.5). Como parte de la red generadora, una red siamesa comparte pesos con el codificador de la red de transformación para captura las características de la imagen de estilo. Luego, la red de transformación codifica la imagen de contenido y la hace coincidir con las estadísticas del mapa de características de la imagen de estilo a través de la capa *CoMatch*. Finalmente, en el tramo de decodificación de la red de transformación, se obtiene como *output* la imagen estilizada.

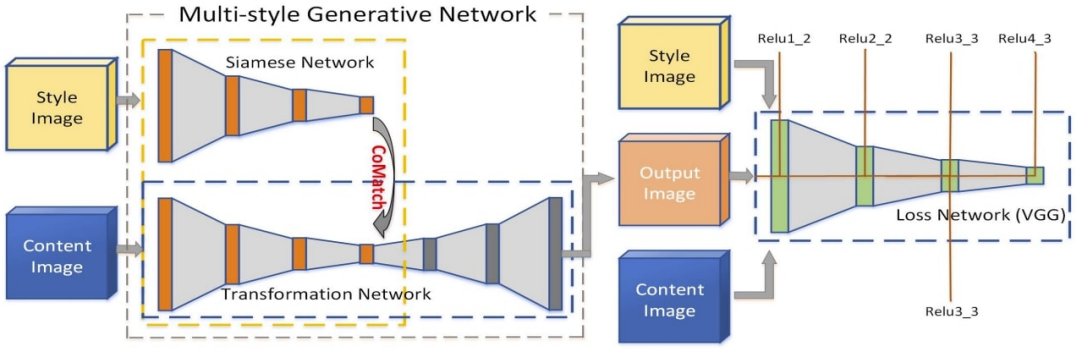


Figura 2.5: Visión general de la arquitectura propuesta por [60]. MSG-Net: *Multi-style Generative Network*.

La capa *CoMatch* está diseñada para ajustar el tamaño de la salida de la red siamesa a fin de que coincida con la dimensión deseada, y también posee una matriz de peso entrenable con el propósito de ajustar los mapas de características para ayudar a minimizar la función objetivo. Durante el entrenamiento, Zhang y Dana [60] entrenan la red con diferentes tamaños de imagen de estilo para aprender diferentes tamaños de pinceladas. Después del entrenamiento, el tamaño del trazo del pincel puede ser una opción para el usuario al cambiar el tamaño de la imagen de estilo de entrada.

El segundo tipo de MSPM aborda la limitación del primer tipo, en que el tamaño del modelo generalmente se hace más grande con el aumento del número de estilos aprendidos. A un costo, la escalabilidad de estilo del segundo tipo de MSPM es mucho más pequeña, ya que solo una red se utiliza para múltiples estilos. Además, algunas limitaciones antes mencionadas en el primer tipo de MSPM todavía existen, es decir, el segundo tipo de algoritmos MSPM aún está limitado en la preservación de la coherencia de las estructuras finas y la información de profundidad.

2.2.2.3. MOB-NST de Estilos Arbitrarios (ASPM-MOB-NST)

La tercera categoría, ASPM-MOB-NST, apunta a un modelo para todos los estilos, es decir, un modelo único para transferir estilos artísticos arbitrarios. También hay dos tipos de ASPM, uno basado en el modelado de texturas no-paramétrico con MRF y el otro basado en el modelado de texturas paramétrico con estadísticas de resumen.

ASPM-MOB-NST No-Paramétrico

Chen y Schmidt proponen el primer algoritmo ASPM [61]. Primero extraen un conjunto de parches de activación de los mapas de características de las imágenes de estilo y contenido calculados en una red VGG previamente entrenada. Luego, coinciden cada parche de contenido con el parche de estilo más parecido, basados en la medida de la correlación cruzada normalizada, y los intercambian (a este procedimiento, [61] lo llama Intercambio de Estilo o *Style Swap*). El resultado estilizado se puede producir reconstruyendo el mapa de activación resultante después del *Style Swap*. El algoritmo de Chen y Schmidt es más flexible que los enfoques anteriores debido a su característica de un modelo único para todos los estilos. Pero los resultados estilizados de [61] son menos atractivos ya que los parches de contenido se intercambian normalmente con los parches de estilo que no son representativos del estilo deseado. Como resultado,

el contenido está bien conservado, mientras que el estilo generalmente no está bien reflejado.

ASPM-MOB-NST Paramétrico

Considerando [57] de la sección 2.2.2.2, el enfoque más sencillo para la transferencia de estilo arbitraria es entrenar una red P adicional para predecir los parámetros γ_s y β_s en la ecuación (2.7) con una serie de estilos de entrenamiento [62]. Dada una imagen de estilo I_s, las capas CIN en la red de transferencia de estilo toman los parámetros γ_s y β_s desde $P(I_s)$, y normalizan la imagen de contenido de entrada al estilo deseado con una sola inferencia.

Huang y Belongie [55] proponen otro enfoque similar basado en [57]. En lugar de entrenar una red de predicción de parámetros, Huang y Belongie proponen modificar la normalización de instancia condicional en la Ecuación (2.7) por la normalización de instancia adaptativa o *adaptive instance normalization* (AdaIN):

$$\text{AdaIN}(\mathcal{F}(I_c), \mathcal{F}(I_s)) = \sigma(\mathcal{F}(I_s)) \left(\frac{\mathcal{F}(I_c) - \mu(\mathcal{F}(I_c))}{\sigma(\mathcal{F}(I_c))} \right) + \mu(\mathcal{F}(I_s)) \tag{2.8}$$

La capa AdaIN recibe una entrada de contenido I_c y una entrada de estilo I_s, y simplemente alinea la media y la varianza de $\mathcal{F}(I_c)$ para que coincida con las de $\mathcal{F}(I_s)$. Al igual que con IN, estas estadísticas se calculan a través de los ejes espaciales. A diferencia de BN, IN y CIN, AdaIN no tiene parámetros afines entrenables. En su lugar, calcula adaptativamente los parámetros afines a partir de la entrada de estilo.

A diferencia de [57], el codificador en la red de transferencia de estilo de [55] es fijo y comprende las primeras capas en la red VGG pre-entrenada. Por lo tanto, $\mathcal{F}$ en [55] es el mapa de activación desde la red VGG. La parte del decodificador necesita ser entrenada con un gran conjunto de imágenes de estilo y contenido para decodificar los mapas de características resultantes después de AdaIN al resultado estilizado: $I = Dec\left(\text{AdaIN}(\mathcal{F}(I_c), \mathcal{F}(I_s))\right)$.

El algoritmo de Huang y Belongie es el primer algoritmo ASPM que logra una estilización en tiempo real. Sin embargo, este algoritmo está basado en datos y limitado en la generalización de los estilos no vistos. Además, el simple hecho de ajustar la media y la varianza de los mapas de activación hace que sea difícil sintetizar patrones de estilo complicados con ricos detalles.

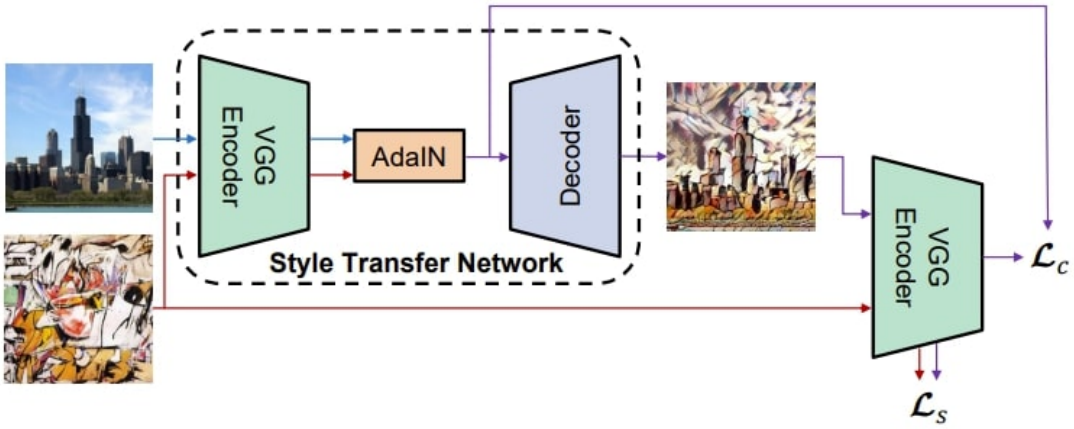


Figura 2.6: Descripción general de la arquitectura propuesta por [55]. Se utilizan las primeras capas de una red VGG-19 fija para codificar el contenido y el estilo. Una capa AdaIN se utiliza para realizar la transferencia de estilo en el espacio de características. Se entrena un decodificador para invertir la salida de la capa AdaIN a los espacios de la imagen. Se utiliza el mismo codificador VGG para calcular la pérdida de contenido y estilo.

Un trabajo más reciente de Li et al. [63] intenta explotar una serie de transformaciones de características para transferir un estilo artístico arbitrario de una manera libre de aprendizaje de estilo. De forma similar a [55], Li et al. utilizan las primeras capas de la red VGG pre-entrenada como codificador y entrenan el decodificador correspondiente. Pero sustituyen la capa de AdaIN [55] entre el codificador y el decodificador por un par de transformaciones de blanqueamiento y coloración o *whitening and coloring transforms* (WCT): $I = Dec\left(\text{WCT}(\mathcal{F}(I_c), \mathcal{F}(I_s))\right)$.

Su algoritmo se basa en la observación de que la transformación de blanqueamiento puede eliminar la información relacionada con el estilo y preservar la estructura del contenido. Por lo tanto, al recibir mapas de activaciones de contenido $\mathcal{F}(I_c)$ desde el codificador, la transformación de blanqueamiento puede filtrar el estilo original de la imagen de contenido de entrada y devolver una representación filtrada con solo información de contenido. Entonces, aplicando la transformación de coloración, los patrones de estilo contenidos en $\mathcal{F}(I_s)$ se incorporan en la representación de contenido filtrado, y el resultado estilizado I se puede obtener decodificando las características transformadas. [63] también extiende esta estilización de un solo nivel a la estilización de varios niveles para mejorar aún más la calidad visual.

El algoritmo de Li et al. [63] es el primer algoritmo ASPM para transferir estilos artísticos de una manera libre de aprendizaje. Por lo tanto, en comparación con [55],

no tiene la limitación de las capacidades de generalización en estilos no vistos. Pero el algoritmo de Li et al. todavía no es eficaz para producir detalles nítidos y trazos finos. Además, carece de consideración en la conservación de la información de profundidad y las variaciones en las pinceladas.

2.2.3. Mejoras y Extensiones

Desde la aparición de los algoritmos NST, también hay algunas investigaciones dedicadas a mejorar los actuales algoritmos NST mediante el control de factores de percepción (por ejemplo, control de tamaño del trazo del pincel, control espacial de la transferencia de estilo o control de color). Además, todos los métodos NST mencionados están diseñados para imágenes fijas generales, y pueden no ser apropiados para tipos especializados de imágenes y videos (por ejemplo, bocetos o *frames* de video). Por lo tanto, una variedad de estudios de seguimiento tiene como objetivo extender algoritmos NST generales a estos tipos particulares de imágenes e incluso extenderlos más allá del estilo de imagen artística (por ejemplo, estilo de audio).

2.2.3.1. Control de los Factores de Percepción en NST

Control Espacial

Los propios Gatys et al. [64] proponen ligeras modificaciones para mejorar su algoritmo anterior [35]. Presentan una estrategia para controlar espacialmente la transferencia de estilo. El objetivo es controlar qué región de la imagen de estilo se utiliza para estilizar cada región en la imagen de contenido. Para llevar a cabo este objetivo se definen canales de guías espacial tanto para el contenido como para la imagen de estilo. Cada uno de ellos es un mapa de imagen de valores en $\{0, 1\}$ especificando qué estilos deben aplicarse a qué región de contenido, es decir, las regiones de contenido donde el canal de guía de contenido r-ésimo es igual a 1 deben representarse con el estilo donde el canal de guía de estilo r-ésimo es igual a 1.

Control de Color

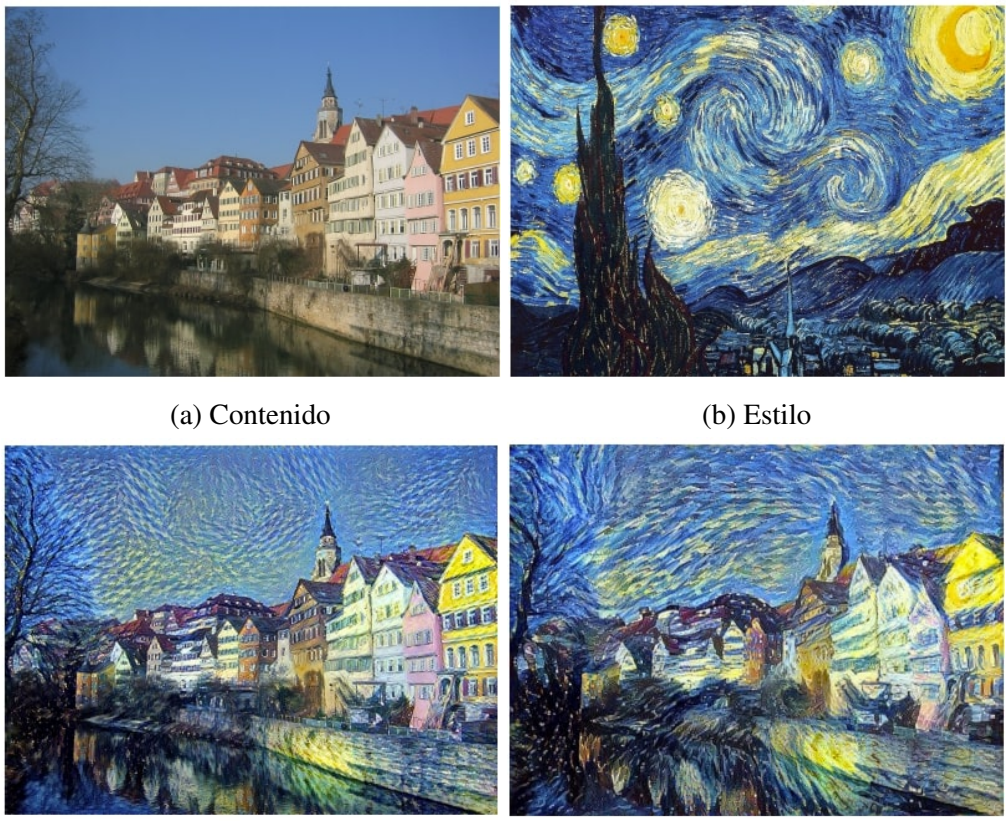
El algoritmo NST original produce imágenes estilizadas con la distribución de color de la imagen de estilo. Sin embargo, a veces se prefiere una transferencia de estilo que preserve el color de la imagen de contenido. La solución correspondiente es primero transformar los colores de la imagen de estilo para que coincidan con los colores de la

imagen de contenido antes de la transferencia de estilo, o bien realizar la transferencia de estilo sólo en el canal de luminosidad.

Control del Tamaño del Trazo

Para el control del tamaño del trazo, el problema es mucho más complejo. En la Figura 2.7 se muestran ejemplos del control de tamaño del trazo. Las discusiones sobre la estrategia de control del tamaño de la carrera deben dividirse en varios casos [65]:

IOB-NST sin imágenes de alta resolución. Dado que las estadísticas de estilo actuales (por ejemplo, estadísticas basadas en matriz de Gram o BN) son sensibles a la escala [65], para lograr diferentes tamaños de trazo, la solución es simplemente redimensionar la imagen de estilo dada a diferentes escalas.



(a) Contenido (b) Estilo

(c) Tamaño de trazo pequeño (d) Tamaño de trazo grande

Figura 2.7: Control del tamaño de trazo en NST. (c) es la salida con un tamaño de pincel más pequeño y (d) con un tamaño de pincel más grande. La imagen de estilo es "La noche estrellada"de Vincent van Gogh.

MOB-NST sin imágenes de alta resolución. Una solución posible es cambiar el tamaño de la imagen de entrada a diferentes escalas antes de la inferencia, lo que inevitablemente perjudica la calidad de estilización. Otra solución posible es entrenar varios modelos con diferentes escalas de una imagen de estilo, lo que requiere mucho tiempo y espacio. Además, esta solución no puede preservar la consistencia del trazo entre los resultados con diferentes tamaños de trazo, es decir, los resultados varían en las orientaciones del trazo, configuraciones del trazo, etc. Para abordar este problema, Jing et al. [65] proponen un algoritmo PSPM controlable. El componente central de su algoritmo es un módulo *StrokePyramid*, que aprende diferentes tamaños de trazos con campos receptivos adaptativos. Sin sacrificar calidad y velocidad, su algoritmo es el primero en utilizar un solo modelo para lograr un control flexible y continuo del tamaño del trazo, preservando la consistencia, y además lograr un control espacial del tamaño del trazo para producir nuevos efectos artísticos. Aunque también se puede utilizar el algoritmo ASPM para controlar el tamaño del trazo, ASPM cambia la calidad y la velocidad. Como resultado, ASPM no es eficaz para producir trazos y detalles finos en comparación con [65].

IOB-NST con imágenes de alta resolución. Para imágenes de alta resolución (por ejemplo, 3000x3000 píxeles en [64]), no se puede lograr un gran tamaño de trazo simplemente redimensionando la imagen de estilo a gran escala. Dado que los campos receptivos en una CNN tienen un tamaño fijo, el resultado de la estilización depende de la resolución de las imágenes de entrada: la estilización ocurre sólo hasta la escala de los campos receptivos en la salida. En la práctica, [64] observan que para la red VGG-19, hay un punto óptimo alrededor de 500x500 píxeles para el tamaño de las imágenes de entrada, de tal manera que la estilización es atractiva pero el contenido está bien conservado. Para una imagen de alta resolución, sin embargo, los campos receptivos son típicamente muy pequeños en comparación con la imagen, por lo que sólo estructuras a muy pequeña escala son estilizadas. Debido a lo anterior, para una red entrenada con imágenes de alta resolución, modificar el tamaño de la imagen de entrada casi no afecta visualmente el tamaño del trazo en la imagen estilizada. Gatys et al. [64] abordan este problema proponiendo un procedimiento *coarse-to-fine* IOB-NST con varios pasos de *downsampling*, estilización, *upsampling* y estilización final.

MOB-NST con imágenes de alta resolución. Similar al caso anterior, el tamaño del trazo en el resultado estilizado no varía con la escala de la imagen de estilo para MOB-NST entrenados con imágenes de alta resolución. La solución también es similar al algoritmo de Gatys et al. en [64], que es un procedimiento de estilización *coarse-to-fine* [66]. La idea es realizar una transferencia multimodal, que comprende múltiples subredes. Cada subred recibe el resultado estilizado sobremuestreado de la subred anterior como entrada, y la estiliza nuevamente con trazos más finos.

Consideración de la Información de Profundidad

Por último, otra limitación perceptiva de los algoritmos NST actuales es que no consideran la información de profundidad contenida en la imagen. Para abordar esta limitación, se propone el algoritmo NST de preservación de profundidad [67]. Su enfoque es añadir una función de pérdida de profundidad basada en [52] para medir la diferencia de profundidad entre la imagen de contenido y la imagen estilizada. La profundidad de la imagen se adquiere aplicando un algoritmo de estimación de profundidad de una sola imagen (por ejemplo, el trabajo de Chen et al. en [68]).

2.2.3.2. *Semantic Style Transfer*

Dado un par de imágenes de estilo y contenido que son similares en el contenido, el objetivo de la transferencia de estilo semántico es construir una correspondencia semántica entre el estilo y el contenido, que mapea cada región de estilo a una región de contenido semánticamente similar correspondiente. Entonces el estilo en cada región de estilo se transfiere a la región de contenido semánticamente similar.

Lu et al. [69] realizan transferencia de estilo semántico basados en su método de reconstrucción de características ligeras (*lightweight feature reconstruction*) dentro de cada región semántica. El estilo y las imágenes de contenido se envían a través de una red de extracción de características. Las máscaras de segmentación semántica también se envían y se superponen en los mapas de características. Luego, con su método de reconstrucción de características ligera [69], reconstruyen el mapa de características dentro de cada región. El mapa de características finalmente se decodifica en una imagen estilizada.

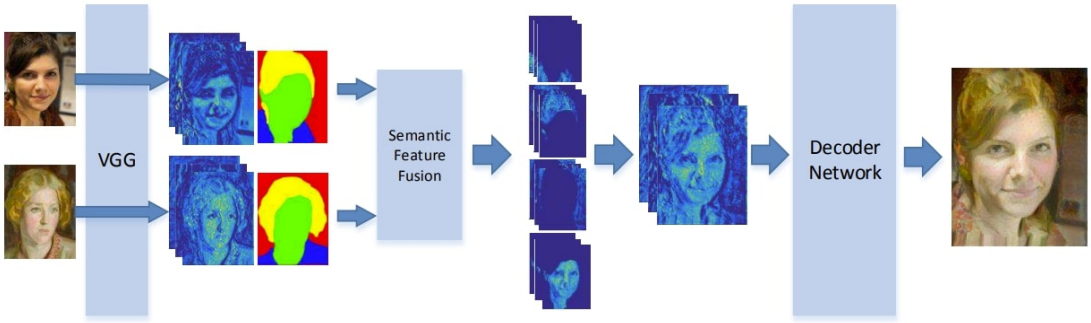


Figura 2.8: Flujo de trabajo del método de Lu et al. [69].

2.2.3.3. *Instance Style Transfer*

La transferencia de estilo de instancias se basa en la segmentación de instancias (*instance segmentation*), y tiene como objetivo estilizar solo un objeto especificado por el usuario dentro de una imagen. El desafío radica principalmente en la transición entre un objeto estilizado y un fondo no estilizado. Castillo et al. [70] abordan este problema añadiendo una pérdida adicional basada en MRF para suavizar y aplicar *anti-aliasing* a los píxeles atípicos cerca de los límites del objeto, para que los objetos estilizados se mezclen naturalmente con su entorno.

2.2.3.4. *Doodle Style Transfer*

Una extensión interesante se puede encontrar en [71], que consiste en aprovechar NST para transformar bocetos en bruto en bellas obras de arte. El método es simplemente descartar el término de pérdida de contenido y usa bocetos como mapa de segmentación para hacer transferencia de estilo semántico.

2.2.3.5. *Stereoscopic Style Transfer*

Impulsados por la demanda de AR/VR, Chen et al. [72] proponen un algoritmo NST estereoscópico para imágenes estereoscópicas. Proponen una pérdida de disparidad para penalizar la disparidad bidireccional. Se muestra que su algoritmo produce trazos más consistentes para diferentes vistas.

2.2.3.6. *Video Style Transfer (VST)*

Los algoritmos NST para secuencias de video se proponen sustancialmente poco después del primer algoritmo NST de Gatys et al. para imágenes fijas [35]. A diferencia de la transferencia de estilo de imagen fija, el diseño del algoritmo de transferencia de estilo de video debe considerar la transición suave entre los fotogramas de video adyacentes. Como antes, los algoritmos relacionados se dividen en VST basados en la optimización de imágenes (IOB-VST) y VST basados en la optimización de modelos (MOB-VST).

IOB-VST

El primer algoritmo de transferencia de estilo de video es propuesto por Ruder et al. [73, 74]. Introducen una pérdida de consistencia temporal basada en el flujo óptico para penalizar las desviaciones a lo largo de trayectorias puntuales. El flujo óptico se calcula utilizando nuevos algoritmos de estimación del flujo óptico [75, 76]. Como resultado, su algoritmo elimina errores temporales entre *frames* y produce videos estilizados suaves. Sin embargo, construyen su algoritmo sobre [35] y necesitan varios minutos para procesar un único *frame*.

MOB-VST

Varios estudios se han dedicado a resolver el problema de estilizar un determinado video en tiempo real. Huang et al. [77] proponen aumentar la pérdida de consistencia temporal de Ruder et al. [73] sobre el algoritmo PSPM. Dados dos *frames* consecutivos, la pérdida de consistencia temporal se calcula directamente utilizando dos salidas correspondientes de la red de transferencia de estilo para reforzar la consistencia en pixeles, y se introduce una estrategia de entrenamiento sinérgico entre dos *frames* correspondientes para el cálculo de la pérdida de consistencia temporal. Otro trabajo simultáneo que comparte una idea similar con [77], pero con una exploración adicional del problema de inestabilidad de estilo se puede encontrar en [78]. A diferencia de [77, 78], Chen et al. [79] proponen una subred para producir un flujo de características e incorporar información de flujo óptico en el espacio de características. Su algoritmo está construido sobre una red de transferencia de estilo pre-entrenada (un par codificador-decodificador) y deforma los mapas de características del codificador de estilización usando el flujo de características obtenido.

2.2.3.7. *Character Style Transfer*

Dada una imagen de estilo que contiene varios caracteres, el objetivo es aplicar la idea de NST para generar nuevos tipos de fuentes y efectos de texto. En [80], Atarsaikhan et al. aplican directamente el algoritmo en [35] para transferir el estilo de fuente y lograr resultados visualmente plausibles. Mientras que Yang et al. [81] proponen primero caracterizar los elementos de estilo y explotar las características extraídas para guiar la generación de efectos de texto. Un trabajo más reciente [82] diseña un modelo GAN condicional para la predicción de la forma de los caracteres, y también una red adicional para la predicción del color y la textura. Mediante la formación conjunta de estas dos redes, la transferencia de tipo de fuente puede realizarse de forma integral.

2.2.3.8. *Audio Style Transfer (AST)*

Además de transferir estilos de imagen, se extiende el dominio del estilo de imagen al estilo de audio, sintetizando nuevos sonidos transfiriendo el estilo deseado desde otro audio. El estudio de la transferencia de estilo de audio también sigue la ruta de la transferencia de estilo de imagen, es decir, AST basado en optimización de audios (AOB-AST) y luego AST basado en optimización de modelos (MOB-AST). Inspirados en el método IOB-NST, Verma y Smith [83] proponen un algoritmo AOB-AST. Inician de una señal de ruido y la optimizan iterativamente usando la *backpropagation*. Mital [84] mejora la eficiencia al transferir audios por medio de una red previamente entrenada produciendo resultados en tiempo real.

Capítulo 3

Alternativas de Solución

En este capítulo se realiza un análisis a las distintas alternativas de solución para los puntos más relevantes de esta memoria. Se evalúan las opciones cuantitativamente de acuerdo con las ventajas y desventajas presentadas en cada punto. Finalmente, se seleccionan las opciones más apropiadas para el desarrollo de esta memoria.

Debido a que la construcción de la instalación interactiva tiene como tecnología central el proceso de transferencia de estilo y, como características esenciales, la usabilidad, el aspecto visual y la interactividad del usuario con la exposición, el análisis de las alternativas de solución se divide en los siguientes aspectos:

- Métodos de transferencia de estilo neuronal
- Herramientas de trabajo para deep learning
- Herramientas de trabajo para el desarrollo de interfaz de usuario e interactividad

3.1. Métodos de Transferencia de Estilo Neuronal

En el Capítulo 2 se proporcionó una descripción general de los avances actuales en NST. En esta sección, se pretende analizar en detalle los métodos más importantes de NST, identificando claramente sus ventajas y desventajas. Por cada método NST, es decir, IOB-NST, PSPM-MOB-NST, MSPM-MOB-NST y ASPM-MOB-NST, se escoge una arquitectura representativa para su análisis.

3.1.1. NST Basado en Optimización de Imágenes (IOB-NST)

Los algoritmos IOB-NST se caracterizan por realizar un proceso de reconstrucción de imagen iterativo, es decir, los pixeles de una imagen (inicialmente ruido blanco) son ajustados iterativamente hasta lograr una estilización aceptable. La imagen resultante poseerá el estilo de una imagen deseada y el contenido de otra. Para llevar a cabo lo anterior, Gatys et al. [35] proponen el algoritmo que se resume en la Figura 3.1.

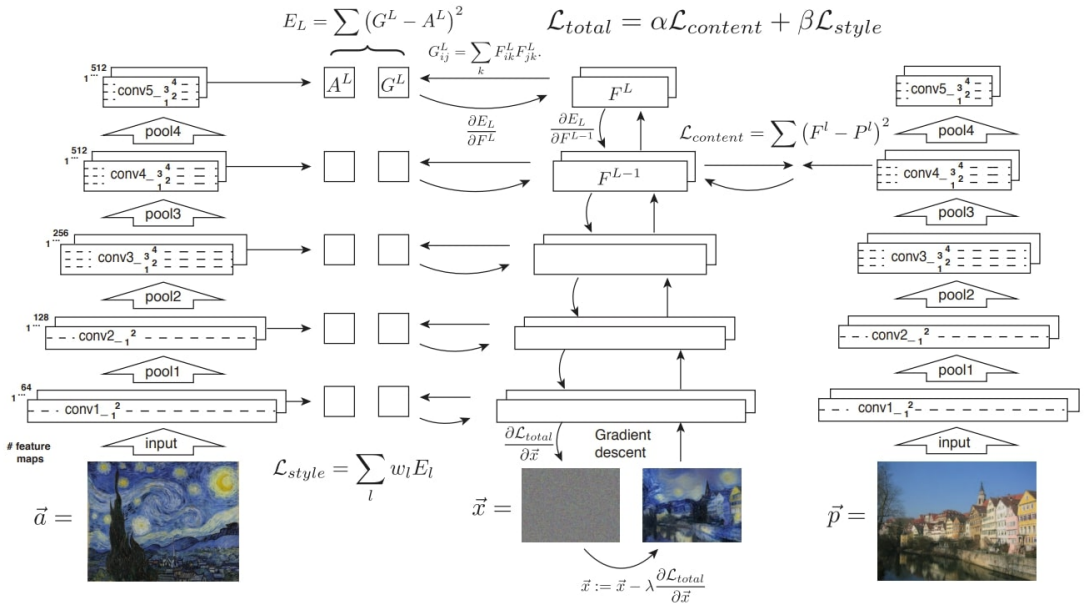


Figura 3.1: Visión general del algoritmo IOB-NST de Gatys et al. [35].

El algoritmo [35], primero extrae y almacena las características de contenido y estilo obtenidas a partir de los mapas de activación de las capas intermedias de la red VGG-19 previamente entrenada. La imagen de estilo $\vec{a}$ es dada como entrada a la red para calcular y almacenar su representación de estilo A^l obtenida de las capas intermedias de la red (Figura 3.1 izquierda). De la misma forma, la imagen de contenido $\vec{p}$ es dada como entrada a la red para almacenar su representación de contenido P^l obtenida a partir de una capa intermedia de la red previamente seleccionada (Figura 3.1 derecha). Luego, una imagen de ruido blanco $\vec{x}$ se introduce como entrada de la red para calcular sus características de estilo G^l y contenido F^l. En cada capa incluida en la representación de estilo, el error cuadrático medio entre G^l y A^l se calcula para obtener la pérdida de estilo $\mathcal{L}_{style}$ (Figura 3.1 izquierda). De la misma forma, el error cuadrático medio entre F^l y P^l se calcula para obtener la pérdida de contenido $\mathcal{L}_{content}$ (Figura

3.1 derecha). Finalmente, la pérdida total $\mathcal{L}_{total}$, es entonces la combinación lineal entre la pérdida de contenido y de estilo. Su derivada con respecto a los valores de pixel se puede calcular utilizando *backpropagation* (Figura 3.1 centro). Este gradiente se utiliza para actualizar iterativamente la imagen $\vec{x}$ hasta que coincida simultáneamente con las características de estilo de $\vec{a}$ y las características de contenido $\vec{p}$ (Figura 3.1 centro, abajo).

El algoritmo IOB-NST de [35] presenta la ventaja de que la imagen estilizada resultante es de mejor calidad que los algoritmos MOB-NST, y es flexible en el sentido de que puede estilizar imágenes con el estilo que se desee. Pero la limitación de este algoritmo es su alto costo computacional, debido al procedimiento iterativo de optimización de imágenes [85].

3.1.2. MOB-NST de Estilo Único (PSPM-MOB-NST)

Los algoritmos PSPM-MOB-NST abordan el problema de la velocidad y el costo computacional de los métodos IOB-NST utilizando una red neuronal adicional para realizar la estilización. Johnson et al. [52] entrenan una red de transformación de imágenes para transferir el estilo artístico a la imagen deseada. Además, emplean una red de pérdida previamente entrenada (VGG-16) a fin de definir funciones de pérdida de percepción que miden las diferencias de percepción en contenido y estilo entre imágenes. Este enfoque de extracción de características de estilo y contenido mediante una red pre-entrenada es el mismo que utiliza Gatys et al. [35] en el método anterior.

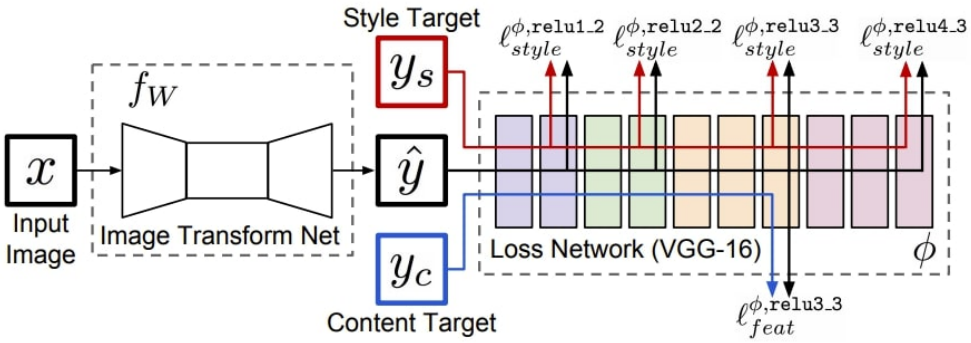


Figura 3.2: Visión general del algoritmo PSPM-MOB-NST de Johnson et al. [52].

