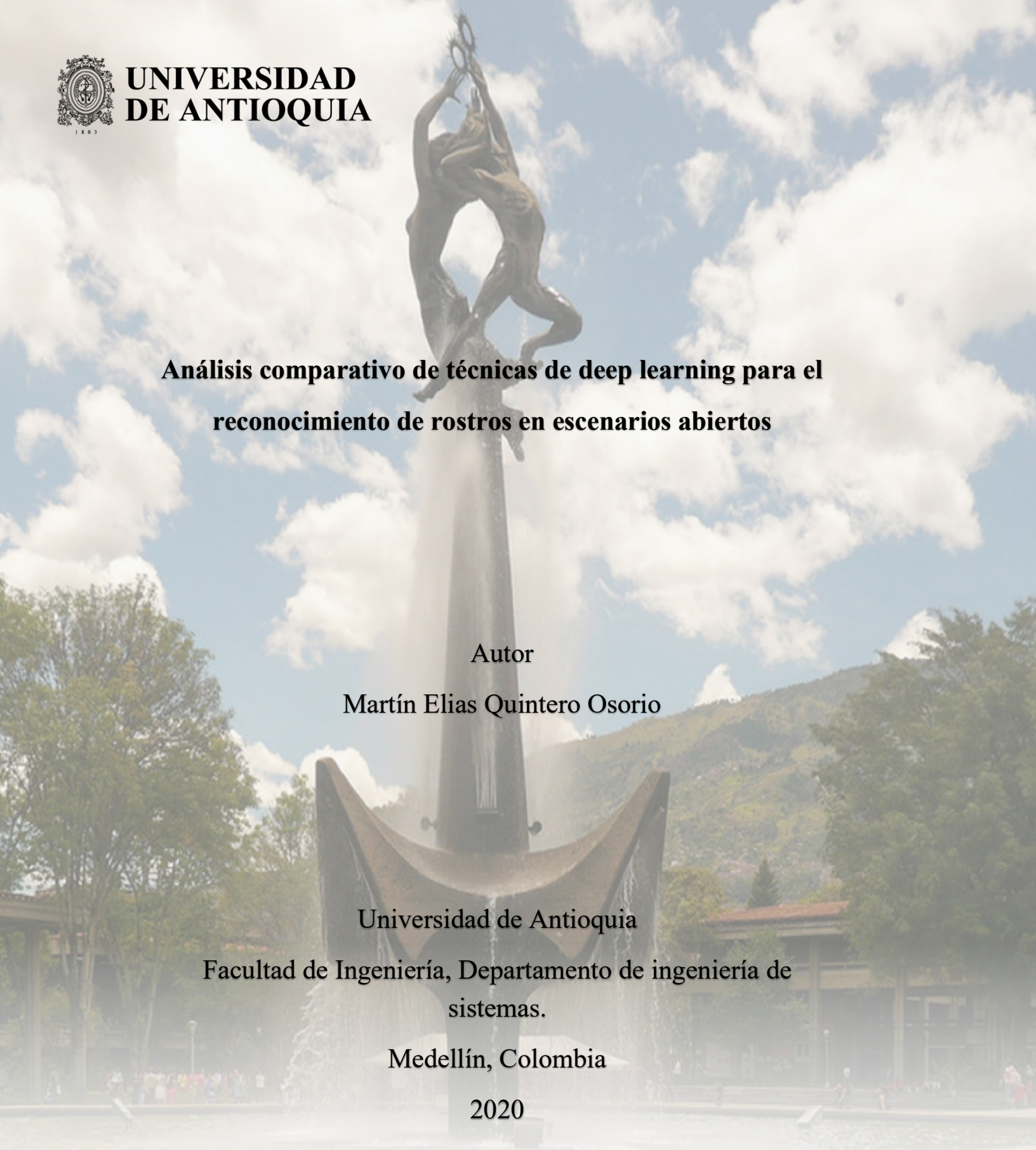


UNIVERSIDAD DE ANTIOQUIA

Análisis comparativo de técnicas de deep learning para el reconocimiento de rostros en escenarios abiertos

Autor

Martín Elias Quintero Osorio

Universidad de Antioquia

Facultad de Ingeniería, Departamento de ingeniería de sistemas.

Medellín, Colombia

2020

Análisis comparativo de técnicas de deep learning para el reconocimiento de rostros en escenarios abiertos

Martín Elias Quintero Osorio

Trabajo de grado
como requisito para optar al título de:
Ingeniero de sistemas

Asesor

Prof. Julián David Arias Londoño, PHD.

Universidad de Antioquia
Facultad de Ingeniería, Departamento de ingeniería de sistemas.
Medellín, Colombia
2020

A mi madre Maria Bernarda Osorio Petro, mi padre Martín Emilio Herrera Mora y mis abuelos Vicente Osorio Fajardo y Marlene Petro Nerio, han sido mi guía y motor desde siempre, gracias por formar la persona que soy. Los amo.

Agradecimientos

En primer lugar, agradezco a mi asesor, Prof. Julian David Arias Londoño, por el apoyo y la confianza que me brindó en este trabajo de grado y durante gran parte de mi formación profesional, su acompañamiento dedicado ha sido esencial en mi formación.

Igualmente, agradezco al grupo de investigación In2Lab y a su director, Prof. Jhon Freddy Duitama Muñoz, por permitir el desarrollo de este proyecto y facilitar muchas de las herramientas necesarias para que fluyera de la mejor manera posible.

Quiero expresar también mi más sincero agradecimiento a Joseph Fabricio Vergel-Becerra quien atendió mis consultas y ha apoyado desde hace años y más importante, por la gran amistad que me ha brindado. Sus consejos y explicaciones fueron cruciales para el desarrollo de este texto.

Al Prof. Leonardo A. Pachón que de forma desinteresada aportó toneladas de información en nuestras conversaciones, también se ha convertido en un colega y amigo muy cercano.

Por la envergadura del proyecto se requieren componentes especiales para su culminación, mi agradecimiento a guane Enterprises que de forma generosa aportó gran parte de ese procesamiento.

A mi colega Andrés Gutiérrez por darme claridad en uno de los temas más complicados y al mismo tiempo más interesantes para mi, los modelos generativos.

A mis compañeros de vida Cristian Andres Bustamante Caro, Christian Camilo Gaviria Castro, Jharol Muñóz Padilla y Johan Ospina Hincapié, su acompañamiento incondicional y paciencia han sido fundamentales para mí.

Finalmente, gracias a mis padres, abuelos y amigos más cercanos por tantos años apoyo.

A todos, infinitas gracias. Sin ustedes, no sería posible este escrito.

Título

Análisis comparativo de técnicas de *deep learning* para la reconocimiento de rostros en escenarios abiertos

Resumen: La detección y reconocimiento de rostros con alta precisión es uno de los aspectos más importante en el desarrollo e implementación de sistemas de seguridad biométricos modernos. Una de las alternativas más robustas para alcanzar esa alta precisión es el uso de redes neuronales convolucionales; sin embargo, los pequeños conjuntos de entrenamiento, el desbalance de clases, las condiciones ambientales y falta de estandarización en los protocolos de captura representan retos mayúsculos. En esta investigación se exploran alternativas para minimizar el impacto del uso de pequeños conjuntos de datos a través de técnicas de aumento de datos (transformaciones genéricas y modelos generativos), en arquitecturas de redes neuronales enfocadas en la detección del rostro. Se encontró que FaceNet fue la arquitectura convolucional que mejor se desempeñó en la tarea de reconocimiento facial haciendo uso del aumento de datos utilizando un autoencoder variacional, además, este enfoque ayuda positivamente a combatir el sobreajuste de los modelos.

Palabras clave: *Small data, deep learning, data augmentation, face recognition.*

Índice general

CAPÍTULO 1

Introducción

Los esfuerzos tecnológicos a lo largo de la historia han estado dirigidos a mejorar la calidad de vida y el confort. A lo largo de sus vidas, las personas pasan gran parte del tiempo en edificaciones, espacios compartidos y evidentemente en sus hogares, motivando así el constante deseo de mejorar estos ambientes. Fue entonces cuando en 1883, Warren S. Johnson profesor del *State Normal School*, cansado y frustrado por no poder regular las temperaturas dentro de su aula de clase, inventó el primer sistema neumático de control de temperaturas; sistema que se popularizaría e implementaría en miles de edificaciones al rededor del mundo, desde hospitales, escuelas, edificios de oficinas y hoteles, a practicamente cualquier tipo de edificación que requiriera regulación de temperatura. Y fue así como desde el siglo XIX se daban los primeros indicios de lo que en el siguiente siglo se acuñaría bajo el término de domótica para referirse a la automatización de edificaciones. Sin embargo, fue el nuevo mileno el que trajo consigo grandes avances tecnológicos en *hardware* y *software* que han repercutido fuertemente en el desarrollo y democratización de la demótica, como es el caso de los sistemas inteligentes (inteligencia artificial o AI por sus siglas en ingles) basados en algoritmos de *machine learning* y *deep learning* que cada vez están más presentes en la cotidianidad de todos.

Sin duda alguna, el *deep learning* ha dominado durante la ultima década las aplicaciones de vision artificial, procesamiento de audio y proceseamiento del lenguaje natural (NLP por sus siglas en ingles) gracias a uno de sus enfoques más llamativos, el *end-to-end learning* que proporciona a través de "profundas y gigantescas" arquitecturas la ejecución implícita de complejos "pipelines" de procesamiento y toda una diversa etapa de ingeniería de características (*feature engineering*), siendo entonces una solución directa y flexible, capaz de vencer a los mejores algoritmos de los ultimos años en tareas de detección de objetos, reconocimiento de voz, segmentación de imágenes, reconocimiento de entidades en texto no estructurado, entre muchas otras más. No obstante, pese al éxito que han tenido las arquitecturas de redes neuronales (*deep learning*), al día de hoy aun persisten gran parte de sus limitaciones y desventajes, como es el caso del demandante poder computacional que requieren en cada entrenamiento o la complejidad que presentan a la hora de desplegar en producción estos robustos sistemas, condicionando casi de manera obligatoria la presencia de unidades gráficas de procesamiento (GPU por sus siglas en ingles), además de su principal necesidad; bastas cantidades de datos para el proceso de entrenamiento.

A pesar que en nuestros días ya existen masivas cantidades de datos, no siempre son de fácil acceso o requieren de arduas tareas de etiquetado, dificultando la correcta generación de sistemas inteligentes basados en *deep learning*. Especificamente en el área del *computer vision* (vision por computadora o vision artificial), la baja cantidad de imágenes de una determinada clase produce conjuntos de datos (*datasets*) desbalanceados que impactan seriamente en las predicciones de las redes neuronales convolucionales (CNN por sus siglas en ingles), algoritmo por excelencia en la mayoría de tareas de procesamiento de imágenes. Por esta razón, sendas investigaciones han propuesto procedimientos de aumento de datos (*data augmentation*) a través de transformaciones básicas y manipulaciones, técnicas de sobremuestreo (*oversampling*) o enfoques más sofisticados para la generación de imágenes sintéticas con algoritmos no supervisados, como es el caso de las redes generativas adversarias (GAN por sus siglas en ingles) y diferentes tipos de *autoencoders*.

Uno de los requerimientos actuales de los sistemas de domótica e internet de las cosas hogares y oficinas, es la posibilidad de identificar unívocamente a la persona que está haciendo uso de un espacio en un determinado momento, para ajustar las condiciones habitacionales de acuerdo con los gustos establecidos por dicho usuario, tales como la temperatura, la iluminación, el canal de musical, el canal de TV, etc. Adicionalmente, esta aproximación puede ser usada para implementar estrategias de seguridad como deshabilitar una estufa o un horno, si quien se encuentra en el espacio de la cocina es un niño pequeño.

Para poder lograr dicho objetivo se han desarrollado algunas aproximaciones asociando dispositivos móviles como smartphones, a la presencia de su propietario. Sin embargo, dicha aproximación tiene grandes limitaciones debido a las fuertes suposiciones que realiza y a las múltiples situaciones en las que dichas suposiciones pueden no cumplirse. Una de las alternativas para poder realizar la identificación de los usuarios en los espacios de uso común, es utilizar las cámaras de vigilancia que están cada vez más diseminadas en muchos espacios, con el objetivo de hacer reconocimiento de las personas por las imágenes de rostro a partir de los videos capturados por las cámaras de vigilancia. Si bien es una alternativa viable, para poder implementar una solución de este tipo, es necesario analizar la capacidad que tiene las tecnologías actuales de reconocimiento de personas a partir de imágenes de rostros, que como ya se dijo en su mayoría están basadas en técnicas de deep learning, de trabajar con la calidad de imáganes y de enfoque que se encuentran en referencias de cámaras típicas de seguridad, particularmente en el contexto local.

Por lo tanto, en este trabajo se realiza un análisis de diferentes arquitecturas de procesamiento de imáganes, en el problema de identificación y reconocimiento de personas en imáganes obtenidas de una cámara de seguridad. Se implementaron varios enfoques de *data augmentation* para el balanceo de un *dataset* propio de imágenes con el que se entrenaron algunas de las más populares arquitecturas *end-to-end learning*, bajo el esquema de *transfer learning*. Esto con el objetivo de realizar reconocimiento facial en condiciones ambientales abiertas. Finalmente, después de la evaluación de resultados y con miras a la implementación en sistemas de seguridad y de acceso en el campo de la demótica, esta investigación define las mejores combinaciones metodológicas para la tarea predicativa de reconocimiento facial en escenarios con condiciones ambientales abiertas.

CAPÍTULO 2

Planteamiento del problema y trabajos relacionados

Los sistemas de reconocimiento facial cada vez tienen mayor presencia en la cotidianidad, como es el caso de los dispositivos de acceso seguro y monitoreo continuo. Su implementación en los teléfonos móviles y televisores inteligentes, en los aeropuertos para el control migratorio, en los supermercados y centros comerciales para el seguimiento de los consumidores, en aeropuertos para la detección temprana de objetos extraños, entre otros lugares y dispositivos, es una novedad hoy en día. Ahora, las condiciones ambientales de exposición agudizan las diferentes complicaciones que pueda tener cualquier sistema inteligente de visión artificial e.g. en las tareas de reconocimiento o detección de objetos. Las variantes en luminosidad, el bajo contrastes, las posibles oclusiones y desenfoques, variaciones a través de las diferentes poses, entre otros, son algunos de los principales agravantes que dificultan las predicciones en escenarios abiertos[1, 2]. Este amplio margen de posibilidades en los problemas de reconocimiento facial ha sido acuñado por la comunidad del *computer vision* bajo la denotación "reconocimiento facial sin restricciones" (*unconstrained face recognition*)[3]. Al mismo tiempo, las repercusiones del *unconstrained face recognition* en la adquisición de imágenes acompañan al latente problema de la difícil obtención de muestras y el desbalanceo de clases para los sistemas de reconocimiento facial[4, 5].

Por lo tanto, gran parte de los avances investigativos y tecnológicos de los últimos años plantean diversas estrategias para la mitigación de estas complicaciones, especialmente en volver más robusto los algoritmos de reconocimiento facial sin restricciones basados en *deep learning*[6] y en aumentar la cantidad de muestras[7, 8]. No obstante, en ambos conjuntos de soluciones se solapan varios de sus objetivos y pasos metodológicos, e.g., el uso de modelos tridimensionales para un data augmentation más realista, genera variaciones de la pose para cada sujeto[9]. Por otra parte, la generación de imágenes sintéticas a través de GANs (*Generative Adversarial Networks*) es uno de los métodos más populares y con mejores resultados, haciendo de los estimadores "invariantes" frente a las multiples poses y en la expresiones de un rostro[10, 11]. Otra opción es el uso de estrategias de data augmentation basadas en autoencoders variacionales[12]. No obstante, a pesar que existen muchas más metodologías, muchas de ellas giran en torno a aumentar el número de muestras y al mismo tiempo eliminar las ligaduras para las predicciones. Otras soluciones que combaten el problema de la baja cantidad de muestras, son los enfoques *one-shot learning* y *few-shot learning*[13, 14, 15], en donde a partir de pocas muestras se realiza un

proceso de *feature transfer learning*[16] para obtener mejores resultados en las predicciones. Finalmente, la integración de estas estrategias en el *workflow* global de los sistemas de reconocimiento facial es un aspecto que varia su dificultad según las particularidades de los datos disponibles[17, 18].

CAPÍTULO 3

Marco Teórico

3.1. Aprendizaje profundo

La base de los sistemas "cognitivos" actuales es el aprendizaje de máquina (*machine learning*), en donde el algoritmo es capaz de aprender a partir de la experiencia y no por un proceso de instrucciones programadas previamente y en donde existen diversos subcampos como se muestra en la Figura 3.1. Uno de estos subconjuntos de técnicas que más ha conseguido despertar un especial interés en la última década es el conocido como aprendizaje profundo, cuya principal característica es su estructura bio-inspirada en las redes neuronales de los sistemas nerviosos del reino animal.

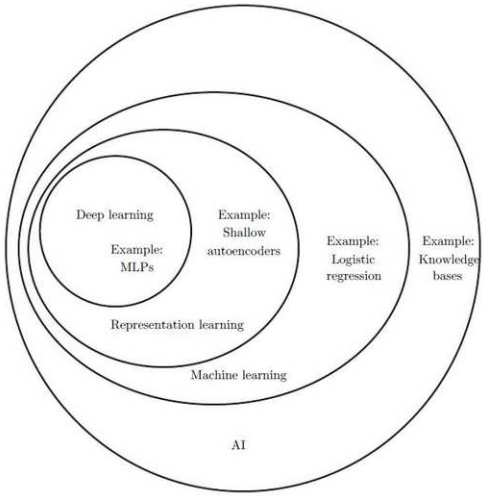


FIGURA 3.1: Diragama jerarquico de la inteligencia artificial y sus subcampos: *machine learning*, *representation learning* y *deep learning*. Fuente: [19].

De manera muy general, una red neuronal artificial (ANN: *Artificial Neural Networks*) es un conjunto de neuronas artificiales (funciones matemáticas conocidas como funciones de activación, tales como tanh, *sigmoide*, $softmax$, etc.) conectadas entre sí mediante pesos estadísticos w_{ij}. La red "aprende" a realizar tareas cuando considera ejemplos, generalmente sin estar programado con ninguna regla de tareas específicas pero que le permiten

ajustar los pesos w_{ij} óptimos para generar una predicción. El diagrama esquemático de la figura 3.2, las neuronas se representan por circunferencias, y en cada una de éstas se programa una función de activación según sea la funcionalidad de la capa a la que pertenezca. R, N y S corresponden al número de entradas(*inputs*), neuronas ocultas(*hidden neurons*) y salidas(*outputs*), respectivamente; x es el vector de entrada de la red, iw y hw son las matrices de entrada y de pesos ocultos, respectivamente. hb y ob_i son el sesgo de la capa oculta y la capa de salida, respectivamente; ho corresponde al vector de salida de la capa oculta y y al vector de salida de la red.

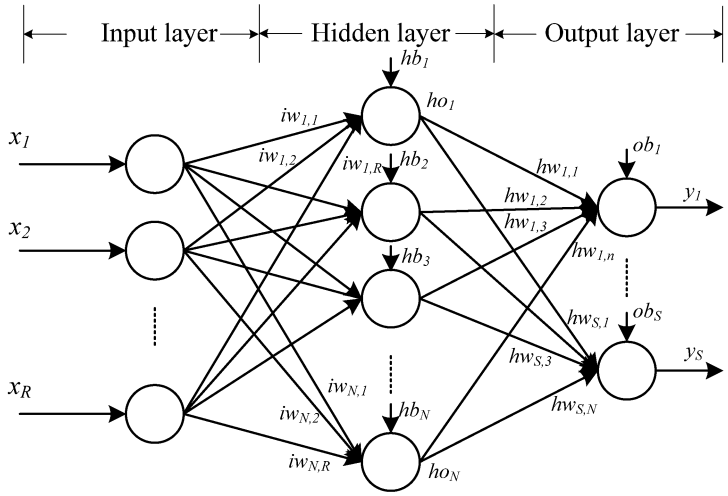


FIGURA 3.2: Diagrama esquemático de una red neuronal artificial (Perceptron multicapa). Fuente: Mdpi.

Las ANNs están constituidas por una capa de entrada, una capa de salida y entre estas dos, un cierto número de capas ocultas que realizan diferentes operaciones no lineales en los datos. Más de tres capas otorgan a la red el carácter profundo y es de allí de donde viene el concepto de aprendizaje profundo. El proceso de aprendizaje estadístico (*statistical learning*) de la red viene de un proceso de optimización del error en las predicciones de la red y en cada ciclo (época o *epoch*) del proceso de retroalimentación de la red (*feedback*), los pesos estadísticos w_{ij} de las conexiones entre las neuronas son recalculados. Este proceso de ajuste sobre los w_{ij} es conocido como propagación hacia atrás (*backpropagation*).

Para la aplicación del proceso de *backpropagation* es necesario definir el tipo de aprendizaje de la red. Para definir este aspecto principalmente se analizan los datos con los que se entrenará la red. Sí para cada conjunto de entrada x_i se posee el conocimiento a priori y_i, se dice que el paradigma de aprendizaje es supervisado. Tomando como base el teorema de Bayes, la mayoría de los algoritmos de aprendizaje supervisado se basan en la estimación de la distribución de probabilidad condicional $p(y|x)$ [19]. En este punto el método de estimación por máxima verosimilitud (*maximum likelihood estimation*) juega un rol fundamental, pues proporciona un mecanismo para la estimación de los parámetros del modelo estadístico, i.e., permite encontrar el mejor vector de parámetros θ para una familia paramétrica de distribuciones $p(y|x;\theta)$. Para los problemas de regresión lineal, la familia de distribuciones de probabilidad condicional está definida por

$$p(y|x;\theta) = \mathcal{N}(y;\theta^T x, I), \tag{3.1}$$

expresado como una distribución condicional Gaussiana en $\mathbf{y}^{(\text{train})}$,

$$p(\mathbf{y}^{(\text{train})}|\mathbf{X}^{(\text{train})},\theta) = \mathcal{N}(\mathbf{y}^{(\text{train})},\mathbf{X}^{(\text{train})}\theta,\mathbf{I}) \tag{3.2}$$

$$\propto \exp\left(-\frac{1}{2}(\mathbf{y}^{(\text{train})} - \mathbf{X}^{(\text{train})}\theta)^T(\mathbf{y}^{(\text{train})} - \mathbf{X}^{(\text{train})}\theta)\right), \tag{3.3}$$

en donde la etiqueta (train) hace referencia al espacio de muestras que conforman el conocimiento a priori del modelo. Por su parte $\mathbf{X}^{(\text{train})}$ es la información o los datos suministrados para el proceso de aprendizaje, junto con su respectivo conjunto de objetivos $\mathbf{y}^{(\text{train})}$ que el algoritmo buscará producir en un nuevo conjunto de muestras $\mathbf{X}^{(\text{test})}$. El proceso de aprendizaje estadístico se enfoca en computar el vector de parámetros θ que para el caso de las ANN está relacionado con la estimación de los pesos estadísticos w_{ij} que unen a las neuronas de las diferentes capas.

Es posible generalizar el concepto de regresión lineal al escenario de los problemas de clasificación definiendo una familia diferente de distribuciones de probabilidad. Sean dos clases bien definidas, la clase a y la clase b, entonces solo es necesario especificar la probabilidad de una de estas clases. La probabilidad de la clase b determina la probabilidad de la clase a, puesto que estos dos valores deben cumplir el axioma de probabilidad de Kolmogorov, i.e., la suma de todas las probabilidades debe ser igual a uno.

3.2. Redes neuronales convolucionales

En los problemas de clasificación, especialmente en el procesamiento de imágenes, es precisamente en donde las redes neuronales convolucionales poseen grandes capacidades en la ejecución de este tipo de tareas. Las CNNs son un conjunto de algoritmos de *deep learning* y *computer vision* ampliamente utilizados en la actualidad en el área del procesamiento digital de imágenes, pues tienen la particularidad que entre sus capas ocultas se realizan operaciones de convolución entre diferentes transformaciones de las imágenes de entrada. Este proceso es similar a la ingeniería de características (*feature engineering*) y se efectúa implícitamente dentro de la arquitectura de la red.

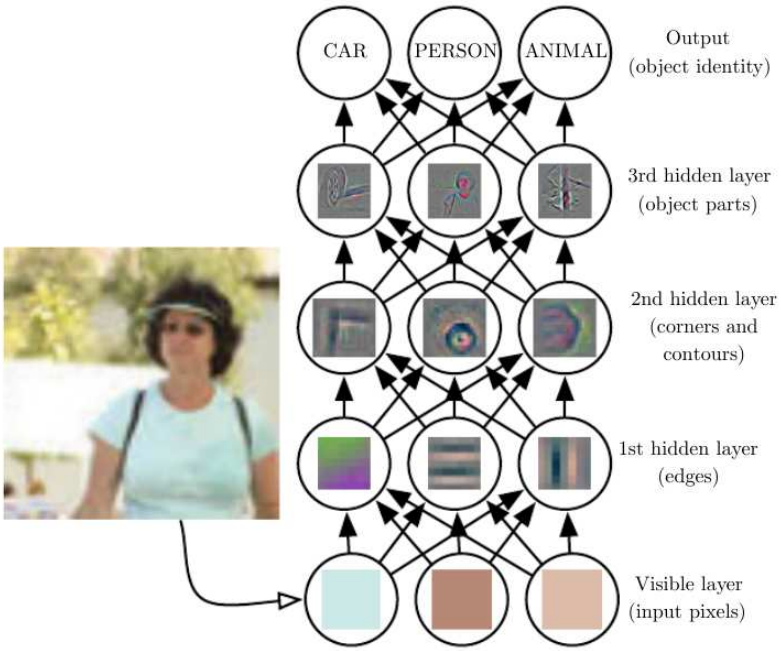


FIGURA 3.3: Descripción a alto nivel del funcionamiento de una CNN. Fuente: Ref [19].

Dichas transformaciones de las características es una de las particularidades que hacen robustas a las redes neuronales convolucionales. La figura 3.3 muestra el diagrama del funcionamiento de una CNN, en donde la capa de entrada (**visible layer**) recibe la información de intensidad de la imagen en una determinada escala de color. Posteriormente, las siguientes tres capas ocultas transforman y extraen características y las convolucionan con un conjunto de núcleos (*kernels*). Estos *kernels*, también conocidos como matrices de convolución, filtros o *feature detectors*, se utilizan para desenfocar, enfocar, grabar, detectar bordes y más. De esta manera se generan representaciones cada vez más abstractas de la imagen.

Específicamente, en el ejemplo citado en la figura 3.3 la primera capa oculta identifica los bordes en la imagen comparando la intensidad de los píxeles vecinos. Una vez extraída la característica de los bordes de la primera capa, la segunda capa oculta busca esquinas y contornos extendidos, que se reconocen como colecciones de bordes (*feature maps*). Ahora, a partir del resultado obtenido por la segunda capa, la tercera capa oculta detecta partes enteras de objetos específicos al encontrar *feature maps* específicos que las identifican. Finalmente, la capa de salida asocia las diferentes *feature maps* calculados al conocimiento a priori de las etiquetas suministradas a la red en la etapa de entrenamiento, i.e., las diferentes clases que definen el proceso de clasificación, con lo que el estimador resultante es capaz de identificar los objetos presentes en la imagen.

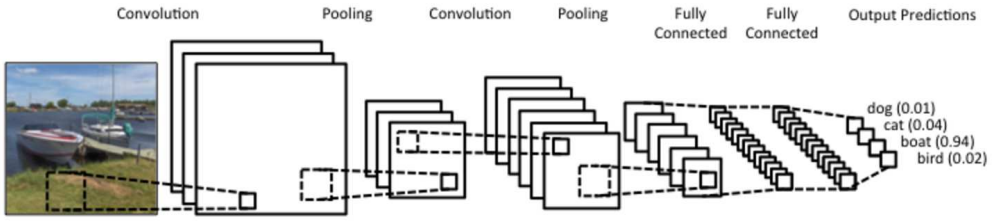


FIGURA 3.4: Estructura de una red neuronal convolucional. Fuente: Ref [19].

A diferencia de las tradicionales *feedforward networks* como el Multi-Layer Perceptron (ver figura 3.2) en donde cada neurona está conectada a todas las neuronas de la capa siguiente, las CNNs efectúan operaciones de convolución entre regiones de la imagen de entrada y el banco de *kernels* como se aprecia en el diagrama esquemático de la figura 3.4. El operador de convolución está definido por[20]:

$$S[i,j] = \sum_{-m}^{m} \sum_{-n}^{n} f\,[m,n]\, x\,[i+m, j+n]\,, \tag{3.4}$$

donde $S[i,j]$ es el *feature map* producto de la convolución, m y n son el tamaño (alto y ancho respectivamente) del *kernel* f y $x[i,j]$ es el valor original del pixel en la región extraída. Estas aplicaciones del operador convolución generan colecciones de "descriptores" llamados *feature maps* o *activation maps* (simbolizados como cuadrados en las diferentes capas de la figura 3.4). Cada filtro compone una región local de características de nivel inferior en una representación de nivel superior, dotando así a la red de cierta composicionalidad[21].