Como se muestra en la Figura 3.2, el sistema de consta de dos componentes: una red de transformación de imágenes f_W y una red de pérdida ϕ que se utiliza para definir varias funciones de pérdida $\ell_1, \ldots, \ell_k$. La red de transformación de imágenes es una red

neuronal convolucional residual parametrizada por los pesos W; transforma las imágenes de entrada x en imágenes de salida $\hat{y}$ a través del mapeo $\hat{y} = f_W(x)$. Cada función de pérdida calcula un valor escalar $\ell_i(\hat{y}, y_i)$, que mide la diferencia entre la imagen de salida $\hat{y}$ y una imagen objetivo y_i. La red de transformación de imágenes se entrena usando descenso del gradiente estocástico (SGD) para minimizar la combinación lineal de las funciones de pérdida:

$$W^* = \arg\min_W \mathbf{E}_{x,\{y_i\}} \left[\sum_{i=1} \lambda_i \ell_i \left(f_W(x), y_i \right) \right] \tag{3.1}$$

Se hace uso de una red ϕ, que ha sido pre-entrenada para la clasificación de imágenes, como una red de pérdida fija con el fin de definir las funciones de pérdida. Esta red debe ser pre-entrenada para clasificación porque de esta manera, la red ya ha aprendido a codificar la información perceptiva y semántica que se desea medir en las funciones de pérdida. Entonces, la red convolucional de transformación de imágenes se entrena utilizando funciones de pérdida que también son redes convolucionales profundas.

La red de pérdida ϕ se utiliza para definir una *pérdida de reconstrucción de características* ℓ^{ϕ}_{feat} y una *pérdida de reconstrucción de estilo* ℓ^{ϕ}_{style}, que miden las diferencias de contenido y estilo entre las imágenes. Para cada imagen de entrada x se tiene un contenido objetivo y_c y un estilo objetivo y_s. Para la transferencia de estilo, el contenido objetivo y_c es la imagen de entrada x y la imagen de salida $\hat{y}$ debe combinar el contenido de $x = y_c$ con el estilo de y_s; se entrena una red por cada objetivo de estilo.

El algoritmo PSPM-MOB-NST de [52] presenta buenos resultados de estilización y una gran eficiencia al realizar la transferencia de estilos, al punto de poder realizarla en tiempo real. Pero su gran limitación es que se deben entrenar tantas redes como estilos se deseen.

3.1.3. MOB-NST de Múltiples Estilos (MSPM-MOB-NST)

Aunque los enfoques de PSPM-MOB-NST pueden producir imágenes estilizadas dos órdenes de magnitud más rápido que los métodos IOB-NST anteriores, se deben entrenar redes generativas por cada imagen de estilo en particular, lo que requiere bastante tiempo. Los algoritmos MSPM-MOB-NST mejoran la flexibilidad al incorporar múltiples estilos en un solo modelo alcanzando una eficiencia similar al método PSMP-MOB-NST.

El algoritmo previo de transferencia basado en un solo estilo utiliza una red de

transformación que toma solo la imagen de contenido como entrada y genera como salida la imagen estilizada, es decir, la red de transformación puede expresarse como $G(x_c)$, que aprende implícitamente las estadísticas de características de la imagen de estilo desde la función de pérdida. Zhang y Dana [60] introducen una Red Generativa Multiestilo (*Multi-style Generative Network*) que toma tanto el contenido como el estilo objetivo como entradas, es decir, $G(x_c, x_s)$. La red propuesta coincide explícitamente las características de estilo en tiempo de ejecución. En la Figura 3.3 se presenta una visión general de la arquitectura propuesta por [60].

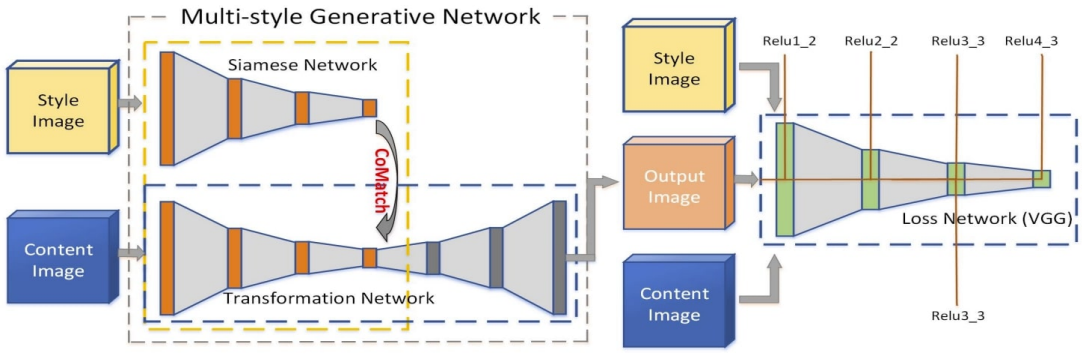


Figura 3.3: Visión general de MSG-NET: Multi-style Generative Network [60].

Como parte de la red generadora, se adopta una red siamesa que comparte pesos con el codificador de la red de transformación, captura las estadísticas de características de la imagen de estilo x_s a diferentes escalas, y produce en la salida las matrices de Gram $\{\mathcal{G}(\mathcal{F}^i(x_s))\}(i = 1, \ldots, K)$, donde K es el número total de escalas. Luego, una red de transformación toma la imagen de contenido x_c y la hace coincidir con las estadísticas de características de la imagen de estilo a múltiples escalas por medio de la capa *CoMatch*. Finalmente, en el tramo de decodificación de la red de transformación se obtiene como output la imagen estilizada. Para el entrenamiento, una red de pérdidas previamente entrenada proporciona el aprendizaje supervisado de MSG-Net minimizando las diferencias de contenido y estilo con los objetivos.

El algoritmo MSPM-MOB-NST de [60], a diferencia del PSPM-MOB-NST, puede generar imágenes en múltiples estilos previamente seleccionados. Además, logra una flexibilidad visual porque permite al usuario modificar el tamaño del trazo del pincel en la imagen resultante al cambiar el tamaño de la imagen de estilo de entrada. Pero el algoritmo MSPM-MOB-NST de [60] aún está limitado en la preservación de la coherencia de las estructuras finas y la información de profundidad.

3.1.4. MOB-NST de Estilos Arbitrarios (ASPM-MOB-NST)

Los algoritmos ASPM-MOB-NST buscan estilizar imágenes con el estilo de imagen que se desee, es decir, un solo modelo para todos los estilos deseados. Huang y Belongie [55] proponen la arquitectura de la Figura 3.4 para conseguir transferencia de estilos arbitrarios. Utilizan las primeras capas de una red VGG-19 fija para codificar el contenido y las imágenes de estilo. Luego, una capa de AdaIN se utiliza para realizar la transferencia de estilo en el espacio de características. Finalmente, se entrena un decodificador para invertir la salida de AdaIN a los espacios de la imagen. Se utiliza el mismo codificador VGG para calcular la pérdida de contenido $\mathcal{L}_c$ y estilo $\mathcal{L}_s$.

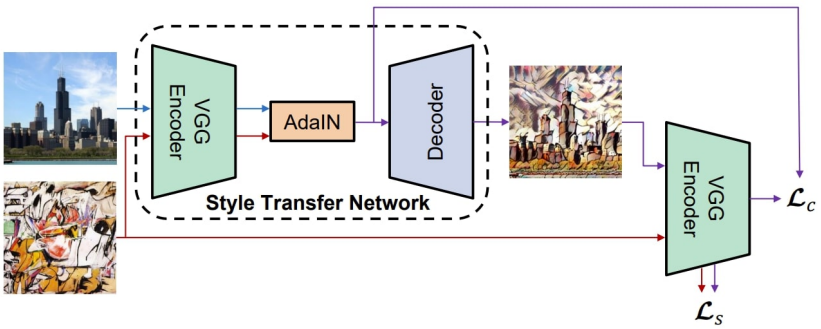


Figura 3.4: Visión general del algoritmo ASPM-MOB-NST de Huang y Belongie [55].

La base para la transferencia de estilo en [55] es la capa AdaIN, una extensión simple de la capa IN. AdaIN recibe una entrada de contenido x y una entrada de estilo y, y simplemente alinea la media y la varianza de x para que coincida con las de y. A diferencia de BN, IN o CIN, AdaIN no tiene parámetros afines entrenables. En su lugar, calcula adaptativamente los parámetros afines a partir de la entrada de estilo:

$$\text{AdaIN}(x, y) = \sigma(y) \left(\frac{x - \mu(x)}{\sigma(x)} \right) + \mu(y) \tag{3.2}$$

En la que simplemente se escala la entrada de contenido normalizada con $\sigma(y)$, y se desplaza con $\mu(y)$. Similar a IN, estas estadísticas se calculan a través de los ejes espaciales.

El algoritmo ASPM-MOB-NST de [55] logra una estilización en tiempo real. Sin embargo, este algoritmo está basado en datos y limitado en la generalización de estilos no vistos. Además, el simple hecho de ajustar la media y la varianza de los mapas de activación aumenta la dificultad de sintetizar patrones de estilo complicados con ricos detalles.

3.2. Herramientas de Trabajo para Deep Learning

En la actualidad, se encuentran disponibles numerosos frameworks de desarrollo en el área de deep learning, que asisten en el proceso de entrenamiento, pruebas y producción de redes neuronales. Para la claridad, en éste ámbito, framework se define como una colección completa de bibliotecas para construir modelos de deep learning. Los frameworks presentan distintas características de modelamiento, interfaces gráficas, soporte de plataformas, interfaces de programación de aplicaciones (APIs), etc. Por lo que es indispensable analizar las diferentes cualidades que proporciona cada uno.

En la Figura 3.5 se presenta un diagrama de alto nivel del flujo de trabajo típico de las herramientas dedeep learning. En el nivel superior se encuentran las APIs de alto nivel, herramientas de gestión de trabajo y visualización, luego están los frameworks y, finalmente, se encuentran las plataformas sobre las que funcionan dichos frameworks.

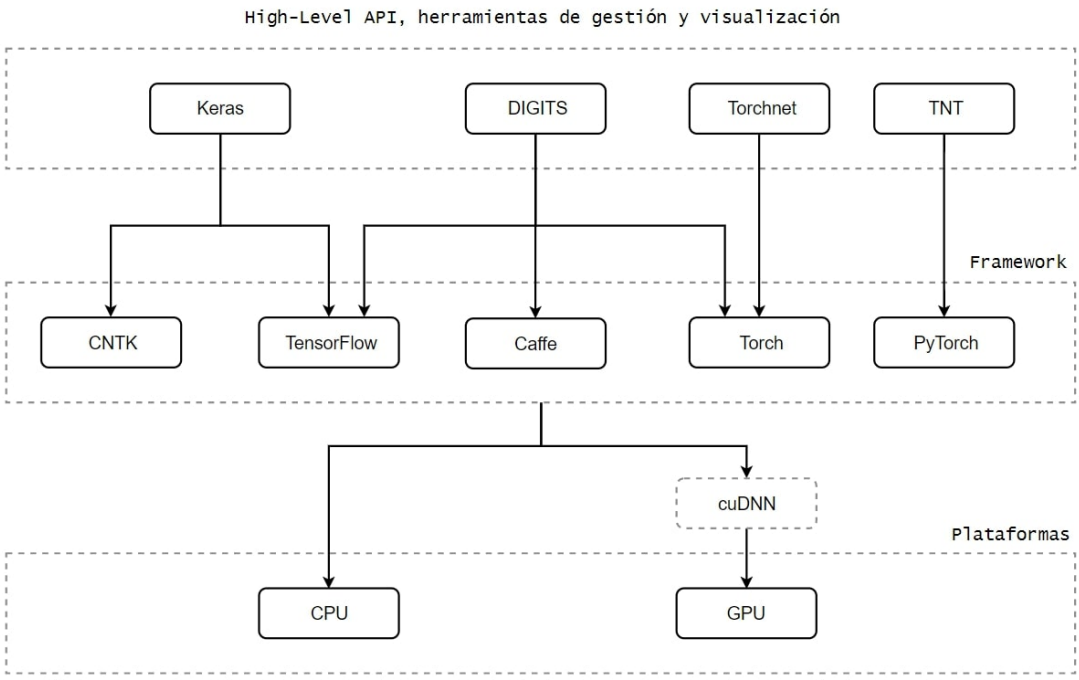


Figura 3.5: Flujo de trabajo de herramientas de deep learning.

3.2.1. High-Level API, Herramientas de Gestión de Trabajo y Visualización

Estas herramientas sirven para facilitar el manejo de las distintas etapas del proceso de las redes neuronales y resultan de gran utilidad en áreas de investigación y desarrollo. No se detallará la evaluación de herramientas de alto-nivel, sin embargo, se deben considerar sus principales características debido a que aportan diferentes cualidades a los frameworks.

3.2.1.1. Keras

Keras [86] es una API de alto nivel para redes neuronales, escrita en Python, y capaz de ejecutarse sobre TensorFlow o CNTK. Fue desarrollado con un enfoque en permitir la experimentación rápida: *"Poder pasar de la idea al resultado con el menor retraso posible es clave para hacer una buena investigación"* [86].

Keras permite una creación de prototipos fácil y rápida (a través de la facilidad de uso, la modularidad y la extensibilidad). La creación de modelos masivos de deep learning en Keras se reduce a funciones de una sola línea, pero esta estrategia hace de Keras un entorno menos configurable que los marcos de bajo nivel. Además, admite redes convolucionales y redes recurrentes, así como combinaciones de las dos, y funciona a la perfección en CPU y GPU.

Keras es la mejor API de deep learning para aquellos que recién están comenzando. Es ideal para aprender y crear prototipos de conceptos simples, para comprender la esencia misma de los diversos modelos y procesos de su aprendizaje.

3.2.1.2. DIGITS

DIGITS, NVIDIA Deep Learning GPU Training System [87], es un wrapper para NVCAffe y TensorFlow con soporte para Torch, que proporciona una interfaz web para estos frameworks en lugar de tratarlos directamente de la linea de comandos.

DIGITS puede ser usado para entrenar rápidamente redes neuronales profundas (DNN) para clasificación de imágenes, segmentación, detección de objetos y más. Simplifica las tareas comunes de deep learning, como la gestión de datos, el diseño y capacitación de redes neuronales en sistemas de multiples GPU, la supervisión de rendimiento en tiempo real con visualizaciones avanzadas, y la selección del modelo con mejor rendimiento para implementación. DIGITS es completamente interactivo para

que los ingenieros puedan enfocarse en diseñar y capacitar redes en lugar de programar y depurar.

3.2.1.3. Torchnet

Torchnet [88] es un framework de código abierto para Torch que proporciona abstracciones y lógica repetitiva para Machine Learning. Fomenta la programación modular y la reutilización de código, lo que reduce la posibilidad de errores, facilita el uso datos y vuelve más eficiente el uso de múltiples GPU. Torchnet está escrito en Lua, lo que facilita su instalación en cualquier arquitectura con una instalación de Torch [89].

En este momento, Torchnet proporciona cuatro conjuntos de clases importantes:

- `Dataset`: Manipulación y preprocesamiento de datos de diversas formas.
- `Engine`: Entrenamiento y prueba de algotimos de Machine Learning.
- `Meter`: Medidor de rendimiento o cualquier otra cantidad.
- `Log`: Manejo de eventos. Proporcionan una forma de monitorear los modelos.

3.2.1.4. TNT

TNT [90] es una biblioteca que proporciona potentes servicios de gestión de datos, registro y visualización para Python. Está estrechamente integrado con PyTorch y está diseñado para permitir una rápida iteración con cualquier modelo o régimen de entrenamiento.

Esta biblioteca proporciona métodos para registrar el rendimiento de los modelos en el módulo *torchnet.meter* y para registrarlos en Visdom (o en el futuro, TensorboardX) con el módulo *torchnet.logger*. Las clases que proporciona TNT son similares a las de Torchnet, estas son:

- `torchnet.dataset`
- `torchnet.engine`
- `torchnet.logger`
- `torchnet.meter`
- `torchnet.utils`

En el futuro, también brindará un sólido soporte para el Aprendizaje Multitarea (MTL, Multitask Learning) y la Transferencia de Aprendizaje (Transfer Learning). Actualmente admite la carga de datos de entrenamiento a través de *torchnet.utils.MultiTaskDataLoader*.

3.2.2. Frameworks

Existen muchos frameworks para deep learning, pero aquí se presentarán como opción solo los más populares.

3.2.2.1. Caffe

Caffe, Convolutional Architecture for Fast Feature Embedding [91], es una herramienta de trabajo para deep learning desarrollado por Berkeley Vision and Learning Center y publicada bajo licencia BSD. Es desarrollado en C++ pero es compatible con interfaces para python y Matlab, y es posible utilizarlo por linea de comandos [92].

Caffe es conocido por su velocidad y aplicabilidad en redes neuronales convolucionles (CNN), la razón de esto, es que posee un gran repositorio de modelos de redes neuronales pre-entrenada llamado “Model Zoo”. Esta es la principal ventaja de este framework, ya que agiliza en gran manera el modelado de las CNNs y proporciona el acceso a la implementación de una gran cantidad de modelos pre-entrenadas para redes populares como VGG, AlexNet, GoogLeNet, entre otras.

Debido a que el enfoque de Caffe es en procesamiento de imágenes, resulta más complicado y menos eficiente el trabajo con redes enfocadas en otras áreas, como por ejemplo, las redes recurrentes.

Otro problema de Caffe, es que la interfaz de Python no se encuentra muy documentada y, si bien es un framework bastante popular, el desarrollo está siendo cada vez menor. La última versión de Caffe fue lanzada en abril del año 2017.

3.2.2.2. TensorFlow

TensorFlow [93] es un framework de código abierto desarrollado por *Google Brain Team* [94], que utiliza un grafo de flujo de datos para realizar los cálculos numéricos, donde cada nodo representa una operación matemática y, las aristas, las estructuras de datos (o tensores, arreglos n-dimensionales) [95]. Está escrito con una API de Python sobre un motor de C/C++ lo que produce que se ejecute más rápido.

Actualmente es el framework más popular para deep learning, es flexible y escalable, posee una comunidad muy activa, y documentación extensa y útil [96]. Además, posee soporte de muchas aplicaciones de alto nivel como Keras, herramientas para facilitar la depuración como Eager Execution, y herramientas para el uso de datos. Estas interfaces simplifican mucho el uso de TensorFlow.

TensorFlow posee dos herramientas que son muy utilizadas. La primera es *Tensorboard*, una interfaz web para la visualización de grafos de flujo de datos, que provee un mejor entendimiento del modelo y brinda mayor facilidad para corregir errores y buscar puntos de optimización. También permite graficar métricas cuantitativas de la ejecución del programa. La segunda herramienta es *Tensorflow Serving*, un sistema de servicio flexible y de alto rendimiento para modelos de machine learning, diseñado para entornos de producción. Una vez que el modelo está entrenado y listo para ser utilizado para la predicción, esta herramienta ayudará a comprobar el buen funcionamiento del modelo, el aislamiento y su seguridad, facilitando el desafío de pasar de la experimentación a la producción.

Un punto en contra de TensorFlow es que debido a su estructura e implementación, es comparativamente más lento que otros frameworks como CNTK [97]. Además, no es muy fácil de usar sin la ayuda de las herramientas adicionales antes mencionadas, y requiere de amplios conocimientos de machine learning para utilizar todo su potencial. Por último, las únicas GPUs compatibles con este framework son las de Nvidia.

3.2.2.3. Torch

Torch [98] es un framework con una API escrita en Lua, con amplio soporte para algoritmos de machine learning que prioriza la implementación en GPUs. Proporciona un rendimiento más rápido en comparación con otros frameworks debido al uso del lenguaje LuaJIT y su implementación en C/CUDA subyacente.

Torch puede usarse junto con *Torchnet* para realizar prototipos de modelos de deep learning de manera más rápida, por lo que es ideal para la experimentación [89]. La última versión de Torch permite el uso con *LoadCaffe* [99], lo que posibilita cargar todos los modelos pre-entrenados disponibles de Caffe.

Dado que Lua no es muy conocido por los ingenieros de software de la comunidad empresarial, ocasiona que Torch no sea popular, a pesar de ser un framework potente, y que posea una comunidad pequeña [100]. Además, contiene poca documentación y está enfocado para desarrollo de algoritmos para áreas de investigación más que para

el uso en producción. La última versión de Torch fue lanzada en febrero del año 2017. A la fecha, el framework ya no está en desarrollo activo y su comunidad está cerca de no existir [101].

3.2.2.4. CNTK

CNTK, Microsoft Cognitive Toolkit [102], es un kit de herramientas de deep learning unificado que describe las redes neuronales como una serie de pasos computacionales a través de un grafo dirigido. De manera similar a TensorFlow, en estos grafos dirigidos los nodos hoja representan valores de entrada o parámetros de red, mientras que los otros nodos representan operaciones matriciales [103].

CNTK también tiene un alto grado de flexibilidad. Utiliza un archivo de configuración para definir la estructura de la red (igual que Caffe) y luego la línea de comandos para realizar el entrenamiento. Otra característica importante es la escalabilidad, CNTK no solo tiene muchos nodos de cómputo (para los grafos), sino que también permite a los usuarios definir sus propios nodos, lo que permite un alto grado de personalización [100]. CNTK agregó soporte para el aprendizaje por refuerzo en septiembre de 2016.

Una de las grandes ventajas de este framework es que posee mejor rendimiento que los demás cuando se utiliza en múltiples GPU. Su rendimiento en CPU también es bueno, tiene una gran ventaja cuando se trata de computación paralela (en CPU o GPU), pero en otros casos, como por ejemplo, cuando se trabaja una sola GPU, tiene peor rendimiento que sus competidores.

CNTK es fácil de usar en comparación con otros frameworks porque no requiere de mucha programación, pero un gran punto en contra de este framework es el débil apoyo de la comunidad, tiene poca documentación y escasas implementaciones. En segundo lugar, CNTK no es compatible con procesadores integrados y también carece de soporte para otras GPU que no sean Nvidia.

3.2.2.5. PyTorch

PyTorch [104] es una biblioteca diseñada para permitir una investigación rápida sobre modelos de deep learning que se construyó en base a algunos proyectos, especialmente Lua Torch, Chainer y HIPS Autograd [105].

PyTorch se ejecuta en python y proporciona principalmente dos características de alto nivel:

- Computación de tensores (como NumPy) con una gran optimización en GPU.
- Redes neuronales profundas construidas en *autograd* (biblioteca para cálculo eficiente de gradientes [106])

A diferencia de TensorFlow, PyTorch ofrece grafos de computación dinámica que le permiten procesar entradas y salidas de longitud variable, y es posible su modificación en tiempo de ejecución, lo cual es bastante valioso para situaciones en las que no se sabe cuánta memoria se necesitará para crear una red neuronal. Éste tipo de arquitectura es bastante útil para trabar, por ejemplo, con RNN. Dada la arquitectura de PyTorch, todo el proceso de modelado es mucho más simple y transparente en comparación con Torch.

Dado que PyTorch está altamente integrado con Python, es posible utilizar naturalmente bibliotecas propias de este lenguaje como NumPy, scikit-learn, SciPy, etc., y sus herramientas de depuración. La popularidad de este framework es cada vez mayor y se considera el gran rival de TensorFlow.

3.2.3. Plataformas

Los frameworks funcionan tanto para CPU como GPU. Las CPUs son una buena alternativa para las aplicaciones con tareas complejas pero secuanciales y bajo paralelismo de datos. Sin embargo, es más adecuado utilizar GPUs para modelos de deep learning, debido a que sus procesos involucran una gran cantidad de operaciones como convoluciones, multiplicación de matrices, funciones de activación, que son fácilmente paralelizables en GPU. La paralelización de operaciones permite aumentar la velocidad de los procesos, lo cual resulta particularmente útil en la etapa de entrenamiento.

Trabajar con GPUs de Nvidia ofrece grandes ventajas debido a que posee una API para programación paralela conocida como CUDA (Compute Unified Device Architecture), que permite desarrollar programas para GPU. CUDA provee de un conjunto de herramientas como depuradores, aceleradores, profilers, etc. A pesar de ser una API muy utilizada en redes neuronales, está bastante desarrollada (con más de diez años de existencia). Posibilita usar la biblioteca acelerada de GPU de Nvidia llamada cuDNN que proporciona implementaciones altamente ajustadas para rutinas de propagación, pooling, normalización, entre otras [107].

3.3. Herramientas de Trabajo para el Desarrollo de Interfaz de Usuario e Interactividad

3.3.1. Herramientas para el Desarrollo de Interfaz de Usuario

Cuando se desarrolla un producto digital e interactivo, es necesario tener una capa de software que permita al usuario interactuar con la aplicación. Ya sea una aplicación móvil, web o de escritorio, una interfaz de usuario (UI) será un componente obligatorio en el desarrollo. De la misma forma, la instalación interactiva de arte digital que se desarrolla en este proyecto, debe poseer una UI que permita al usuario interactuar con la exposición. Esta UI puede estar presente en la misma unidad de procesamiento o puede ser algún componente externo que este comunicado con la unidad de procesamiento, por ejemplo, una Tablet.

Electroveja Labs, empresa con la cual se desarrolla esta instalación interactiva, posee capacidades para el desarrollo de UI. El software que utilizan es Unity, un motor de videojuegos disponible como plataforma de desarrollo para Microsoft Windows, Mac OS y Linux. En la actualidad, Unity es utilizado por un amplio número de desarrolladores de videojuegos para realizar una variedad de simulaciones interactivas que van desde pequeños juegos para navegador y dispositivos móviles, a juegos de consola de alto presupuesto y experiencias en Realidad Aumentada y Realidad Virtual.

Históricamente, Unity ha tenido un enfoque en hacer de este motor de desarrollo una herramienta de propósito general, que pueda ser utilizado por usuarios con diferentes niveles de experiencia, y que sea versátil para desarrollar distintos tipos de videojuegos. La flexibilidad de Unity, lo convierten en la plataforma ideal para la creación de distintos tipos de experiencias, no solo videojuegos. Es por esto, que su utilización en el desarrollo de este proyecto otorgará flexibilidad en el diseño de la instalación, permitiendo crear módulos de visualización para los algoritmos de IA, e interfaces para la interacción con el usuario, mejorando en gran medida la experiencia vivida en la instalación interactiva.

3.3.2. Herramientas para la Interactividad

El objetivo general del proyecto es desarrollar una instalación de arte digital que sea interactiva. En este sentido, dado que se utiliza un algoritmo de NST como base

tecnológica de la instalación, es necesaria la presencia de una cámara para poder capturar la escena del usuario, y un componente de visualización, como un proyector o una pantalla LED, para exhibir el resultado de la transferencia de estilo. Mientras mayor sea la calidad de estos componentes, mejor será la experiencia del usuario en la instalación. Una pantalla de gran tamaño o un proyector de alta potencia permitirán una mayor inmersión para el espectador. Por otro lado, el usuario se volverá protagonista de la exposición por medio de la interacción con la instalación, por ello, una cámara que permita capturar la escena en alta resolución y con una gran cantidad de FPS, también mejorará en gran medida la experiencia del usuario. La elección del hardware para este proyecto dependerá completamente de la empresa Electroveja Labs, por lo que su selección no ser analizada en este capítulo.

Además, como se debe utilizar una cámara, es obligatorio utilizar una biblioteca de software que permita comunicar la cámara con la unidad de procesamiento y también permita realizar operaciones de procesamiento de imágenes. En visión por computador, OpenCV es la biblioteca más popular y versátil de todas. Es fácil de usar y cubre todas las técnicas y algoritmos necesarios para realizar varias tareas de procesamiento de imágenes. Además, es multiplataforma siendo compatible con Windows, Linux, Mac OS, iOS y Android, y tiene interfaces para C, C++, Python y Java.

OpenCV tiene una estructura modular, lo que significa que el paquete incluye varias bibliotecas compartidas dentro de ella. Es posible encontrar bibliotecas para trabajar con arreglos multidimensionales, procesamiento de imágenes y video, visión en 3D, detección de objetos, etc. Dada la gran comunidad y su activo desarrollo, las implementaciones de los algoritmos en las bibliotecas de OpenCV se encuentran muy optimizados tanto para CPU como para GPU. Además, OpenCV es compatible con framworks para deep learning como TensorFlow, PyTorch y Caffe.

3.4. Alternativa Seleccionada

En esta sección se evalúan cuantitativamente los 3 aspectos analizados en las secciones anteriores. Se determinan los criterios más relevantes para cada uno asignando una ponderación de acuerdo con su grado de importancia. Luego, estos criterios se evalúan con una calificación de 0 a 100 para su posterior elección.

3.4.1. Selección del Método de Transferencia de Estilo Neuronal

Para las alternativas de métodos NST se consideran los siguientes criterios de evaluación (el detalle de la calificación se encuentra en la sección A.1 del apéndice):

- **Eficiencia:** Dado que el objetivo es desarrollar una intervención para el museo Artequin, evaluar este punto es de vital importancia porque no será del agrado de los visitantes esperar mucho tiempo para ver el resultado de la transferencia de estilos. Se investigan optimizaciones existentes para los modelos y se evalúa la posibilidad de ejecutar en tiempo real. Aquellos modelos que se prevea no poder optimizar deberán ser descartados como posible opción de solución y se calificaran en este ítem con puntuación 0.
- **Calidad Visual:** Se evalúa cualitativamente el resultado del método NST. Dado que el objetivo general del proyecto es introducir la inteligencia artificial como fenómeno artístico, una mayor calidad visual permitirá que la red se perciba más artística, atrayente e interesante. El puntaje está dado a criterio del autor al estudiar los resultados presentados en los artículos científicos respectivos.
- **Flexibilidad:** Se evalúa la capacidad del método NST de poder controlar la calidad visual del resultado estilizado considerando varios estilos distintos. Además, se evalúa qué tan flexible es su arquitectura para recibir modificaciones que mejoren el resultado final. En este aspecto se evalúa la posibilidad de controlar el tamaño del trazo del pincel, controlar espacialmente la transferencia de estilo, controlar el color, considerar la profundidad, etc.

Criterio	**Ponderación**	IOB-NST	PSPM-MOB-NST	MSPM-MOB-NST	ASPM-MOB-NST
Velocida	40 %	0	100	90	80
Calidad Visual	40 %	100	95	90	80
Flexibilidad	20 %	100	100	90	90
Total	**100 %**	**60**	**98**	**90**	**82**

Tabla 3.1: Calificaciones de los métodos de transferencia de estilo neuronal.

En la Tabla 3.1 se muestran las calificaciones de los métodos NST. Se aprecia que el método más conveniente para este proyecto es el PSPM-MOB-NST. El mayor beneficio de esta alternativa es que se puede lograr un buen resultado artístico en base una arquitectura simple. Luego, es posible mejorar iterativamente la red agregando distintas extensiones para mejorar la estilización, lo que permite diseñar y desarrollar la

solución en torno a una metodología iterativa que se ajuste a los tiempos e imprevistos en el desarrollo del PMV de la instalación interactiva.

3.4.2. Selección de Framework para Deep Learning

Para las alternativas de framework se consideran los siguientes criterios de evaluación (el detalle de la calificaciones se encuentra en la sección A.2 del apéndice):

- **Bibliotecas de modelos pre-entrenados:** Se examina la extensión de las bibliotecas de modelos pre-entrenados en los sitios web de cada framework. Los modelos pre-entrenados son relevantes porque sirven para acelerar el proceso de entrenamiento, y reducen la cantidad necesaria para el *dataset* de entrenamiento. En vez de entrenar una red con parámetros inicializados aleatoriamente, se pueden utilizar los parámetros de una red pre-entrenada (de propósito similar) como punto de partida.

  Es de suma importancia que el framework proporcione bibliotecas con modelos pre-entrenados ya que el modelo de red escogido para dar solución al problema propuesto esta basado en un tipo de CNN muy conocida llamada VGG-Net. En caso de no poseer este modelo pre-entrenado, se califica con puntuación cero y se descarta su posible utilización.

- **Popularidad:** Se evalúa si el framework es conocido y utilizado por la comunidad de desarrolladores e investigadores, debido a que esta característica generalmente implica una comunidad más activa con mayor cantidad de información. También, el nivel de uso y la popularidad pueden llegar a determinar el crecimiento y continuidad del framework.

  La métrica utilizada en este punto es el número de *stars* (cantidad de personas que siguen el repositorio del proyecto) y la cantidad de contribuyentes en Github.

- **Actividad en Stack Overflow:** Stack Overflow es una comunidad abierta de preguntas y respuestas para programadores y profesionales de la informática [108]. Desde hace tiempo que se ha convertido en la comunidad en línea más grande y de mayor confianza para programadores. Se ha vuelto de suma importancia para los programadores porque en ella se pueden encontrar respuestas a muchas problemáticas de programación, en las cuales la documentación de los lenguajes de programación no da abasto. Por esta razón se agrega como criterio de selección la

actividad que posea el framework en dicha comunidad, pues el hecho de tener una gran actividad en ella puede agilizar el desarrollo del proyecto e incluso suplir el hecho de que exista poca documentación sobre el framework. La métrica usada es la cantidad de preguntas en Stack Overflow con la etiqueta del framework.

- **Documentación:** Una buena documentación, clara y sencilla, es una cualidad esencial para cualquier tipo de framework. Dado el rápido avance del deep learning en este último tiempo es recurrente ver que las documentaciones de este tipo de framework no estén completas o sean bastante desordenadas, teniendo la mayor parte de su contenido en GitHub y no en una página web. En este ítem se evalúa la documentación del framework en cuanto a su orden, completitud, claridad y detalle, además de todo la información adicional que contenga, como guías para empezar, tutoriales, ejemplos de implementación, buenas prácticas, etc.

- **Soporte de técnicas de paralelismo:** Se revisa el soporte para CUDA, OpenMP y OpenCL, lo cual posibilita la portabilidad a otra plataformas. Si posee la calificación máxima, significa que tiene soporte para las tres APIs.

- **Herramientas de visualización:** Se valoran las herramientas de visualización que pueden ser utilizadas junto a los distintos frameworks. La visualización facilita la corrección de errores de los programas y sirve de apoyo para encontrar posibles puntos de optimización.

- **Rendimiento:** Se consideran *benchmarks* realizados para los distintos frameworks de deep learning en pruebas de redes neuronales convolucionales en una sola GPU [97, 109]. Si bien es influyente, se presenta que la diferencia de tiempo no es tan significativa entre frameworks, sino que varía de mayor manera según la configuración de la biblioteca acelerada de GPU cuDNN [110], por lo que la ponderación de este ítem no es tan significativa.

En la Tabla 3.2 se muestran las calificaciones de los frameworks. Se aprecia que el framework más conveniente para este proyecto es Tensorflow, que supera con bastante diferencia a sus competidores. El punto más fuerte de Tensorflow es que actualmente existe mucho desarrollo sobre él y, por lo tanto, posee mayor cantidad documentación, implementaciones, proyectos en marcha, herramientas, etc. Por otro lado, a pesar de ser un framework complicado de trabajar, se puede utilizar con Keras, una API de alto nivel altamente ligada a TensorFlow que agiliza la programación pero le quita versatilidad.