El tamaño del *kernel* define la región que se "desliza" sobre la imagen, generalmente centrada o alineada sobre cada píxel y su desplazamiento está definido por el número de pasos también conocidos como *strides*. Las condiciones de frontera para las convoluciones en los límites de la imagen son tratados con la anulación de la aplicación del *kernel* por fuera de la imagen, procedimiento al que se le conoce como *padding* y además de evitar indeterminaciones en el cálculo de los *feature maps*, preserva el tamaño espacial y contribuye a mejorar el rendimiento manteniendo información de los bordes[22]. Debido a la manera en cómo se recorre y generan los *feature maps*, no importa el lugar en el que se encuentre un determinado objeto, simplemente será capturado en una cantidad n de regiones. Esto se conoce como invarianza traslacional y es una de las principales bondades de las CNNs[23]. Una vez obtenida una colección de *feature maps*, estos son transformados por capas intermedias a los bloques convolucionales efectuando operaciones de agrupación (*pooling*). A grandes rasgos, las *pooling layers* submuestrean los mapas según un criterio determinado. Entre los operadores *pooling* más comunes están: 1) la suma total de los píxeles (*sum-pooling*), 2) el valor promedio de los píxeles que componen la región de desplazamiento (*average-pooling*) y 3) el valor maximo de los pixeles en dicha región (*max-pooling*). Los operadores *max-pooling* y *average-pooling* se definen como[24, 25]:

$$y_i = \text{máx}\{x_j, j \in [1, m^2]\} \;\; \& \;\; y_i = \frac{1}{m^2} \sum_{j=1}^{m^2} x_j, j \in [1, m^2], \tag{3.5}$$

en donde i es el índice de la ventana de agrupación, m es el tamaño de la ventana de agrupación y x_j es el j-esimo valor del arreglo $\mathbf{x}$ de pixeles que conforman la ventana de agrupación. Precisamente el operador *pooling* es quien aumenta la invariancia espacial de una red[24] y evita el sobreajuste (*overfiting*). Cada *feature map* está conectado a una neurona de las capas completamente conectadas (*fully connected*) a traves de una funcion de activación no lineal tipo *ReLU* o *tanh*. Dichas capas *fully conected* junto a la capa de salida se comportan muy similar al MLP mostrado en la figura 3.2 y con ellas se define un clasificador capaz de discriminar entre las diferentes clases del conjunto de entrenamiento.

Este robusto procedimiento de transformación de características es una de las principales razones de porque las redes neuronales convolucionales son tan poderosas a la hora de procesar imágenes. Después de su primera victoria en Imagenet[26] en el año 2012 con la arquitectura AlexNet[27], las CNNs han proliferado y dominado el campo del *computer vision* en cuanto a tareas de predicción se refiere. Al mismo tiempo, las arquitecturas convolucionales renovaron dicho campo al reemplazar el tradicional *feature engineering* por arquitecturas cada vez más profundas que implícitamente efectúan esta etapa de transformación de características de manera automática, convirtiendo a las CNNs en el referente de una nueva técnica del *deep learning* denominada *end-to-end learning*.

3.3. Arquitecturas convolucionales destacadas

La técnica del aprendizaje de extremo a extremo o *end-to-end learning* es un tema muy activo en el campo del *deep learning* que aprovecha la estructura de las redes neuronales profundas (DNN por sus siglas en inglés) y su carácter de profundidad para resolver problemas complejos[28]. Cada capa de una DNN, bloque (grupo de capas del mismo operador) o módulo (conjunto de capas que pueden contener diferentes operadores) puede especializarse en tareas intermedias necesarias para resolver un determinado problema. Tobias Glasmachers en [29] asegura que si todos los módulos de una DNN son diferenciables con respecto a todos los parámetros ajustables (pesos), entonces el entrenamiento de estos sistemas como un todo se eleva al estado de principios, i.e., el sistema se entrena de manera integral basado en un principio único.

Aunque elegante y directa, el *end-to-end learnig* sigue siendo un método por fuerza bruta (al igual que la mayoria de técnicas agrupadas por el *deep learning*), que se ha popularizado por reemplazar parcialmente la tradicional ingeniería de características (*feature engineering*). Dado que su objetivo es lanzar tuberías (*pipelines*) de procesamiento complejas con la versatilidad y flexibilidad que caracterizan el *deep learning*, el *end-to-end learning* es capaz de difuminar los límites clásicos que diferencian a estos sistemas inteligentes de otros componentes de procesamiento[2]. Por consiguiente, bajo la premisa en la que se ha convertido el *end-to-end learning*, las redes neuronales convolucionales se han rediseñado y reinventado basándose en principios de optimización de parámetros, regularización, reformulación estructural, entre otros[2]. La figura 3.5 muestra la taxonomía de las CNNs más populares de la última década, donde destacan algunas de las arquitecturas que serán presentadas en las siguientes subsecciones.

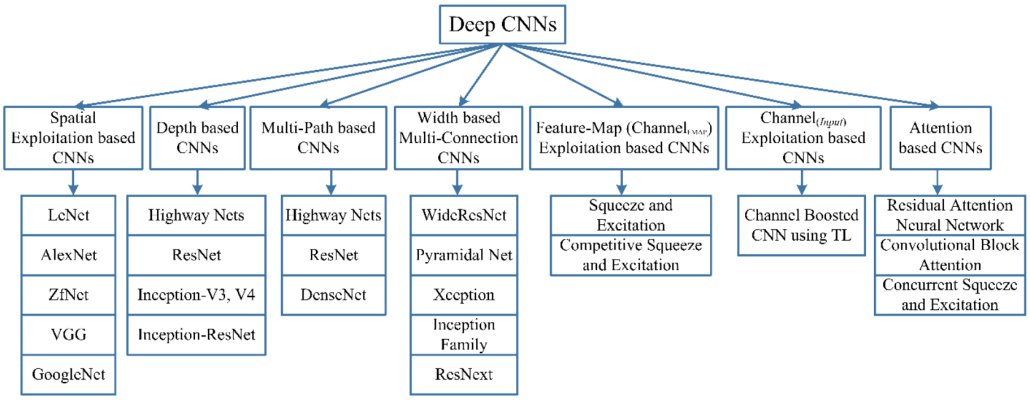


FIGURA 3.5: Clasificación de la redes nueronales convolucionales. Fuente: [2].

3.3.1. VGG16

El *Visual Geometry Group* (VGG) de la Universidad de Oxford inventó la VGG16[30] que cuenta con 13 capas convolucionales y 3 densas (completamente conectadas) con funciones de activación *ReLU* y un total de 138 millones de parámetros. La figura 3.6 muestra la arquitectura VGG16, en donde los bloques "conv 3x3" definen convoluciones por un banco de filtros(*kernels*) de 3x3 píxeles y los bloques "max-pool 2x2" aplicaciones del operador *max-pooling* en ventanas de agrupación de 2x2 píxeles. Por otra parte, la etiqueta "R" denota funciones de activación tipo *ReLU*[31] y la etiqueta "S" define la función $Softmax$ usada en problemas de clasificación multiclase. Estas convenciones prevalecen para los componentes homólogos de las arquitecturas subsecuentes.

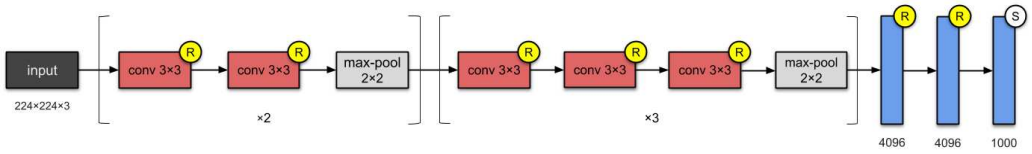


FIGURA 3.6: Arquitectura de la red nueronal convolucional VGG16. Fuente: [32, 30].

3.3.2. Inception-v3

Inception-v3[33] es un sucesor de Inception-v1[28], con 24M de parámetros. Inception-v3 reporta mejoras en el optimizador, función de pérdida y adición de normalización por lotes, en comparación a sus antecesores. Inception-v3 evita cuellos de botella representativos, i.e., reduce drásticamente las dimensiones de entrada de la siguiente capa y de esta manera obtiene cálculos más eficientes mediante el uso de métodos de factorización. La figura 3.9 muestra la arquitectura global de Inception-v3 y sus diferentes módulos que combinan factorizaciones de convoluciones simétricas y asimétricas.

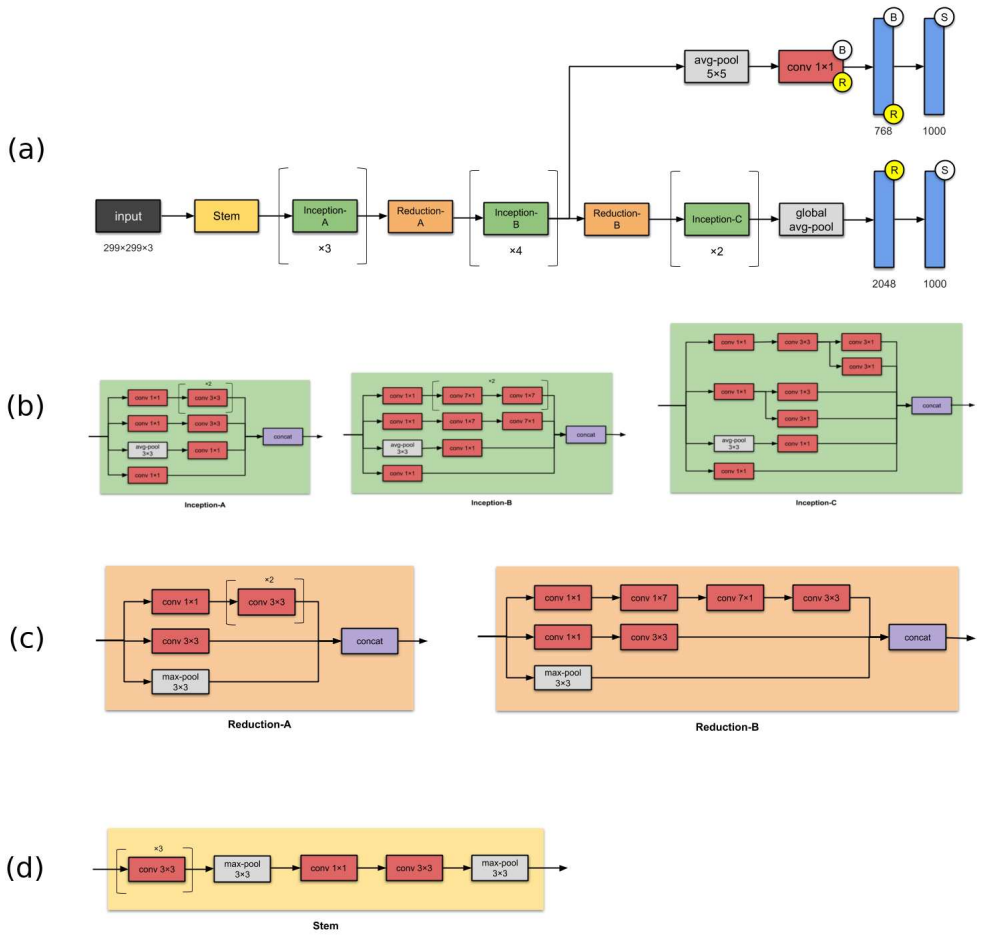


FIGURA 3.7: Inception-v3: (a) Arquitectura global. (b) Modulos *Inception*. (c) Modulos de reduccion. (d) Modulo *Stem*. Fuente: [32, 33].

3.3.3. ResNet-50

La arquitectura ResNet-50 se enfoca en atacar el problema de la degradación[34], en dónde a medida que se aumenta la profundidad de la red, el entrenamiento acrecienta su demanda computacional, la precisión se satura y posteriormente se degrada abruptamente. Para solucionar este inconveniente, He et al. en [35] proponen el aprendizaje residual profundo (*deep residual learning*) que define bloques de convolución y bloques de mapeos de identidad[36] que implementan conexiones de salto (*skip*), también conocidas como *shortcut connections* o conecciones residuales. Este esquema permite entrenar redes aún más profundas, e.g., ResNet-50 está conformada por un total de 152 capas. Al mismo tiempo, ResNet-50 también implementa la normalización de lotes(*batch normalization*)[37] que normaliza la distribución de entradas en cada capa, facilitando así el entrenamiento de DNNs.

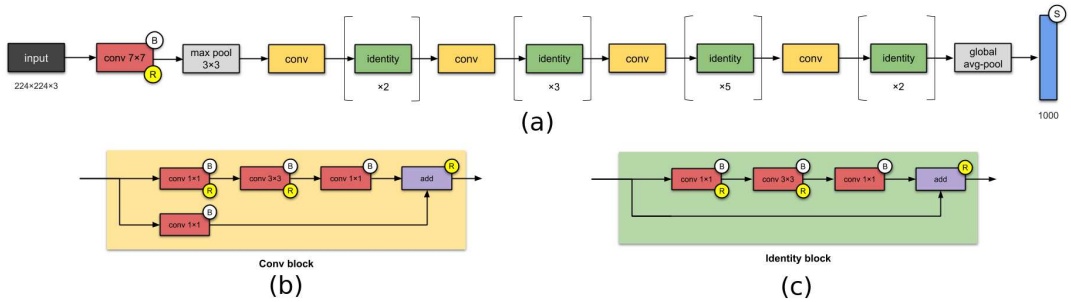


FIGURA 3.8: ResNet-50: (a) Arquitectura global. (b) Bloque convolucional. (c) Bloque de identidad. Fuente: [32, 35].

3.3.4. Inception-ResNet-v2

La implementación de los *skip connections* mejora considerablemente el entrenamiento de las redes neuronales profundas y su proceso de aprendizaje. Por consiguiente, la arquitectura Inception-v3 y ResNet-50 fueron combinadas[38]. En comparación a Inception-v3, la arquitectura de la Inception-ResNet-v2 convierte los módulos de *Inception* en bloques de *Residual Inception*. Además, la nueva arquitectura agrega más módulos de *Inception*. Por otra parte, el módulo *Inception-A* fue agregado después del módulo *Stem*.Finalmente, Inception-ResNet-v2 es una arquitectura de 56 millones de parámetros.

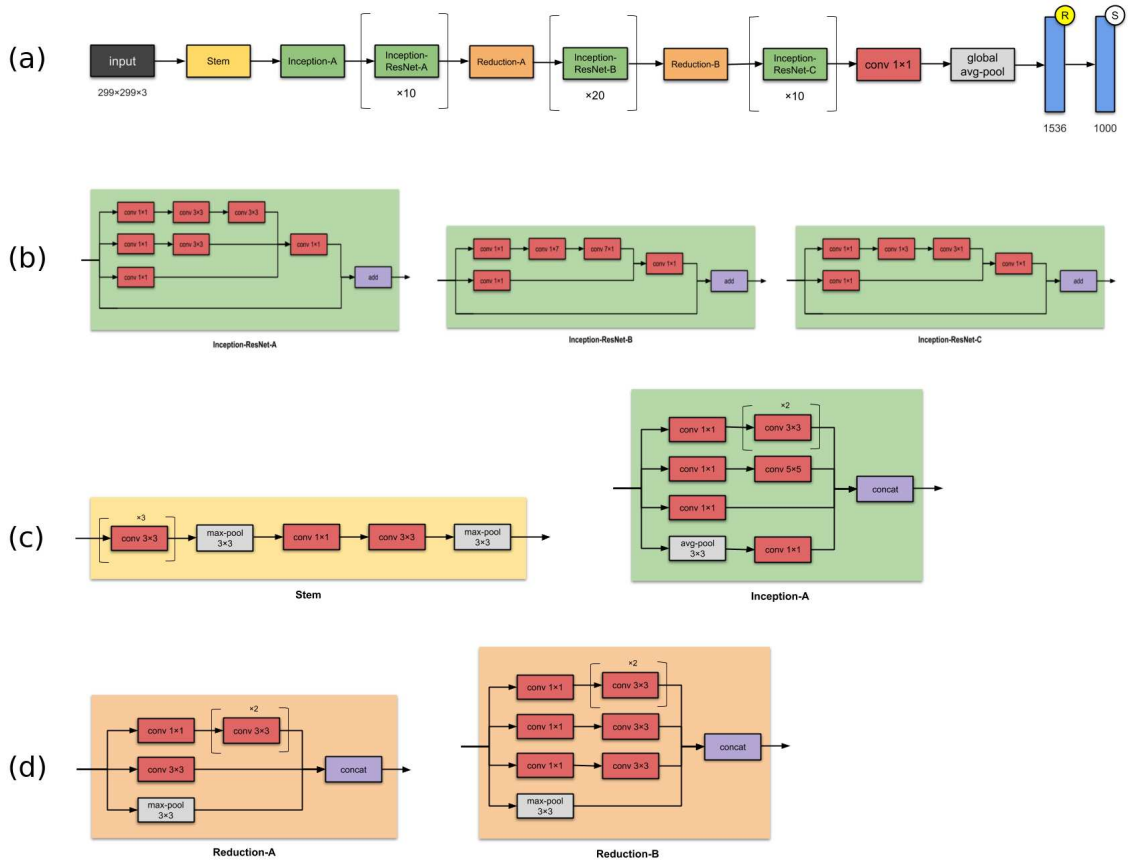


FIGURA 3.9: Inception-ResNet-v2: (a) Arquitectura global. (b) Modulo stem. (c) Modulo *Stem* y modulo *Inception-A*. (a) Modulos *Reduction-A* y *Reduction-B*. Fuente: [32, 38].

Todas las arquitecturas acá presentadas fueron entrenadas con Imagenet[26] y sus pesos estadísticos fueron liberados bajo licencia *Creative Commons*. Es posible cargar dichos pesos y por un procedimiento conocido como *transfer learning* volver a entrenar parcialmente la DNN y así realizar una nueva tarea predictiva.

3.4. Transfer learning

En esencia, el *transfer learning* es el proceso por el cual se toma el aprendizaje de una tarea y se ajusta en una tarea similar. Cuanto más similares son las dos tareas, más fácil es adaptar el aprendizaje de una tarea a otra. Para ello se comienza por obtener los pesos estadísticos de un modelo previamente entrenado, e.g., las arquitecturas anteriormente descritas y así reutilizar la red rápidamente. Posteriormente, si el objetivo es transferir el conocimiento de un modelo a otro, es necesario reutilizar más capas genéricas (las capas más cercanas a los *inputs* de la red) y menos capas específicas de la tarea (las capas más cercanas a los *outputs*). En otras palabras, se deben eliminar las últimas capas (generalmente las capas *fully-connected*) para que la CNN utilice las más genéricas y de esta manera, agregar capas que estén orientadas a la tarea de clasificación específica que se tiene como objetivo. Una vez que comienza el reentrenamiento, las capas genéricas (que conforman la mayoría del modelo) se mantienen congeladas (i.e., no se modifican), mientras que las capas de tareas específicas se modifican[39, 40].

Por lo general, en el *transfer learning* se agregan tan solo dos o tres capas *fully-connected* después de las capas genéricas para crear un nuevo modelo clasificador. No obstante, si el objetivo es alcanzar una mayor precisión, es necesario entrenar más capas específicas para la tarea de clasificación que se desea obtener, i.e., descongelar algunas de las capas finales de la arquitectura que de otro modo se habrían congelado en el *transfer learning*. A este procedimiento se le conoce como ajuste fino (*fine tuning*)[39, 40]. La figura 3.10 muestra el procedimiento de *fine tuning* sobre una red neuronal convolucional.

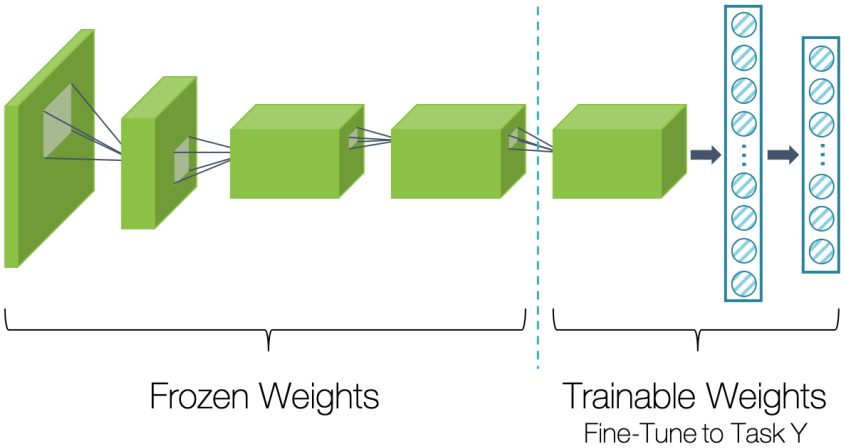


FIGURA 3.10: *Fine tuning* de una red neuronal convolucional pre-entrenada. Fuente: [40].

Por lo tanto, debido a que se adaptan un mayor número de capas a la nueva tarea de clasificación en comparación con el *transfer learning*, el *fine tuning* puede lograr mayor

	Alta similitud entre conjuntos de datos	**Baja similitud entre conjuntos de datos**
Gran cantidad de datos de entrenamiento	Afinar todas las capas	Entrenar desde cero o afinar todas las capas
Pequeña cantidad de datos de entrenamiento	Afinar las últimas capas	Entrenar en una red más pequeña con *data augmentation* o recolectar más datos

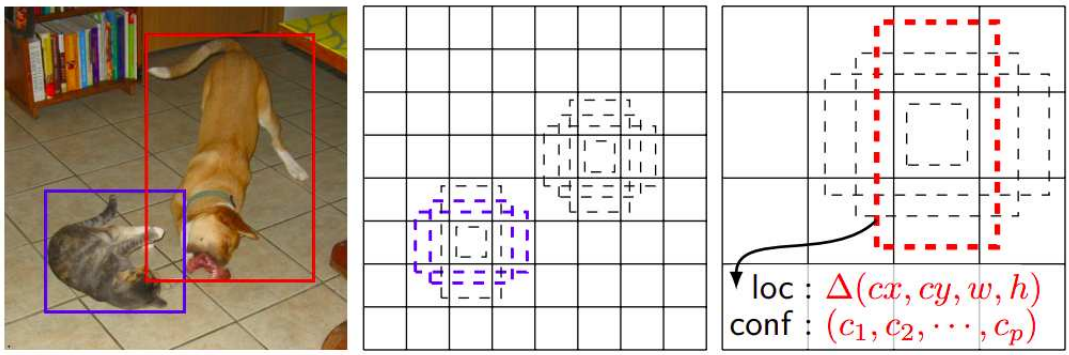
TABLA 3.1: Cuando y como utilizar *fine tuning*[40].

precisión. Sin embargo, determinar cuántas capas ajustar depende de la cantidad de datos disponibles, así como de la similitud de la tarea objetivo con el conjunto de datos original en el que se entrenó el modelo previamente entrenado[39, 40]. La tabla 3.1 describe los diferentes escenarios así como sus correspondientes acciones para implementar un *fine tuning* eficientemente.

El *transfer learning* y el *fine tuning* son técnicas tratadas indistintamente en la comunidad del *deep learning* pero con diferencias sutiles en sus implementaciones y ampliamente utilizadas en tareas de clasificación de imagenes de grano fino (*fine-grained classification*), segmentación semantica (*semantic segmentation*), *image captioning*, detección de objetos (*object detection*), entre otros.

3.5. Detectores de disparo único

En la escena del *deep learning*, el *transfer learning* es parte fundamental de la aparición de nuevas técnicas que robustecen las tareas predictivas en el campo del *computer vision*. Un caso particular del beneficio de esta estrategia es el *object detection*. Esta vez, no solo basta con identificar a qué clase pertenece una muestra, si no también, predecir su posición dentro de la imagen. Las propuestas previas a los detectores de disparo único dividían la solución en dos problemas: una red para realizar la detección de objetos y un clasificador para determinar la clase del objeto. Una de las falencias más importantes es el enorme esfuerzo computacional que conlleva entrenar y desplegar este tipo de sistemas, especialmente, si en la discusión se encuentran las aplicaciones en tiempo real. La particularidad de las arquitecturas de disparo único es su versatilidad para realizar ambas tareas utilizando únicamente una arquitectura de red neuronal.



(a) Image with GT boxes (b) 8×8 feature map (c) 4×4 feature map

FIGURA 3.11: SSD: Single Shot Detector. Fuente: [41].

Las etiquetas que requieren este tipo de detectores para entrenarse son más complejas en comparación a las utilizadas en una arquitectura de clasificación de imágenes. En la figura 3.11.(a) puede observarse cómo en una escena (imagen) pueden coexistir dos clases estadísticamente independientes con sus coordenadas bien definidas por sus respectivas cajas delimitadoras(*bounding boxes*). Así, para una muestra x_i existe una etiqueta y_i que tiene la forma:

$$y_i = (l_j, c_x, c_y, w, h), \tag{3.6}$$

donde l_j es la clase a la que pertenece la muestra x_i; c_x y c_y son las coordenadas del centro de la caja delimitadora en el eje horizontal y vertical respectivamente; w y h son el ancho y el alto de la caja delimitadora, respectivamente.

Los detectores de disparo único como el SSD (*Single Shot Detector*)[41] o el YOLO(*You Only Look Once*)[42], en términos generales están compuestos por una red pre-entrenada con un gran conjunto de datos que sirve como extractor de características y capas convoluciones auxiliares que permiten reducir[41] o aumentar[42] el espacio de búsqueda. Estas DNNs combinan en su predicción una tarea de regresión para la localización de la caja delimitadora y una tarea de clasificación para predecir la clase del objeto. Por consiguiente, se define la función de pérdida

$$L_{total} = \frac{1}{N} \left(L_{\text{conf}} + \alpha L_{\text{loc}} \right), \tag{3.7}$$

donde L_{conf} es la perdida de la clasificacion e.g. *categorical crossentropy* y L_{loc} es una perdida utilizada en problemas de regresión e.g. *Normal* L_1.

Al igual que muchas otras tareas de *computer vision*, la detección de objetos se enfrentan a los latentes problemas del *deep learning*, como lo son la alta demanda computacional y la necesidad de grandes conjuntos de datos. Para este último se han propuesto sendas alternativas que mitigan parcialmente dicha necesidad, en donde el término *data augmentation* agrupa este conjunto de técnicas que permiten mejores resultados en la predicción.

3.6. Data augmentation

Una de las brechas que más afecta los modelos basados en *deep learning* es la cantidad de datos necesarios para lograr obtener buenos resultados. Más concretamente, las redes neuronales convolucionales requieren de estos grandes bancos de imágenes[26] para lograr capturar las representaciones necesarias plasmadas en la fase de entrenamiento(*feacture extraction*). Es por ello que aumentar los datos existentes a partir de transformaciones que aumentan la varianza del conjunto, es el objetivo del *data augmentation* y brinda una solución que evita el *overfiting* y aumenta la precisión de los modelos[7, 43, 44, 45]. El diagrama de la figura 3.12 desglosa la taxonomía de las diferentes técnicas de *data augmentation*, en donde la manipulación básica de imágenes también conocidas como transformaciones genéricas son ampliamente utilizadas para tareas de clasificación de imágenes general.

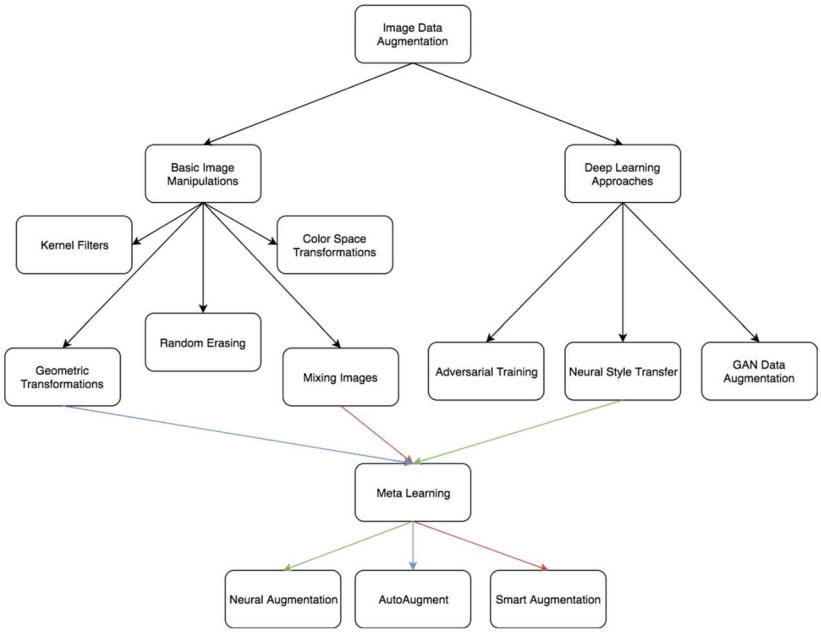


FIGURA 3.12: Tipos de aumento de datos. Fuente: [7].

Por lo general, los métodos genéricos transforman la imagen completa e ignoran los contenidos de alto nivel, mientras que los métodos específicos de la cara se centran en los componentes o atributos de los rostros y son capaces de transformar atributos como la edad, el maquillaje, el peinado, el uso de accesorios, etc. La figura 3.13 muestra una descripción general de las transformaciones de datos faciales de uso más común[8]. Por su parte, los enfoques de *data augmentation* basados en *deep learning* como las GANs[10, 11] u otros modelos generativos como los VAEs(*Variational Autoencoders*)[12] son mucho más robustos, pues incorporan dentro de sus esquemas diversas tareas específicas dentro del *data augmentation*, pero al mismo tiempo son mucho más demandantes en cuanto a su implementación y ejecución. No obstante, ofrecen excelentes resultados para tareas de clasificación de grano fino, como es el caso del reconocimiento facial(*face recognition*)[8].