Generalmente se recomienda iniciar con Keras y luego de adquirir más conocimiento, sumergirse en TensorFlow.

Criterio	**Ponderación**	CNTK	TensorFlow	Caffe	Torch	PyTorch
Pre-entrenados	30 %	90	100	100	100	100
Popularidad	20 %	6	58	10	4	22
Stack Overflow	15 %	1	85	5	2	7
Documentación	10 %	100	100	50	50	100
Soporte Paralelismo	10 %	66	66	100	100	66
Visualización	10 %	100	100	80	0	80
Rendimiento	5 %	100	85	100	95	95
Total	**100 %**	**59.95**	**85.2**	**60.75**	**50.85**	**64.8**

Tabla 3.2: Calificaciones de los frameworks para deep learning.

3.4.3. Selección de Herramientas de Trabajo para el Desarrollo de Interfaz de Usuario e Interactividad

Como se mencionó en el análisis de estas herramientas, se utiliza Unity para el desarrollo de la UI y la interfaz de visualización. Por otro lado, las cámaras para propósitos específicos contienen un SDK desarrollado por el proveedor para la comunicación del hardware con el software, por lo tanto, se utiliza dicha herramienta para la conexión de la cámara y se deja OpenCV como biblioteca de software para la lectura y procesamiento de datos de video. Se debe aclarar que estas selecciones son a modo general, posteriormente, en las etapa de diseño y desarrollo de la instalación interactiva, se definen de forma detallada cada elemento de la instalación con sus componentes de hardware y software y, en consecuencia, del propósito específico para cada una de las herramientas.

Capítulo 4

Diseño de la Instalación Interactiva

El espacio de instalación dispuesto por el museo Artequin de Viña del Mar se encuentra actualmente contiguo a una representación real de la pintura "Dormitorio en Arlés" de Vincent van Gogh (Figura 4.1). Debido a esto, se decide que la instalación interactiva esté basada en una inteligencia artificial que extraiga los estilos artísticos de las pinturas de Vincent van Gogh para modificar videos capturados por la propia instalación y permita a los visitantes mezclarse en las pinturas de este pintor. Este proyecto es nombrado **Van Gogh te pinta**, una instalación interactiva de arte digital con inteligencia artificial que pinta al estilo de Vincent van Gogh.

En este capítulo se presenta el diseño completo y detallado de la solución. Se expone la solución desde lo más general hasta llegar al detalle de cada módulo en particular. Se definen los requisitos del sistema y se detalla la arquitectura completa del sistema, relacionando cada requerimiento funcional con los módulos de la instalación instalación que permiten realizar dicha función. Finalmente, se detalla respecto a la interacción entre módulos y el flujo completo del sistema.

4.1. Solución Propuesta

Van Gogh te pinta es una instalación interactiva de arte digital que permite a los visitantes de un museo mezclarse en la escena de distintas pinturas de Vincent van Gogh por medio de videos breves en bucle de no más de 8 segundos; como los *boomerangs* de Instagram [112]. En este caso en particular, el desarrollo está enfocado en el museo Artequin, pero está diseñada para dar valor a todas las instituciones culturales que quieran innovar en la experiencia ofrecida a sus visitantes.



(a) Pintura Original



(b) Representación Real

Figura 4.1: Dormitorio en Arlés de Vincent van Gogh. (a) es la pintura original que se encuentra en "Van Gogh Museum", Ámsterdam [111]. (b) es la representación real de la pintura en el Museo Artequin de Viña del Mar.

El usuario interactúa con la instalación mediante una tablet, en la cual selecciona una pintura de Vincent van Gogh e inicia la grabación de video. En paralelo, la instalación proyecta en la pared del espacio una interfaz de visualización, que guía al usuario en cada etapa del proceso. Mientras se graba, el usuario puede realizar movimientos, gestos, o lo que desee hacer frente a la cámara. Posteriormente, cuando finaliza la grabación, el video es procesado mediante NST para aplicar el estilo artístico de la pintura de van Gogh seleccionada. Mientras el video es procesado, en la interfaz de visualización se despliegan datos interesantes de la vida del pintor para estimular la curiosidad de los visitantes y despertar el interés por la persona que hay detrás de la pintura seleccionada. Al finalizar el procesamiento, se obtiene un cuadro de Vincent van Gogh en movimiento que es proyectado en la interfaz de visualización que queda en exhibición hasta el próximo uso de la instalación.

4.2. Alcance de la Propuesta de Solución

El nivel de desarrollo tecnológico que se busca alcanzar en el desarrollo de esta memoria es una solución al nivel de un PMV, no de un producto final. El PMV o producto mínimo viable es la versión de un nuevo producto que permite a un equipo recolectar la máxima cantidad de aprendizaje validado sobre clientes al menor coste

[113]. La idea detrás de un PMV es desarrollar una versión de prueba del producto, con una inversión financiera y de tiempo mínimas y que, al mismo tiempo, sea capaz de ofrecer los mismos valores que el producto ya terminado. A través del PMV se pude validar el producto en su primera aproximación al mercado, identificando claramente las fortalezas del producto así como sus flaquezas, estando a tiempo de poder mejorarlo.

En definitiva, el diseño de la instalación interactiva que se plantea en este capítulo tiene el alcance de un producto mínimo viable, es decir, se diseña una primera versión de la instalación interactiva Van Gogh te pinta con la capacidad de ofrecer el mismo valor que el producto ya terminado, esto con la intención de conocer qué es lo que quieren los clientes y aplicarlo para optimizar el ciclo de desarrollo del producto.

En conclusión, en esta memoria solo se pretende construir la primera visión del PMV de Van Gogh te pinta. Entonces, es necesario entender que el diseño y solución planteados aquí sufrirán modificaciones en un futuro por parte de la startup Electroveja Labs. Debido a lo anterior, es importante tener claro que este desarrollo es un paso importante en el alcance del producto final, y que el aprendizaje que se obtiene en la puesta en marcha en el Museo Artequin es información necesaria para definir los pasos a seguir en el alcance del producto ideal que se ofrecerá a las organizaciones culturales del país.

4.3. Requisitos del Sistema

Teniendo en consideración la solución propuesta y el alcance del desarrollo, en esta sección se da a conocer una completa descripción de los requerimientos de la instalación interactiva Van Gogh te pinta. Estas funcionalidades fueron recogidas a partir de las reuniones de trabajo con el equipo gestor y el cliente, y complementadas con una investigación de las necesidades del sector cultural del país.

4.3.1. Requisitos Funcionales

En la Tabla 4.1 se detallan los requisitos funcionales (RF) de la instalación interactiva Van Gogh te pinta.

Requisito Funcional	**Descripción**
RF1	El sistema debe permitir a los usuarios escoger entre 3 pinturas de Vincent van Gogh: La Noche Estrellada, Autorretrato y Dormitorio en Arlés.
RF2	El sistema debe permitir a los usuarios iniciar la grabación de video desde la UI.
RF3	Antes de iniciar la grabación, el sistema debe mostrar una cuenta regresiva de 3 segundos.
RF4	Al iniciar la grabación, el sistema debe indicar al usuario, por medio de la UI y la interfaz de visualización, que se está grabando el video.
RF5	El sistema debe permitir a los usuarios grabarse durante 8 segundos.
RF6	Mientras se graba el video, el sistema debe mostrar al usuario la vista desde la cámara en la interfaz de visualización.
RF7	Mientras se está procesando el video grabado, el sistema debe mostrar en la interfaz de visualización datos curiosos de la vida de Vincent van Gogh.
RF8	Al finalizar el procesamiento del video, el sistema debe mostrar en la interfaz de visualización el video estilizado en bucle.
RF9	El sistema debe permitir estilizar entre 3 pinturas de Vincent van Gogh: La Noche Estrellada, Autorretrato y Dormitorio en Arlés.
RF10	El sistema debe mostrar a los visitantes el último video estilizado si es que no está siendo utilizado.
RF11	El sistema debe guardar el video grabado en formato MP4.
RF12	El sistema debe guardar el video estilizado en formato MP4.
RF13	El sistema debe reproducir una música de fondo mientras esté en funcionamiento.
RF14	Todas las comunicaciones entre aplicaciones cliente/servidor del sistema deben realizarse vía protocolo TCP/IP en una red local.

Tabla 4.1: Requisitos funcionales.

4.3.2. Requisitos No Funcionales

En la Tabla 4.2 se detallan los requisitos no funcionales (RNF) de la instalación interactiva Van Gogh te pinta.

Requisito No Funcional	**Descripción**
RNF1	El sistema debe operar continuamente durante el horario de visita del museo.
RNF2	El sistema debe prenderse en el horario de apertura del museo y apagarse en el horario de cierre del museo automáticamente.
RNF3	La estructura de protección del sistema debe ser lo suficientemente firme y segura para evitar robos y daños del sistema electrónico.
RNF4	La estructura de instalación y la disposición espacial de los elementos del sistema debe estar adecuada para ser utilizada por niños y niñas.
RNF5	La UI y la interfaz de visualización deben ser intuitivas para niños y niñas.
RNF6	La UI debe funcionar en el sistema operativo Android.
RNF7	La interfaz de visualización debe funcionar en el sistema operativo Windows 10.

Tabla 4.2: Requisitos no funcionales.

4.4. Diagrama de Contexto

Considerando la recopilación de los requerimientos funcionales y no funcionales del sistema, en la Figura 4.2 se presenta el diagrama de contexto de la instalación interactiva Van Gogh te pinta. Este diagrama define los límites entre el sistema y su ambiente, mostrando las entidades que interactúan con él en una vista de alto nivel.

Los usuarios de Van Gogh te pinta son los visitantes del museo, quienes pueden interactuar con la instalación seleccionando una pintura en la que deseen mezclarse, luego iniciar la grabación y divertirse frente a la cámara. Finalmente, la instalación interactiva proyecta el resultado estilizado mostrando un cuadro animado en que el

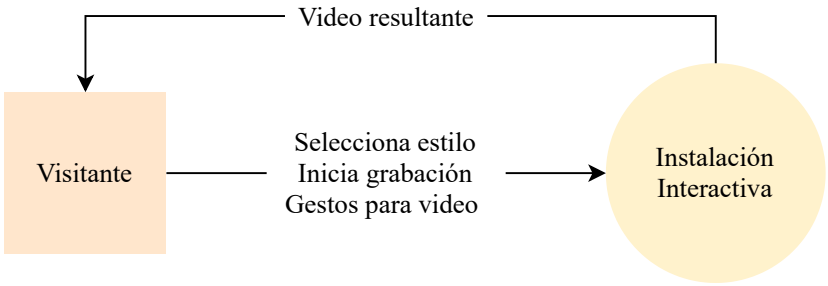


Figura 4.2: Diagrama de contexto.

visitante forma parte de la pintura de Vincent van Gogh. En el caso particular del museo Artequin, los visitantes son principalmente niños y niñas, por lo tanto, Van Gogh te pinta debe ser una instalación interactiva enfocada en este tipo de usuarios.

La instalación no presenta interacción con administradores o personal del museo porque el sistema se diseña para que sea autónomo. Se diseña para no necesitar mantenimiento o atención de parte del museo para su correcto funcionamiento, y de esta manera facilitar la adopción de la tecnología.

4.5. Arquitectura de Componentes

La instalación interactiva esta conformada por componentes de hardware y módulos de software, en esta sección se presenta la conexión de los componentes físicos y luego, en la siguiente sección, se profundiza en la interconexión de los módulos de software. A continuación, se muestra la arquitectura de componentes de la instalación interactiva.

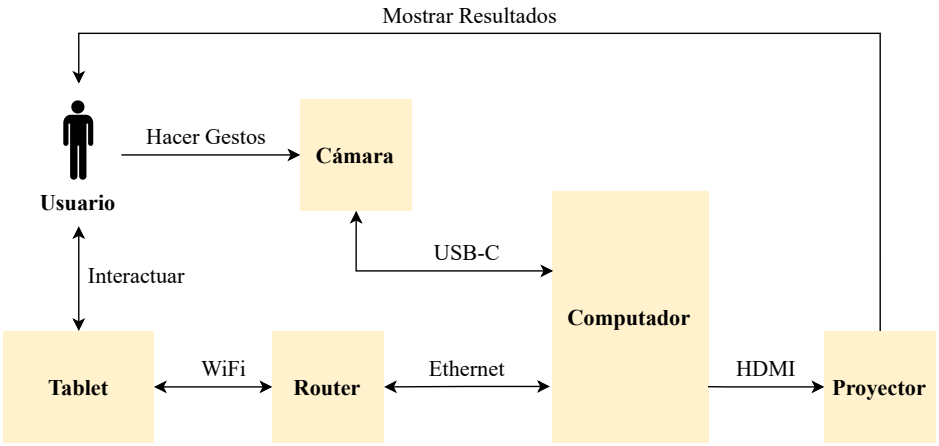


Figura 4.3: Diagrama de arquitectura de componentes.

Uno de los objetivos específicos del proyecto es la implementación de un algoritmo de transferencia de estilo, lo que restringe de forma natural la estructura de componentes de la solución. El diagrama de arquitectura de componentes de la Figura 4.3 muestra la conexión de los elementos físicos del sistema y presenta con mayor detalle la interacción del usuario con el sistema. A continuación, se realiza una descripción general de los componentes del sistema:

- **Tablet:** El visitante selecciona la pintura e inicia la grabación por medio de la tablet. Es el principal medio de interacción para el usuario junto con la cámara.
- **Cámara:** Se utiliza una cámara para grabar videos de los usuarios que luego son estilizados con el algoritmo NST utilizando las pinturas Vincent van Gogh.
- **Proyector:** Se utiliza un proyector para mostrar la interfaz de visualización y el resultado estilizado.
- **Computador:** Naturalmente, todos los componentes deben estar conectados a una unidad de procesamiento, en este caso, un computador, que ejecutará la inteligencia artificial y dará control a todo el sistema.
- **Router:** Se utiliza para comunicar la tablet con el computador. Se configura una red local conectando la tablet vía WiFi y el computador mediante un cable Ethernet.

En el Capítulo 3 se mencionó que la elección un componente del sistema u otro está determinado por la capacidad de la startup Electroveja Labs de disponer de dichos elementos. Lo anterior, explica la razón de la elección de un proyector para mostrar los resultados de la estilización. Además, el proyector, frente a la opción de utilizar una pantalla para mostrar los resultados, tiene el beneficio de ofrecer un mayor tamaño de exposición y mayor seguridad, con un bajo costo de inversión.

Note que la cámara es conectada al computador mediante un USB-C debido a que la cámara dispuesta por la empresa Electroveja Labs para el desarrollo de este proyecto utiliza ese conector[1]. De la misma forma, el proyecto utiliza conector HDMI para conectarse al computador.

[1]El detalle de los componentes se abordará en el siguiente capítulo.

4.6. Arquitectura de Componentes y Software

En la Figura 4.4 se presenta el diagrama de arquitectura de componentes y software que permite visualizar la estructura de alto nivel del sistema respecto a software y hardware. Este diagrama detalla la interconexión de cada módulo de software describiendo como interactúan con los agentes externos y los componentes del sistema.

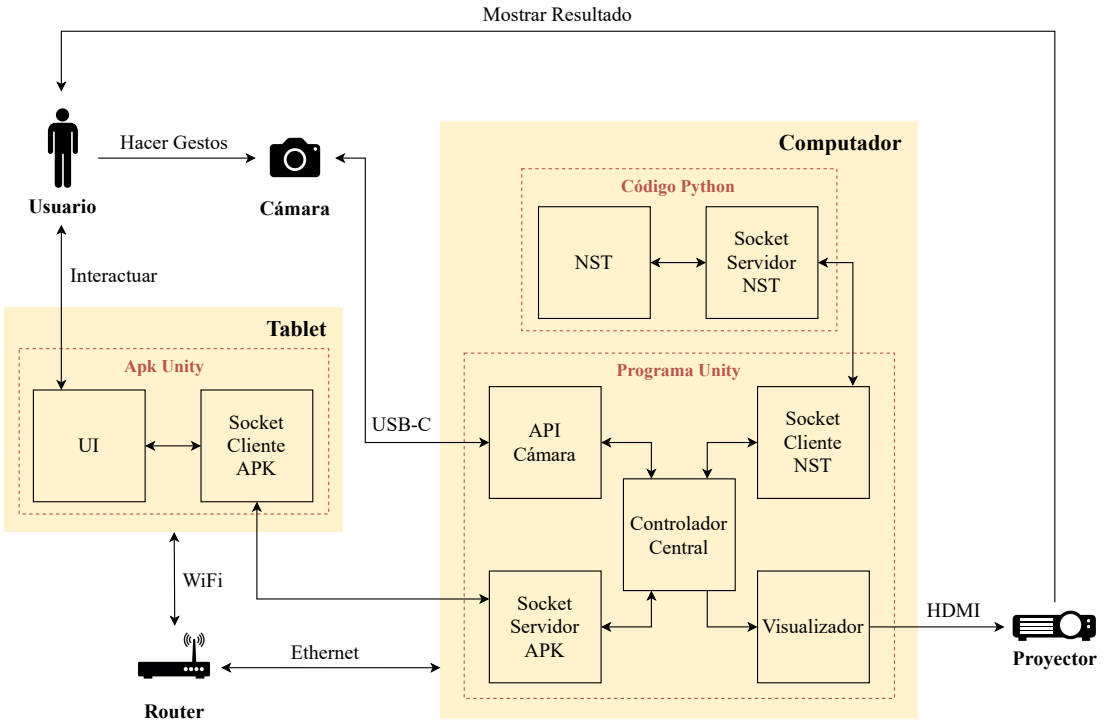


Figura 4.4: Diagrama de arquitectura de componentes y software.

Considerando el diagrama de arquitectura de la Figura 4.4, en la Tabla 4.3 se resumen los módulos de alto nivel de la arquitectura de software de la instalación interactiva. Por cada módulo se entrega un breve párrafo descriptivo de su propósito y de la sección en donde se especifica el módulo en detalle.

Módulo	**Propósito**	**Sección**
UI	Permitir la interacción del usuario con el sistema.	4.7.1
Socket Cliente APK	Comunicar la UI con el Controlador Central. Parte del cliente.	4.7.2

Socket Servidor APK	Comunicar la UI con el Controlador Central. Parte del servidor.	4.7.2
Controlador Central	Gestionar el flujo de trabajo del sistema.	4.7.3
API Cámara	Permitir la conexión funcional entre la cámara y el sistema.	4.7.4
Socket Cliente NST	Comunicar el módulo NST con el Controlador Central. Parte del cliente.	4.7.5
Socket Servidor NST	Comunicar el módulo NST con el Controlador Central. Parte del servidor.	4.7.5
NST	Realizar la transferencia de estilo neuronal.	4.7.6
Visualizador	Mostrar la interfaz de visualización y los resultados del módulo NST.	4.7.7

Tabla 4.3: Módulos de software de la arquitectura del sistema.

4.7. Descripción de Módulos

4.7.1. Módulo UI

La interfaz de usuario es la vista que permite al usuario interactuar de manera efectiva con el sistema. Es la suma de una arquitectura de información, patrones de interacción y elementos visuales. La UI de la instalación interactiva Van Gogh te pinta permite al usuario seleccionar una de las tres pintura de Vincent van Gogh definidas en los requisitos funcionales: La Noche Estrellada, Autorretrato o Dormitorio en Arlés. Además, permite iniciar la grabación de video que posteriormente será estilizada con la pintura seleccionada. El módulo UI junto con el módulo Socket Cliente APK, constituyen la aplicación que se instala en la tablet (archivo .apk).

Una de las consideraciones que debe tener este módulo es la experiencia del usuario (UX), aquello que el usuario percibe al interactuar con la instalación interactiva. Se logra una buena UX al enfocarse en diseñar productos útiles, usables y deseables, lo cual influye en que el usuario se sienta satisfecho, feliz y encantado. Los visitantes del museo Artequin son en su mayoría niños y niñas, por lo tanto, el diseño UI/UX debe estar enfocado en ellos.

El módulo UI funciona mediante el trabajo en conjunto con el módulo Socket Cliente APK que permite transmitir las selecciones del usuario al computador, donde se generará todo el procesamiento de los datos.

4.7.2. Módulo Socket Cliente-Servidor APK

Se describen los módulos Socket Cliente APK y Socket Servidor APK simultáneamente, porque en conjunto representan el canal de comunicación entre la tablet y el computador. De acuerdo a los requisitos funcionales de la Tabla 4.1, esta comunicación se realiza mediante el protocolo TCP/IP que permite que las aplicaciones se comuniquen en forma segura independiente de las capas inferiores. Esta comunicación es posible gracias a la red local generada por el router.

Estos dos módulos permiten que las interacciones del usuario recopilada en la tablet, esto es, selección de pintura y evento de inicio de grabación, sean transmitidas a la unidad de procesamiento. Notar que mientras el usuario no inicie la grabación, la interfaz de visualización proyectada solo mostrará el resultado estilizado anterior en bucle, por lo tanto, este canal de comunicación es fundamental para el correcto funcionamiento del sistema, porque si falla la transmisión de los datos, el ciclo de trabajo no podrá iniciar. Esto último, es una de las razones para utilizar el protocolo TCP/IP para comunicar la tablet con el computador.

Estos dos módulos trabajan en conjunto con el módulo UI y el módulo Controlador Central. Estos 4 módulos conforman el camino para recibir las interacciones del usuario y transmitirlas al módulo Controlador Central, el cual inicia el flujo de trabajo del sistema, comenzando por la grabación de video, luego el procesamiento del video con el algoritmo NST y finalmente, la muestra de los resultados en la interfaz de visualización.

4.7.3. Módulo Controlador Central

El módulo Controlador Central es el que guía todos los cambios de estado del sistema. El estado de reposo o sin utilizar de la instalación interactiva mantiene reproduciendo el ultimo video estilizado en bucle. Cuando el usuario interactúa con la instalación interactiva, el sistema cambia de estado para grabar el video, luego procesar el video con las redes neuronales y finalmente, mostrar los resultados. Este módulo se preocupa de guiar el flujo de datos para que cada módulo funcione correctamente en el momento adecuado del ciclo general del sistema.

El módulo Controlador Central interactúa con todos los módulos del sistema, directa o indirectamente, para llevar a cabo con éxito el ciclo de trabajo de la instalación.

4.7.4. Módulo API Cámara

Este módulo se encarga de comunicar la cámara con el computador y el resto de los módulos del sistema. En este módulo se utiliza OpenCV, la herramienta de visión por computador seleccionada en el Capítulo 3, para la lectura y el procesamiento de los datos de video. Además, se hace uso de las herramientas provistas por el proveedor del hardware, facilitando aún más la tarea de procesar los datos de video. Asimismo, este módulo se encarga de guardar los videos grabados para su posterior procesamiento.

4.7.5. Módulo Socket Cliente-Servidor NST

De la misma forma que en la sección 4.7.2, se describen los módulos Socket Cliente NST y Socket Servidor NST en conjunto porque representan el canal de comunicación entre el programa en Unity y el código en Python para las redes neuronales. Esta canal de comunicación permite a ambos programas sincronizar sus procesos; el algoritmo NST debe esperar a que el video sea grabado para comenzar el procesamiento y, por otro lado, el programa en Unity debe esperar a que la transferencia de estilo este completa para poder mostrarla.

Como ambos programas se encuentran en el computador, la comunicación se realiza mediante *localhost* TCP/IP pues la transmisión de datos tiene como destino el propio host. Por esta razón, en el diagrama de la Figura 4.4 la comunicación entre el Socket Cliente NST y el Socket Servidor NST es procesada internamente sin transitar por el router.

Estos dos módulos trabajan en conjunto con el módulo NST y el módulo Controlador Central. Estos 4 módulos conforman el camino para adherir la transferencia de estilo neuronal al flujo de trabajo de la instalación interactiva.

4.7.6. Módulo NST

Este módulo se encarga de gestionar y ejecutar las redes neuronales para la transferencia de estilo. En el Capítulo 3 se escogió el modelo NST de estilo único, por lo tanto, este módulo debe gestionar las tres redes neuronales convolucionales que trans-

fieren los estilos de “La Noche Estrellada”, el “Autorretrato” y “Dormitorio en Arlés”, al video grabado por el usuario.

Por otro lado, para que este módulo pueda ejecutar correctamente el algoritmo NST, es necesario realizar las lecturas correspondientes al archivo de video grabado por el sistema. En este sentido, para realizar dicho procesamiento de datos de video, el módulo NST hace uso de la biblioteca de OpenCV para poder obtener cada frame de video con el formato correcto y poder alimentar la red neuronal de transferencia de estilo correspondiente. Bajo esta misma idea, este módulo también utiliza OpenCv para guardar el video estilizado en el formato definido en los requisitos del sistema.

Este módulo trabaja en conjunto con los módulos Socket Cliente NST, Socket Servidor NST y Controlador Central. El socket le permite comunicarse con el Controlador Central para sincronizar el procesamiento de video. Por otro lado, el módulo Controlador Central le indica cuando iniciar el algoritmo NST y, al finalizar el procesamiento, el módulo NST responde para que el ciclo de trabajo del sistema continúe.

4.7.7. Módulo Visualizador

Este módulo es el encargado de mostrar el resultado de todo el proceso del sistema. Este módulo es principalmente una interfaz de visualización, que muestra distintas escenas dependiendo el estado del sistema, por ejemplo, grabando video, procesando video, mostrando resultado estilizado, etc. De la misma manera que en el modulo UI (sección 4.7.1), el diseño UX y el diseño de la interfaz deben ser desarrollados asumiendo que los usuarios serán, en su mayoría, niños y niñas que desean disfrutar y divertirse haciendo arte.

A diferencia del módulo UI, el módulo Visualizador solo entrega información a los usuarios y no permite la interacción con él. La proyección de la interfaz de visualización siempre estará presente, independiente de si el sistema es utilizado o no. Se podría decir que tiene una doble función, por un lado, atraer a los visitantes a interactuar con el sistema y, por otro, ser una exposición viva en el museo.

Este módulo trabaja en conjunto con el módulo Controlador Central y el proyector. El primero, permite guiar al módulo Visualizador en las escenas que debe mostrar. Por otro lado, el proyector es el elemento que permite proyectar la interfaz y que sea visible para el usuario. El trabajo unido de estos tres elementos permite mostrar al usuario una visualización continua y coherente de cada estado del sistema en forma sincronizada con sus acciones.

4.8. Matriz de Requisitos Funcionales y Módulos

Cada uno de los requisitos funcionales de la Tabla 4.1 es implementado por uno o más módulos de la arquitectura del sistema de la Figura 4.4. A continuación, se presenta la matriz que relaciona los requisitos funcionales con sus respectivos módulos de la arquitectura del sistema.

	Módulos								
Requisitos Funcionales	UI	Socket Cliente APK	Socket Servidor APK	Controlador Central	API Cámara	Socket Cliente NST	Socket Servidor NST	NST	Visualizador
RF1	x	x	x	x					
RF2	x	x	x	x	x				
RF3				x					x
RF4	x	x	x	x					x
RF5				x	x				
RF6					x				x
RF7				x					x
RF8				x		x	x	x	x
RF9								x	
RF10				x					x
RF11					x				
RF12						x	x	x	
RF13									x
RF14		x	x			x	x		

Tabla 4.4: Matriz de requisitos funcionales y módulos.

4.9. Flujo de Datos Entre Módulos

Considerando la arquitectura de la instalación interactiva Van Gogh te pinta, se define el flujo de datos que debe manejar el sistema para lograr un correcto funcionamiento y dar cumplimiento a los requisitos del sistema. En la Figura 4.5 se muestra el diagrama de flujo de datos que refleja de forma clara y precisa los procesos que conforman el sistema. Este diagrama presenta la transmisión de información a nivel de módulos, identificando de forma simple las entradas y salidas de cada uno.

Para presentar de una forma simple y precisa los proceso que conforman el sistema, en el diagrama de flujo de datos se han simplificado las conexiones entre los módulos

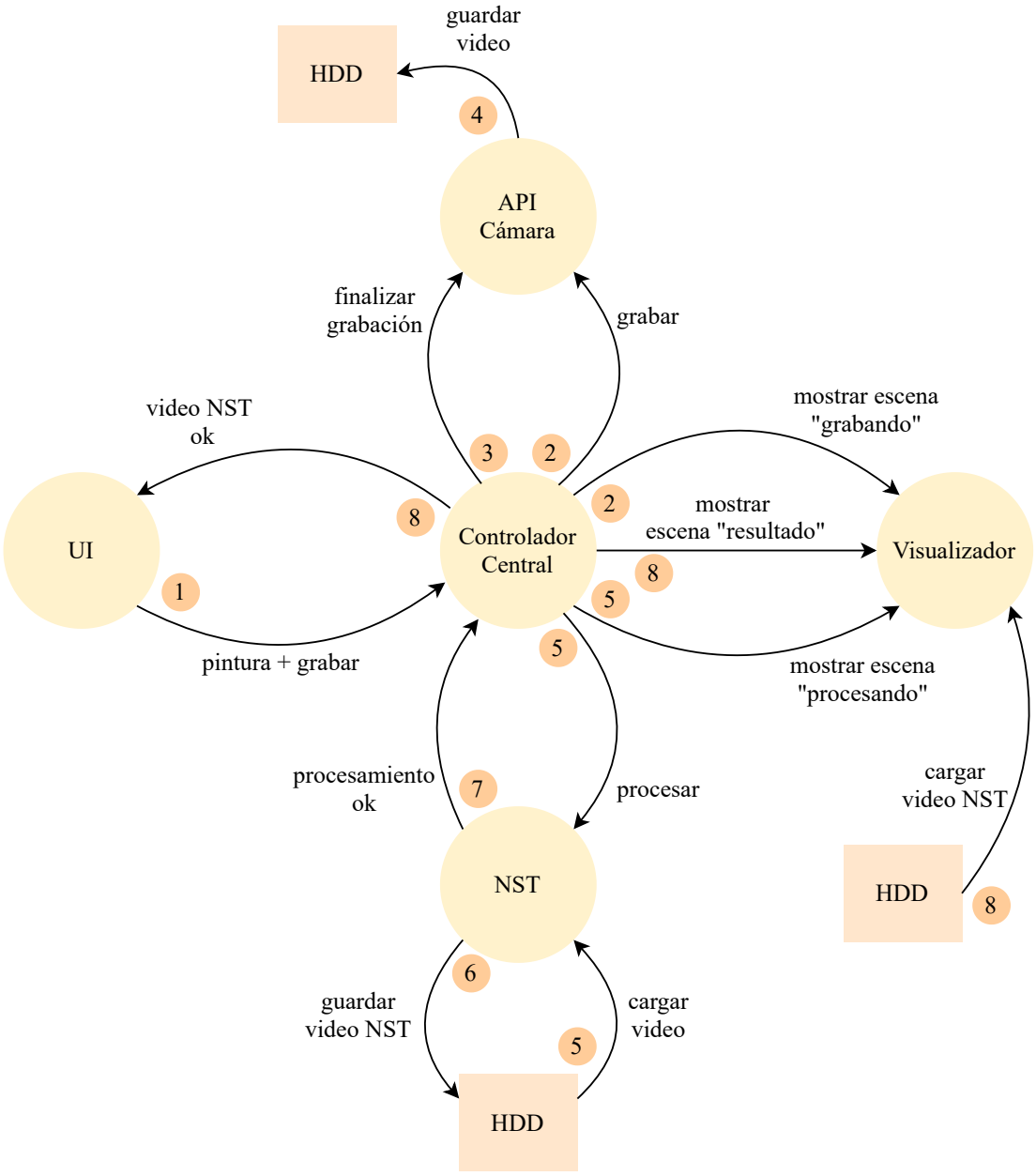


Figura 4.5: Diagrama de flujo de datos.

respecto del diagrama de arquitectura de componentes y software de la Figura 4.4. En el diagrama de flujo de datos solo se muestran los módulos de software, y los módulos Socket Cliente-Servidor APK y Socket Cliente-Servidor NST son omitidos porque su propósito es solo lograr la comunicación de los módulos adyacentes a ellos. Además, el flujo de información se muestra en términos de acciones para proporcionar un entendimiento intuitivo del diagrama.

El diagrama de flujo de datos presenta una numeración que indican la secuencia de cada proceso. Siguiendo la numeración de la Figura 4.5, el flujo de información se detalla de la siguiente manera:

1. El usuario interactúa con la UI para seleccionar la pintura e iniciar la grabación. En seguida, esta información es transmitida al módulo Controlador Central.

2. Luego, el Controlador Central le indica al módulo API Cámara que inicie la grabación y, paralelamente, el Visualizador debe mostrar la escena "grabando" que indica al usuario que se ha iniciado la grabación y muestra la vista desde la cámara en la proyección.

3. Luego de transcurrir 8 segundos de grabación, el Controlador Central le indica al módulo API Cámara que finalice la grabación.

4. Al finalizar la grabación, se guarda el video en el disco duro del computador.

5. A continuación, el Controlador Central le ordena al módulo NST que inicie el procesamiento porque el video ya se encuentra grabado, y le indica el estilo (pintura) a transferir. El módulo NST toma el video grabado desde el disco duro del computador y lo procesa. En paralelo, el módulo Visualizador debe mostrar la escena "procesando" mientras el video no termine de procesarse.

6. Al terminar la transferencia de estilo de video, el resultado se guarda en el disco duro del computador.

7. Y se le indica al Controlador Central que la transferencia de estilo ha finalizado.

8. Finalmente, el Visualizador debe mostrar la escena "resultado" que carga el video estilizado desde el disco duro y lo reproduce en bucle. En paralelo, el Controlador Central le indica al módulo UI que el video ha sido procesado, esto permite que la interfaz de usuario pueda iniciar nuevamente otro procesamiento.

El diagrama de flujo de datos expone que la instalación interactiva Van Gogh te pinta es un ciclo de procesos (1 al 8 en la Figura 4.5) que funciona una y otra vez mientras la instalación se encuentra en funcionamiento. Este ciclo de trabajo no avanzará hasta que un usuario seleccione una pintura e inicie la grabación (1 en la Figura 4.5). En consecuencia, la instalación interactiva se mantendrá mostrando el video estilizado (8 en la Figura 4.5) hasta que un visitante interactúe con la UI.

Capítulo 5

Desarrollo de la Instalación Interactiva

Considerando el diseño de la instalación interactiva (Capítulo 4) y las herramientas propuestas para su implementación (Capítulo 3), en este capítulo se detalla el desarrollo del PMV de la instalación interactiva de arte digital Van Gogh te pinta.

5.1. Consideraciones para el Desarrollo de la Solución

Como se mencionó en la sección 4.2 del capítulo anterior, el alcance del desarrollo de la instalación interactiva es construir un PMV, esto implica que el diseño planteado en el Capítulo 4 y el desarrollo que se describe en este capítulo, tendrán módulos que adicionar o modificar para llegar al producto final. Estos análisis y mejoras futuras quedan fuera del alcance de esta memoria. Aquí solo se pretende alcanzar la primera versión del PMV y su posterior instalación en el museo Artequin Viña del Mar.

Otra consideración que se debe tener presente para el desarrollo de la solución es el lugar de instalación. Las dimensiones del espacio son 230 cm de alto, 120 cm de ancho y 332 cm de profundidad, y es acondicionado para la instalación. Esta información es importante pues influye en la elección, construcción y disposición de los elementos de la instalación interactiva. Por una parte, el espacio de instalación es bastante reducido, por lo que es de suma importancia escoger componentes pequeños para la implementación de la instalación. Por otro lado, se deben construir estructuras que contengan las unidades de procesamiento, tablet, etc., que a su vez no interfieran en la puesta en escena.

Finalmente, se debe dejar en claro que el diagrama de componentes de la solución siempre será el mismo, pero la disposición de estos elementos en el espacio del cliente

será distinto uno de otro. La construcción de las estructuras físicas que contengan el hardware de la instalación y su posterior montaje en el espacio de instalación se detallan en en el Capítulo 6.

5.2. Componentes del Sistema

De acuerdo al diagrama de arquitectura de componentes de la Figura 4.3 del Capítulo 4, la instalación interactiva consta de 5 componentes. Electroveja Labs dispone de cada uno de ellos para el desarrollo de la instalación. A continuación, se realiza una descripción general de cada componente.

5.2.1. Cámara

Se utiliza una Intel® RealSense™ Depth Camera D435 (Figura 5.1). Si bien es una cámara de profundidad, posee un sensor RGB bastante bueno. Por otro lado, la startup solo poseía este modelo de cámaras, por lo tanto, como cumple con los requerimientos del sistema, se opta por usarla y evitar aumentar los costos del proyecto comprando otra cámara.



Figura 5.1: Intel® RealSense™ Depth Camera D435.

Una de los requerimientos más importantes de la cámara son los fps en función de la resolución de la imagen. La cámara RealSense presenta buenas especificaciones técnicas en este aspecto. La Tabla 5.1 muestra un resumen de estas especificaciones para su sensor RGB.

Resolución	**FPS**
1920×1080	$6, 15, 30$
1280×720	$6, 15, 30$
640×480	$6, 15, 30, 60$

Tabla 5.1: Velocidad de fotogramas en función de la resolución de la imagen para el sensor RGB de la cámara RealSense [114].

Para más detalles de las especificaciones técnicas de la cámara RealSense ingresar a la web del proveedor [114].

5.2.2. Tablet

Se utiliza una Tablet Samsung Galaxy Tab E (Figura 5.2). La startup Electroveja Labs pone a disposición del proyecto esta tablet debido a su buena resolución de pantalla (1280×800 (WXGA)) y su gran tamaño ($9{,}6''$ ($243{,}4$ mm)). Estas características permiten que la UI instalada en ella sea legible y de fácil interacción para el usuario.



Figura 5.2: Tablet Samsung Galaxy Tab E [115].

Para más detalles de las especificaciones técnicas de la tablet Samsung Galaxy Tab E ingresar a la web del proveedor [115].

5.2.3. Router

Se utiliza un router DIR-608 Wireless N 150 (Figura 5.3). Es un router básico, pero que cumple con los requerimientos del sistema, esto es, configurar una red local con dos hosts, un computador y una tablet. Además, tiene la ventaja de ser pequeño, sus dimensiones son $180{,}56 \times 91{,}07 \times 25{,}37$ mm (largo x ancho x alto), lo que beneficia en gran manera al momento de realizar el montaje del sistema.



Figura 5.3: Router DIR-608 Wireless N 150 [116].

5.2.4. Computador

Se utiliza un Mini PC Zotac ZBOX QK5P1000 (Figura 5.4). Electroveja Labs dispone de este Mini PC porque sus dimensiones son pequeñas, $184{,}6 \times 184{,}6 \times 71{,}5$ mm (largo x ancho x alto), y tiene buenas especificaciones técnicas [117]. Lo primero facilita la instalación del sistema, y lo segundo es necesario para ejecutar las redes neuronales. Notar que el Mini PC posee una GPU NVIDIA Quadro P1000 que posee 640 CUDA cores y 4GB de memoria, unas especificaciones aceptables para ejecutar con buen rendimientos las redes neuronales convoluciones de los algoritmos NST.



Figura 5.4: Mini PC Zotac ZBOX QK5P1000 [117].

Para más detalles de las especificaciones técnicas del Mini PC Zotac ZBOX QK5P1000 ingresar a la web del proveedor [117].

5.2.5. Proyector

Se utiliza un proyector láser LG ProBeam HF80LG (Figura 5.5) de alta resolución (1920x1080) e intensidad (2000 lumen). El proyector es pequeño, $108 \times 252{,}3 \times 140$ mm (ancho x profundidad x altura), lo que permite ocultarlo fácilmente en el techo del espacio de instalación.



Figura 5.5: Proyector láser LG ProBeam HF80LG [118].

Para más detalles de las especificaciones técnicas del proyector LG ProBeam HF80LG ingresar a la web del proveedor [118].

5.3. Consideraciones para el Desarrollo de los Módulos de Software

El desarrollo de los módulos de software de la arquitectura del sistema (Figura 4.4) es un proceso iterativo y en conjunto con los miembros del equipo de Electroveja Labs. Cada módulo que se detalla en las siguientes secciones ha sido modificado una y otra vez, pero para claridad de la memoria se expone el resultado final del desarrollo de cada módulo. En particular, el desarrollo de la red neuronal convolucional para realizar la transferencia de estilo neuronal sigue el flujo de trabajo común de un proyecto de machine learning. Esto consiste en un ciclo iterativo de entrenamiento y prueba del algoritmo escogido, complementado con posibles optimizaciones a la arquitectura de la red, para finalmente llegar al despliegue del modelo para su uso en un entrono real.

5.4. Módulo NST

El módulo NST es el primer módulo en ser explicado porque es la tecnología central de la solución. Además, es el módulo más complejo y requiere de ciertas optimizaciones que son explicados en el apéndice del presente escrito.

Tomando en consideración lo expuesto en el Capítulo 3, el módulo NST implementa el método PSPM-MOB-NST de Johnson et al. [52], pero con algunas modificaciones. A continuación, se detalla la implementación del método de transferencia de estilo neuronal para los 3 estilos de pintura definidos en los requisitos del sistema. Se detalla la teoría del método en paralelo con su implementación en TensorFlow (TF). Además,

se debe aclarar que la arquitectura de red neuronal utilizada es producto de una serie de optimizaciones realizadas, las cuales se detallan en el Apéndice B.

5.4.1. Método

Como se muestra en la Figura 5.6, el método consta de dos componentes: una *red de transformación de imágenes* f_W y una *red de pérdidas* ϕ que se utiliza para definir varias *funciones de pérdida* $\ell_1, \ldots, \ell_k$. La red de transformación de imágenes es una red neuronal convolucional residual profunda parametrizada por los pesos W; transforma las imágenes de entrada x en imágenes de salida $\hat{y}$ mediante el mapeo $\hat{y} = f_W(x)$. Cada función de pérdida calcula un valor escalar $\ell_i(\hat{y}, y_i)$ que mide la diferencia entre la imagen de salida $\hat{y}$ y una imagen objetivo y_i.

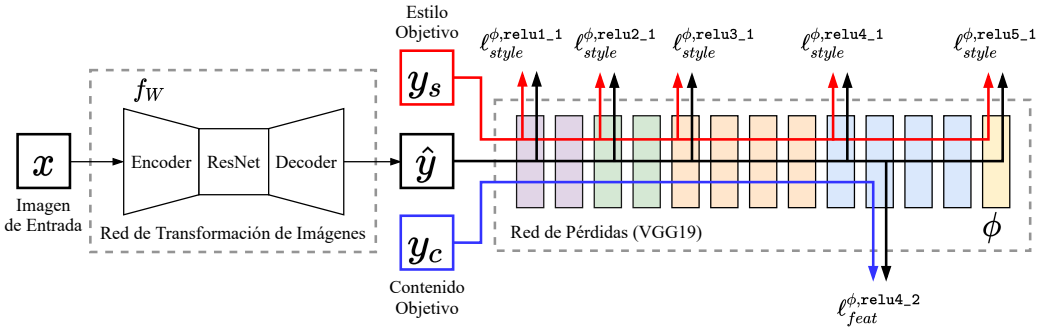


Figura 5.6: Visión general del algoritmo de transferencia de estilo neuronal implementado

La red de transformación de imágenes se entrena usando descenso de gradiente para minimizar una combinación ponderada de funciones de pérdida:

$$W^* = \arg\min_W \mathbf{E}_{x,\{y_i\}} \left[\sum_{i=1} \lambda_i \ell_i \left(f_W(x), y_i \right) \right] \tag{5.1}$$

La idea clave de estos métodos es que las redes neuronales convolucionales pre-entrenadas para la clasificación de imágenes ya han aprendido a codificar la información perceptiva y semántica que se gustaría medir en las funciones de pérdida. Por lo tanto, se hace uso de una red ϕ que ha sido pre-entrenada para la clasificación de imágenes como una red de pérdidas fija con el fin de definir las funciones de pérdida. Entonces, la red de transformación de imágenes se entrena usando funciones de pérdida que también son redes convolucionales profundas.

La red de pérdidas ϕ se utiliza para definir una *pérdida de reconstrucción de características* ℓ^{ϕ}_{feat} y una *pérdida de reconstrucción de estilo* ℓ^{ϕ}_{style}, que miden las diferencias de contenido y estilo entre las imágenes respectivamente. Para cada imagen de entrada x se tiene un contenido objetivo y_c y un estilo objetivo y_s. Para la transferencia de estilo, el contenido objetivo y_c es la imagen de entrada x y la imagen de salida $\hat{y}$ debe combinar el contenido de $x = y_c$ con el estilo de y_s; se entrena una red por cada estilo objetivo.

5.4.2. Red de Transformación de Imágenes

Las redes de transferencia de estilo usan la arquitectura mostrada en la Tabla 5.2. En esta tabla "$C \times H \times W$ conv" denota una capa convolucional con C filtros de tamaño $H \times W$, seguida inmediatamente por la normalización de instancias [54] y una no linealidad ReLU, con la excepción de la capa de salida, que en su lugar utiliza la función Tanh escalada para asegurar que la imagen de salida tenga pixeles en el rango $[0, 255]$.

Capa	Tamaño de activación
Entrada	$3 \times 256 \times 256$
Padding	$3 \times 272 \times 272$
$32 \times 9 \times 9$ conv, stride 1	$32 \times 272 \times 272$
$64 \times 3 \times 3$ conv, stride 2	$64 \times 136 \times 136$
$128 \times 3 \times 3$ conv, stride 2	$128 \times 68 \times 68$
Bloque residual, 128 filtros	$128 \times 64 \times 64$
$64 \times 3 \times 3$ conv, stride $1/2$	$64 \times 128 \times 128$
$32 \times 3 \times 3$ conv, stride $1/2$	$32 \times 256 \times 256$
$3 \times 9 \times 9$ conv, stride 1	$3 \times 256 \times 256$

Tabla 5.2: Arquitectura de red utilizada para las redes de transferencia de estilo.

La arquitectura de red utilizada para las redes de transferencia de estilo presenta algunas diferencias respecto a la red utilizada por [52]. Esta arquitectura es el resultado de una serie de pruebas y optimizaciones realizadas para obtener tiempos de inferencia menores sin perder la calidad en el resultado estilizado. Las pruebas realizadas se encuentran detalladas en el Apéndice B. Además, se han utilizado las mejoras de [54] que produce mejores resultados visuales al utilizar normalización de instancias en lugar de

normalización por lotes. La implementación de la red de transformación de imágenes en TensorFlow 1 se muestra en la Figura 5.7.

```
def net(self, image):
    image_p = self._reflection_padding(image)
    conv1 = self._conv_layer(image_p, 32, 9, 1, name='conv1')
    conv2 = self._conv_layer(conv1, 64, 3, 2, name='conv2')
    conv3 = self._conv_layer(conv2, 128, 3, 2, name='conv3')
    resid1 = self._residual_block(conv3, 3, name='resid1')
    conv_t1 = self._conv_transpose_layer(resid3, 64, 3, 2, name='convt1')
    conv_t2 = self._conv_transpose_layer(conv_t1, 32, 3, 2, name='convt2')
    conv_t3 = self._conv_layer(conv_t2, 3, 9, 1, relu=False, name='convt3')
    with tf.name_scope("Tanh"):
        preds = (tf.nn.tanh(conv_t3) + 1) * (255. / 2)
        return preds
```

Figura 5.7: Implementación de la red de transformación de imágenes en TF.

Entradas y salidas. Para el entrenamiento, la entrada y la salida de la red de transformación de imágenes son ambas imágenes de color de dimensiones $3 \times 256 \times 256$. Esto quiere decir que la red no modifica las dimensiones de la imagen de entrada respecto a la salida, solo modifica sus valores. Para la inferencia, la imagen de entrada puede tener las dimensiones que se deseen.

Padding. Se agregan filas y columnas a la imagen (padding) para que la salida de la red, luego de todas las operaciones, posea las mismas dimensiones que la entrada, y permita calcular las operaciones de pérdida pertinentes.

```
def _reflection_padding(self, net):
    with tf.name_scope("reflection_padding"):
        return tf.pad(net,[[0, 0],[8, 8],[8, 8], [0, 0]], "REFLECT")
```

Figura 5.8: Implementación de la capa de padding en TF.

Capa de convolución. La red utiliza dos convoluciones con stride 2 para reducir el tamaño de la entrada. La implementación de la capa convolucional se muestra en la Figura 5.9.

La capa de convolución realiza una operación de convolución seguida inmediatamente por la normalización de instancias y una no linealidad ReLU. En [119] se define la normalización de instancias de la siguiente manera: Sea $x \in \mathbb{R}^{T \times C \times W \times H}$ un tensor

```
def _conv_layer(self, net, num_filters, filter_size, strides, padding='SAME', relu=True, name=None):
    with tf.name_scope(name):
        weights_init = self._conv_init_vars(net, num_filters, filter_size, name=name)
        strides_shape = [1, strides, strides, 1]
        net = tf.nn.conv2d(net, weights_init, strides_shape, padding=padding)
        net = self._instance_norm(net, name=name)
        if relu:
            net = tf.nn.relu(net)
        return net
```

Figura 5.9: Implementación de la capa convolucional en TF.

de entrada que contiene un lote de imágenes T. Sea x_{tijk} el elemento $tijk$, donde k y j abarcan las dimensiones espaciales, i es el canal de características (canal de color si la entrada es una imagen RGB), y t es el índice de la imagen en el batch, entonces la normalización de instancias se obtiene de la siguiente manera:

$$\hat{x}_{tijk} = \frac{x_{tijk} - \mu_{ti}}{\sqrt{\sigma_{ti}^2 + \epsilon}}, \quad \mu_{ti} = \frac{1}{HW}\sum_{l=1}^{W}\sum_{m=1}^{H} x_{tilm}, \quad \sigma_{ti}^2 = \frac{1}{HW}\sum_{l=1}^{W}\sum_{m=1}^{H}(x_{tilm} - \mu_{ti})^2 \tag{5.2}$$

Luego, como en [120], todos los métodos de normalización (BN, IN, etc.) aprenden una transformación lineal por canal para compensar la posible pérdida de capacidad de representación:

$$y_{tijk} = \gamma \hat{x}_{tijk} + \beta \tag{5.3}$$

Donde γ y β son parámetros entrenables de escala y desplazamiento respectivamente (indexado por canal en todos los casos, que se omite para simplificar la notación). A continuación, se muestra la implementación de la normalización de instancias.

```
def _instance_norm(self, net, name=None):
    batch, rows, cols, channels = [i.value for i in net.get_shape()]
    var_shape = [channels]
    mu, sigma_sq = tf.nn.moments(net, [1,2], keep_dims=True)
    with tf.variable_scope(name, reuse=self.reuse):
        shift = tf.get_variable('shift', initializer=tf.zeros(var_shape), dtype=tf.float32)
        scale = tf.get_variable('scale', initializer=tf.ones(var_shape), dtype=tf.float32)
    epsilon = 1e-3
    normalized = (net-mu)/(sigma_sq + epsilon)**(.5)
    return scale * normalized + shift
```

Figura 5.10: Implementación de la normalización de instancias en TF.

Bloque residual. Las conexiones residuales facilitan que la red aprenda la función identidad, esta es una propiedad atractiva para las redes de transformación de imágenes, ya que en la mayoría de los casos la imagen de salida debe compartir estructura con la imagen de entrada. El cuerpo de la red consiste así en un bloque residual, el cual tiene la siguiente forma:

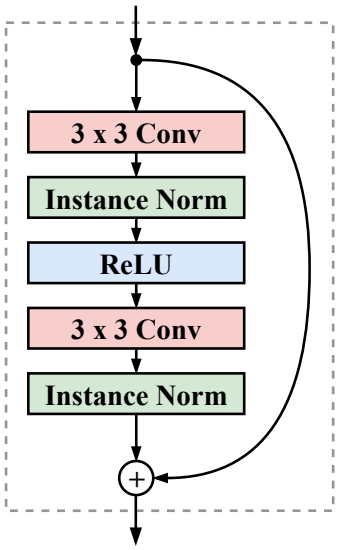


Figura 5.11: Diseño del bloque residual.

El bloque residual contiene dos capas convolucionales con filtros de tamaño 3×3, dos capas de normalización de instancias y una no linealidad ReLU. La implementación del bloque residual se muestra a continuación.

```
def _residual_block(self, net, filter_size=3, name=None):
    batch, rows, cols, channels = [i.value for i in net.get_shape()]
    tmp = self._conv_layer(net, 128, filter_size, 1, padding='VALID', relu=True, name=name + "_conv1")
    tmp2 = self._conv_layer(tmp, 128, filter_size, 1, padding='VALID', relu=False, name=name + "_conv2")
    with tf.variable_scope(name + "_add"):
        return tmp2 + tf.slice(net, [0,2,2,0], [batch,rows-4,cols-4,channels])
```

Figura 5.12: Implementación del bloque residual en TF.

Capa de convolución transpuesta. Una capa convolucional transpuesta se utiliza normalmente para el upsampling, es decir, para generar un mapa de características de salida que tiene una dimensión espacial mayor que la del mapa de características de entrada. Al igual que la capa convolucional estándar, la capa convolucional transpuesta también está definida por el padding y el stride. Estos valores de padding y stride son

los que hipotéticamente se utilizaron en la salida para generar la entrada, es decir, si se toma la salida y se realiza una convolución estándar con stride y padding definidos, generará la dimensión espacial igual que la de la entrada.

En la Tabla 5.2, la capa convolucional con stride fraccionado hace referencia a una convolución transpuesta de stride inverso, por ejemplo, la capa de convolución con stride $1/2$ es equivalente a la capa de convolución transpuesta con stride 2. De la misma forma, la capa de convolución transpuesta realiza una una operación de convolución transpuesta seguida por al normalización de instancias y una no linealidad ReLU. A continuación, se muestra la implementación de la capa de convolución transpuesta en TensorFlow.

```
def _conv_transpose_layer(self, net, num_filters, filter_size, strides, name=None):
    with tf.name_scope(name):
        weights_init = self._conv_init_vars(net, num_filters, filter_size, transpose=True, name=name)

        batch_size, rows, cols, in_channels = [i.value for i in net.get_shape()]
        new_rows, new_cols = int(rows * strides), int(cols * strides)

        new_shape = [batch_size, new_rows, new_cols, num_filters]
        tf_shape = tf.stack(new_shape)
        strides_shape = [1,strides,strides,1]

        net = tf.nn.conv2d_transpose(net, weights_init, tf_shape, strides_shape, padding='SAME')
        net = self._instance_norm(net, name=name)
        return tf.nn.relu(net)
```

Figura 5.13: Implementación de la capa de convolución transpuesta en TF.

5.4.3. Funciones de Pérdida

Se definen dos funciones de pérdida perceptiva que miden las diferencias perceptivas y semánticas de alto nivel entre imágenes. Estas funciones utilizan una red de pérdidas ϕ pre-entrenada para la clasificación de imágenes, lo que significa que estas funciones de pérdida perceptiva son en sí mismas redes neuronales convolucionales profundas. Específicamente, la red ϕ es la red VGG de 19 capas [47] pre-entrenada en el conjunto de datos ImageNet [121].

Pérdida de reconstrucción de características. En lugar de fomentar a los pixeles de la imagen de salida $\hat{y} = f_W(x)$ para que coincidan exactamente con los pixeles de la imagen objetivo y, se incita a que tengan representaciones de características similares a las calculadas por la red de pérdidas ϕ. Sea $\phi_j(x)$ las activaciones de la j-ésima

capa de la red ϕ al procesar la imagen x; si j es una capa convolucional entonces $\phi_j(x)$ será un mapa de características de dimensiones $C_j \times H_j \times W_j$. La pérdida de reconstrucción de características se define como la distancia euclidiana al cuadrado (normalizada) entre los mapas de característica:

$$\ell^{\phi,j}_{feat}(\hat{y}, y) = \frac{1}{C_j H_j W_j} \|\phi_j(\hat{y}) - \phi_j(y)\|_2^2 \tag{5.4}$$

Como se demuestra en [52] y se muestra en la Figura 5.14, encontrar una imagen $\hat{y}$ que minimice la pérdida de reconstrucción de características para capas tempranas tiende a producir imágenes que son visualmente indistinguibles de y. A medida que se reconstruye a partir de capas superiores, el contenido de la imagen y la estructura espacial general se conservan, pero el color, la textura y la forma exacta no. El uso de una pérdida de reconstrucción de características para entrenar las redes de transformación de imágenes permite que la imagen de salida $\hat{y}$ sea perceptualmente similar a la imagen objetivo y, pero no la obliga a coincidir exactamente.

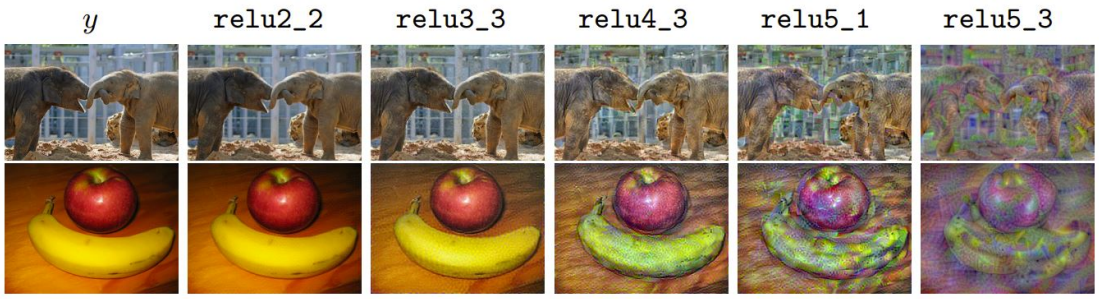


Figura 5.14: En [52] se utiliza la optimización para encontrar una imagen $\hat{y}$ que minimiza la pérdida de reconstrucción de características $\ell^{\phi,j}_{feat}(\hat{y}, y)$ para varias capas j de la red de pérdida ϕ. A medida que se reconstruye a partir de capas superiores, el contenido de la imagen y la estructura espacial general se conservan, pero el color, la textura y la forma exacta no.

Pérdida de reconstrucción de estilo. La pérdida de reconstrucción de característica penaliza la imagen de salida $\hat{y}$ cuando se desvía del contenido de la imagen objetivo y. También se desea penalizar las diferencias de estilo: colores, texturas, patrones comunes, etc. Para lograr este efecto, Gatys et al [40, 46] proponen la siguiente pérdida de reconstrucción de estilo.

Como arriba, sea $\phi_j(x)$ las activaciones en la j-ésima capa de la red ϕ para la entrada x, que es un mapa de características de la forma $C_j \times H_j \times W_j$. Se define la

matriz de Gram $G_j^{\phi}(x)$ como la matriz $C_j \times C_j$ cuyos elementos están dados por

$$G_j^{\phi}(x)_{c,c'} = \frac{1}{C_j H_j W_j} \sum_{h=1}^{H_j} \sum_{w=1}^{W_j} \phi_j(x)_{h,w,c}\, \phi_j(x)_{h,w,c'} \tag{5.5}$$

La matriz de Gram puede ser calculada eficientemente transformando $\phi_j(x)$ en una matriz $\psi_j(x)$ de dimensiones $C_j \times H_j W_j$. Entonces la matriz de Gram se calcula como

$$G_j^{\phi}(x) = \frac{1}{C_j H_j W_j} \psi_j(x) \psi_j^T(x) \tag{5.6}$$

De acuerdo a la ecuación anterior, la matriz de Gram representa el producto escalar de un conjunto de vectores. Esto captura la similitud (o correlación) entre cada vector, ya que si dos vectores son similares entre sí, entonces su producto escalar será grande y, por lo tanto, la matriz de Gram será grande. La Figura 5.15 muestra gráficamente lo explicado anteriormente.

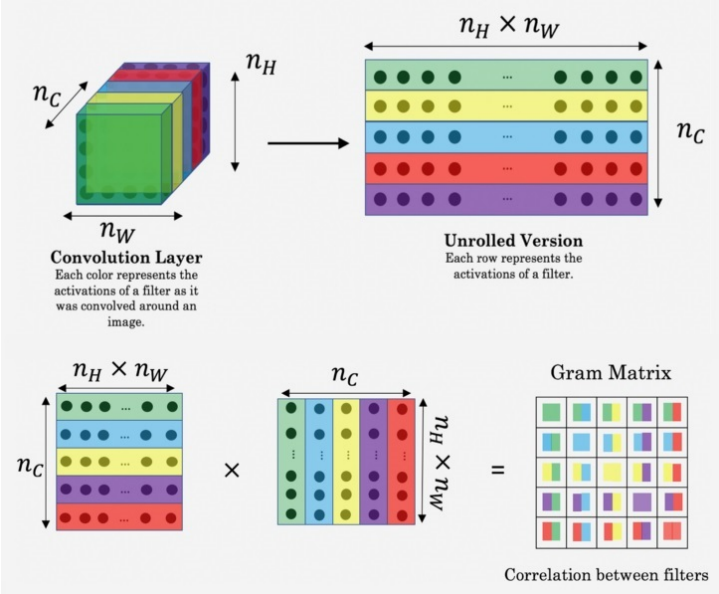


Figura 5.15: Representación intuitiva de la matriz de Gram.

La pérdida de reconstrucción de estilo es entonces la distancia euclidiana al cuadrado entre las matrices Gram de las imágenes de salida y objetivo:

$$\ell_{style}^{\phi,j}(\hat{y}, y) = \left\| G_j^{\phi}(\hat{y}) - G_j^{\phi}(y) \right\|_2^2 \tag{5.7}$$

La pérdida de reconstrucción de estilo está bien definida incluso cuando $\hat{y}$ e y tienen

diferentes tamaños, ya que sus matrices de Gram tendrán las mismas dimensiones.

La matriz de Gram básicamente captura la "distribución de características" de un conjunto de mapas de características en una capa determinada. Al tratar de minimizar la pérdida de estilo entre dos imágenes, básicamente se están haciendo coincidir la distribución de características entre las dos imágenes.

Como se demuestra en [52] y se muestra en la Figura 5.16, generar una imagen $\hat{y}$ que minimiza la pérdida de reconstrucción de estilo, preserva las características estilísticas de la imagen objetivo, pero no preserva su estructura espacial. La reconstrucción a partir de capas más altas transfiere una estructura de mayor escala desde la imagen objetivo.

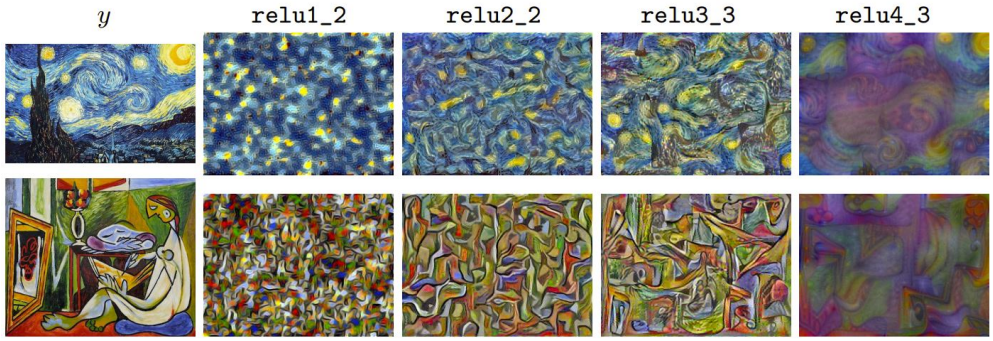


Figura 5.16: En [52] se utiliza la optimización para encontrar una imagen $\hat{y}$ que minimiza la pérdida de reconstrucción de estilo $\ell^{\phi,j}_{style}(\hat{y}, y)$ para varias capas j de la red de pérdida ϕ. Las imágenes $\hat{y}$ conservan características estilísticas pero no la estructura espacial.

Para efectuar la reconstrucción del estilo a partir de un conjunto de capas J en lugar de una sola capa j, se define $\ell^{\phi,J}_{style}(\hat{y}, y)$ como la suma de las pérdidas para cada capa $j \in J$.

Regularización de variación total. Además de las pérdidas perceptivas definidas anteriormente, también se define una función de pérdida simple que depende sólo de la información de pixel de bajo nivel. Para fomentar la suavidad espacial en la imagen de salida $\hat{y}$ se hace uso del regularizador de variación total $\ell_{TV}(\hat{y})$. En [44] este factor se aproxima como

$$\ell_{TV}(\hat{y}) = \sum_{i,j} \left((\hat{y}_{i,j+1} - \hat{y}_{ij})^2 + (\hat{y}_{i+1,j} - \hat{y}_{ij})^2 \right)^{\frac{\beta}{2}} \tag{5.8}$$

Donde β es un factor que ajusta la suavidad de la imagen de salida.

5.4.4. Red de Pérdidas

Para la red de pérdidas ϕ se utiliza la red VGG de 19 capas [47] pre-entrenada en el conjunto de datos ImageNet [121]. La arquitectura de la red VGG-19 se muestra en la Figura 5.17.

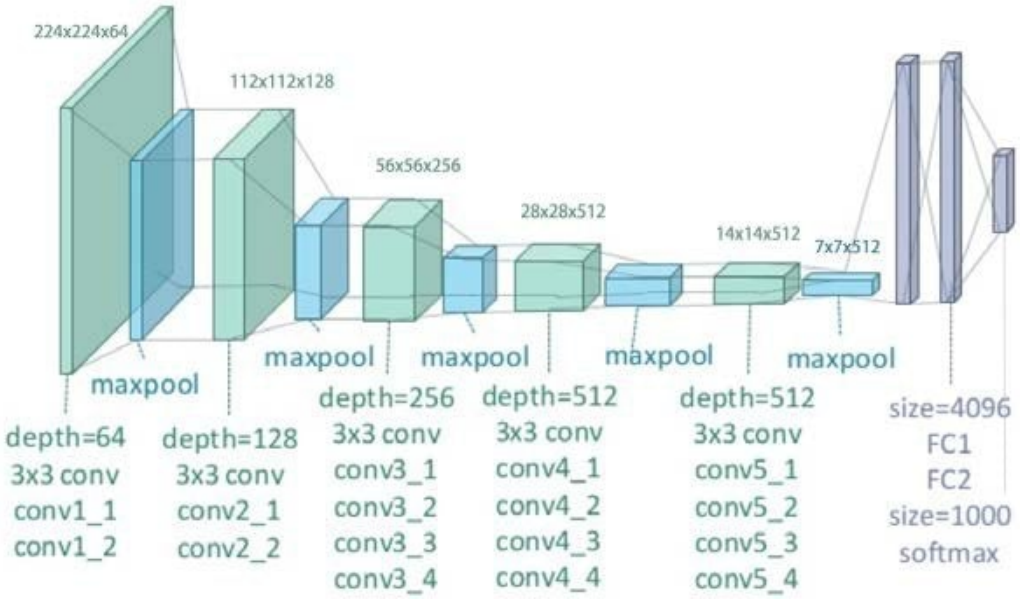


Figura 5.17: Arquitectura de red del modelo VGG-19.

La implementación de la red VGG-19 se lleva a cabo por medio de la creación de una clase en Python que genere la arquitectura de red y los métodos de preprocesamiento necesarios para manipular las entradas y salidas de la red. Esta implementación se muestra en la Figura 5.18.

Los pesos de la red son descargados desde [122] en un archivo *.mat*. Luego, son cargados y almacenados en la variable *weights*. Por otro lado, la red original ha sido entrenada preprocesando los datos de entrada restando la media de cada canal de color del set de datos de entrenamiento (ImageNet). Esta es la razón de la existencia de la variable *mean_pixel* y el método de preprocesamiento que resta los valores de la media (ver Figura 5.18).