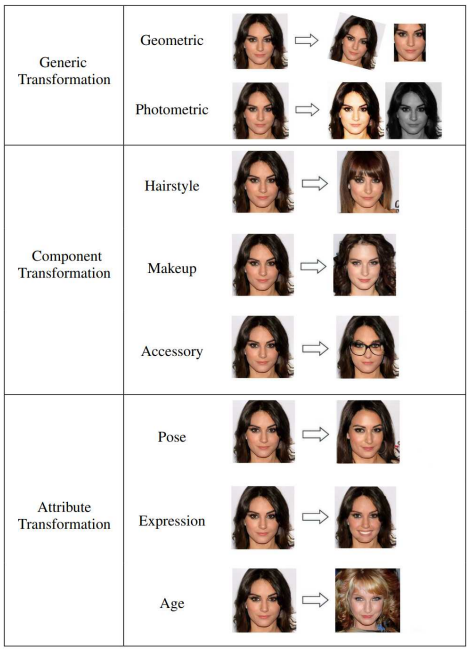


FIGURA 3.13: Descripción general de algunas transformaciones genericas y especificas para *face recognition*. Fuente: [8].

3.7. Autoencoder Variacional

Los modelos generativos son un campo ampliamente utilizado en el *deep learning* y dentro de sus aplicaciones más atractivas se encuentran diversas tareas de *data augmentation*. Una división importante entre los algoritmos de *machine learning* se da al considerar las diferencias entre los modelos generativos y los discriminativos. Estos últimos tienen como objetivo la separación o discriminación entre diferentes tipos de datos, mientras que los modelos generativos tienen como foco el aprendizaje de la distribución conjunta de un grupo de datos, es decir, dado un conjunto de datos X con etiquetas y:

- **Modelo discriminativo:** El modelo captura la probabilidad condicional $p(Y|X)$.
- **Modelo generativo:** Captura la probabilidad conjunta $p(X, Y)$. En caso de no existir etiquetas, el modelo aprende una representación de $p(X)$.

En los modelos generativos, se asume que la variable observada X es una muestra aleatoria de un proceso desconocido, cuya distribución de probabilidad verdadera $p(X)$ es desconocida. Los modelos generativos tienen como objetivo el aproximar la función desconocida mediante un modelo paramétrico $p_\theta(X)$, tal que para cualquier X, se cumpla que:

$$p_\theta(X) \approx p(X). \tag{3.8}$$

El conjunto de parámetros θ, se pueden calcular mediante técnicas de aprendizaje automático [46].

En los últimos años, los modelos generativos basados en aprendizaje profundo han ganado un amplio reconocimiento debido a sus prometedoras aplicaciones en áreas tan diversas

como la astronomía, física de partículas [47, 48], la producción de piezas de contenido altamente realistas (imágenes, textos y sonidos) [49] y la reconstrucción 3D de objetos [50]. Dentro de los modelos generativos, podemos identificar dos tipos principales: Las *GAN's* [51] y los modelos basados en *likelihood*, como los modelos autoregresivos [52], y los *VAE* [46], técnica usada en este trabajo.

Los métodos basados en el *likelihood* optimizan el NLL (*negative log-likelihood*) de los datos de entrenamiento. Dado que la distribución de probabilidad $p(X)$ que el modelo asigna a todos los conjuntos de entrenamiento se maximiza, se mitiga el problema de la falta de diversidad observado en las GAN's [53].

Para obtener una representación de la probabilidad conjunta $p(X)$, los autoencoders variacionales se aprovechan de la capacidad que poseen los autoencoders[54] para encontrar representaciones eficientes y de menor tamaño de un grupo de datos específicos. En ese sentido, la arquitectura de un autoencoder se compone de dos partes:

- **Encoder:** Convierte el conjunto de entrada en una representación reducida denominada espacio latente. Las variables latentes z, son variables que forman parte del modelo, siendo estas una representación del conjunto de datos de entrada.
- **Decoder:** Reconstruye los datos de entrada con base en la representación latente.

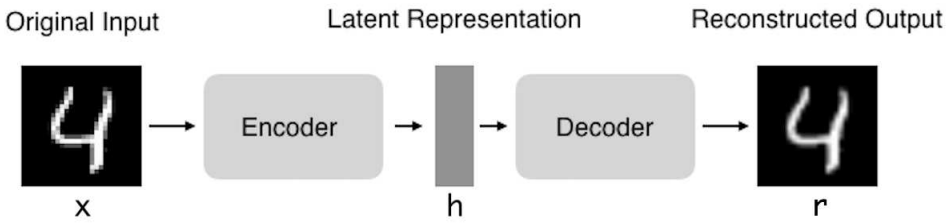


FIGURA 3.14: Estructura de un autoencoder. Fuente: Ref [41].

La rama izquierda en la figura 3.14, representa el *encoder*, el cual puede ser representado por una función $g_\phi()$, dónde ϕ es un grupo de parámetros. La rama derecha corresponde al *decoder*, representado por una función $f_\theta()$. Los parámetros ϕ y θ son hallados durante el entrenamiento de la red, siendo los valores que permiten al autoencoder, el ser capaz de reconstruir los datos de entrada. Dicho de otra forma, el autoencoder aprende la función identidad $X \approx f_\theta(g_\phi(x))$. Para cuantificar la diferencia entre los datos de entrada y los datos reconstruidos, se puede usar la función de perdida Mean Squared Error (MSE), o cualquier otra que permita cuantificar la diferencia entre dos vectores:

$$L(\phi, \theta) = \frac{1}{n}\sum_{i=1}^{n}(\mathbf{x}^i - f_\theta(g_\phi(x)))^2. \tag{3.9}$$

Por otra parte, los VAE constituyen una generalización de los autoencoders tradicionales. La idea central de dicha generalización se encuentra profundamente relacionada con los métodos de inferencia Bayesiana [55]: En lugar de representar los datos de entrada X en un vector latente fijo (ver figura 3.14), se desea representarlos en una distribución p_θ, de tal forma que se pueda establecer una relación entre en dataset de entrenamiento y el vector

latente de la siguiente forma: a) prior $p_\theta(z)$, b) likelihood $p_\theta(x|z)$ y c) posterior $p_\theta(z|x)$. El conjunto de parámetros θ que parametriza las distribuciones anteriores se obtienen mediante el entrenamiento de una red neuronal. Es posible reescribir lo anterior usando el teorema de Bayes [55]:

$$p(x|z) = \frac{p(z|x)p(x)}{p(z)}. \tag{3.10}$$

Por tal motivo, se puede interpretar el espacio latente desde el marco de la estadística Bayesiana: este representa las creencias previas asociadas a las variables observadas [55]. Por ejemplo, si se asume que x corresponde a una distribución gaussiana multivariada, la variable oculta z podría representar la media y la varianza de la distribución gaussiana. En ese orden de ideas, la distribución sobre los parámetros $P(Z)$ es entonces una distribución previa a $P(X)$.

Para generar una muestra que se asemeje a los datos de entrada reales X^i, se muestrea a partir de $p_\theta(z)$ un elemento del espacio latente z^i. A partir de lo anterior, se calcula un nuevo elemento $x^{'i}$ de la probabilidad condicional $p_\theta(x|z = z^i)$. Los parámetros θ que permiten obtener el nuevo elemento $x^{'i}$ se obtienen durante el entrenamiento de la red [56].

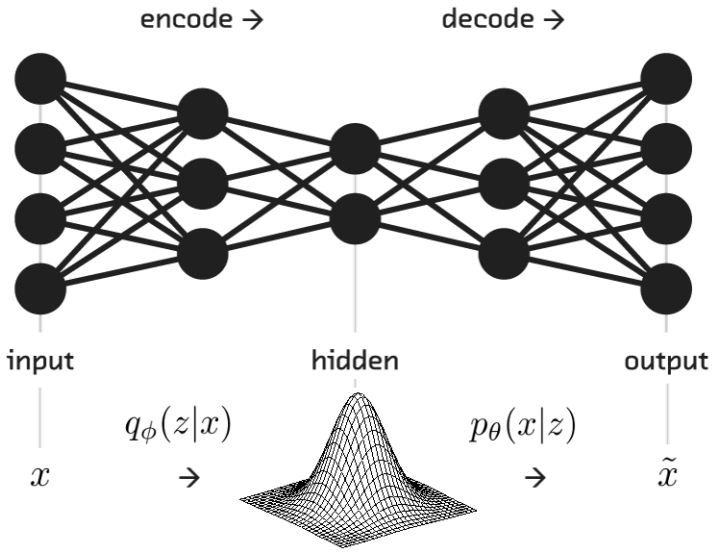


FIGURA 3.15: Variational Autoencoder Fuente: Ref [41].

Sean x los datos que se desean modelar, z la variable latente que se obtiene en la representación comprimida del conjunto de datos y $p(z)$ su correspondiente distribución de probabilidad. $p(X)$ es la distribución de probabilidad de los datos de entrada. Podemos expresar la distribución de probabilidad de los datos dada la variable latente como $p(x|z)$. El objetivo es inferir $p(z)$ a partir de $p(z|x)$, que representa la distribución de probabilidad que proyecta nuestros datos en el espacio latente [56]. La distribución $p(z|x)$ es desconocida, pero puede ser estimada usando el método de inferencia variacional: para hallar la probabilidad a posteriori, se asumirá que la distribución de probabilidad a priori tiene una forma paramétrica conocida sencilla $Q_\phi(z|x)$ (E.j distribución gaussiana). Modificando los

parámetros ϕ, es posible acercar dicha distribución a $p(z|x)$ tanto como sea posible. Para establecer una medida entre la distribución de probabilidad a priori y a posteriori y hallar los valores óptimos de los parámetros ϕ, se hace uso de la divergencia de Kullback-Leibler (KL en adelante) [56] con el fin de minimizar la diferencia entre el prior y el posterior, como se ve en la eq. 3.11:

$$D_{KL}(Q_\phi(z|x)||P(z,x)) = \sum_{z \in Z} q_\phi(z|x) log \frac{q_\phi(z|x)}{p_(z|x)}. \tag{3.11}$$

De lo anterior, es posible hallar la función de pérdida para los VAE [56, 55]:

FIGURA 3.16: Autoencoder variacional: Reconstrucción de los datos de entrada Fuente: [41].

Los VAE se usarán en el presente trabajo para generar rostros de personas, con el fin de obtener más muestras de los rostros de los participantes. Para lograrlo, es necesario crear un encoder que aprenda los parámetros ϕ (media y varianza, en este caso) de la distribución latente subyacente. Se debe generar una función para realizar un muestreo del espacio latente y un decodificador que pueda convertir la muestra obtenida del espacio latente nuevamente en una imagen. Lo anterior se ilustra en la figura 3.17.

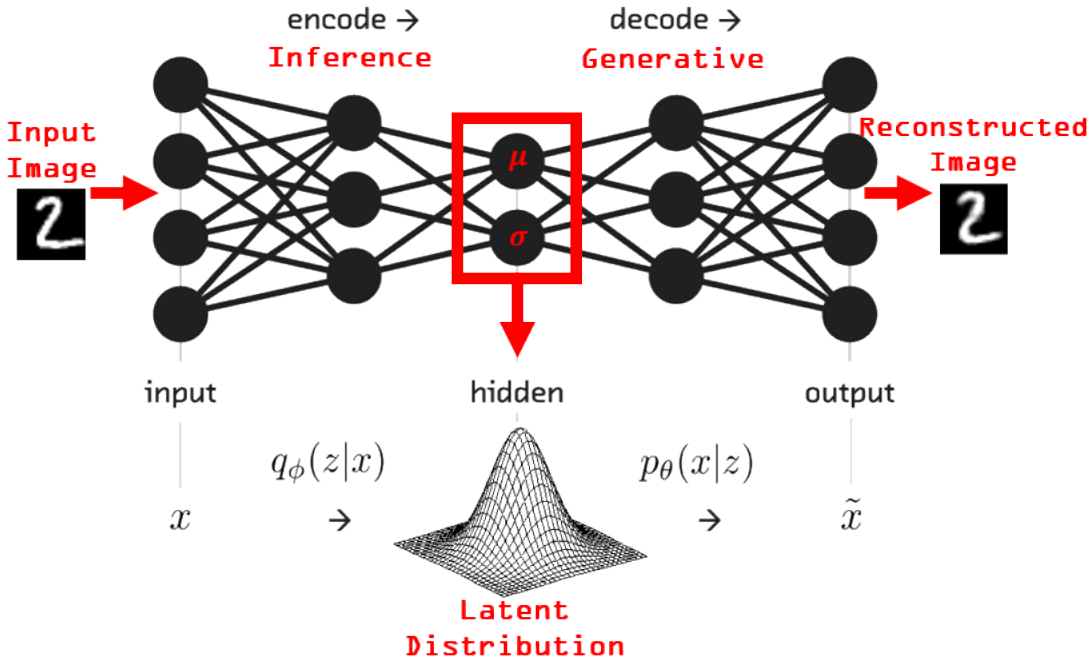


FIGURA 3.17: Autoencoder variacional: Espacio latente y generación de imágenes. Fuente: [41].

En algunas configuraciones, tanto el encoder como el decoder son redes convolucionales, debido a su idoneidad para extraer características relevantes a partir de datos que cuenten con correlaciones espaciales como las imágenes. La arquitectura del encoder [57], encargado de generar la representación latente se puede ver en la figura 3.18

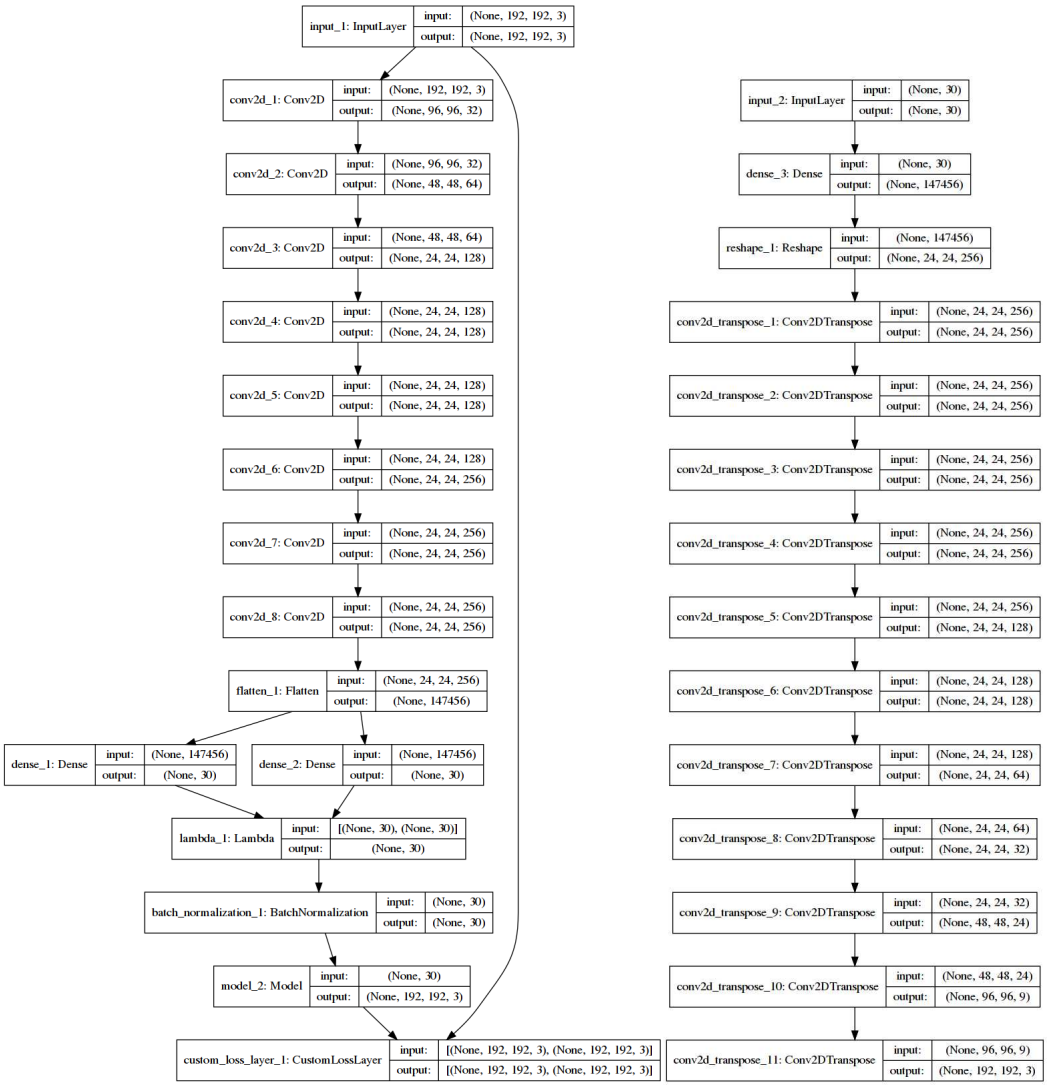


FIGURA 3.18: Arquitecturas convolucionales para el encoder(L.H.S.) y decoder(R.H.S.).

Cabe resaltar que el encoder tiene dos capas densas de salida, que se concatenan usando una función lamda, usada para definir una capa que obtiene como entradas muestras de la distribución definidas por los parámetros z aprendidos por el encoder. Cada una de estas capas densas3.17, representan la media de distribución latente z_μ y su varianza (z_σ).

El decoder se encarga de realizar la reconstrucción de las imágenes de entrada a partir del espacio latente. Por tal motivo, debe reconstruir un tensor del mismo orden que el inicial. Para ello, se usa la función $Conv2DTranspose$ para revertir las convoluciones del encoder. La implementación usada puede consultarse en la figura 3.18. Durante el entrenamiento del autoencoder variacional, es necesario usar dos funciones de pérdida

- Pérdida de reconstrucción: Se usa la entropía cruzada que describe los errores entre las muestras decodificadas de la distribución latente y las entradas originales.
- Divergencia KL: La divergencia Kullback-Liebler se calcula entre la distribución latente y la distriución a priori.

CAPÍTULO 4

Metodología

Para las tareas de clasificación de grano fino se requieren elaborados flujos de trabajo (*workflows*) que transportan la información por diferentes componentes predicativos del sistema y transforman los datos de entrada en características mas robustas que facilitan la tarea del clasificador final. En esta investigación se abordó el problema de clasificación de grano fino de reconocimiento facial, que habitualmente constan de múltiples predictores con funcionalidades especificas tales como: detección de rostros, detección de marcas faciales, localización y alineamiento,identificación, asociación y apareamiento, entre otros, como lo muestran Mei Wang y Weihong Deng en [17].

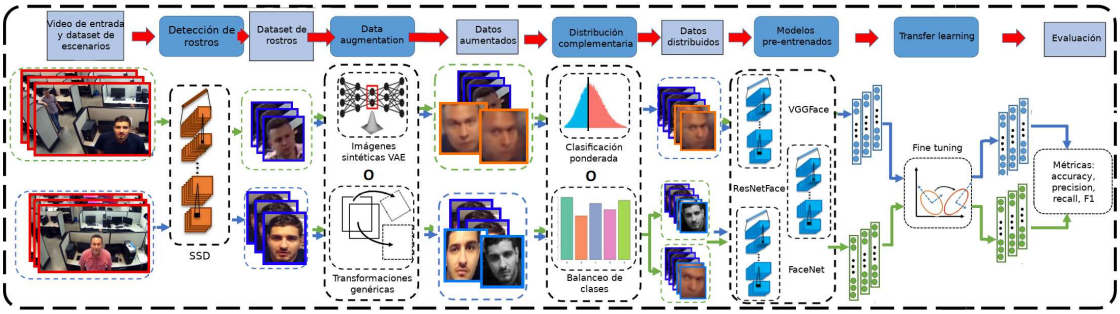


FIGURA 4.1: Flujo de trabajo sobre el sistema de reconocimiento facial implementado.

Para objeto de esta investigación se diseño un *workflow* de trabajo y un *pipeline* de implementación descritos por el esquema de la figura 4.1, en donde el principal objetivo es estudiar los efectos de la estrategia mas común de *data augmentation* conocida como transformaciones genéricas y el aumento de datos con imágenes sintéticas producto del modelo generativo *autoencoder* variacional. Al mismo tiempo se propuso una etapa de distribución de las muestras aumentadas en donde se diseñó un balanceo de las clases minoritarias y al mismo tiempo un esquema rígido de aumento de datos en donde todas las clases son aumentadas en la misma proporción. Los pasos metodológicos de esta investigación para la obtención de un algoritmo de reconocimiento facial se definen y describen a continuación:

1. **Adquisicion y recolección de vídeo.** El registro fílmico se extrajo de un dispositivo de seguridad ubicado en el grupo de investigación *in2lab* del Departamento de Ingeniería de Sistemas, adscrito a la Universidad de Antioquia, en Medellín (Co-

lombia). El equipo de grabación incorpora software del fabricante para la correcta extracción de fragmentos de vídeo guardados en un servidor local. El sistema cuenta con dos métodos de grabación, uno basada en la detección por movimiento y el otro es un modo de grabación continua, ambos tratados de manera indistinta por el software. El criterio de extracción principal fue encontrar los fragmentos de vídeo donde los sujetos se encontraban de frente a la cámara de seguridad en cualquier posición. También se solicitó a los sujetos caminar por los espacios que cubrían el ángulo de la cámara haciendo diferente tipo de gestos.

2. **Separación en fotogramas y definición del *dataset* de escenarios.** Por cada vídeo, extraemos sus *frames* a una tasa de captura especifica. Esto ultimo con el fin de tener el mayor número de muestras posibles y que a la vez, si la tasa es muy baja (i.e., se toman muchas imágenes), es bastante probable que para intervalos de tiempo muy cortos no se perciba diferencia entre las imágenes extraídas.

3. **Detección y extracción del *dataset* de rostros.** Cada *frame* extraído pasa por un detector de rostros que retorna la ubicación de cuatro puntos en la imagen y posterioremente, definen el *bounding box* del rostro. Se utilizo un detector de disparo único (*SSD: Single Shot Detectors*) para la detección y consecuentemente, se conformo el *dataset* de rostros.

4. **Estrategias de aumento de datos.** Se fijo un porcentaje de aumento de datos del 50 % para cada clase, en comparación al tamaño del dataset original. Subsecuentemente, se utilizaron las técnicas de *data augmentation* por transformaciones genéricas en las que se aplicaron quince diferentes operadores de transformación sobre las imágenes originales y el aumento de datos por generación de imágenes sintéticas, utilizando un *autoencoder* variacional.

5. **Distribución de muestras aumentadas.** Una vez los conjuntos de datos aumentados se obtuvieron, se sobremuestreo(*oversampling*) el dataset bajo dos enfoques: balanceo de clases(BC) y clasificación ponderada (CP). El primero de estos, propuesto en el presente trabajo fue reducir la brecha o *gap* entre las clases minoritarias y la clase mayoritaria. Para ello, se agregan a la clase mayoritaria el 5 % del 50 % total de datos aumentados y el 45 % restante se distribuye de manera variable a las clases minoritarias, bajo la ligadura que entre estas clases no existiera una diferencia mayor al 20 % respecto al tamaño final alcanzado por la clases mayoritaria, obteniendo clases más balanceadas de forma manual. La segunda estrategia de distribución de datos aumentados consistió en agregar a cada clase el 50 % de sus muestras y usar para esta distribución el esquema de clasificación ponderada que asigna un mayor peso estadístico en el *threshhold* de decisión a las clases que tiene una menor cantidad de muestras.

6. **Selección de arquitecturas de redes neuronales.** Para los experimentos se seleccionaron tres modelos pre-entrenados: (a) VGGFace que utiliza la arquitectura propuesta por la VGG16, (b) ResNetFace que implementa la arquitectura de ResNet50 y (c) la FaceNet que hace uso de la robusta arquitectura Inception. Esta selección se fundamentó en utilizar modelos pre-entrenados con la mayor similaridad posible con la tarea de reconocimiento facial a efectuar.

7. ***Transfer learning* y *fine tuning*.** Para este procedimiento, se descongeló la ultimo bloque convolucional para cada una de las tres arquitecturas seleccionadas para el re-

calculo de los pesos estadísticos. Adicionalmente, se configuró un clasificador con una capa *fully-connected* de 128 neuronas conectadas directamente a una capa $Softmax$.

8. **Entrenamiento y escenarios de pruebas.** Para el reentrenamiento parcial de las arquitecturas se utilizaron cuatro *datasets* de rostros, definidos a partir de todas las posibles combinaciones de las dos estrategias de *data augmentation* y de distribución de datos, i.e., transformaciones genéricas y balanceo de clases (TG + BC) con 1676 muestras, transformaciones genéricas y clasificación ponderada (TG + CP) con 2234 muestras, imáganes sintéticas y balanceo de clases (AS + BC) con 1676 muestras y finalmente, imágenes sintéticas y clasificación ponderada (AS + CP) tambien con 2234 muestras. Al mismo tiempo, como línea base se entrenaron las tres arquitecturas sobre el conjunto de datos crudos (1117 muestras), es decir, sin aplicar ninguna técnica de *data augmentation* y ningún enfoque de distribución. Para el entrenamiento se utilizaron las tasas 80 % y 20 %, para el conjunto de entrenamiento y el conjunto de validación, respectivamente. Como función de perdida se uso la *categorical cross-entropy* y como optimizador el *Adam Optimizer* con su configuración sugerida por defecto. Finalmente, se configuro la tecnica *early stopping* con el *accuracy* del conjunto de validación como metrica de monitoreo, con parametro de paciencia igual a veinte epocas y con criterio de minima variación fijado en 0.0001 para el *accuracy* del conjunto de validación.

9. **Evaluación.** Despues de la etapa de entrenamiento, los modelos fueron exportados en HDF5 y posteriormente evaluados bajo un conjunto de *test* de 240 muestras, nunca antes vistas por el modelo. Se reportarón las metricas de *accuracy*, *precision*, *recall* y $F_1-Score$, por ser las más relevantes para el problema.

Los experimentos fueron desarrollados en Python 3.7.6 usando los *frameworks* TensorFlow 1.14, Keras 2.0 y scikit-learn 0.22.1. La máquina que procesó los experimentos contaba con una CPU 8-core, 30GB de memoria RAM DDR4, un SSD de 40GB y una Nvidia Tesla V100, desplegada en la nube en GCP(*Google Cloud Platform*).

CAPÍTULO 5

Resultados

5.1. Analísis probabilistico de las técnicas de *data augmentation*

Las figura 5.1 contienen ejemplos de rostros generados por el *random zoom* y el *auto-encoder* variacional (VAE) para el mismo sujeto. Allí, el efecto de las transformaciones es evidente, especificamente, el *random zoom* genera dilataciones y contracciones en regiones aleatorias de la imagen mientras que el VAE puede enterderse como la “gaussianización"de la imagen alrededor del máximo de la intensidad. Para entender el efecto en términos cuantitativos es útil hacer uso del concepto de probabilidad conjunta, que dados dos eventos aleatorios X y Y, permite establecer la probabilidad de que estos ocurran de forma simultánea. Para una misma imagen del sujeto de la figura 5.1, además de presentar la imagen original y sus versiones transformadas (panel a mano izquerda), la figura 5.2 presenta las probabilidades conjutas y sus respectivas probabilades marginles (panel central) y las probabilades condicionales (panel a mano derecha) de cada versión de la imagen.

En relación a las probabilidades conjuntas de la imagen generada por el *random zoom*, para la imagen escogida para la discusión en la figura 5.2, se observa una redistribución de la probabilidad hacia los valores más grandes de las escalas horizontales y verticales de la imagen, esta redistribución se confirma en las probabilidades marginales. La redistribución, aunque hacia diferentes regiones de la imagen, se confirmó para todas las imágnes de todos los sujetos del estudio.

Para las imagénes generadas a través del VAE, la probabilidad tiene a concentrarse hacia las regiones de más alta probabilidad en la imagen original y hacia valores pequeños de las escalas horizontales y verticales de la imagen, al igual que en el caso anterior, este comportamiento se confirma en las probabilides marginales y para todas las imágenes de todos los sujetos en el estudio.