Para calcular las funciones de pérdida es necesario obtener los mapas de características de ciertas capas intermedias de la red ϕ. Para lograr esto, se implementa un método que retorna un diccionario con el cual se puede acceder a todas las capas de la red y extraer su respectivo mapa de características (Ver Figura 5.19). Notar que en este método se aplican los respectivos pesos de la red pre-entrenada a las capas convolucionales.

```python
class VGG19:
    layers = (
        'conv1_1', 'relu1_1', 'conv1_2', 'relu1_2', 'pool1',

        'conv2_1', 'relu2_1', 'conv2_2', 'relu2_2', 'pool2',

        'conv3_1', 'relu3_1', 'conv3_2', 'relu3_2', 'conv3_3',
        'relu3_3', 'conv3_4', 'relu3_4', 'pool3',

        'conv4_1', 'relu4_1', 'conv4_2', 'relu4_2', 'conv4_3',
        'relu4_3', 'conv4_4', 'relu4_4', 'pool4',

        'conv5_1', 'relu5_1', 'conv5_2', 'relu5_2', 'conv5_3',
        'relu5_3', 'conv5_4', 'relu5_4'
    )

    def __init__(self, data_path):
        data = scipy.io.loadmat(data_path)

        self.mean_pixel = np.array([123.68, 116.779, 103.939])

        self.weights = data['layers'][0]

    def preprocess(self, image):
        return image-self.mean_pixel

    def undo_preprocess(self,image):
        return image+self.mean_pixel
```

Figura 5.18: Implementación red VGG-19 en TF.

```python
def feed_forward(self, input_image, scope=None):
    net = {}
    current = input_image

    with tf.variable_scope(scope):
        for i, name in enumerate(self.layers):
            kind = name[:4]
            if kind == 'conv':
                kernels = self.weights[i][0][0][2][0][0]
                bias = self.weights[i][0][0][2][0][1]

                # matconvnet: weights are [width, height, in_channels, out_channels]
                # tensorflow: weights are [height, width, in_channels, out_channels]
                kernels = np.transpose(kernels, (1, 0, 2, 3))
                bias = bias.reshape(-1)

                current = _conv_layer(current, kernels, bias)
            elif kind == 'relu':
                current = tf.nn.relu(current)
            elif kind == 'pool':
                current = _pool_layer(current)
            net[name] = current

    assert len(net) == len(self.layers)
    return net
```

Figura 5.19: Método para extraer los mapas de características de las capas de la red VGG-19.

5.4.4.1. Implementación de las Funciones de Pérdida

La Figura 5.20 muestra el cálculo de las activaciones de la red VGG-19 dado una imagen objetivo de entrada. Notar que se tiene un estilo objetivo y un contenido objetivo.

```
# graph input
self.y_c = tf.placeholder(tf.float32, shape=self.batch_shape, name='content')
self.y_s = tf.placeholder(tf.float32, shape=self.y_s0.shape, name='style')

# preprocess for VGG
self.y_c_pre = self.net.preprocess(self.y_c)
self.y_s_pre = self.net.preprocess(self.y_s)

# get content-layer-feature for content loss
content_layers = self.net.feed_forward(self.y_c_pre, scope='content')
self.Ps = {}
for id in self.CONTENT_LAYERS:
    self.Ps[id] = content_layers[id]

# get style-layer-feature for style loss
style_layers = self.net.feed_forward(self.y_s_pre, scope='style')
self.As = {}
for id in self.STYLE_LAYERS:
    self.As[id] = self._gram_matrix(style_layers[id])
```

Figura 5.20: Activaciones de la red de pérdida $\hat{y}$ dado una imagen de contenido objetivo y una imagen de estilo objetivo.

En la Figura 5.20, *net* es un objeto de clase VGG19, que se utiliza para obtener las activaciones del contenido objetivo (variable *y_c*) y del estilo objetivo (variable *y_s*).

De la misma forma que en [40], se utiliza la capa `relu4_2` de la red de pérdidas ϕ para la reconstrucción de contenido. El diccionario *Ps* contiene las activaciones $\phi_j(y_c)$ para la entrada de contenido objetivo y_c, donde $j =$ `relu4_2`.

Asimismo, se utilizan las capas `relu1_1`, `relu2_1`, `relu3_1`, `relu4_1` y `relu5_1` de la red de pérdida ϕ para la reconstrucción de estilo. El diccionario *As* contiene la matriz de Gram $G_j^{\phi}(y_s)$ de las activaciones $\phi_j(y_s)$ para la entrada de estilo objetivo y_s, donde $j =$ `relu1_1`, `relu2_1`, `relu3_1`, `relu4_1`, `relu5_1`. La matriz de Gram es calculada mediante la ecuación (5.6) y su implementación se muestra en la Figura 5.21.

Por otro lado, es necesario calcular las mismas activaciones de la red de pérdidas ϕ, pero en este caso, se utiliza como imagen de entrada la salida de la red de transformación de imágenes $\hat{y} = f_W(x)$. En la Figura 5.22 se muestra la implementación para obtener todas las activaciones de la red de pérdidas ϕ dada la entrada $\hat{y}$. Luego, es posible obtener las funciones de pérdida con las variables *Ps* y *As*.

```python
def _gram_matrix(self, tensor, shape=None):
    if shape is not None:
        B = shape[0]  # batch size
        HW = shape[1] # height x width
        C = shape[2]  # channels
        CHW = C*HW
    else:
        B, H, W, C = map(lambda i: i.value, tensor.get_shape())
        HW = H*W
        CHW = W*H*C

    # reshape the tensor so it is a (B, 2-dim) matrix
    # so that 'B'th gram matrix can be computed
    feats = tf.reshape(tensor, (B, HW, C))

    # leave dimension of batch as it is
    feats_T = tf.transpose(feats, perm=[0, 2, 1])

    # paper suggests to normalize gram matrix by its number of elements
    gram = tf.matmul(feats_T, feats) / CHW
    return gram
```

Figura 5.21: Implementación cálculo matriz de Gram en TF.

```python
# result of image transform net
self.x = self.y_c/255.0
self.y_hat = self.transform.net(self.x)

# get layer-values for x
self.y_hat_pre = self.net.preprocess(self.y_hat)
self.Fs = self.net.feed_forward(self.y_hat_pre, scope='mixed')
```

Figura 5.22: Activaciones de la red de pérdida ϕ dada la entrada $\hat{y}$.

Implementación pérdida de reconstrucción de característica. La Figura 5.23 muestra el cálculo de la pérdida de reconstrucción de característica considerando la ecuación (5.4). En este caso, la imagen objetivo es la imagen de contenido $x = y_c$ y la imagen de salida de la red de transformación es $\hat{y}$, por lo tanto, la función queda definida como $\ell^{\phi,j}_{feat}(\hat{y}, y_c)$. En el caso del contenido se obtiene solo un valor de pérdida, por lo tanto, el factor de ponderación a la pérdida total de contenido (variable w) es igual a 1.

```python
if id in self.CONTENT_LAYERS:
    ## content loss ##

    F = self.Fs[id]             # content feature of y_hat
    P = self.Ps[id]             # content feature of y_c

    b, h, w, d = F.get_shape()  # first return value is batch size (must be one)
    b = b.value                 # batch size
    N = h.value*w.value         # product of width and height
    M = d.value                 # number of filters

    w = self.CONTENT_LAYERS[id] # weight for this layer

    L_content += w * 2 * tf.nn.l2_loss(F-P) / (b*N*M)
```

Figura 5.23: Implementación pérdida de reconstrucción de característica en TF.

Implementación pérdida de reconstrucción de estilo. La Figura 5.24 muestra el cálculo de la pérdida de reconstrucción de estilo considerando la ecuación (5.7). En este caso, la imagen objetivo es la imagen de estilo y_s y la imagen de salida de la red de transformación es $\hat{y}$, por lo tanto, la función queda definida como $\ell_{style}^{\phi,J}(\hat{y}, y_s)$. En este caso se obtienen 5 valores de pérdida, dado que son 5 capas de las cuales se extraen los mapas de características, por lo tanto, el factor de ponderación a la pérdida total de estilo (variable w) es igual a $0{,}2$.

```
elif id in self.STYLE_LAYERS:
    ## style loss ##

    F = self.Fs[id]

    b, h, w, d = F.get_shape()          # first return value is batch size (must be one)
    b = b.value                         # batch size
    N = h.value * w.value               # product of width and height
    M = d.value                         # number of filters

    w = self.STYLE_LAYERS[id]           # weight for this layer

    G = self._gram_matrix(F, (b,N,M))   # style feature of y_hat
    A = self.As[id]                     # style feature of y_s

    L_style += w * 2 * tf.nn.l2_loss(G - A) / (b * (M ** 2))
```

Figura 5.24: Implementación pérdida de reconstrucción de estilo en TF.

Implementación regularización de variación total. La Figura 5.25 muestra el cálculo del regularizador de variación total considerando la ecuación (5.8).

```
def _get_total_variation_loss(self, img):
    b, h, w, d = img.get_shape()
    b = b.value
    h = h.value
    w = w.value
    d = d.value
    tv_y_size = (h-1) * w * d
    tv_x_size = h * (w-1) * d
    y_tv = tf.nn.l2_loss(img[:, 1:, :, :] - img[:, :self.batch_shape[1] - 1, :, :])
    x_tv = tf.nn.l2_loss(img[:, :, 1:, :] - img[:, :, :self.batch_shape[2] - 1, :])
    loss = 2. * (x_tv / tv_x_size + y_tv / tv_y_size) / b

    loss = tf.cast(loss, tf.float32)
    return loss
```

Figura 5.25: Implementación regularización de variación total en TF.

Notar que para todas las pérdidas calculadas anteriormente se considera un batch de tamaño b y son normalizadas por los tamaños de los mapas de características.

Implementación pérdida total. La pérdida total considera una suma ponderada de todas las pérdidas obtenidas anteriormente. La función de pérdida total queda definida como:

$$\ell^{\phi}_{total}(\hat{y}, y_c, y_s) = \alpha \ell^{\phi,j}_{feat}(\hat{y}, y_c) + \beta \ell^{\phi,J}_{style}(\hat{y}, y_s) + \gamma \ell_{TV}(\hat{y}) \tag{5.9}$$

Donde las constantes α, β y γ, son hiperparámetros que se ajustan manualmente en la etapa de entrenamiento. La Figura 5.26 muestra el cálculo de la pérdida total en TensorFlow.

```
# Total loss
alpha = self.content_weight
beta = self.style_weight
gamma = self.tv_weight

self.L_content = alpha*L_content
self.L_style = beta*L_style
self.L_tv = gamma*L_tv
self.L_total = self.L_content + self.L_style + self.L_tv
```

Figura 5.26: Implementación de la pérdida total en TF.

5.4.5. Entrenamiento

Dado el estilo objetivo y_s, el contenido objetivo y_c y las capas j y J para realizar la reconstrucción de características y estilo, se genera una imagen $\hat{y}$ resolviendo el problema

$$\hat{y} = \arg\min_{y} \alpha \, \ell^{\phi,j}_{feat}(y, y_c) + \beta \, \ell^{\phi,J}_{style}(y, y_s) + \gamma \, \ell_{TV}(y) \tag{5.10}$$

Donde α, β y γ son constantes, $y_c = x$ una imagen de entrada e $y = f_W(x)$ que cuando llega al óptimo es $\hat{y}$.

La red de transformación de imágenes se entrena en el conjunto de datos Microsoft COCO [123] como imágenes de entrada x. Los estilos escogidos son 3 pinturas de Vincent van Gogh: La Noche Estrellada, Autorretrato y Dormitorio en Arlés (Ver Figura 5.27). Se redimensionan cada una de las 80.000 imágenes de entrenamiento a 256×256, y se entrenan las redes con un tamaño de batch de 32 imágenes en 20 épocas sobre los datos de entrenamiento. Se utiliza el optimizador de Adam [124] con una tasa de aprendizaje de 10^{-3}. Para el entrenamiento de todas las redes se calcula la pérdida de

reconstrucción de características en la capa `relu4_2` y la pérdida de reconstrucción de estilo en las capas `relu1_1`, `relu2_1`, `relu3_1`, `relu4_1` y `relu5_1` de la red de pérdidas VGG-19.

El entrenamiento se lleva a cabo en un servidor con sistema operativo Ubuntu 18.04.3 LTS y GPU Nvidia Tesla P100. La implementación del entrenamiento se realiza con TensorFlow 1.15, Python 3.6.8, CUDA 10.0 y cuDNN 7.5.6; el entrenamiento dura aproximadamente 19 horas en una sola GPU.



(a) La Noche Estrellada (b) Autorretrato (c) Dormitorio en Arlés

Figura 5.27: Imágenes de estilo objetivo.

La Noche Estrellada. El entrenamiento de este estilo se lleva a cabo con los siguientes hiperparámetros: $\alpha = 5$, $\beta = 5 \times 10^2$ y $\gamma = 2 \times 10^2$. La Figura 5.28 muestra la variación de las funciones de pérdida en función del número de iteraciones.

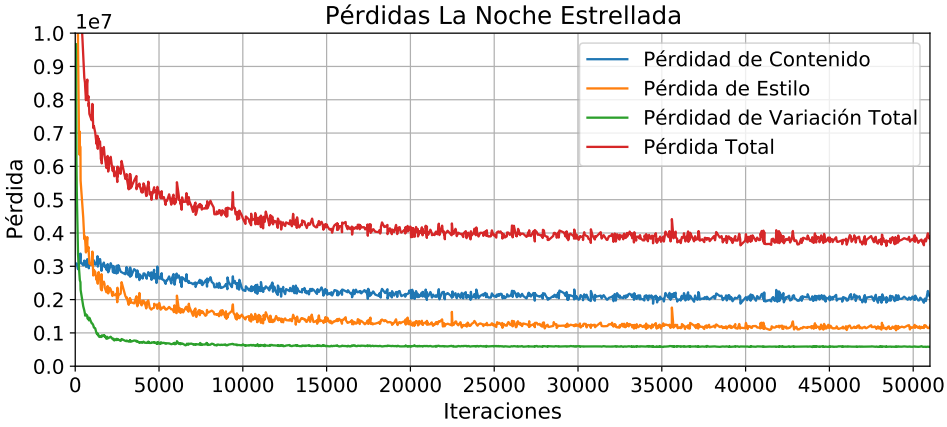


Figura 5.28: Gráfico de pérdidas en función del número de iteraciones de la red transformación de imágenes para el estilo “La Noche Estrellada”.

Autorretrato. Los valores de los hiperparámetros para el entrenamiento de la red correspondiente al estilo "Autorretrato" son: $\alpha = 10$, $\beta = 10^3$ y $\gamma = 2 \times 10^2$. La Figura 5.29 muestra la variación de las pérdidas en función del número de iteraciones.

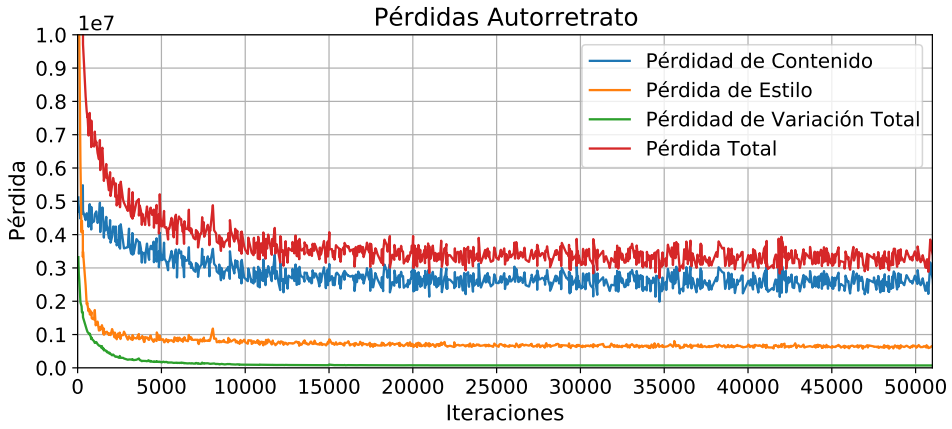


Figura 5.29: Gráfico de pérdidas en función del número de iteraciones de la red transformación de imágenes para el estilo "Autorretrato".

Dormitorio en Arlés. El entrenamiento de la red que transfiere el estilo "Dormitorio en Arlés" se lleva a cabo con los siguientes hiperparámetros: $\alpha = 7{,}5$, $\beta = 5 \times 10^2$ y $\gamma = 2 \times 10^2$. La Figura 5.30 muestra la variación de las pérdidas en función del número de iteraciones.

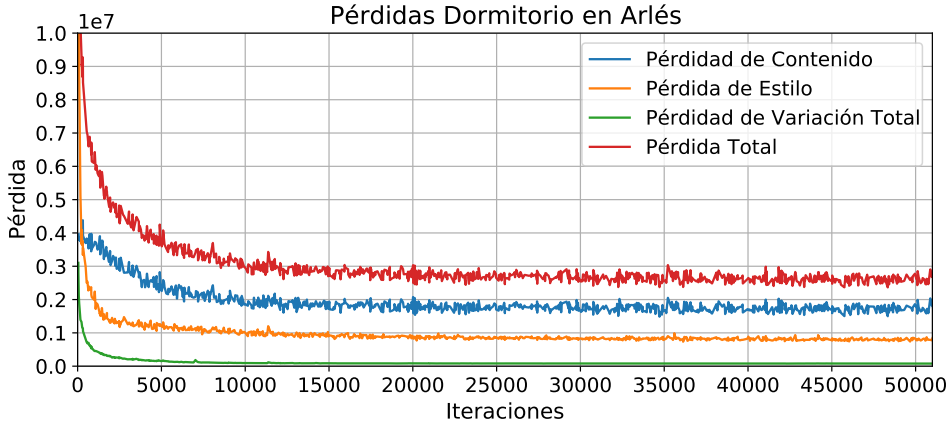


Figura 5.30: Gráfico de pérdidas en función del número de iteraciones de la red transformación de imágenes para el estilo "Dormitorio en Arlés".

5.4.6. Resultados Cuantitativos

La arquitectura de red utilizada en este módulo es analizada en el Apéndice B. En la sección B.1.1 se comparan los tiempos de cada arquitectura de red. En este caso, se utiliza la arquitectura de red de 1 bloque residual, debido a esto, el tiempo de inferencia de las redes es de $0{,}1969$ segundos para imágenes de entrada de 640×640 pixeles.

5.4.7. Resultados Cualitativos

En las Figuras 5.31 y 5.32 se muestran ejemplos cualitativos que comparan las estilizaciones producidas por las 3 redes entrenadas. Se utilizan imágenes entrada de paisajes, ciudades, personas, interiores, etc., para abarcar todo tipo de casos y analizar como son representados por cada red neuronal. Aunque los modelos están entrenados con imágenes de 256×256 pixeles, se pueden aplicar a imágenes de cualquier tamaño.

Las imágenes de las Figuras 5.31 y 5.32 se pueden ver en tamaño original en el siguiente link de Google Drive: https://drive.google.com/drive/folders/1C8OX-oSg1a4s3RwiEa9iazXJ0p6vqI95?usp=sharing.

La red de transformación de imágenes que transfiere el estilo de "Dormitorio en Arlés", se caracteriza por agregar texturas verticales a ciertas partes del fondo o el entorno de la imagen. Es interesante observar que ciertos elementos de la imagen, independiente del tamaño o su posición en la escena, no son texturizados con estas estructuras, como si la red entendiera la semántica de la imagen. Por ejemplo, las personas, no son alteradas con estas texturas. Por otro lado, esta red aumenta el contraste de la imagen, dando fuerza a los bordes, y modifica el color, volviendo la imagen de un tono amarillo.

Cuando se utiliza la pintura "Autorretrato" como estilo objetivo de la red, todos los resultados tienen una tonalidad turquesa. Además, agrega pequeñas pinceladas a la imagen enfatizando ciertos detalles de las imágenes. En general, esta red no es tan invasiva al alterar la estructura de la imagen, mantiene el contenido original alterando algunos detalles pequeños.

Finalmente, la red de transformación de imágenes, entrenada con la pintura "La Noche Estrellada", se caracteriza por presentar resultados con una pincelada prominente, propia del estilo aprendido, en toda la imagen sin importar la semántica. Respecto a los colores en la imagen resultante, la mayor parte de ella lleva tonalidades entre azul y negro. Además, resulta interesante observar que la red logra entender que el color amarillo va en secciones luminosas de la imagen, aunque en algunos casos puede equivocarse.



Figura 5.31: Primer conjunto de ejemplo cualitativos para las 3 redes de transformación de imágenes del módulo NST.



Figura 5.32: Segundo conjunto de ejemplo cualitativos para las 3 redes de transformación de imágenes del módulo NST.

5.4.8. Implementación en Video

Para realizar transferencia de estilo a videos, se implementa el código de la Figura 5.33.

```
# start video processing
print('start video processing')
vid_array = []
frame_id = 0
start_time = timer()
# Read until video is completed
while(cap.isOpened()):
    # Capture frame-by-frame
    ret, frame = cap.read()
    if ret == True:
        frame_id += 1
        print('process frame {}'.format(frame_id))
        # to rgb
        frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
        # get transformed image
        output_image = transformer.test(frame)
        # to bgr
        output_image = cv2.cvtColor(output_image, cv2.COLOR_RGB2BGR)
        # Convert to bytes
        output_image = output_image.astype(np.uint8)
        # add to list
        vid_array.append(output_image)
    # Break the loop
    else:
        break

# When everything done, release the video capture object
cap.release()
# report execution time
end_time = timer()
print('Execution time for a inference: %f sec' % (end_time - start_time))
```

Figura 5.33: Implementación del procesamiento de video en TF.

La transferencia de estilo en video inicia con la extracción de los frames de video por medio de la biblioteca OpenCV. Luego, se procesa cada frame extraido con la red de transformación de imágenes. Finalmente, se crea un video con los frames estilizados manteniendo los mismos FPS que el video original (ver Figura 5.34).

```
# save video
fourcc = cv2.VideoWriter_fourcc(*'mp4v')
out = cv2.VideoWriter(args.output, fourcc, fps, (output_image.shape[1], output_image.shape[0]))

for i in range(len(vid_array)):
    out.write(vid_array[i])
out.release()

# close session
sess.close()
```

Figura 5.34: Guardado de video estilizado.

El resultado aplicar el algoritmo NST a un video con cada una de las 3 redes entrenadas se puede ver en el siguiente link de Google Drive: https://drive.google.com/drive/folders/18PKvLe3s-WhmK-5DRL-SQRJRFjt8o9Cy?usp=sharing.

5.5. Módulo Controlador Central

Este módulo se preocupa de guiar el flujo de datos para que cada módulo funcione correctamente en el momento adecuado del ciclo general del sistema. Para llevar a cabo lo anterior, se implementa el diagrama de estados que se muestra en la Figura 5.35.

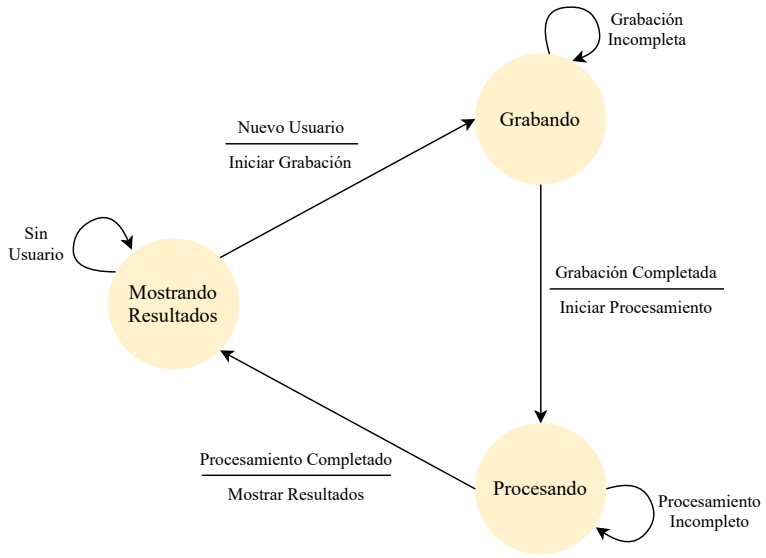


Figura 5.35: Diagrama de estados Van Gogh te pinta.

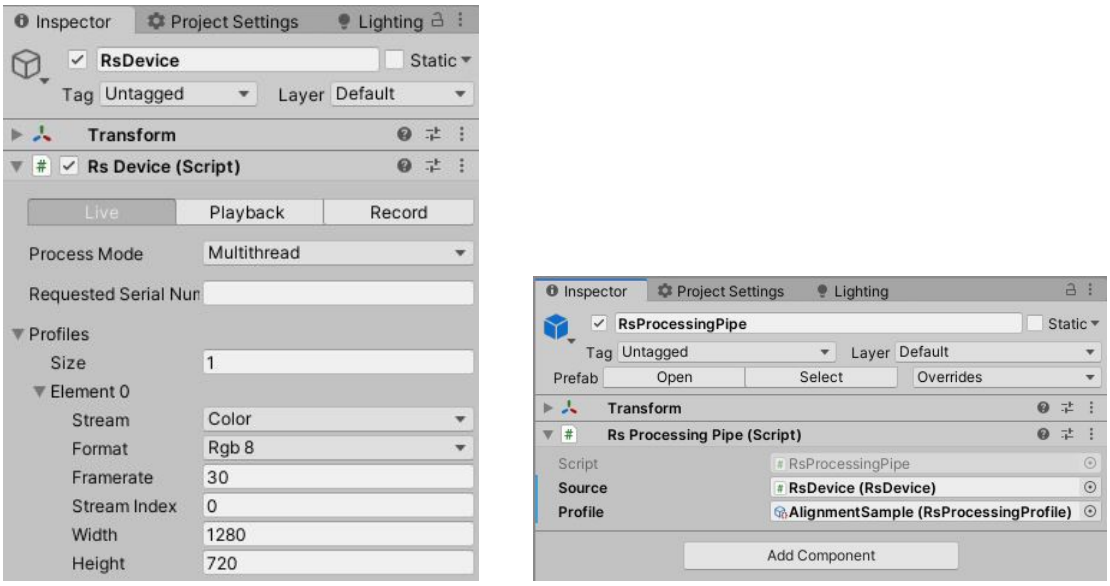
Van Gogh te pinta es un proceso cíclico. Cuando el sistema se encuentra en reposo o sin ser utilizado, se encuentra en el estado “Mostrando Resultados”. Este estado mantiene en exposición el último video estilizado y estará a la espera de una nueva interacción de los visitantes. Cuando un usuario interactúa con el sistema, el módulo UI le indica al Controlador Central que inicie la grabación, esta acción provoca el cambio de estado a “Grabando”. En un comienzo se inicia una cuenta regresiva de 3 segundos para que el usuario se prepare para ser grabado y luego, se comunica al módulo API Cámara que debe iniciar la grabación. El Controlador Central espera durante 8 segundos para que la grabación termine. Al finalizar, el video es guardado para dar paso al siguiente estado. Cuando el sistema se encuentra en “Procesando”, se da inicio al algoritmo NST mediante la comunicación con el script de Python. El sistema espera la respuesta de que el procesamiento ha terminado para pasar al estado “Mostrando Resultado”, lo que provoca el inicio de un nuevo ciclo de trabajo.

5.6. Módulo API Cámara

Este módulo se preocupa de recibir las órdenes del módulo Controlador Central en el estado "Grabando". En particular, otorga las funcionalidades de grabar un video y guardarlo en el momento en que el Control Central lo solicite.

El módulo API Cámara se desarrolla en Unity utilizando el SDK de la cámara Intel RealSense para acceder a los mapas de color, profundidad e infrarrojo sin la necesidad de ocupar OpenCV. Específicamente, se utiliza el *wrapper* de Unity para el SDK 2.0 de Intel RealSense que permite añadir secuencias de cámaras Intel RealSense a sus escenas. El objetivo de este *wrapper* es proporcionar un *prefab*[1] fácil de usar que permita la configuración del dispositivo utilizando el inspector Unity, sin tener que escribir una sola línea de código. Con esta herramienta, se puede obtener una transmisión en tiempo real de los mapas de color, profundidad e infrarrojo.

El SDK de Intel RealSense realiza gran parte del trabajo del módulo. De él se utilizan los *prefabs RsDevice* y *RsProcessingPipe* (ver Figura 5.36) que permiten configurar los parámetros de la cámara, procesar los datos y mostrar la imagen vista desde la cámara RealSense en la escena de Unity.



(a) RsDevice

(b) RsProcessingPipe

Figura 5.36: Vista desde el inspector de Unity de los objetos utilizados del SDK 2.0 de Intel RealSense.

[1] Objetos reutilizables en Unity. Para más detalle revisar el glosario.

Finalmente, se crea el objeto *RecordCamera* que permite grabar los frames de la cámara y guardarlos en formato de video. Desde el inspector de Unity se puede modificar el formato de video, la resolución de grabación, fps, etc (ver Figura 5.37).

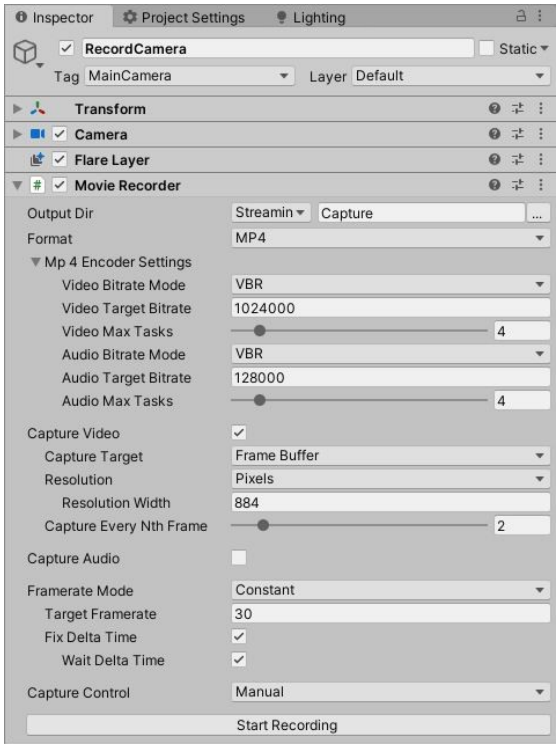
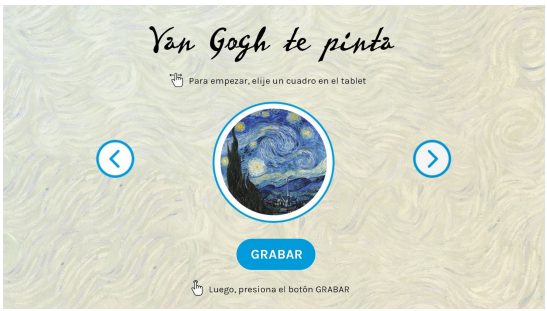


Figura 5.37: Objeto RecordCamera que permite grabar y guardar los videos de la cámara.

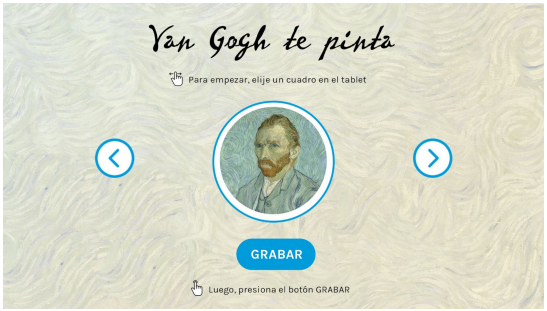
Notar que el módulo API Cámara es un conjunto de objetos en Unity que permiten la comunicación con el módulo Controlador Central para realizar las tareas requeridas por el sistema.

5.7. Módulo UI

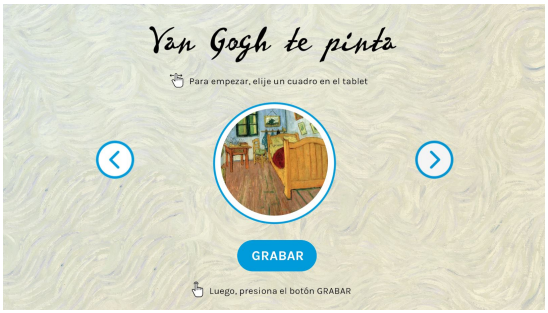
La UI de la instalación interactiva Van Gogh te pinta permite al usuario seleccionar una entre tres pintura de Vincent van Gogh: La Noche Estrellada, Autorretrato o Dormitorio en Arlés. Además, permite iniciar la grabación de video que posteriormente será estilizada con la pintura seleccionada. Este módulo se desarrolla en Unity y se genera una APK para ser instalada en la tablet. Las vistas de la aplicación se muestran en la Figura 5.38.



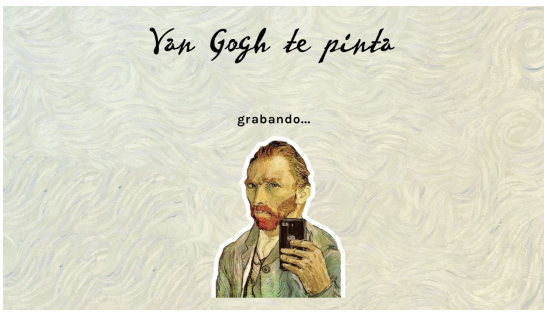
(a) Selección La Noche Estrellada.



(b) Selección Autorretrato.



(c) Selección Dormitorio en Arlés.



(d) Grabando video.

Figura 5.38: Vistas del módulo UI.

La aplicación es bastante sencilla, la misma interfaz muestra las instrucciones de lo que el usuario debe realizar. Primero, se debe escoger un cuadro presionando las flechas (Figura 5.38 (a), (b) y (c)). Luego, se inicia la grabación al presionar el botón "GRABAR". Esta última acción cambia el estado del sistema (módulo Controlador Central Figura 5.35) de "Mostrando Resultados" a "Grabando". Cuando el sistema se encuentra grabando al usuario, la aplicación muestra una escena que indica dicha acción que está realizando el sistema (ver Figura 5.38d).

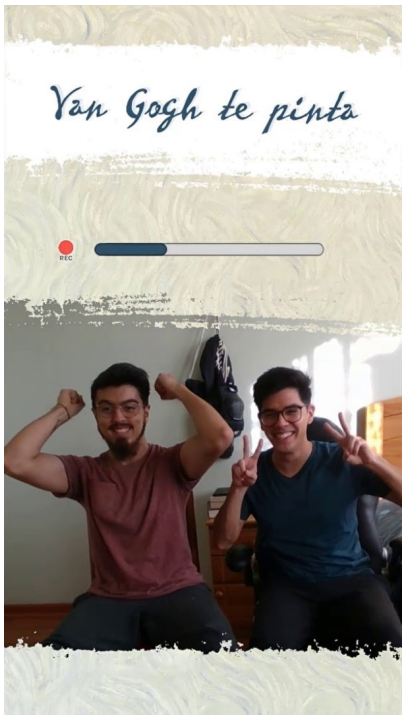
5.8. Módulo Visualizador

Este módulo es el encargado de mostrar el resultado de todo el proceso del sistema, es principalmente una interfaz de visualización que muestra distintas escenas dependiendo el estado del sistema. El módulo Visualizador es desarrollado en Unity y es parte del ejecutable que se genera para el mini-PC. Las vistas del módulo Visualizador se muestran en la Figura 5.39.

Cada estado del módulo Controlador Central (Figura 5.35) está asociado con una o más escenas de la interfaz de visualización. Cuando un usuario inicia la grabación el



(a) Cuenta regresiva



(b) Grabando



(c) Procesando



(d) Mostrando Resultados

Figura 5.39: Vistas del módulo Visualizador.

sistema se encuentra en el estado “Grabando”. En un inicio el visualizador muestra una cuenta regresiva de 3 segundo para que el usuario se prepare para ser grabado (Figura 5.39a). Luego, mientras el visitante es grabado se muestra la vista desde la cámara en la interfaz con un indicador de la grabación (Figura 5.39b).

Después de terminar la grabación, el sistema pasa al estado “Procesando”, y el visualizador muestra la escena de la Figura 5.39c, donde aparece Vincent Van Gogh al estilo de su pintura “Autorretrato”, con una flor de girasol en un lienzo como el indicador de carga. Además, en esta escena, mientras el sistema se encuentra procesando el video, se muestran datos interesantes de la vida del pintor en una pequeña sección de la escena llamada “Sabias que”. Los datos que se muestran son los siguientes:

- Nací en Holanda el 30 de Marzo de 1853.
- Comencé a pintar a los 27 años de edad. Antes de eso quise seguir los pasos de mi padre y convertirme en sacerdote.
- En toda mi vida, vendí solo una pintura.
- Pinté aproximadamente 900 pinturas en solo 10 años.
- Mi mejor amigo fue Paul Gauguin, otro famoso pintor.
- Pinté más de 30 autorretratos entre 1886 y 1889.
- Fui conocido por usar velas en el sombrero para poder pintar de noche.

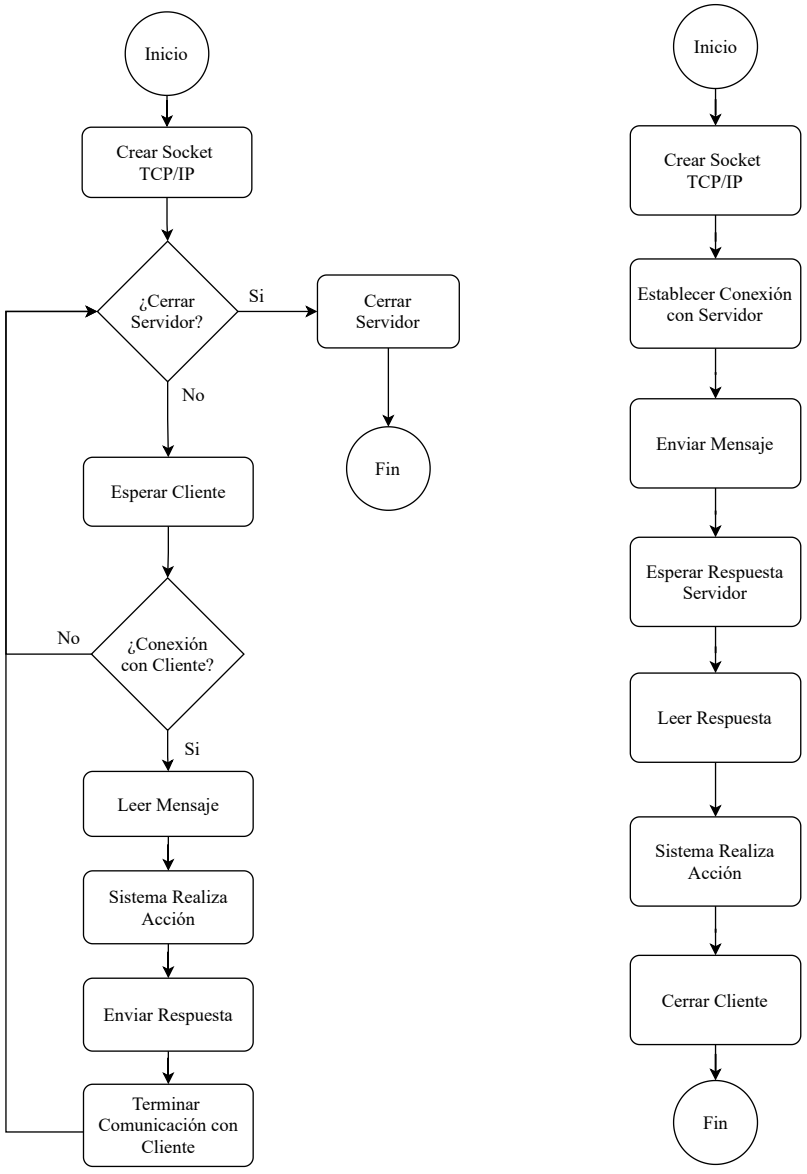
Finalmente, cuando se ha terminado de aplicar el algoritmo NST al video grabado, el sistema pasa al estado “Mostrando Resultados” y se muestra la escena de la Figura 5.39d, donde se expone el video estilizado en loop. Esta última escena queda en exposición hasta la próxima interacción de un usuario.

El visualizador, durante todos los estados, mantiene una música de fondo para acompañar la exposición. La música que se utiliza para este propósito es “Crosswire” de Blue Dot Sessions.

5.9. Integración y Comunicación entre Módulos

Para llevar a cabo la integración de los módulos es necesario comunicar los procesos del sistema. En este caso, se debe comunicar la tablet con el mini-PC y los programas

en Python y Unity que se encuentran ejecutando paralelamente en el mini-PC. Para realizar lo anterior, se desarrollan los módulos de comunicación descritos en el capitulo anterior: Módulos Socket Cliente-Servidor APK (sección 4.7.2) y Módulos Socket Cliente-Servidor NST (4.7.5). Estos módulos comprenden la misma estructura cliente-servidor mediante protocolo TCP/IP que se puede resumir en los diagramas de flujo de la Figura 5.40.



(a) Servidor (b) Cliente

Figura 5.40: Diagramas de flujo cliente y servidor.

Para ambas conexiones (tablet-computador y Python-Unity), el servidor se encontrará activo durante todo el tiempo que el sistema se encuentre prendido, a la espera de que un nuevo cliente desee establecer conexión con él. Por otro lado, el cliente es un hilo creado solo para comunicarse con el servidor respectivo, luego de la comunicación el cliente es destruido y se necesitará crear un nuevo cliente para establecer una nueva conexión con el servidor.

Para el caso de la comunicación tablet-computador, se tiene que el socket de la tablet es el cliente y el socket del mini-pc es el servidor, que se encontrará a la espera de un nuevo cliente mientras el sistema se encuentre en el estado "Mostrando Resultados", en otro caso, el servidor estará conectado a un cliente y no se podrán realizar otras conexiones con este servidor.

En cambio, para la comunicación Python-Unity, se tiene que el cliente es el programa en Unity y el servidor es el programa en Python. El programa de Python ejecuta el algoritmo NST y estará a la espera de que el módulo Controlador Central (cliente Unity) le indique el momento en que debe ejecutarse. Luego de ejecutarse, deberá esperar a que el Controlador Central le orden cuando operar nuevamente.

Por último, para que esta conexión de módulos se ejecute correctamente es necesario configurar correctamente las IPs y los puertos a utilizar. Para la comunicación Python-Unity, se utiliza la IP del localhost y el puerto asignado es el 10000. Para el caso tablet-computador, es necesario dejar fija la IP del mini-pc, en este caso se deja en 192.168.11.2 y el puerto en 10001. Con este enfoque se pueden dejar fijas las configuraciones mediante código y la comunicación se realizará automáticamente al iniciar el sistema.

En este punto, el sistema ya puede funcionar correctamente. Los 3 programas desarrollados (APK, aplicación Unity para mini-PC y programa en Python) se deben ejecutar al inicio para que todas las configuraciones sean cargadas y los servidores se encuentren a la espera de una nueva conexión. Luego, con la interacción por medio de la tablet se inicia el ciclo de trabajo de la instalación. El resultado del sistema funcionando se muestra en el siguiente link (solo se muestran las vistas del módulo UI y el módulo Visualizador): https://drive.google.com/file/d/1bzAgVdkcYkqBRybi1C9wUc9rOXBSqo_P/view?usp=sharing

Capítulo 6

Puesta en Marcha y Resultados

La etapa final del desarrollo de la instalación interactiva Van Gogh te pinta, consiste en el montaje del PMV en el espacio de instalación dispuesto por el museo Artequin para el desarrollo de este proyecto. Si bien la integración de los elementos se encuentra en correcto funcionamiento, aún falta efectuar la puesta en escena de la instalación Van Gogh te pinta, teniendo en consideración la usabilidad de los usuarios, la seguridad de los elementos y del público, el atractivo de la instalación, etc. En este capítulo se muestra el proceso de montaje y puesta en marcha de la instalación Van Gogh te pinta, detallando los distintos trabajos realizados para lograr el correcto funcionamiento de la exposición en el museo Artequin. Al final del capítulo, se presentan los resultados de algunos visitantes utilizando la instalación interactiva.

6.1. Espacio de instalación

El lugar de montaje de la instalación interactiva Van Gogh te pinta es un espacio de 230 cm de alto, 120 cm de ancho y 332 cm de profundidad, que es acondicionado para la instalación. En la Figura 6.1 se muestra una foto del espacio previo a la instalación y a los trabajos del museo Artequin de adaptar el lugar para la instalación.

El trabajo del museo Artequin para preparar el espacio de instalación consiste en pintar las paredes laterales del lugar de color negro, y cerrar la parte trasera con una pared blanca para evitar la entrada de luz al espacio. Esta última consideración es necesaria debido a que se cuenta con un proyector para la visualización de los resultados, y la pared blanca de fondo es utilizada como pantalla de proyección.



Figura 6.1: Espacio de instalación museo Artequin Viña del Mar.

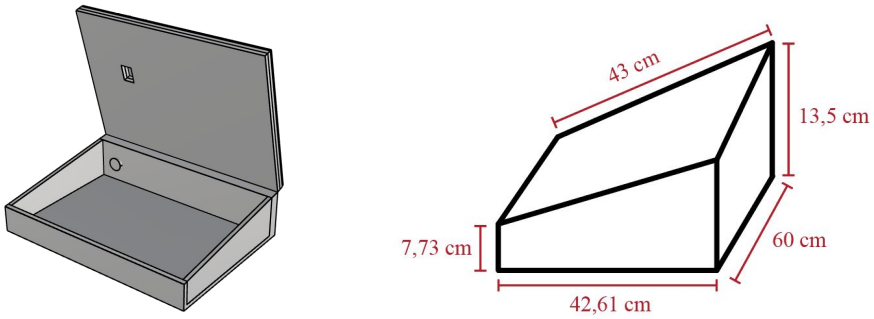
6.2. Diseño de Estructuras de Instalación

Para la instalación se deben diseñar dos estructuras para contener el hardware. Para la primera estructura, se decide construir una mesa de madera y acrílico que contenga todos los elementos de la instalación interactiva, menos la cámara y el proyector. Este mueble contenedor es lo suficientemente seguro para evitar el robo del mini-PC, tablet, o algún otro elemento dispuesto para la exposición. Por otro lado, la segunda estructura que se diseña es una pequeña caja que contiene la cámara RealSense. Este elemento cumple la función de proteger y permite adherir la cámara a la pared para captar de la mejor manera la escena del usuario, permitiendo a personas altas, niños pequeños, e incluso personas en situación de discapacidad, utilizar la instalación interactiva. El proyector se adhiere al techo mediante un soporte para el montaje.

Vale la pena mencionar que el diseño y la construcción de estos elementos fue tercerizado, debido a la falta de capacidades en el equipo para realizar este tipo de tareas. A continuación se detallan las dos elementos diseñados para el montaje de la instalación interactiva.

6.2.1. Mueble Contenedor de Hardware

La Figura 6.2 muestra el diseño 3D del mueble contenedor de hardware junto a sus medidas. Este mueble de madera y acrílico es el que contiene en su interior el mini-PC, el router, periféricos y sus conectores. Además, la parte superior del mueble es de acrílico, y tiene un espacio en el centro para que la tablet vaya encajada en él. Este mueble va empotrado al suelo y a la pared del espacio de instalación por medio de unas estructuras de fierro. Por seguridad, el mueble contiene dos cerraduras para que solo los administradores del museo lo puedan abrir.

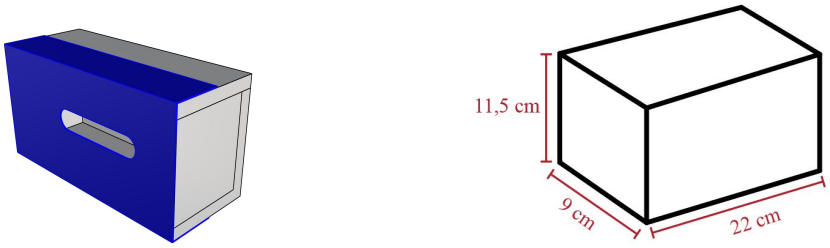


(a) Modelo 3D (b) Medidas

Figura 6.2: Modelo 3D del mueble contenedor de hardware y sus medidas.

6.2.2. Caja Protectora de Cámara

La Figura 6.3 muestra el diseño 3D de la caja contenedora de la cámara RealSense y sus medidas. Está hecha a la medida de la cámara y tiene un espacio frontal para que esta pueda grabar. Esta caja va fijada a la pared por medio de adhesivos de doble contacto para montaje.



(a) Modelo 3D (b) Medidas

Figura 6.3: Modelo 3D de la caja protectora de la RealSense y sus medidas.

6.3. Diseño de Elementos Gráficos

Para dar un buen aspecto visual a la instalación interactiva se diseñan gráficas para la parte superior del mueble contenedor que son posicionadas sobre el acrílico. La Figura 6.4 muestra la gráfica diseñada.



Figura 6.4: Gráfica para mueble.

Además de las gráficas para el mueble, se otras adicionales para situar en las paredes del espacio de instalación. Como el lugar se encuentra en un pequeño espacio del museo, el cual no está a la vista de los visitantes, el objetivo de estas gráficas es guiar a las a personas hacia la instalación interactiva. La Figura 6.5 muestra las gráficas del espacio interactivo.



(a) Gráfica pared lateral (b) Gráfica pared superior

Figura 6.5: Gráficas para situar en la pared y mostrar la posición de la instalación interactiva.

6.4. Montaje

Luego de que los elementos que habilitan la instalación se encuentran diseñados y construidos, y el espacio de instalación ha sido acondicionado, se procede a realizar el montaje de Van Gogh te pinta en el museo Artequin. La Figura 6.6 muestra el montaje del mueble contenedor de hardware y la caja protectora de la cámara en el espacio de instalación. Además, estas estructuras se muestran con los elementos gráficos instalados.



(a) Mueble y caja cámara (b) Gráfica pared lateral (c) Gráfica pared superior

Figura 6.6: Montaje del mueble contenedor de hardware y caja protectora de cámara en el espacio de instalación junto con las gráficas para mostrar la posición de la instalación interactiva.

Notar que el espacio de instalación ha sido cerrado (comparar con la Figura 6.1). Las paredes laterales son pintadas de color negro y la pared trasera de color blanco para favorecer la proyección de la instalación interactiva (ver Figura 6.6c).

La arquitectura del espacio se aprovecha para ocultar el proyector en el techo. Este se instala detrás de la pared superior que dice “Espacio interactivo” (Figura 6.6c), el cual permite tener un gran tamaño de proyección. Por otro lado, durante el montaje se utilizaron molduras de color negro para esconder los cables de conexión entre componentes. Las dos ideas anteriores permiten ocultar gran parte de los componentes del sistema, dejando a la vista del usuario solo los elementos estéticos de la instalación.

La Figura 6.6a muestra la cámara adherida a la pared por sobre el mueble. Esto se realiza mediante adhesivos para montaje, que soportan una gran cantidad de peso. Desde su instalación, la caja protectora no ha sufrido inconvenientes.

La Figura 6.7 muestra la vista completa del espacio de instalación de Van Gogh te pinta en el museo Artequin.



Figura 6.7: Vista panorámica del espacio de instalación de Van Gogh te pinta.

6.5. Puesta en Marcha en Museo Artequin

Luego del montaje, la instalación interactiva Van Gogh te pinta queda en funcionamiento en el museo Artequin de Viña del Mar de forma permanente. La Figura 6.8, muestra la vista de la instalación en funcionamiento.

Notar en la Figura 6.7 que el espacio de instalación es bastante oscuro y dificulta la grabación de video por la iluminación de la imagen recibida. Por esta razón, se instalan luces en la parte superior del espacio (ver Figura 6.8), ubicándolas de tal manera que permanezcan ocultas pero iluminen de buena manera la escena vista desde la cámara. Se utilizan luces direccionales porque es necesario mantener iluminada la sección del espacio que observa la cámara, pero se debe mantener oscura la parte donde se proyecta la visualización de los resultados.

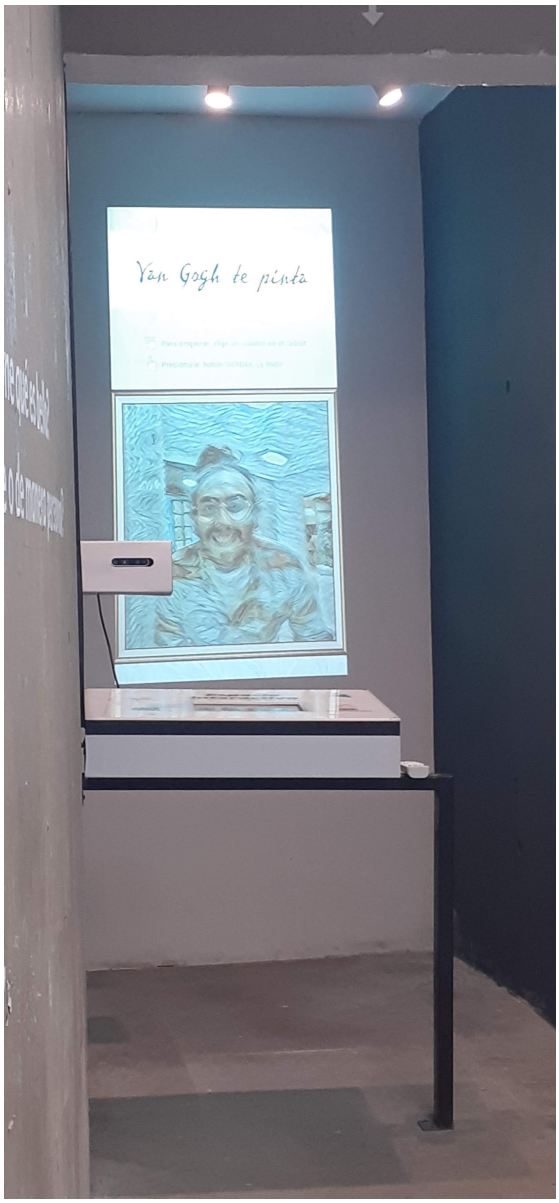


Figura 6.8: Instalación interactiva Van Gogh te pinta.

Además, es sorprendente observar que la mayoría de los componentes del sistema se encuentran en la mesa contenedora de hardware, esto es, mini-PC, router, periféricos, cables, alargador para la energía, etc. La buena elección de los componentes en el Capítulo 5 permiten un montaje sencillo y estético de la instalación interactiva Van Gogh te pinta.

Uno de los ajustes finales de la instalación es la configuración de la resolución de la grabación. Primero es necesario entender que la dimensión efectiva del video

en la proyección es bastante reducida. Esto queda claro en la Figura 6.9, que muestra la relación entre la dimensión total de proyección, la dimensión del visualizador y la dimensión del video. Debido a esto, se configura el sistema para grabar a una resolución de 460×432 pixeles. Además, para generar el efecto "boomerang" de Instagram, se disminuye la cantidad de fotogramas grabados por la cámara para que la reproducción de video se vea aproximadamente $2{,}5$ veces más rápida. Bajo estas condiciones, el tiempo total de procesamiento de una grabación de 8 segundos es de $11{,}3919$ segundos. Un periodo de tiempo aceptable para que el usuario logre leer los datos curiosos de la vida de Vincent van Gogh y no se aburra de esperar por el resultado.

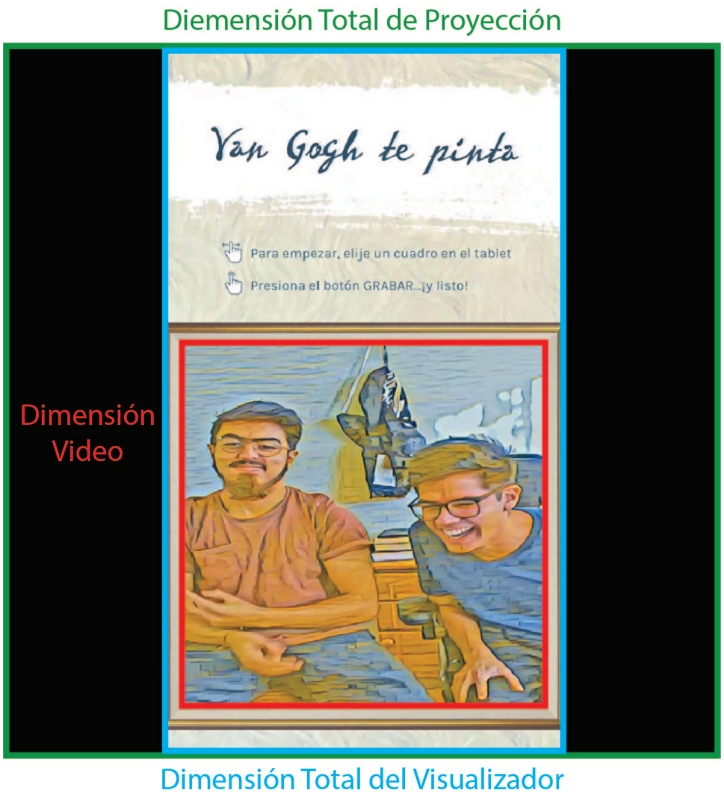
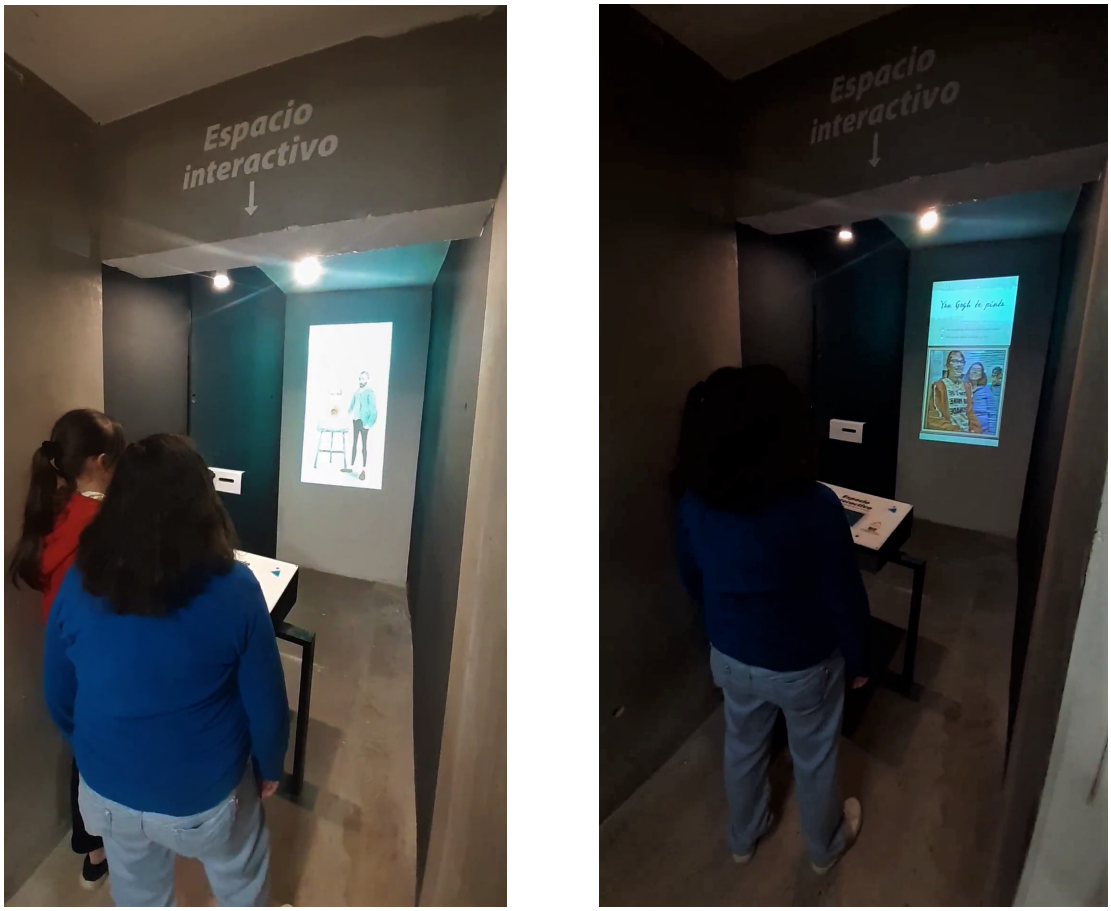


Figura 6.9: Dimensión efectiva del video en la proyección.

Finalmente, para que el sistema se mantenga operativo durante los horarios de visita del museo, se configura el sistema operativo del mini-PC para que se prenda y apague automáticamente en dichos horarios. Por otro lado, la tablet se mantiene siempre prendida y el proyector es apagado por el personal del museo. Además, se configuran los programas de Unity (Visualizador) y Python (módulo NST) para que inicien junto al sistema operativo. Todo lo anterior, vuelve al sistema casi completamente autónomo.

6.6. Resultados

Durante su funcionamiento, la instalación Van Gogh te pinta tiene buen recibimiento por parte del público. Los visitantes sienten la curiosidad y cierto miedo por interactuar con la instalación. La Figura 6.10 muestra fotos de dos niñas que visitan el museo Artequin e interactúan con la instalación. Estas imágenes son parte de un video grabado como análisis de los resultados del proyecto. El video completo puede ser visto en el siguiente link: https://drive.google.com/file/d/1NfsrY2emLPZA4Ily-8eeZPR7YkcLcQ_3/view?usp=sharing



(a) (b)

Figura 6.10: Instalación interactiva Van Gogh te pinta siendo utilizado por visitantes del museo Artequin.

La Figura 6.11 muestra fotos de algunos resultados obtenidos producto de la interacción de los visitantes con la instalación Van Gogh te pinta. Los videos completos se encuentran en el siguiente link: https://drive.google.com/drive/folders/1dXxgwdSNPv81glxTrUQT9TMprQPoYLa_?usp=sharing



(a) (b)

(c) (d)

Figura 6.11: Resultados de la interacción de los visitantes con la instalación interactiva Van Gogh te pinta.

Como término de desarrollo del proyecto, se crea un video promocional de la instalación interactiva Van Gogh te pinta, que tiene como objetivo mostrar el funcionamiento de la instalación con la finalidad de ser ofrecida a otras instituciones culturales. El video presenta un paso a paso corto y preciso de cómo utilizar la instalación interactiva con foco comercial. El video completo puede ser visto en el siguiente link de Google Drive: https://drive.google.com/file/d/1jlWAUgEqyTeERn4J7I4ELupmEFxbjW0m/view?usp=sharing

Capítulo 7

Conclusiones y Trabajos Futuros

Los resultados obtenidos en este proyecto de memoria cumplen el objetivo de introducir la inteligencia artificial como fenómeno artístico y cultural. La instalación interactiva de arte digital "Van Gogh te pinta" es montada en el Museo Artequin de Viña del Mar, apareciendo como una experiencia nueva e innovadora, ofreciendo un nuevo paradigma tecnológico a la experiencia del usuario dentro de las dependencias del museo.

Lamentablemente, la pandemia mundial producida por el virus COVID-19, ha privado al país de las libertades de reunir grandes cantidades de personas en lugares cerrados. En particular, el Museo Artequin ha cerrado sus puertas al público masivo teniendo que pausar sus actividades cotidianas teniendo que reinventarse en este nuevo paradigma de pandemia, donde todo se realiza remotamente. En este contexto, la instalación interactiva ha sido pausada a poco tiempo de iniciar su funcionamiento.

Sorprendentemente, a pesar de estar unas semanas en funcionamiento, la instalación interactiva tuvo un buen recibimiento por parte del público del Museo Artequin. En los resultados mostrados en la sección 6.6 del Capítulo 6, se aprecia como los niños y niñas, que tuvieron la posibilidad de interactuar con la instalación, disfrutaron de esta nueva experiencia tecnológica.

Debido al corto tiempo en exposición, no se pudo medir el interés de los visitantes sobre este nuevo tipo de arte. Tampoco se pudo medir el nivel de curiosidad o aprendizaje que produce este tipo de exposiciones artísticas. Por otro lado, el pronto cierre del museo postergó hasta nuevo aviso la inauguración de este espacio de exposición en colaboración con la UTFSM.

Independiente de los inconvenientes, respecto a los objetivos planteados en esta

memoria, se puede establecer que se ha cumplido lo siguiente:

- Se ha diseñado, optimizado e implementado satisfactoriamente el algoritmo de transferencia de estilo PSPM-MOB-NST basado en una red de transferencia de estilo. En base a esta arquitectura de red, se han entrenado 3 redes neuronales para transferir los estilos de 3 pinturas de Vincent van Gogh: La Noche Estrellada, Autorretrato y Dormitorio en Arlés.

- Se ha diseñado e implementado la UI y la interfaz de visualización que permiten a la instalación interactiva intercambiar información con los visitantes del museo. Por medio de la UI el usuario puede manejar el flujo de trabajo de la instalación y, por otro lado, la interfaz de visualización permite mostrar los resultados de video estilizados a los usuarios.

- Se ha desarrollado, por medio de la interfaz de visualización, una capa de contenido educativo para los visitantes. Por medio de datos curiosos de la vida del van Gogh se logra captar la atención y estimular la curiosidad de los usuarios.

- Se ha logrado comunicar los módulos del sistema satisfactoriamente por medio del protocolo TCP/IP bajo la arquitectura de cliente/servidor, permitiendo el correcto flujo de datos del sistema.

- Se ha realizado el montaje de la instalación interactiva Van Gogh te pinta en el espacio dispuesto por el Museo Artequin. Esto ha sido permitido por el diseño y construcción de las estructuras que contienen los componentes del sistema.

- Se ha construido la primera versión del PMV de la instalación interactiva Van Gogh te pinta siguiendo un desarrollo iterativo modular, cumpliendo con los respectivos requisitos del sistema y siendo validado técnicamente en el Museo Artequin Viña del Mar.

De este proyecto, se concluye que las innovaciones tecnológicas ejercen un papel importante en el buen desempeño de las organizaciones culturales. Además, los avances tecnológicos habilitan nuevas formas de expresar y vivir el arte permitiendo reducir las brechas de desigualdad presentes en el país, dando así, la posibilidad de que todos puedan participar del arte y la cultura.

7.1. Trabajos Futuros

Respecto a las posibles mejoras a implementar en la instalación interactiva Van Gogh te pinta, se pueden listar los siguientes trabajos futuros:

- Mejorar el método NST con las distintas extensiones estudiadas en el estado del arte (Capítulo 2). En este sentido, se busca mejorar la calidad visual del resultado considerando características como la profundidad, coherencia temporal del video, etc., sin perder la velocidad de inferencia.
- Extender la cantidad de estilos entrenando más redes neuronales que transfieran estilos de otras pinturas de Vincent van Gogh.
- Desarrollar nuevas escenas para transferir estilos de fotos en vez de video, con la finalidad de que el usuario reciba el resultado en su correo o red social y asi dar difusión al proyecto.
- Trabajar en la capa de contenido educativo de la instalación. Es posible adicionar más contenido educativo que permita al usuario entender de forma más clara el propósito de la exposición. En este sentido, se pueden agregar nuevas escenas donde el visitante pueda aprender más sobre tecnología y arte.
- Desarrollar un modelo de negocios y propuesta de valor que permitan comercializar la instalación interactiva.
- Continuar el desarrollo del PMV para lograr un producto validado comercial y técnicamente.

Apéndice A

Detalles de las Calificaciones de las Alternativas de Solución

A.1. Métodos de Transferencia de Estilo Neuronal

A.1.1. Velocidad

El análisis de la velocidad se basa en la investigación de Jing et al. [85] que realizan una revisión de los métodos NST, y uno de sus criterios de evaluación es la velocidad de estilización.

- **IOB-NST:** Este método se caracteriza por ser computacionalmente costoso, debido al procedimiento iterativo de optimización de imágenes en que se basa su algoritmo. Es el método más lento de los cuatro analizados.
- **PSPM-MOB-NST:** Este método es el más veloz de los cuatro analizados. Su estructura simple de una sola red de transformación de imagen le permite superar a los demás métodos.
- **MSPM-MOB-NST:** El método de múltiples estilos es un poco más lento que el algoritmo PSPM-MOB-NST porque en su arquitectura de red se agrega una red siamesa para extraer las características de estilo.
- **ASPM-MOB-NST:** El método de transferencia arbitraria es el método más lento de los 3 algoritmos MOB-NST porque debe procesar tanto el contenido como el

estilo en la primera mitad de la red. Además, la capa AdaIN, que recibe como entrada el contenido y el estilo, realiza cálculos estadísticos también que ralentizan el proceso.

A.1.2. Flexibilidad

El análisis de la flexibilidad también se basa en la investigación de Jing et al. [85]. Como se vio en el Capítulo 2 existen extensiones a los métodos base de NST, las cuales se pueden implementar en la mayoría de los métodos que se analizan, pero existen pequeñas limitaciones dependiendo del método.

- **IOB-NST:** Este algoritmo iterativo tiene la facultad de estilizar perfectamente cualquiera de los estilos que se tomen como entrada. Se puede extender para controlar el tamaño del trazo del pincel, considerar la profundidad de las imágenes, etc.

- **PSPM-MOB-NST:** Este método, que aprende un estilo por modelo, brinda la posibilidad de ajustar cada red cuidadosamente a cada estilo que se desee. En este sentido, la única limitación es el costo en tiempo de entrenar cada red, pero con el beneficio de obtener resultados estilizados mucho mejores que con los demás métodos MOB-NST. También es posible extender su arquitectura sin perder velocidad.