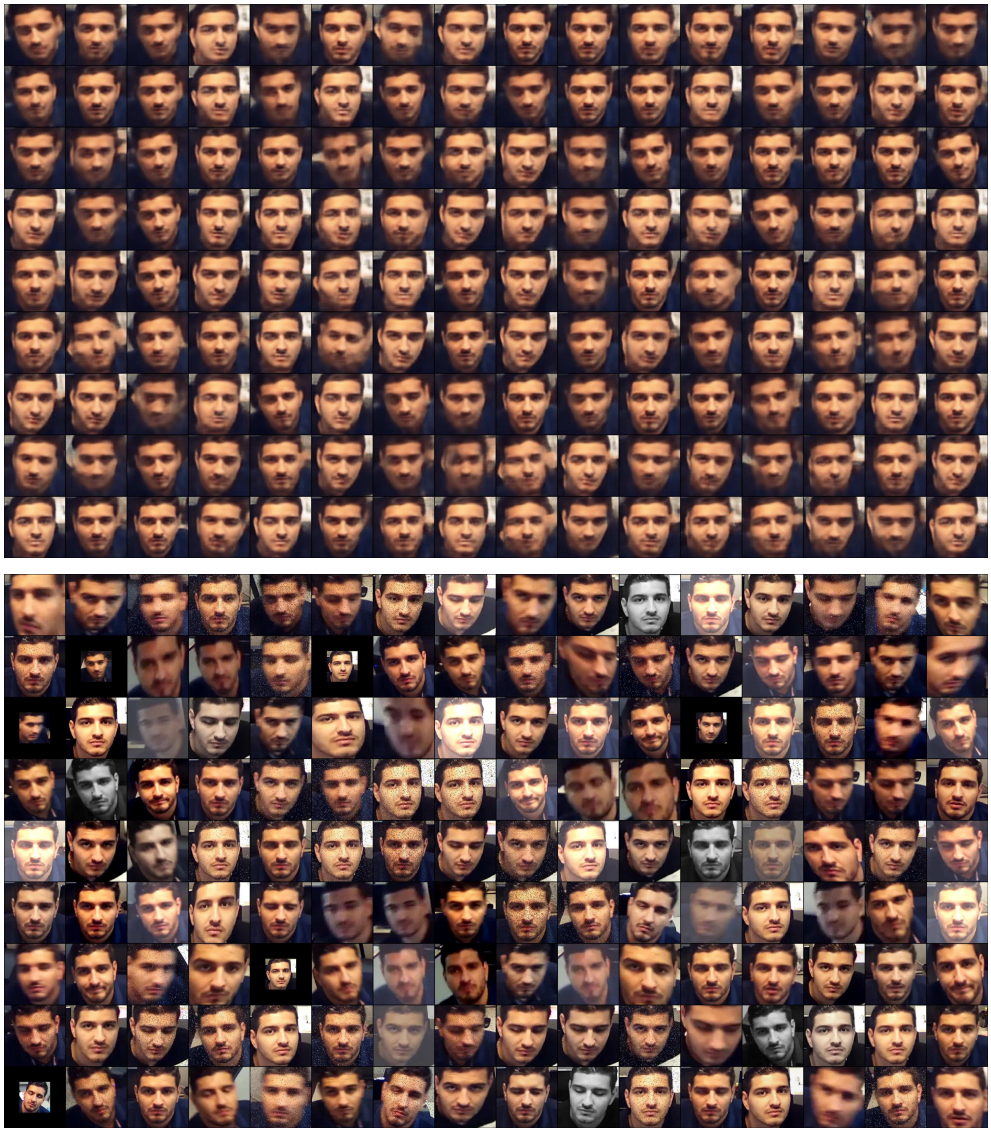


FIGURA 5.1: Rostros generados con el *random zoom* (panel superior) y con el *autoencoder* variacional (panel inferior) para uno de los sujetos (clases) del *dataset* de entrenamiento.

La discusión presentada arriba se confirma al analizar las probabilidades condicionales para tres cortes transversales de las imágnes: líneas rojas (píxel número 191 en la escala vertical), azules (píxel 117 número en la escala vertical) y cyan (píxel 63 en la escala vertical), ver paneles central y a mano derecha de la figura 5.2. Para la imagen generada por el *random zoom*, se observa una mayor concentración de la probabilidad en $p(x|191)$, regiones altas en la escala vertical, que para $p(x|127)$ y $p(x|63)$ (regiones bajas en la escala vertical). Para la imagen generada por el VAE se observa que el área bajo la curva en los tres casos, $p(x|191)$, $p(x|127)$ y $p(x|63)$ la probabilidad aumenta a medida que se acerca hacia la región de valores bajos de la escala tanto horizontal como vertical.

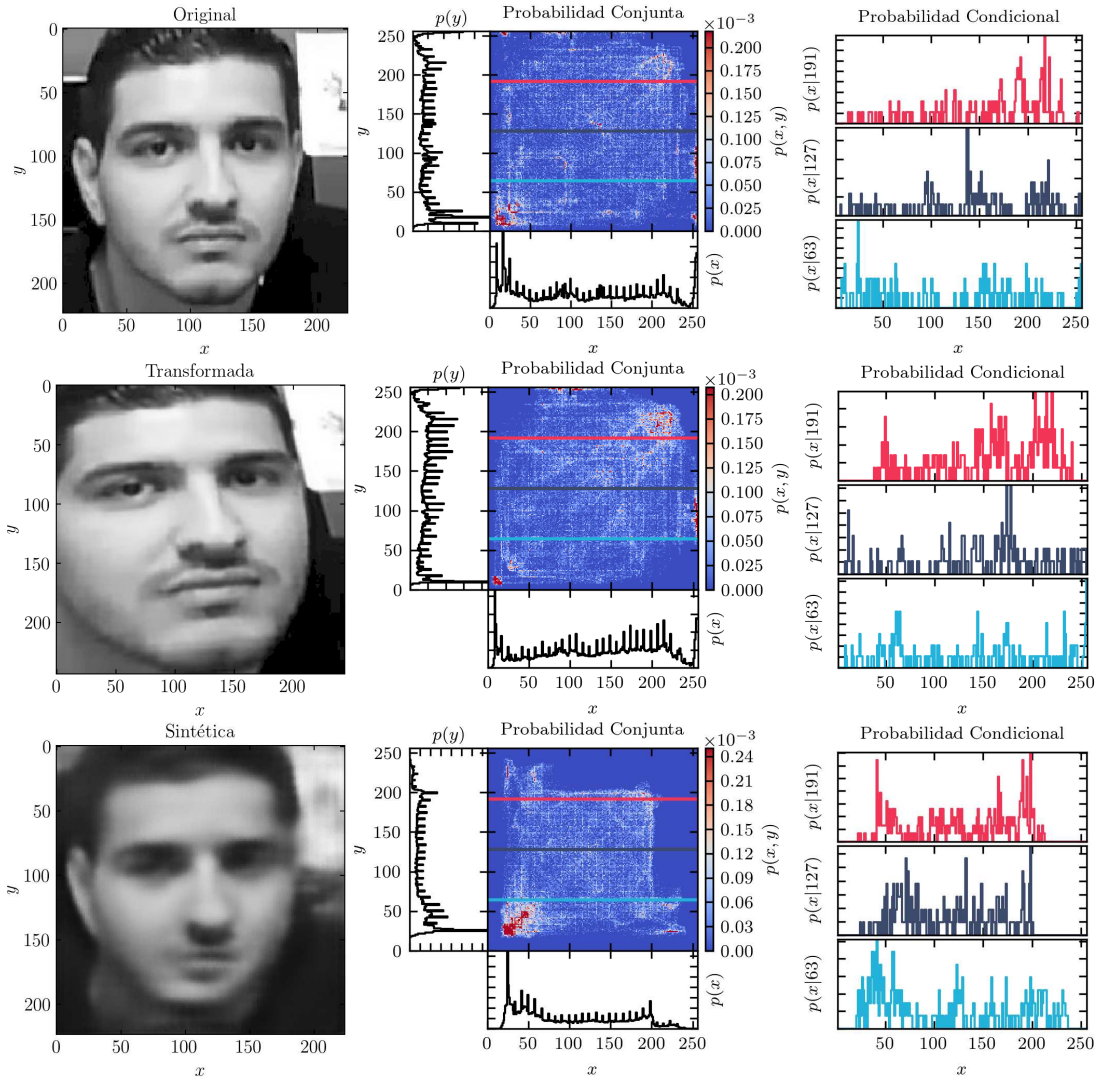
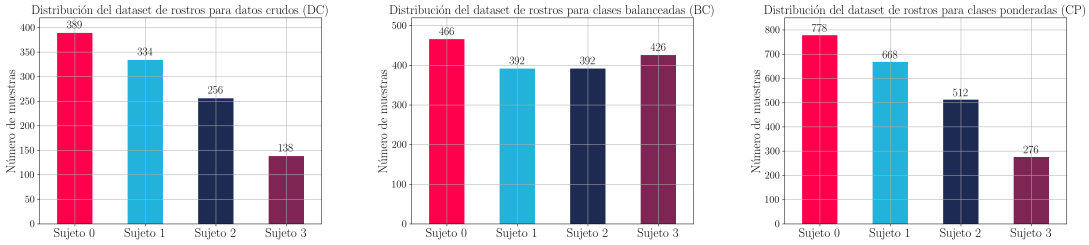


FIGURA 5.2: Analísis probabilístico de una (i) imagen original (panel superior), (ii) imagen bajo la transformacion "*random zoom*" (panel central) y de una imagen sintética generada con VAE (panel inferior).

5.2. Clasificación para reconocimiento facial

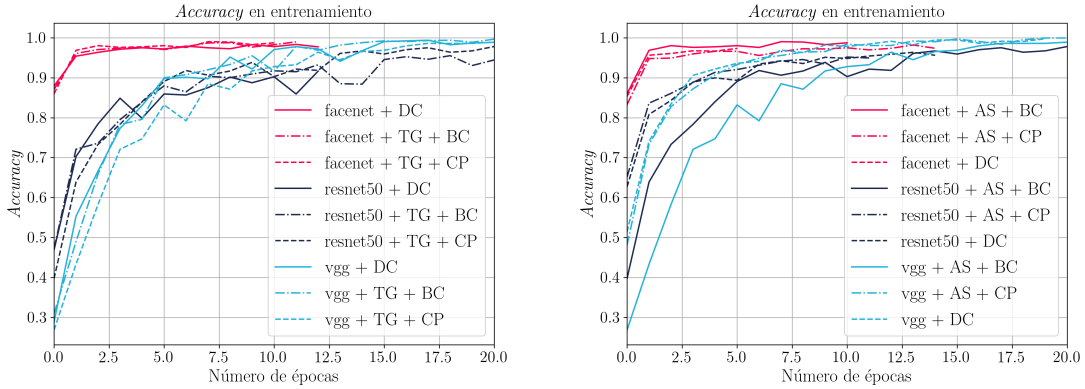
Despues de la ejecución del *workflow* descrito en la sección 4 (ver figura 4.1), se obtuvieron tres tipos de *datasets* de rostros: (a) datos crudos, (b) balanceo de clases y (c) clasificación ponderada. Las distribuciones de dichos conjuntos de datos generados en esta investigación, son mostradas en la figura 5.3.

El estado del arte define la métrica *accuracy* como un estándar para las tareas de reconocimiento facial en grandes volúmenes de datos. Por esta razón se monitoreó dicha métrica en los conjuntos de entrenamiento, obteniendo los mejores resultados para la arquitectura FaceNet en todos los escenarios posibles para las combinaciones de las estrategias de *data augmentation* y de distribución de datos, como se evidencia en la figura 5.4. De la misma forma, el *accuracy* en el conjunto de validación es mostrado en la figura 5.5.



(A) Datos crudos (B) Balanceo de clases (BC) (C) Clasifi. ponderada (CP)

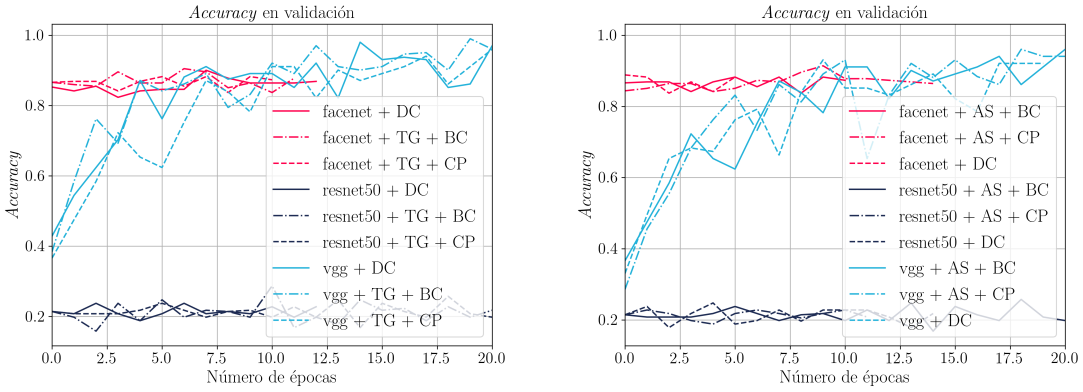
FIGURA 5.3: Distribución de imágenes para los diferentes conjuntos de datos.



(A) Transformaciones genéricas (TG). (B) Imagenes sintéticas (AS).

FIGURA 5.4: Comparación del *accuracy* en entrenamiento entre los conjuntos de aumentados (TG y AS) y el conjunto de datos crudos (DC), para los métodos de balance de clases (BC) y clasificación ponderada (CP).

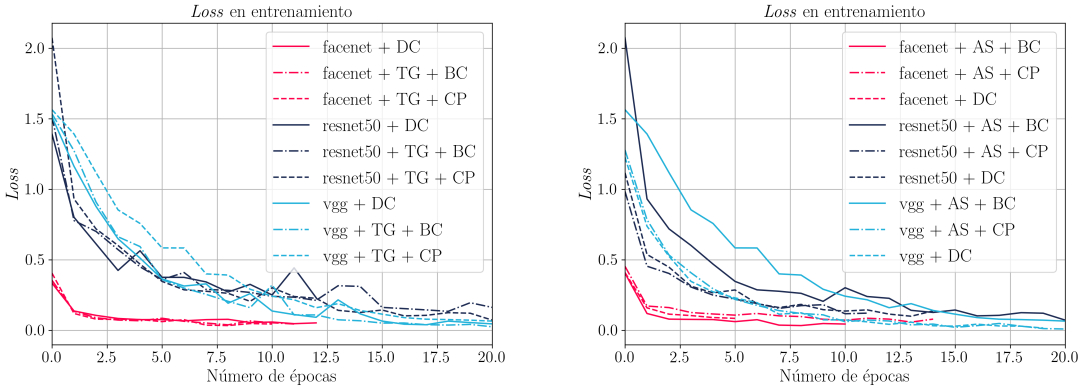
En este punto ya fueron notorias las diferencias en la precisión de las arquitecturas despues de culminado el proceso de reentrenamiento. La figura 5.5 muestra un pésimo rendimiento para la arquitectura ResNetFace a pesar de haber tenido un rendimiento aceptable en el conjunto de entrenamiento. Esto se debe a la combinación de las conexiones residuales de la ResNet50 y el uso de la *cross-entropy* para la clasificación de los *feature embeddings*, como lo exponen Zhang y Sabuncu en [58]. La función de perdida *cross-entropy* se sesga a la distribución de las muestras de entrenamiento. A diferencia de las funciones de perdida habitualmente usadas en el estado del arte como lo son la *contrasstive loss*[59] y la *tripletloss*[60] que atienden específicamente a muestras duras, la pérdida *cross-entropy* maximiza la probabilidad condicional de todas las muestras en un *batch* dado. Por lo tanto, se adapta bien a los rostros de buena calidad, ignorando las caras difíciles o raras que pueden estar en un *batch* de validación. Este hecho repercute con mayor fuerza en la ResNetFace por el enfoque de *residual learning* que define a la ResNet50.



(A) Transformaciones genéricas (TG). (B) Imagenes sintéticas (AS).

FIGURA 5.5: Comparación del *accuracy* en el conjunto de validación.