- **MSPM-MOB-NST y ASPM-MOB-NST:** Si bien estos métodos son posibles de extender, tiene la problemática de que al aprender varios estilos en un modelo, no es posible ajustar cada uno de ellos por separados. No es posible ajustar, en el entrenamiento de la red, los parámetros de la arquitectura para cada estilo, volviéndose dependiendo de la capacidad de generalización de la red para obtener buenos resultados estilizados para todos los estilos. Por otro lado, en caso de no existir tanto tiempo para entrenar tantos modelos, estos suelen ser una mejor opción que el método PSPM-MOB-NST.

A.2. Frameworks para Deep Learning

A.2.1. Bibliotecas de modelos pre-entrenados

- **CNTK:** Dispone de un repositorio que contiene modelos pre-entrenados que pueden usarse como DNN o como base para transferencia de aprendizaje [125]. Algunos de los modelos están entrenados desde cero por CNTK, y algunos otros se convierten de otros conjuntos de herramientas como Caffe. Los modelos están categorizados por sus dominios de aplicación.

- **Tensorflow:** Posee una gran cantidad de modelos pre-entrenados oficiales que están bien mantenidos, probados y actualizados con la última API estable de TensorFlow. Además, posee modelos de investigación, que estan completamente soportados por investigadores, los cuales brindan apoyo y dan solución a bugs que se puedan ir presentando en sus modelos [126]. También posee *TensorFlow Hub*, una biblioteca para la publicación, descubrimiento y consumo de partes reutilizables de modelos de aprendizaje automático [127]. Finalmente, recordar que **Keras**, una API de alto nivel que funciona con TensorFlow, también ofrece una buena cantidad de modelos pre-entrenados [128].

- **Caffe:** Dispone de *Model Zoo*, un gran repositorio de modelos pre-entrenados creado por investigadores e ingenieros. Estos modelos ya estan implementados y solo se deben descargar para ser utilizados [129].

- **Torch7:** La última versión de **Torch**, permite el uso de *Load Caffe*, que permite cargar todos los modelos pre-entrenados disponibles en Caffe.

- **PyTorch:** Contiene *Pytorch Hub*, un repositorio de modelos pre-entrenados diseñado para facilitar la reproducibilidad de la investigación. También posee *torchvision* [130], un paquete que consta de conjuntos de datos populares, arquitecturas de modelos y transformaciones de imagen comunes para visión por computador. En él hay un subpaquete con definiciones de modelos pre-entrenados para abordar diferentes tareas, que incluyen: clasificación de imágenes, segmentación semántica de pixeles, detección de objetos, segmentación de instancias y detección de puntos clave de personas [131].

A.2.2. Popularidad

Se estudia la popularidad de los frameworks en base a la cantidad de stars (Figura A.1) y de contribuyentes (Figura A.2). Esta información es obtenida directamente de la página de los repositorios de GitHub de cada framework [103, 132, 133, 101, 134].

La asignación de puntaje esta dada por el promedio de los porcentajes de stars y contribuyentes de cada framework. El porcentaje de stars del framework se calcula dividendo la cantidad de stars en total de stars de los cinco frameworks. De la misma manera se calcula el porcentaje de los contribuyentes por framework. Finalmente, se promedian los porcentajes y se obtiene el puntaje de la Tabla 3.2.

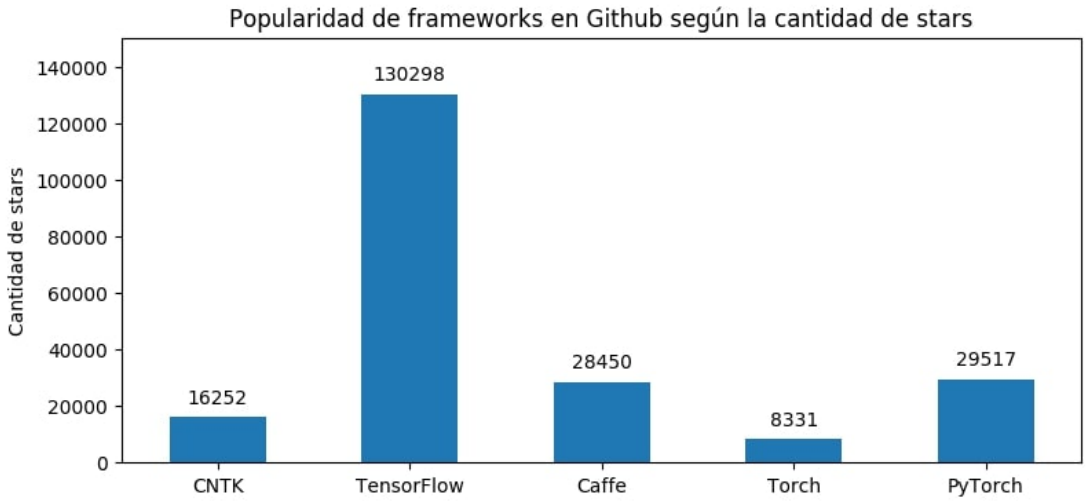


Figura A.1: Cantidad de stars en GitHub de cada framework.

A.2.3. Actividad en Stack Overflow

Stack Overflow tiene una sección de "Tags" en su página (`https://stackoverflow.com/tags`), en ella es posible buscar cualquier etiqueta y encontrar una sección dedicada a ella con todas las preguntas relacionadas.

En la Figura A.3 se muestra un gráfico con la cantidad de preguntas con la etiqueta de cada framework estudiado.

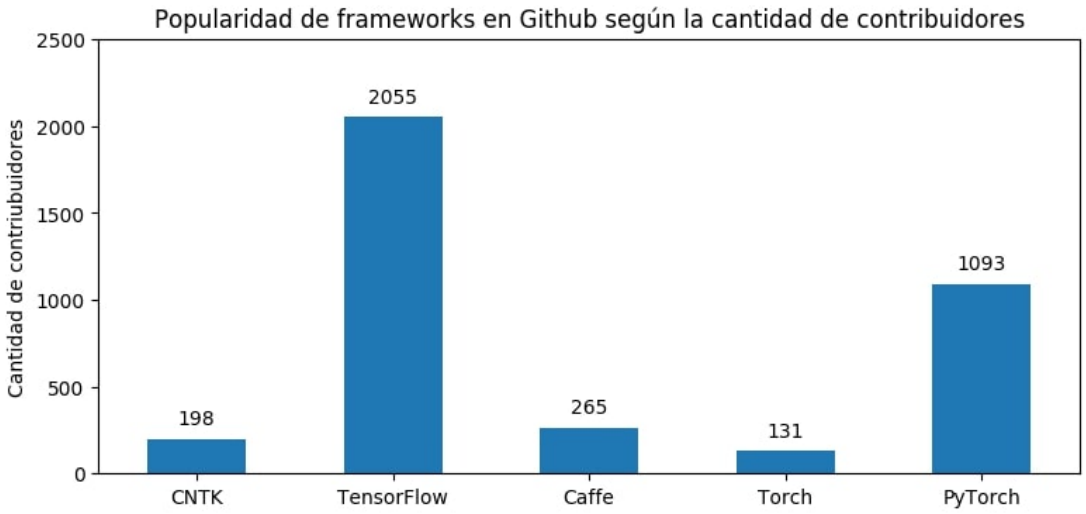


Figura A.2: Cantidad de contribuyentes en GitHub de cada framework.

A.2.4. Documentación

Un rápido estudio de las documentaciones muestra que **TensorFlow** [93], **CNTK** [135] y **PyTorch** [104], poseen muy buenas documentaciones, son fáciles de estudiar y poseen guías detalladas que permiten ir avanzando poco a poco en la creación de modelos de deep learning. Además, poseen links que redireccionan a las herramientas adicionales sin necesidad de navegar exhaustivamente en su repositorio de GitHub para encontrarlas.

Por el contrario, **Torch** [98] y **Caffe** [91] poseen una documentación muy escasa, poco clara y desordenada. Torch tiene algunos ejemplos y tutoriales, pero nada muy detallado. Caffe tiene una página web muy extraña, complicada de navegar, es más fácil estudiar su documentación mediante GitHub. Ambos poseen muy pocos ejemplos de implementación, lo que produce complicaciones para comenzar a programar en estos frameworks.

A.2.5. Soporte de Técnicas de Paralelismo

La Tabla A.1 resume el soporte de CUDA, OpenCL y OpenMP en los frameworks para deep learning.

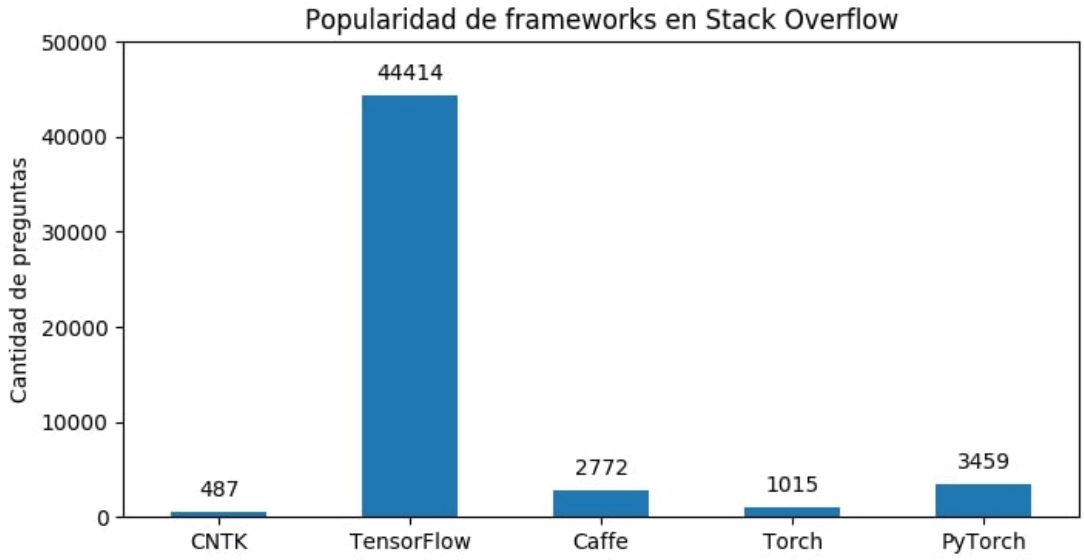


Figura A.3: Cantidad de preguntas en Stack Overflow que tienen como etiqueta cada framework.

Framework	CUDA	OpenCL	OpenMP
CNTK	✓[103]	✗	✓[136]
TensorFlow	✓[132]	✓[137]	✗
Caffe	✓[133]	✓[138]	✓[133]
Torch	✓[139]	✓[139]	✓[139]
PyTorch	✓[134]	✗	✓[134]

Tabla A.1: Soporte de técnicas de paralelismo en frameworks para deep learning.

A.2.6. Herramientas de Visualización

- **TensorFlow:** En este framework los cálculos que se utilizan (como el entrenamiento de una red neuronal profunda) pueden ser complejos y confusos. Para facilitar la comprensión, la depuración y la optimización de los programas, TensorFlow incluye un conjunto de herramientas de visualización llamadas **TensorBoard** [140], que se puede usar para visualizar sus grafos, trazar métricas cuantitativas sobre la ejecución de su grafos y mostrar datos adicionales como las imágenes que pasan a través de él.

- **CNTK:** Se proporciona una forma sencilla y simple de visualizar el grafo subyacente de un modelo utilizando **Graphviz** [141], un software de visualización de código abierto. También tiene compatibilidad con la herramienta de visualización

TensorBoard, la misma que utiliza TensorFlow.

- La opción de visualización para **Caffe** es **Netscope** [142], una herramienta web para visualizar arquitecturas de redes neuronales (o técnicamente, cualquier grafo acíclico dirigido). También es posible visualizar las arquitecturas de redes neuronales con Nvidia **DIGITS** [87].
- **Torch:** No posee una herramienta de visualización propia o externa.
- **PyTorch:** Se ha desarrollado una herramienta de visualización llamada **PyTorch-Viz** [143], la cual permite una visualización de grafos muy sencilla y no tiene más funcionalidades.

Apéndice B

Optimización Red de Transformación de Imágenes

B.1. Modificación de la Arquitectura de Red

Dado que se necesita que la transferencia de estilo en video ocurra lo más rápido posible, se desea que el tiempo necesario para realizar la transferencia de estilo a cada frame sea lo más bajo posible. Como cada imagen tiene que ser procesada por la red de transferencia de estilo, conviene hacer que el modelo sea lo más pequeño posible y, al mismo tiempo, conservar la mayor parte de la función f_W, así se logra reducir el tiempo de inferencia manteniendo la calidad del resultado estilizado. La arquitectura de red utilizada por Johnson et al. [52] es la que se utiliza generalmente para realizar transferencia de estilo con un bajo tiempo de inferencia. En esta sección se muestran las modificaciones realizadas a este modelo para llegar a la arquitectura de red utilizada en la instalación interactiva Van Gogh te pinta.

La red de transferencia de estilo de Johnson et al. [52] se define en el material suplementario de su articulo científico [144] y se expone detalladamente en la Tabla B.1. La arquitectura consiste en una primera etapa de downsampling, con dos convoluciones de stride igual a 2 para reducir la entrada, seguida de varios bloques residuales y luego dos capas de convolución transpuesta con stride igual a 2 para mantener la dimensión en la salida respecto de la entrada. Aunque la entrada y la salida tienen el mismo tamaño, existen varios beneficios para las redes que reducen la resolución y luego la aumentan.

El primero es computacional. Realizando un análisis superficial, una convolución de C filtros de 3×3 para una entrada de tamaño $C \times H \times W$ requiere $9HWC^2$

Capa	Tamaño de activación
Entrada	$3 \times 256 \times 256$
$32 \times 9 \times 9$ conv, stride 1	$32 \times 256 \times 256$
$64 \times 3 \times 3$ conv sep, stride 2	$64 \times 128 \times 128$
$128 \times 3 \times 3$ conv sep, stride 2	$128 \times 64 \times 64$
Bloque residual, 128 filtros	$128 \times 64 \times 64$
Bloque residual, 128 filtros	$128 \times 64 \times 64$
Bloque residual, 128 filtros	$128 \times 64 \times 64$
Bloque residual, 128 filtros	$128 \times 64 \times 64$
Bloque residual, 128 filtros	$128 \times 64 \times 64$
$64 \times 3 \times 3$ conv, stride $1/2$	$64 \times 128 \times 128$
$32 \times 3 \times 3$ conv, stride $1/2$	$32 \times 256 \times 256$
$3 \times 9 \times 9$ conv, stride 1	$3 \times 256 \times 256$

Tabla B.1: Arquitectura de red utilizada por [52] para transferencia de estilo.

multiplicaciones, que es el mismo costo que requiere una convolución de DC filtros de 3×3 para una entrada de tamaño $DC \times H/D \times W/D$. Esto quiere decir que, cuando se usa una etapa de downsampling, se puede utilizar una una red más grande por el mismo costo computacional.

El segundo beneficio tiene que ver con el tamaño efectivo del campo receptivo. La transferencia de estilo de alta calidad requiere cambiar grandes partes de la imagen de manera coherente; por lo tanto, es ventajoso que cada pixel de la salida tenga un gran campo receptivo efectivo en la entrada. Sin downsampling, cada capa convolucional de 3×3 adicional aumenta el tamaño del campo receptivo efectivo en 2. Por otro lado, si se reduce la resolución en un factor de D, cada convolución de 3×3 aumentar el tamaño de campo receptivo efectivo en $2D$, dando mayores campos receptivos efectivos con el mismo número de capas.

Respecto a las conexiones residuales, estas facilitan que la red aprenda la función identidad, esta es una propiedad atractiva para las redes de transformación de imágenes, ya que en la mayoría de los casos la imagen de salida debe compartir ciertas estructuras con la imagen de entrada.

Tomando en consideración lo anterior, el enfoque de la optimización está en la eliminación de ciertas capas residuales para reducir el tamaño de la red. Si bien esto puede

afectar el tamaño del campo receptivo de las capas y el aprendizaje de las características de estilo y contenido, se comprueban empíricamente los efectos adversos de realizar dichos cambios en la red, comparando los resultados cualitativos y cuantitativos para cada nuevo modelo. En las siguientes secciones se muestran los análisis cualitativos y cuantitativos de quitar bloques residuales a la arquitectura de red original; se toma como inicio una arquitectura de red con 5 bloques residuales y se quitan uno a uno hasta llegar a una arquitectura de red con 0 bloques residuales.

B.1.1. Análisis Cuantitativo

Para la comparación de las diferentes arquitecturas de red, se utiliza el estilo "Dormitorio en Arlés" y los siguientes hiperparámetros: $\alpha = 7{,}5$, $\beta = 5 \times 10^2$ y $\gamma = 2 \times 10^2$. Para las arquitecturas con un número de bloques residuales entre 0 y 3, se utiliza un tamaño de batch de 32 imágenes, y para las arquitecturas con 4 y 5 bloques, se usa 16, esto se debe a la memoria del sistema donde se entrena. Además, se entrenan en el conjunto de datos Microsoft COCO [123] redimensionando cada una de las 80.000 imágenes de entrenamiento a un tamaño de 256×256. Se utiliza una tasa de aprendizaje de 10^{-3} en 20 épocas sobre los datos de entrenamiento.

Para calcular los tiempos de inferencia, se toma la duración total de procesar un video de una resolución de 640×640 con una cantidad total de 420 frames. Luego, el tiempo de inferencia para un frame será el tiempo total dividido la cantidad de imágenes procesadas. En la Tabla B.2 se muestran los tiempos obtenidos para cada arquitectura.

Bloques Residuales	Tiempo Total [s]	Tiempo por Frame [s]	Optimización
0	77.7172	0.18504	31.44 %
1	82.6866	0.1969	27.047 %
2	89.9977	0.2143	20.6 %
3	97.3936	0.2319	14.079 %
4	105.2941	0.2507	7.11 %
5	113.3427	0.2699	-

Tabla B.2: Comparación de tiempos para redes de transformación con distinto número de bloques residuales.

De los tiempo obtenidos se tiene que la red sin bloques residuales es la más rápida, y la que tiene 5 bloques es la más lenta. Esto es de esperar, la arquitectura de red

neuronal más pequeña realiza menos cálculos al momento de efectuar una inferencia y, por lo tanto, demora menos tiempo en ejecutar dicho proceso. En la siguiente sección, se analizan los resultados cualitativos de cada una de las arquitecturas de red.

B.1.2. Análisis Cualitativo

Como se mencionó anteriormente, la red de transformación de imágenes consiste en una primera etapa de downsampling, seguida de varios bloques residuales y, finalmente, una etapa de upsampling. La primera etapa de reducción de tamaño permite obtener una representación comprimida de la imagen de entrada, también es conocida como etapa de codificación. Por el contrario, la etapa de upsampling devuelve las representaciones a las dimensiones de la imagen de entrada; también es conocida como decodificación. La parte intermedia de la red, es decir, los bloques residuales, se pueden considerar como una pequeña red ResNet [145], que procesan la imagen de entrada en un espacio de características comprimido (debido al downsampling).

Las modificaciones que se realizan para optimizar el modelo de red afectan la parte intermedia de la arquitectura, por lo tanto, se esperaría que la calidad del resultado disminuyera. Esto se debe a que los campos receptivos efectivos disminuyen su tamaño y la complejidad de la red también disminuye, pudiendo dificultar el aprendizaje de las características de estilo y contenido en la etapa de entrenamiento de la red.

En la Figura B.2 se muestran resultados estilizados para distintas imágenes de entrada utilizando las mismas redes de la Tabla B.2. Estas redes son entrenadas con el estilo de la pintura “Dormitorio en Arlés” (Figura B.1)



Figura B.1: Estilo utilizado para comparar cualitativamente el efecto de modificar la cantidad de bloques residuales en la arquitectura de red.

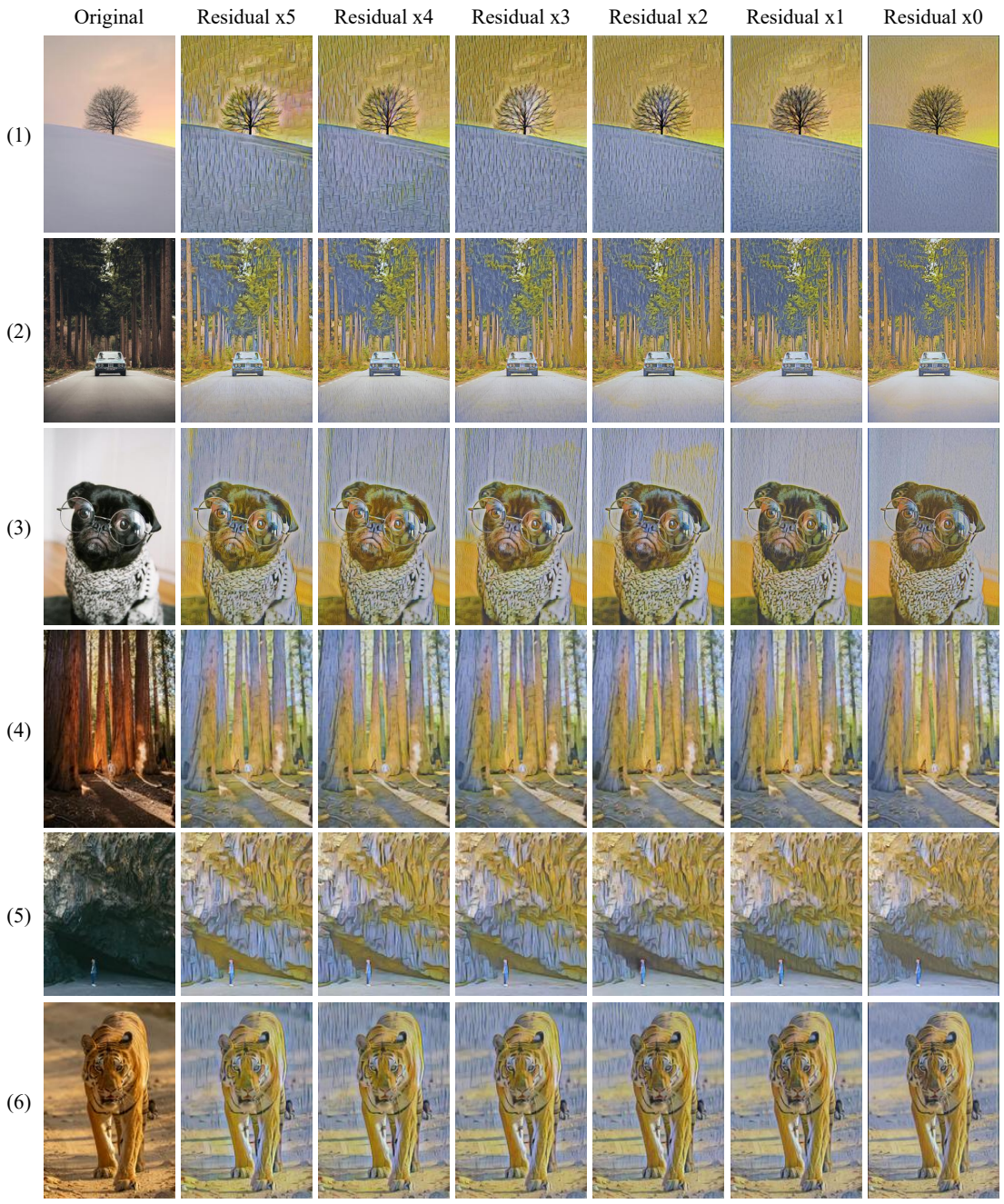


Figura B.2: Resultados cualitativos para las distintas redes de transferencia de estilo.

Los resultados de la Figura B.2 muestra que el hecho de reducir la cantidad de bloques residuales afecta algunas características de la imagen resultante. El efecto más claro sobre la imagen resultante es la disminución del tamaño de lineas verticales que se observan sobre la imagen. La primera fila de la Figura B.2, muestra como se reduce el grosor y el largo de estas franjas sobre la imagen, que terminan casi desapareciendo

cuando la arquitectura de red no posee bloques residuales. Este mismo efecto se puede ver en el fondo de las filas 3 y 6. Por otro lado, la prominencia de los bordes también se ve afectado (ver Figura B.2 fila 5). Finalmente, es posible ver que el color prácticamente se mantiene en todos los resultados, pero cuando la arquitectura no contiene bloques residuales, este se vuelve más opaco.

La instalación interactiva funciona con los visitantes del museo, por lo tanto, se realiza un análisis cualitativo de las redes de transformación considerando imágenes de personas como entrada. Además, con el propósito de mostrar en más detalle el efecto de los bloques residuales en la red de transformación de imágenes, el análisis se realiza para las arquitecturas de red con 5, 1 y 0 bloques residuales. Lo anterior se debe a que son los extremos de los análisis, la arquitectura con 5 bloques residuales es la que presenta un mejor resultado estilizado, y las arquitecturas con 1 bloque o ninguno, son las más rápidas.



(a) Original

(b) 5 bloques residuales

(c) 1 bloque residual

(d) Sin bloques residuales

Figura B.3: Resultados cualitativos para distintas redes de transferencia de estilo con imágenes de entrada de personas.



(a) Original

(b) 5 bloques residuales

(c) 1 bloque residual

(d) Sin bloques residuales

Figura B.4: Resultados cualitativos para distintas redes de transferencia de estilo con imágenes de entrada de personas.

Es posible notar que la red neuronal de transformación de imágenes, en este estilo en particular, realza el contraste y da fuerza a los bordes de la imagen. En particular, afecta en gran manera los detalles de la ropa de las personas en la escena. Se aprecia que la red con 5 bloques residuales es capaz de resaltar detalles muy específicos, por ejemplo, pequeños dobleces en los pantalones o chaquetas, y agrega tonos amarillos a toda la imagen, independiente de la semántica. Las Figuras B.3b y B.4b, muestran como la red agrega tonos amarillentos a secciones de la imagen que son más oscuras, como en las camisetas negras y pantalones oscuros. Por otro lado, la red con 1 bloque residual, logra resaltar algunos detalles y sus resultados se acercan a los que se obtienen con la red de 5 bloques residuales. Se puede mencionar que esta red no agrega excesivos tonos amarillentos a la imagen, y las lineas verticales de textura no son tan prominentes como en la red original. Finalmente, la red sin bloques residuales pierde bastantes detalles,

dejando planas ciertas partes de la imagen en cuanto a color y detalle. Y en el entorno de la imagen, las estructuras delgadas parecen ser ruido en la imagen y no la adición de un estilo de pintura.

El entorno de la escena deja en claro que la presencia de bloques residuales permite agregar estructuras gruesas y largas, con una mezcla de colores prominentes. En constaste, la ausencia de bloques residuales causa que las estructuras sean pequeñas y delgadas, acompañadas de colores con baja saturación.

B.1.3. Resolución de la Arquitectura de Red

De acuerdo a los análisis cualitativos y cuantitativos de las secciones anteriores, se concluye que la arquitectura de red con 1 bloque residual es la más adecuada para ser utilizada en la instalación interactiva. Esta arquitectura reduce en un $27{,}047\,\%$ el tiempo de inferencia a la vez que mantiene buenos resultados de estilización. De las 6 arquitecturas expuestas, esta es la que posee la mejor relación velocidad-calidad en los análisis de resultados.

B.2. Optimización de la Operación de Convolución

Las redes neuronales convolucionales son modelos computacionales complejos. Cuanto más profundo sea el modelo, mayor será la complejidad. Debido a esta desafortunada propiedad, no es trivial usar estos modelos para propósitos de tiempo real. En el modelo de Xception [146], Chollet muestra el funcionamiento de la convolución separable en profundidad (*depthwise separable convolution*) en arquitecturas de aprendizaje profundo. El uso de estas convoluciones resulta en una enorme reducción en el número de parámetros pero con un rendimiento de modelo similar, lo que ayuda a acelerar la velocidad de una operación de convolución. Esto resultó ser uno de los factores cruciales para conseguir redes neuronales convolucionales modernas eficientes e implementarlas en dispositivos periféricos de baja computación en tiempo real.

La segunda opción de optimización que se evalúa para el modelo NST de Van Gogh te pinta, es la sustitución de las convoluciones estándar por convoluciones separables en profundidad. A continuación, se detalla teóricamente la razón de dicha optimización.

B.2.1. Convoluciones Separables en Profundidad

Para propósitos de comparación se considera el siguiente caso de convolución estándar. Sea una imagen de entrada de $7 \times 7 \times 3$ y un filtro de $3 \times 3 \times 3$, la operación de convolución sin padding y un stride de 1 resulta en: $(7 \times 7 \times 3) * (3 \times 3 \times 3) = 5 \times 5 \times 1$ (ver Figura B.5).

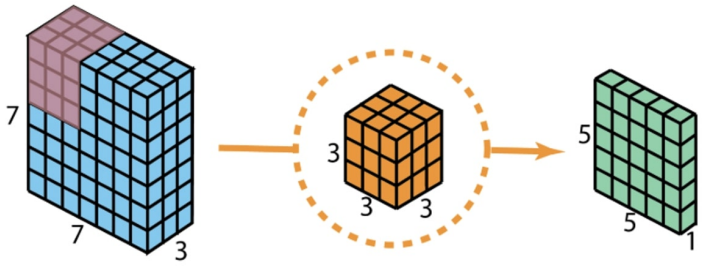


Figura B.5: Convolución 2D estándar con 1 filtro.

Las capas de convolución generalmente operan con más de 1 filtro. Entonces, se considera el mismo caso anterior pero con 128 filtros. La operación de convolución es: $(7 \times 7 \times 3) * (3 \times 3 \times 3 \times 128) = 5 \times 5 \times 128$ (ver Figura B.6).

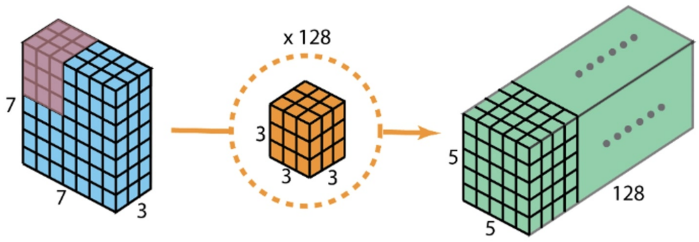


Figura B.6: Convolución 2D estándar con 128 filtros.

Una convolución separable en profundidad separa el proceso de una convolución estándar en 2 partes: una convolución en profundidad (*depthwise convolution*) y una convolución puntual (*pointwise convolution*). A continuación, se explican ambos pasos utilizando el mismo caso anterior.

Primera Etapa - Convolución en Profundidad. En esta primera parte, se aplica un filtro a cada canal de la imagen de entrada. Para una imagen de entrada de $7 \times 7 \times 3$ se tienen 3 filtros de $3 \times 3 \times 1$. Luego, se divide la imagen en tres canales diferentes para aplicar la convolución entre el filtro y el canal de correspondiente de la imagen.

Finalmente, se apilan los resultados para obtener una imagen de $5 \times 5 \times 3$. Este proceso se muestra en la Figura B.7.

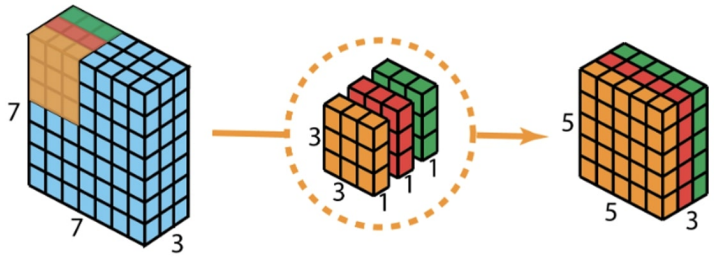


Figura B.7: Primera etapa de la convolución separable en profundidad.

Segunda Etapa - Convolución Puntual. La convolución puntual se llama así porque usa un filtro de 1×1, o un filtro que itera a través de cada punto. Este filtro tiene una profundidad igual a la imagen de entrada, en este caso, de profundidad 3. Por lo tanto, se itera el filtro de $1 \times 1 \times 3$ a través de la imagen resultante de $5 \times 5 \times 3$, para obtener una imagen de $5 \times 5 \times 1$ (Figura B.8 arriba). Luego, para obtener el mismo resultado que la convolución estándar, se utilizan 128 filtros de $1 \times 1 \times 3$ para generar una imagen resultante de $5 \times 5 \times 128$ (Figura B.8 abajo).

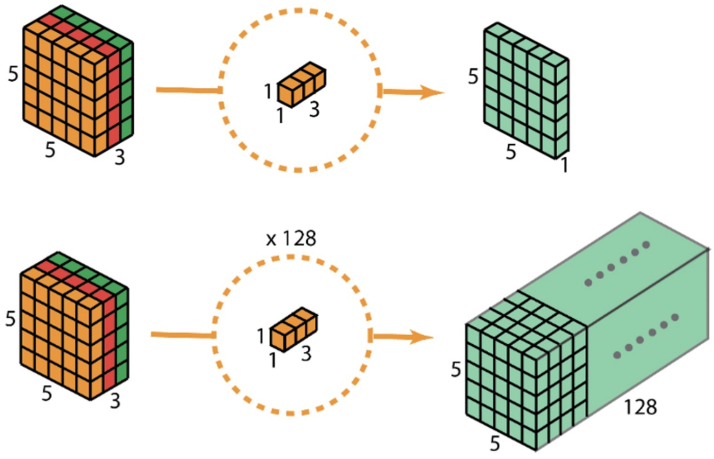


Figura B.8: Segunda etapa de la convolución separable en profundidad.

El beneficio de utilizar convoluciones separables en profundidad en vez de convoluciones estándar está en la reducción del numero de operaciones que se debe realizar para obtener el mismo resultado. Para demostrar con exactitud la afirmación anterior se

calcula el número de multiplicaciones realizadas para los dos casos expuestos anteriormente.

Para la convolución estándar se tienen 128 filtros de $3 \times 3 \times 3$ que se mueven 5×5 veces, esto es: $128 \times 3 \times 3 \times 3 \times 5 \times 5 = 86.400$ multiplicaciones.

En el caso de la convolución en profundidad se tienen 3 filtros de $3 \times 3 \times 1$ que se mueven 5×5 veces, esto es: $3 \times 3 \times 3 \times 1 \times 5 \times 5 = 675$. Por otro lado, en la convolución se tienen 128 filtros de $1 \times 1 \times 3$ que se mueven 5×5 veces, esto es: $128 \times 1 \times 1 \times 3 \times 5 \times 5 = 9.600$. Si se suman ambos resultados se obtienen 10.275 multiplicaciones.

10.275 multiplicaciones es mucho menos que 86.400 multiplicaciones. Con menos cálculos, la red se vuelve más rápida. La diferencia de eficiencia se vuelve más clara mientras más grandes sean las dimensiones de las operaciones.

B.2.2. Entrenamiento

Se implementa el cambio de convolución estándar a convolución separable en profundidad en la arquitectura de red resultante de la sección B.1. Notar que los bloques residuales también utilizan convoluciones separables en profundidad, no así las convoluciones transpuestas.

Capa	Tamaño de activación
Entrada	$3 \times 256 \times 256$
Padding	$3 \times 272 \times 272$
$32 \times 9 \times 9$ conv sep, stride 1	$32 \times 272 \times 272$
$64 \times 3 \times 3$ conv sep, stride 2	$64 \times 136 \times 136$
$128 \times 3 \times 3$ conv sep, stride 2	$128 \times 68 \times 68$
Bloque residual, 128 filtros	$128 \times 64 \times 64$
$64 \times 3 \times 3$ conv, stride $1/2$	$64 \times 128 \times 128$
$32 \times 3 \times 3$ conv, stride $1/2$	$32 \times 256 \times 256$
$3 \times 9 \times 9$ conv, stride 1	$3 \times 256 \times 256$

Tabla B.3: Arquitectura de red utilizada para comprobar la eficiencia de las convoluciones separables en profundidad.

La red anterior se entrena con los siguientes hiperparámetros: $\alpha = 5$, $\beta = 10^2$ y $\gamma = 8 \times 10^2$. Además, el parámetro de la convolución separable en profundidad

channel_multiplier se deja en 1. Se utiliza un batch de tamaño 32 y se deja en ejecución hasta que las pérdidas converjan, esto es, aproximadamente unas 18.000 iteraciones. La Figura B.9 muestra la variación de las pérdidas en función del número de iteraciones.

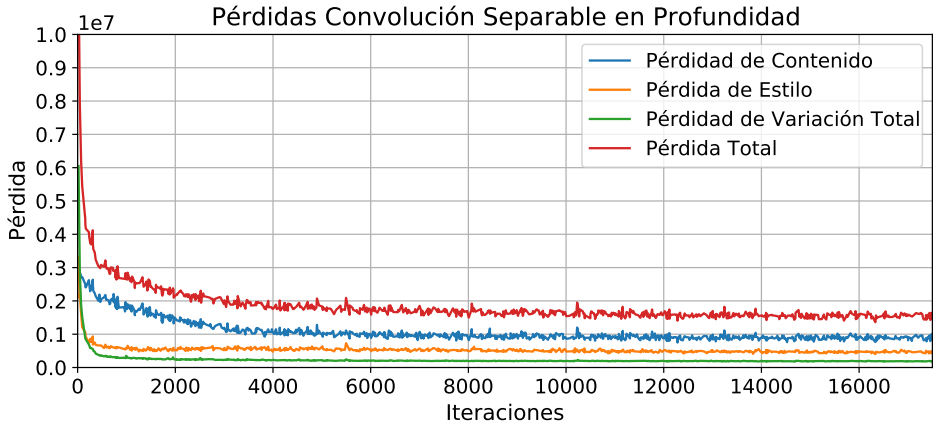


Figura B.9: Gráficos de pérdida en función del número de iteraciones de la red transformación de imágenes con convolución separable en profundidad.

B.2.3. Análisis Cuantitativo

Para comparar cuantitativamente el efecto de utilizar capas de convolución separable en profundidad, se calculan los tiempos que demora la red en procesar 420 imágenes de 640×640 pixeles. Además, las redes tienen la misma arquitectura de una sola capa residual. Los tiempos obtenidos se muestran en la Tabla B.4.

Tipo de Convolución	**Tiempo Total [s]**	**Tiempo por Imagen [s]**
Estándar	82,7082	0,1969
Separable en Profundidad	62,0672	0,1478

Tabla B.4: Comparación de tiempos para red de transformación de imágenes con convoluciones estándar y convoluciones separables por profundidad. Ambas redes tienen una capa residual y procesan 420 imágenes de 640×640.

De acuerdo a los resultados obtenidos, utilizar convoluciones separables en profundidad disminuye el tiempo de inferencia aproximadamente un $25\,\%$, lo cual se corresponde con la reducción de parámetros explicados en la sección anterior.

B.2.4. Análisis Cualitativos

Los resultados cualitativos, al modificar la capas de convolución en la arquitectura de red, son un indicador necesario para evaluar la presente opción de optimización. Si la red es rápida pero no produce buenos resultados estilizados, la opción se descartará. Dado que la convolución separable en profundidad disminuye la cantidad de parámetros de la red, es de esperar que los resultados sean cualitativamente peores que con convolución estándar. Lo que se busca es que esta diferencia cualitativa no sea tan grande.

En la Figura B.10 se muestran resultados para distintas imágenes de entradas utilizando los dos tipos de convolución. Dado que la resolución de la imagen de entrada afecta la forma en que la red estiliza (por ejemplo, ciertas estructuras se vuelven más pequeñas), se utilizan imágenes de entrada de distintos tamaños para analizar la mayor cantidad de casos posibles.

Los resultados de la Figura B.10 muestran que la arquitectura de red con convoluciones separables en profundidad no agrega texturas a ciertas partes de la imagen. Por ejemplo, en el caso de la fila 1 en la Figura B.10, la convolución estándar produce ciertas texturas verticales en las paredes de la pieza que en el otro caso son casi imperceptibles. Además, el uso de convoluciones separables en profundidad produce una pérdida de contraste y color respecto de la convolución estándar. Se puede notar en los detalles de los resultados, que se pierde en gran manera el efecto de realzar las texturas de los materiales en los elementos de la imagen. Por otro lado, los colores pierden saturación o se vuelven más opacos. En general, la arquitectura de red con convoluciones separables en profundidad logra modificar, en cierta medida, el color de la imagen original al de la imagen de estilo, pero no consigue modificar el contraste. Asimismo, la calidad de las texturas son bastante malas en comparación con la convolución estándar. Esto era de esperar, pues la disminución de parámetros debido al cambio de convolución es bastante grande, y la red, que ya ha sido optimizada una vez (sección B.1), no alcanza a aprender las características de estilo deseados.

Como estudio adicional, se decide analizar el caso para una arquitectura de red con 5 bloques residuales y con convoluciones separables en profundidad, esto sería no considerar la optimización de la sección B.1. Esta opción se analiza porque puede ser mejor optimizar la red solo modificando las convoluciones sin modificar la arquitectura de red original [52].

Para las mismas condiciones de análisis que en los estudios anteriores, se obtiene

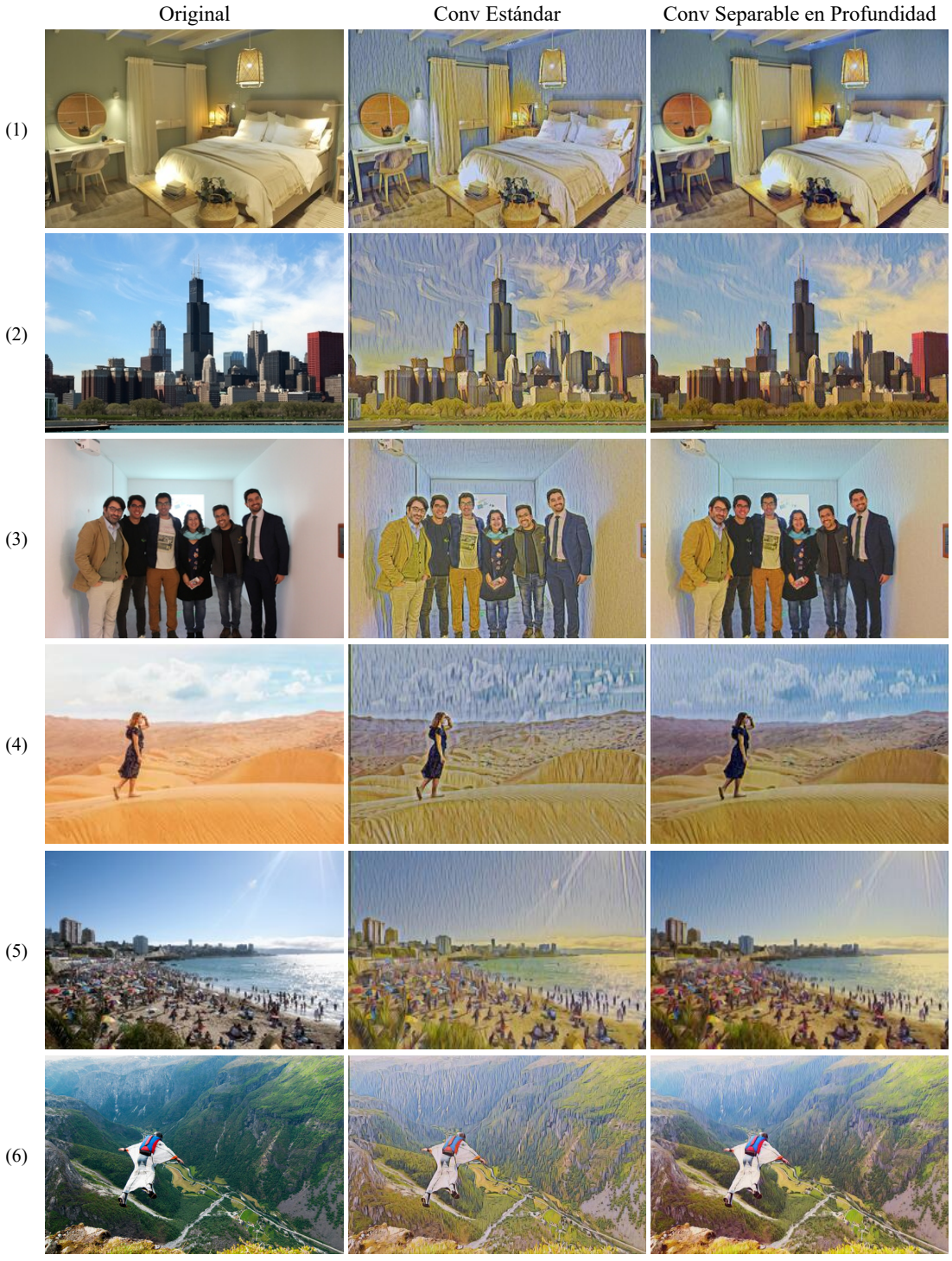


Figura B.10: Resultados cualitativos que comparan las convoluciones estándar con las convoluciones separables en profundidad.

que la velocidad de inferencia de la red es de $0{,}2101$ segundos, la cual es un poco más lenta que la red con 1 bloque residual y convoluciones estándar. Por otro lado, en la Figura B.11 se muestran los resultados cualitativos de este caso. Se puede ver que la calidad del resultado mejora al agregar 4 bloques residuales adicionales, pero sigue siendo peor que la estilización obtenida con la arquitectura de red de 1 bloque residual con convoluciones estándar (ver Figura B.10 columna central).



(a) Original (b) Estilizada

Figura B.11: Resultado cualitativo para una arquitectura de red con 5 bloques residuales y capas de convolución separable en profundidad.

B.2.5. Resolución Optimización de Convoluciones

Sustituir las convoluciones estándar por convoluciones separables en profundidad mejora bastante la velocidad de inferencia de la red. Lamentablemente, los resultados cualitativos que se obtienen con este cambio no son lo suficientemente buenos para ser utilizados en la instalación interactiva, donde la calidad de lo que se observa debe tener la capacidad de sorprender y atraer a los visitantes. Por esta razón, se decide mantener la arquitectura de red con 1 bloque residual y convoluciones estándar.

Referencias

[1] N. Rosenblum. *A World History of Photography*. 3.ª ed. Abbeville Pr, 1997.

[2] *Universe of Water Particles on a Rock where People Gather | teamLab Borderless Tokyo Official Site :MORI Building DIGITAL ART MUSEUM*. URL: https://borderless.teamlab.art/es/ew/iwa-waterparticles/.

[3] J. McCarthy. *Professor John McCarthy*. 2012. URL: http://jmc.stanford.edu/index.html.

[4] K. Hao. *What is AI? We drew you a flowchart to work it out*. Inglés. 2 de abr. de 2020. URL: https://www.technologyreview.com/2018/11/10/139137/is-this-ai-we-drew-you-a-flowchart-to-work-it-out/.

[5] T. Mitchell. *Machine Learning*. Inglés. New York, Estados Unidos: McGraw-Hill Education, 1997.

[6] E. Alpaydin. *Introduction to Machine Learning, Fourth Edition*. 4th ed. MIT Press, 2020.

[7] Y. Bengio, A. Courville y P. Vincent. «Representation learning: A review and new perspectives». En: *IEEE transactions on pattern analysis and machine intelligence* 35.8 (2013), págs. 1798-1828.

[8] Y. LeCun, Y. Bengio y G. Hinton. «Deep learning». En: *Nature* 521.7553 (2015), págs. 436-444. DOI: 10.1038/nature14539.

[9] *Obvious – Art & AI*. Francés. URL: https://obvious-art.com/.

[10] *Edmond de Belamy, from La Famille de Belamy*. Inglés. URL: https://www.christies.com/Lotfinder/lot_details.aspx?sid=&intObjectID=6166184&T=Lot&language=en.

[11] O. Hicks. *ART-ificial Intelligence: The Curious Case of Edmond De Belamy*. Inglés americano. 19 de oct. de 2019. URL: https://isismagazine.org.uk/2019/03/art-ificial-intelligence-the-curious-case-of-edmond-de-belamy/.

[12] B. Rosado. *Sotheby's subasta por primera vez una obra generada por inteligencia artificial*. 6 de mar. de 2019. URL: https://www.expansion.com/fueradeserie/cultura/2019/03/06/5c7ea1dd46163fa03e8b460a.html.

[13] *Mario Klingemann MEMORIES OF PASSERSBY I*. 6 de mar. de 2019. URL: https://www.sothebys.com/en/auctions/ecatalogue/2019/contemporary-art-day-auction-l19021/lot.109.html.

[14] *What Is the Definition of Art?* Inglés. URL: https://www.vangoghgenova.it/what-is-the-definition-of-art.html.

[15] *Art*. En: URL: https://www.lexico.com/definition/art.

[16] *Art*. En: URL: https://www.merriam-webster.com/dictionary/art.

[17] *Pablo Picasso Quotes*. URL: https://www.goodreads.com/author/quotes/3253.Pablo_Picasso.

[18] C. Medred. *Several Alaska events scheduled for Autism Awareness Month*. Inglés. 30 de sep. de 2016. URL: https://www.adn.com/arts/article/several-alaska-events-scheduled-autism-awareness-month/2011/04/01/.

[19] A. Martinez. *Art for Darfur*. Inglés. URL: https://scholar.smu.edu/big_ideas_2010_proposals/7/.

[20] B. Mathema. *Trash to treasure: Turning Mt. Everest waste into art*. 16 de ene. de 2013. URL: https://edition.cnn.com/2013/01/15/world/asia/everest-trash-art/.

[21] A. Irizarry. *Creativity in Conservation*. Inglés americano. 25 de sep. de 2020. URL: https://bowseat.org/impact/creativity-in-conservation/.

[22] E. Talongang. *La importancia del arte como herramienta para la sociedad*. 27 de mayo de 2019. URL: https://www.elperiodico.com/es/entre-todos/participacion/la-importancia-del-arte-como-herramienta-para-la-sociedad-191849.

[23] Consejo Nacional de la Cultura y las Artes". 1.ª ed. Ministerio de las Culturas, las Artes y el Patrimonio, 2018.

[24] *What is Human Development? | Human Development Reports*. Inglés. URL: http://hdr.undp.org/en/content/what-human-development.

[25] Ministro de Cultura Presidencia de la Nación". *Encuesta Nacional de Consumos Culturales 2017*. 2017.

[26] Departamento Administrativo Nacional de Estadística – DANE". *Encuesta de Consumo Cultural 2017*. 2017. URL: https://www.dane.gov.co/index.php/estadisticas-por-tema/cultura/consumo-cultural/informacion-historica-encuesta-de-consumo-cultural.

[27] Ministerio de Cultura y Deporte". *Encuesta de Hábitos y Prácticas Culturales En España 2018-2019*. 2019.

[28] *The Taking Part Survey 2016/17*. 2017. URL: https://www.artscouncil.org.uk/taking-part-survey.

[29] E. Vicente, C. Camarero y M. J. Garrido. «Insights into Innovation in European Museums». En: *Public Management Review* 14.5 (2012), págs. 649-679. DOI: 10.1080/14719037.2011.642566.

[30] C. Carmen y G. María José. «The role of technological and organizational innovation in the relation between market orientation and performance in cultural organizations». En: *European Journal of Innovation Management* 11.3 (2008), págs. 413-434. DOI: 10.1108/14601060810889035.

[31] Y. Jing et al. «Neural style transfer: A review». Inglés. En: *IEEE transactions on visualization and computer graphics* 26.11 (2019), págs. 3365-3385.

[32] B. Gooch y A. Gooch. *Non-Photorealistic Rendering*. USA: A. K. Peters, Ltd., 2001. ISBN: 1568811330.

[33] T. Strothotte y S. Schlechtweg. *Non-Photorealistic Computer Graphics: Modeling, Rendering, and Animation*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2002. ISBN: 1558607870.

[34] P. Rosin y J. Collomosse. *Image and Video-Based Artistic Stylisation*. Springer Publishing Company, Incorporated, 2012. ISBN: 1447145186.

[35] L. A. Gatys, A. S. Ecker y M. Bethge. «Image Style Transfer Using Convolutional Neural Networks». En: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, págs. 2414-2423. DOI: 10.1109/CVPR.2016.265.

[36] A. A. Efros y W. T. Freeman. «Image Quilting for Texture Synthesis and Transfer». En: New York, NY, USA: Association for Computing Machinery, 2001. ISBN: 158113374X. DOI: 10.1145/383259.383296. URL: https://doi.org/10.1145/383259.383296.

[37] I. Drori, D. Cohen-Or y H. Yeshurun. «Example-based style synthesis». En: *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings.* Vol. 2. 2003, págs. II-143. DOI: 10.1109/CVPR.2003.1211464.

[38] O. Frigo et al. «Split and Match: Example-based Adaptive Patch Sampling for Unsupervised Style Transfer». En: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Peer-reviewed paper presented at IEEE International Conference on Computer Vision and Pattern Recognition (CVPR), June 2016, Las Vegas, United States. Las Vegas, United States, jun. de 2016. URL: https://hal.archives-ouvertes.fr/hal-01280818.

[39] M. Elad y P. Milanfar. «Style transfer via texture synthesis». En: *IEEE Transactions on Image Processing* 26.5 (2017), págs. 2338-2351.

[40] L. A. Gatys, A. S. Ecker y M. Bethge. «A neural algorithm of artistic style». En: *arXiv preprint arXiv:1508.06576* (2015).

[41] P. Labs. *Prisma: Turn memories into art using artificial intelligence*. 2016. URL: https://www.artscouncil.org.uk/taking-part-survey.

[42] *Ostagram*. URL: https://www.ostagram.me/static_pages/lenta?last_days=1000&locale=en.

[43] A. J. Champandard. *Deep forger: Paint photos in the style of famous artists*. 2015. URL: http://deepforger.com.

[44] A. Mahendran y A. Vedaldi. «Understanding deep image representations by inverting them». En: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, págs. 5188-5196.

[45] A. Mahendran y A. Vedaldi. «Visualizing deep convolutional neural networks using natural pre-images». En: *International Journal of Computer Vision* 120.3 (2016), págs. 233-255.

[46] L. A. Gatys, A. S. Ecker y M. Bethge. «Texture synthesis using convolutional neural networks». En: *arXiv preprint arXiv:1505.07376* (2015).

[47] K. Simonyan y A. Zisserman. «Very deep convolutional networks for large-scale image recognition». En: *arXiv preprint arXiv:1409.1556* (2014).

[48] Y. Li et al. «Demystifying neural style transfer». En: *arXiv preprint arXiv:1701.01036* (2017).

[49] E. Risser, P. Wilmot y C. Barnes. «Stable and controllable neural texture synthesis and style transfer using histogram losses». En: *arXiv preprint arXiv:1701.08893* (2017).

[50] S. Li et al. «Laplacian-steered neural style transfer». En: *Proceedings of the 25th ACM international conference on Multimedia*. 2017, págs. 1716-1724.

[51] C. Li y M. Wand. «Combining markov random fields and convolutional neural networks for image synthesis». En: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, págs. 2479-2486.

[52] J. Johnson, A. Alahi y L. Fei-Fei. «Perceptual losses for real-time style transfer and super-resolution». En: *European conference on computer vision*. Springer. 2016, págs. 694-711.

[53] A. Radford, L. Metz y S. Chintala. «Unsupervised representation learning with deep convolutional generative adversarial networks». En: *arXiv preprint arXiv:1511.06434* (2015).

[54] D. Ulyanov, A. Vedaldi y V. Lempitsky. «Improved texture networks: Maximizing quality and diversity in feed-forward stylization and texture synthesis». En: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, págs. 6924-6932.

[55] X. Huang y S. Belongie. «Arbitrary style transfer in real-time with adaptive instance normalization». En: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, págs. 1501-1510.

[56] C. Li y M. Wand. «Precomputed real-time texture synthesis with markovian generative adversarial networks». En: *European conference on computer vision*. Springer. 2016, págs. 702-716.

[57] V. Dumoulin, J. Shlens y M. Kudlur. «A learned representation for artistic style». En: *arXiv preprint arXiv:1610.07629* (2016).

[58] D. Chen et al. «Stylebank: An explicit representation for neural image style transfer». En: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, págs. 1897-1906.

[59] Y. Li et al. «Diversified texture synthesis with feed-forward networks». En: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, págs. 3920-3928.

[60] H. Zhang y K. Dana. «Multi-style generative network for real-time transfer». En: *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*. 2018.

[61] T. Q. Chen y M. Schmidt. «Fast patch-based style transfer of arbitrary style». En: *arXiv preprint arXiv:1612.04337* (2016).

[62] G. Ghiasi et al. «Exploring the structure of a real-time, arbitrary neural artistic stylization network». En: *arXiv preprint arXiv:1705.06830* (2017).

[63] Y. Li et al. «Universal style transfer via feature transforms». En: *arXiv preprint arXiv:1705.08086* (2017).

[64] L. A. Gatys et al. «Controlling perceptual factors in neural style transfer». En: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, págs. 3985-3993.

[65] Y. Jing et al. «Stroke controllable fast style transfer with adaptive receptive fields». En: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, págs. 238-254.

[66] X. Wang et al. «Multimodal transfer: A hierarchical deep convolutional neural network for fast artistic style transfer». En: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, págs. 5239-5247.

[67] X.-C. Liu et al. «Depth-aware Neural Style Transfer». En: *Non-Photorealistic Animation and Rendering (NPAR)*. 2017, 4:1-4:10. ISBN: 978-1-4503-5081-5. DOI: 10.1145/3092919.3092924.

[68] W. Chen et al. «Single-image depth perception in the wild». En: *arXiv preprint arXiv:1604.03901* (2016).

[69] M. Lu et al. «Decoder Network over Lightweight Reconstructed Feature for Fast Semantic Style Transfer». En: *2017 IEEE International Conference on Computer Vision (ICCV)*. 2017, págs. 2488-2496. DOI: 10.1109/ICCV.2017.270.

[70] C. Castillo et al. «Son of zorn's lemma: Targeted style transfer using instance-aware semantic segmentation». En: *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2017, págs. 1348-1352.

[71] A. J. Champandard. «Semantic style transfer and turning two-bit doodles into fine artworks». En: *arXiv preprint arXiv:1603.01768* (2016).

[72] D. Chen et al. «Stereoscopic neural style transfer». En: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, págs. 6654-6663.

[73] M. Ruder, A. Dosovitskiy y T. Brox. «Artistic style transfer for videos». En: *German conference on pattern recognition*. Springer. 2016, págs. 26-36.

[74] M. Ruder, A. Dosovitskiy y T. Brox. «Artistic style transfer for videos and spherical images». En: *International Journal of Computer Vision* 126.11 (2018), págs. 1199-1219.

[75] P. Weinzaepfel et al. «DeepFlow: Large displacement optical flow with deep matching». En: *Proceedings of the IEEE international conference on computer vision*. 2013, págs. 1385-1392.

[76] J. Revaud et al. «Epicflow: Edge-preserving interpolation of correspondences for optical flow». En: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, págs. 1164-1172.

[77] H. Huang et al. «Real-Time Neural Style Transfer for Videos». En: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, págs. 7044-7052. DOI: 10.1109/CVPR.2017.745.

[78] A. Gupta et al. «Characterizing and improving stability in neural style transfer». En: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, págs. 4067-4076.

[79] D. Chen et al. «Coherent online video style transfer». En: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, págs. 1105-1114.

[80] G. Atarsaikhan et al. «Neural Font Style Transfer». En: *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*. Vol. 05. 2017, págs. 51-56. DOI: 10.1109/ICDAR.2017.328.

[81] S. Yang et al. «Awesome typography: Statistics-based text effects transfer». En: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, págs. 7464-7473.

[82] S. Azadi et al. «Multi-content gan for few-shot font style transfer». En: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, págs. 7564-7573.

[83] P. Verma y J. O. Smith. «Neural style transfer for audio spectograms». En: *arXiv preprint arXiv:1801.01589* (2018).

[84] P. Mital. «Time Domain Neural Audio Style Transfer». En: *ArXiv* abs/1711.11160 (2017).

[85] Y. Jing et al. «Neural style transfer: A review». En: *IEEE transactions on visualization and computer graphics* 26.11 (2019), págs. 3365-3385.

[86] K. Team. *Keras: the Python deep learning API*. Inglés. URL: https://keras.io/.

[87] *NVIDIA DIGITS*. Inglés. 4 de jun. de 2019. URL: https://developer.nvidia.com/digits.

[88] F. *facebookarchive/torchnet*. Inglés. URL: https://github.com/facebookarchive/torchnet.

[89] R. Collobert, L. V. D. Maaten y A. Joulin. «Torchnet: An Open-Source Platform for (Deep) Learning Research». En: 2016.

[90] P. *pytorch/tnt*. Inglés. URL: https://github.com/pytorch/tnt.

[91] *Caffe | Deep Learning Framework*. URL: http://caffe.berkeleyvision.org/.

[92] Y. Jia et al. «Caffe: Convolutional architecture for fast feature embedding». En: *Proceedings of the 22nd ACM international conference on Multimedia*. 2014, págs. 675-678.

[93] *TensorFlow*. Inglés. URL: https://www.tensorflow.org/.

[94] M. Abadi et al. «Tensorflow: Large-scale machine learning on heterogeneous distributed systems». En: *arXiv preprint arXiv:1603.04467* (2016).

[95] A. Shatnawi et al. «A comparative study of open source deep learning frameworks». En: *2018 9th International Conference on Information and Communication Systems (ICICS)*. 2018, págs. 72-77. DOI: 10.1109/IACS.2018.8355444.

[96] A. Parvat et al. «A survey of deep-learning frameworks». En: *2017 International Conference on Inventive Systems and Control (ICISC)*. 2017, págs. 1-7. DOI: 10.1109/ICISC.2017.8068684.

[97] S. Shi et al. «Benchmarking state-of-the-art deep learning software tools». En: *2016 7th International Conference on Cloud Computing and Big Data (CCBD)*. IEEE. 2016, págs. 99-104.

[98] *Torch | Scientific computing for LuaJIT*. URL: http://torch.ch/.

[99] S. *szagoruyko/loadcaffe*. Inglés. URL: https://github.com/szagoruyko/loadcaffe.

[100] Z. Wang et al. «Various Frameworks and Libraries of Machine Learning and Deep Learning: A Survey». En: *Archives of Computational Methods in Engineering* (feb. de 2019). ISSN: 1886-1784. DOI: 10.1007/s11831-018-09312-w. URL: https://doi.org/10.1007/s11831-018-09312-w.

[101] T. *torch/torch7*. Inglés. URL: https://github.com/torch/torch7.

[102] *Microsoft Cognitive Toolkit (CNTK)*. URL: https://cntk.azurewebsites.net/.

[103] microsoft. *microsoft/CNTK*. Inglés. URL: https://github.com/microsoft/CNTK.

[104] *PyTorch*. Inglés. URL: https://pytorch.org/.

[105] A. Paszke et al. «Automatic differentiation in PyTorch». En: (2017).

[106] H. *HIPS/autograd*. Inglés. URL: https://github.com/HIPS/autograd.

[107] *NVIDIA cuDNN*. Inglés. 8 de abr. de 2021. URL: https://developer.nvidia.com/cudnn.

[108] *Stack Overflow - Where Developers Learn, Share, & Build Careers*. URL: https://stackoverflow.com/.

[109] W. Rosinski. *Deep Learning Frameworks Speed Comparison*. Inglés. 22 de nov. de 2017. URL: https://wrosinski.github.io/deep-learning-frameworks/.

[110] H. Kim et al. «Performance analysis of CNN frameworks for GPUs». En: *2017 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*. 2017, págs. 55-64. DOI: 10.1109/ISPASS.2017.7975270.

[111] *The Bedroom - Vincent van Gogh*. Inglés. URL: https://www.vangoghmuseum.nl/en/collection/s0047V1962.

[112] *Boomerang from Instagram*. 30 de abr. de 2018. URL: https://apps.apple.com/es/app/boomerang-from-instagram/id1041596399.

[113] E. Ries. *The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses*. Illustrated. Currency, 2011.

[114] Intel RealSense". *Depth Camera D435*. Inglés americano. 23 de mar. de 2021. URL: https://www.intelrealsense.com/depth-camera-d435/.

[115] F. *facebookarchive/torchnet*. Inglés. URL: https://github.com/facebookarchive/torchnet.

[116] *Router DIR-608 Wireless N 150*. URL: https://la.dlink.com/la/routers/dir-608/.

[117] *ZBOX QK5P1000 (Barebone)*. Inglés. 28 de oct. de 2020. URL: https://www.zotac.com/us/product/mini_pcs/qk5p1000.

[118] *Proyector LG LED HF80LG*. URL: https://www.lg.com/pe/proyectores/lg-HF80LG#.

[119] D. Ulyanov, A. Vedaldi y V. Lempitsky. «Instance normalization: The missing ingredient for fast stylization». En: *arXiv preprint arXiv:1607.08022* (2016).

[120] Y. Wu y K. He. «Group normalization». En: *Proceedings of the European conference on computer vision (ECCV)*. 2018, págs. 3-19.

[121] O. Russakovsky et al. «Imagenet large scale visual recognition challenge». En: *International journal of computer vision* 115.3 (2015), págs. 211-252.

[122] *Pretrained CNNs - MatConvNet*. Inglés. URL: http://www.vlfeat.org/matconvnet/models/imagenet-vgg-verydeep-19.mat.

[123] T.-Y. Lin et al. «Microsoft COCO: Common Objects in Context». En: *Computer Vision – ECCV 2014*. Ed. por D. Fleet et al. Cham: Springer International Publishing, 2014, págs. 740-755. ISBN: 978-3-319-10602-1.

[124] D. P. Kingma y J. Ba. «Adam: A method for stochastic optimization». En: *arXiv preprint arXiv:1412.6980* (2014).

[125] microsoft. *microsoft/CNTK*. Inglés. URL: https://github.com/microsoft/CNTK/tree/master/PretrainedModels.

[126] T. *tensorflow/models*. Inglés. URL: https://github.com/tensorflow/models.

[127] *TensorFlow Hub*. Inglés. URL: https://www.tensorflow.org/hub.

[128] K. Team. *Keras documentation: Keras Applications*. Inglés. URL: https://keras.io/api/applications/.

[129] *Caffe | Model Zoo*. URL: https://caffe.berkeleyvision.org/model_zoo.html.

[130] *torchvision — Torchvision master documentation*. Inglés. URL: https://pytorch.org/vision/stable/index.html.

[131] *torchvision.models — Torchvision master documentation*. Inglés. URL: https://pytorch.org/vision/stable/models.html.

[132] T. *tensorflow/tensorflow*. Inglés. URL: https://github.com/tensorflow/tensorflow.

[133] BVLC". *BVLC/caffe*. Inglés. URL: https://github.com/BVLC/caffe.

[134] P. *pytorch/pytorch*. Inglés. URL: https://github.com/pytorch/pytorch.

[135] C. *The Microsoft Cognitive Toolkit - Cognitive Toolkit - CNTK*. Inglés americano. 22 de ene. de 2017. URL: https://docs.microsoft.com/en-us/cognitive-toolkit/.

[136] microsoft. *How to train a model using multiple machines? · Issue #59 · microsoft/CNTK*. Inglés. URL: https://github.com/Microsoft/CNTK/issues/59#issuecomment-178104505.

[137] T. *OpenCL support · Issue #22 · tensorflow/tensorflow*. Inglés. URL: https://github.com/tensorflow/tensorflow/issues/22.

[138] BVLC". *BVLC/caffe*. Inglés. URL: https://github.com/BVLC/caffe/tree/opencl.

[139] T. *torch/torch7*. Inglés. URL: https://github.com/torch/torch7/wiki/Cheatsheet#cuda.

[140] *TensorBoard*. Inglés. URL: https://www.tensorflow.org/tensorboard.

[141] M. *Using Graphviz for Visualization - Cognitive Toolkit - CNTK*. Inglés americano. 15 de dic. de 2017. URL: https://docs.microsoft.com/en-us/cognitive-toolkit/using-graphviz-for-visualization.

[142] *Quick Start — Netscope*. URL: https://ethereon.github.io/netscope/quickstart.html.

[143] S. *szagoruyko/pytorchviz*. Inglés. URL: https://github.com/szagoruyko/pytorchviz.

[144] J. Johnson, A. Alahi y L. Fei-Fei. «Perceptual losses for real-time style transfer and super-resolution: supplementary material». En: *European conference on computer vision*. Department of Computer Science, Stanford University. 2016.

[145] K. He et al. «Deep residual learning for image recognition». En: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, págs. 770-778.

[146] F. Chollet. «Xception: Deep learning with depthwise separable convolutions». En: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, págs. 1251-1258.