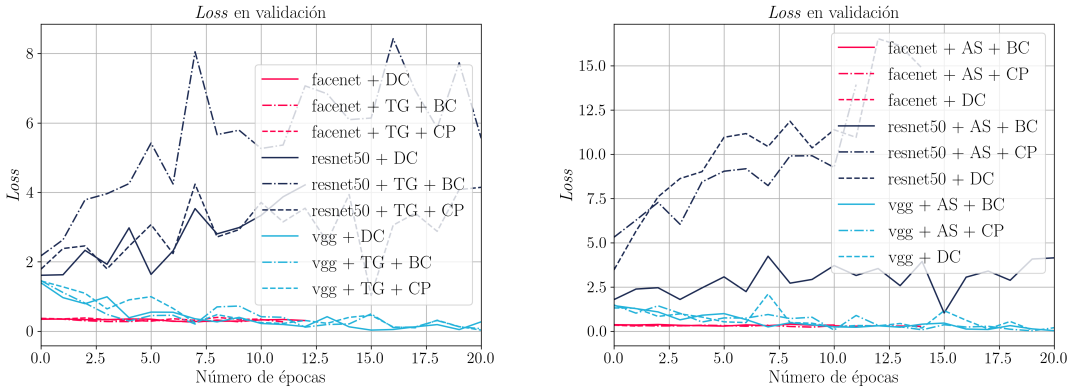
Despues de monitorear la perdida *cross-entropy* en el proceso de entrenamiento, las figura 5.6 muestra una clara predominancia de la eficiencia en la convergencia al mínimo "global"para la FaceNet y un comportamiento esperado para la VGGFace y la ResNetFace. No obstante, en el conjunto de validación (ver figura 5.7) esta última nuevamente ratifica el bajo rendimiento detectado en el *accuracy* (ver figura 5.5), por lo que se eliminó dicha arquitectura de la figura 5.8.



(A) Transformaciones genéricas (TG). (B) Imagenes sintéticas (AS).

FIGURA 5.6: Comparación del *accuracy* en entrenamiento entre los conjuntos de aumentados (TG y AS) y el conjunto de datos crudos (DC), para los métodos de balance de clases (BC) y clasificación ponderada (CP).

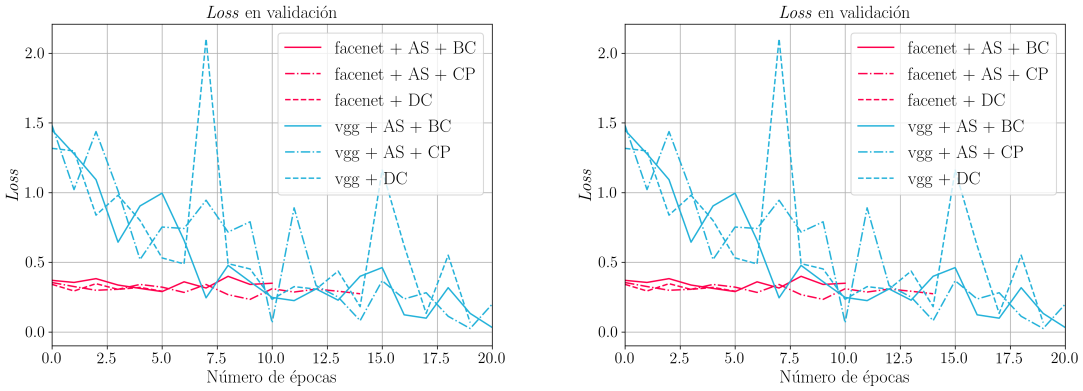
Cabe resaltar la buena convergencia, precisión y rendimiento que obtuvo la arquitectura FaceNet en todos los escenarios propuestos, superando ampliamente a una resagada VGGFace, como se evidencia en la figura 5.8.



(A) Transformaciones genéricas (TG). (B) Imagenes sintéticas (AS).

FIGURA 5.7: Comparación del *accuracy* en el conjunto de validación.

Al analizar la perdida *cross-entropy* y el *accuracy*, se corrobora que la *cross-entropy* no es la mejor elección para problemas de clasificación *fine-grained*, como ya lo desmotrarón Srivastava et al. en [61].



(A) Transformaciones genéricas (TG). (B) Imagenes sintéticas (AS).

FIGURA 5.8: Comparación del *accuracy* en el conjunto de validación sin considerar la ResNet50.

Por su parte, la etapa de evaluación reporta las principales métricas para un problema de clasificación y fueron consignadas para las diferentes arquitecturas y configuraciones en la tabla 5.1. El mejor rendimiento para la metodologia de *data augmentation* basada en transformaciones genéricas la obtuvo la configuración FaceNet y la distribución de imágenes por balanceo de clases. En contraste, para la estrategia de *data augmentation* por imágenes sintéticas producidas con un *autoencorder* variacional, la configuración levemente ganadora fue la FaceNet con clasificación ponderada, seguida de la FaceNet con balanceo de clases. Es notorio que el *gap* entre el *precision* y el *recall* es mayor para la configuración FaceNet con balanceo de clases en comparación a la mejor configuración (FaceNet con clasificación ponderada). Sin embargo, un mayor valor de *precision* indica una mejor tasa de aciertos. No obstante, basándose en la aplicación misma de reconocimiento facial, esto implicaría que

para sistemas de seguridad dicha configuración seria el modelo menos adecuado, mientras que para sistemas de monitoreo continuo en domótica, si sería la mejor elección.

	Accuracy	***Precision***	***Recall***	**F_1-Score**
Datos crudos				
VGG16	0.990	0.991	0.982	1.000
Resnet50	0.237	0.273	0.273	0.273
FaceNet	0.857	0.879	0.831	0.853
Transformaciones genéricas y balanceo de clases				
VGG16	0.990	0.991	0.991	0.991
Resnet50	0.247	0.897	0.262	0.262
FaceNet	**0.890**	**0.901**	**0.822**	**0.859**
Transformaciones genéricas y clasificación ponderada				
VGG16	0.861	0.880	0.835	0.860
Resnet50	0.237	0.253	0.253	0.253
FaceNet	0.746	0.772	0.670	0.717
Imágenes sintéticas y balanceo de clases				
VGG16	0.920	0.960	0.900	0.927
Resnet50	0.247	0.282	0.282	0.282
FaceNet	0.881	0.927	0.793	0.852
Imágenes sintéticas y clasificación ponderada				
VGG16	0.960	0.963	0.955	0.959
Resnet50	0.207	0.246	0.246	0.246
FaceNet	**0.894**	**0.898**	**0.890**	**0.894**

TABLA 5.1: Métricas sobre los conjuntos de validación para cada una de las arquitecturas y configuraciones establecidas en esta investigación.

5.3. Rendimiento en ambiente de pruebas

Una vez definida la FaceNet como la mejor arquitectura implementada en esta investigación, se procedio a computar y evaluar sus metricas sobre el conjunto de *test* conformado por imagenes nunca vistas por la arquitectura. Las métricas fueron consignadas en la tabla 5.2. Definiendo como criterio global la obtención de los mayores *precision* y *recall*, pero a su vez su menor diferencia, los resultados entregaron que la mejor configuración para la FaceNet fue el entrenamiento con *data augmentation* basado en la incorporacion de imagenes sinteticas, con la tecnica de clasificacion ponderada.

A continuación se presentan las curvas ROC (*Receiver Operating Characteristic*) para la FaceNet, la mejor arquitectura en los diferentes escenarios de configuración. Por lo tanto, se evaluó el rendimiento de la FaceNet en el conjunto de testeo para 5 modelos diferentes. Como linea base, la figura 5.9 muestra el rendimiento del clasificador sobre el conjunto de datos crudos (DC). Para el "sujeto 1" es evidente el sobreajuste y despues de una inspección visual en los *datasets* de rostros, se observo que el detector de rostros incluyó en las imágenes información perteneciente al cuerpo del sujeto. Posteriormente, en el *dataset* de escenarios, gran parte de las capturas de dicho sujeto se realizaron un mismo día, por

	Accuracy	***Precision***	***Recall***	**F_1-Score**
DC	0.771	0.811	0.667	0.730
TG + BC	**0.791**	**0.828**	**0.671**	**0.739**
TG + CP	0.747	0.773	0.671	0.717
AS + BC	0.739	0.847	0.570	0.679
AS + CP	**0.795**	**0.810**	**0.783**	**0.796**

TABLA 5.2: Métricas sobre el conjunto de prueba (*testing set*) para la FaceNet en todas sus configuraciones.

lo que su vestimenta no cambio considerablemente. Esta notoriedad puede impactar en el sobreajuste de la clase.

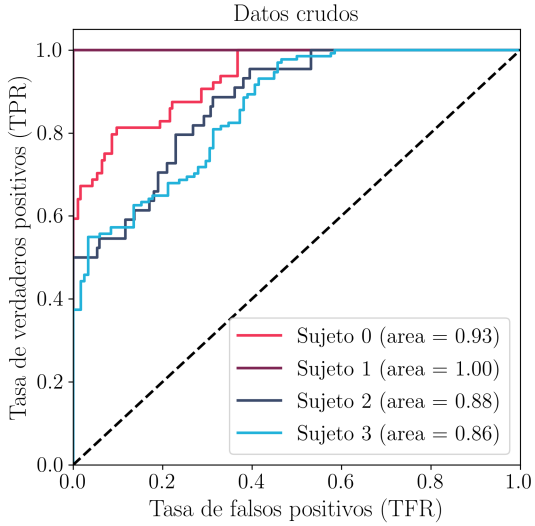
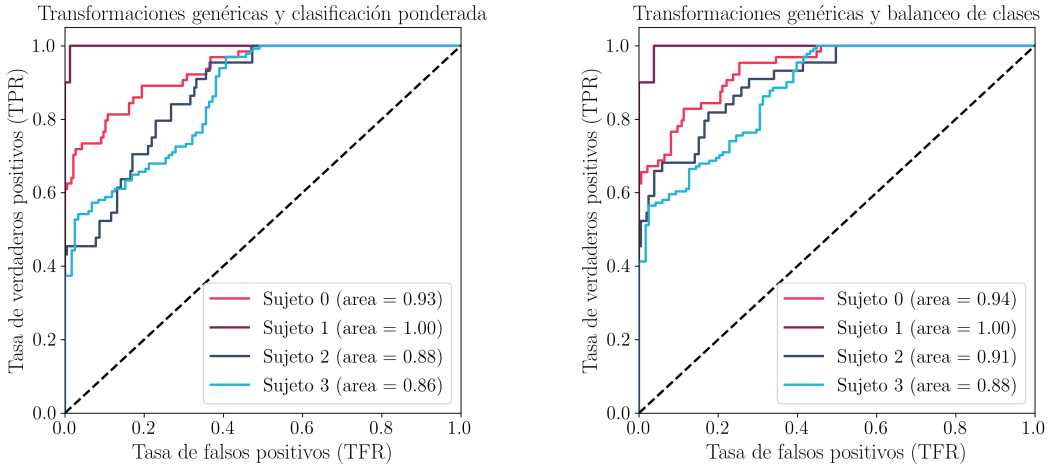


FIGURA 5.9: Curva ROC sobre el *testing set* del clasificador FaceNet entrenado con el conjunto de datos crudos (DC).

Para la estrategia de *data augmentation* bajo transformaciones genéricas el sobreajuste en la clase "sujeto 1" se mantiene, pues dichas transformaciones son poco "ìnvasivas" debido los operadores efectúan traslaciones, rotaciones, cambios de contraste, ampliaciones y reducciones, que modifican la información espacial, mas no en las probabilidades conjuntas para las intensidades verticales y horizontales de los píxeles de las imágenes, como se analizó en la sección 5.1, especificamente en las figura 5.2. Por otra parte, para el esquema de transformaciones genéricas el balance de clases parece ser mas eficiente que su contraparte no balanceada y que implementa una clasificación ponderada, ya que el área bajo cada curva ROC es mayor. Sencillamente, a pesar de aumentar la cantidad de muestras el hecho que provengan de la misma fuente (*dataset* original) no favorece o resalta las características necesarias (*feature maps* con información mas fácil de discriminar) para una tarea de clasificación de grano fino sin restricciones, como la estudiada en esta investigación.

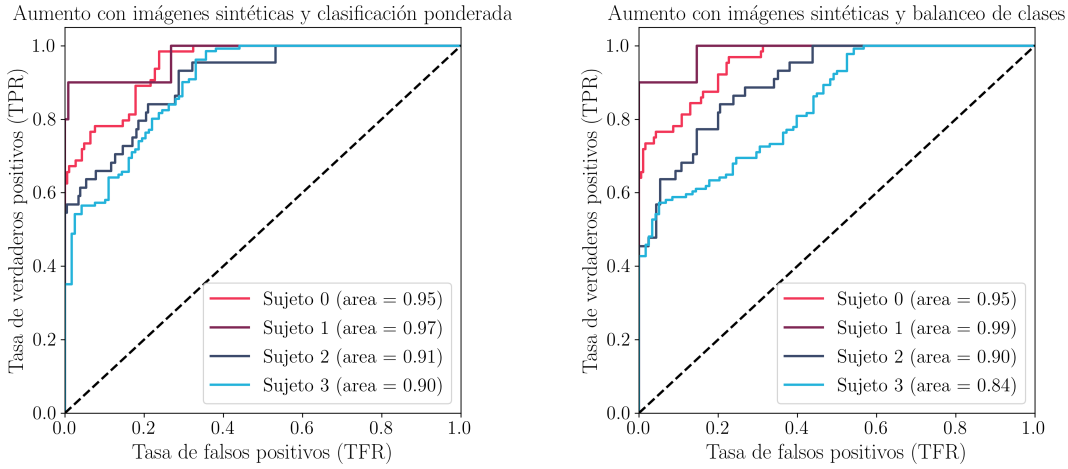
Finalmente, para el *data augmentation* con imágenes sintéticas la mejor configuración se da con la clasificación ponderada, pues como se evidencio en la figura 5.2 la probabilidad conjunta de la imagen sufre un efecto de "gaussianización" que aumenta la variabilidad



(A) Clasificación ponderada (CP). (B) Balanceo de clases (BC).

FIGURA 5.10: Curva ROC sobre el *testing set* del clasificador FaceNet entrenado con *data augmentation* de transformaciones genericas (TG).

del conjunto de muestras. Al mismo tiempo, este hecho aumenta la posibilidad de evitar el *overfiting* en cualquiera de las clases. Por su parte, al ponderar las probabilidades en la clasificación, las pobrabilidades *a posteriori* parecen ser mas fáciles de discriminar, en comparación a las otras configuraciones. Esto sumado a un conjunto de imágenes relativamente mas grande (una diferencia de 558 imágenes), repercute favorablemente en las métricas para todas las clases. No obstante, nuevamente en el *testing set*, las imágenes sintéticas y el balance de clases obtuvieron la máxima *precision*, característica que podría ser útil según la aplicación en la que se desplegará el modelo de reconocimiento facial al contar con la mejor tasa de verdaderos positivos.



(A) Clasificación ponderada (CP). (B) Balanceo de clases (BC).

FIGURA 5.11: Curva ROC sobre el *testing set* del clasificador FaceNet entrenado con *data augmentation* de imágenes sintéticas (AS).

CAPÍTULO 6

Conclusiones

El *unconstrained face recognition* es un problema abierto que actualmente impone un reto en investigación. En los experimentos realizados se demostró que la metodología de *data augmentation* mediante la incorporación de imágenes sintéticas producidas con *autoencoders* variacionales (VAE), logra mejorar sustancialmente el rendimiento de un clasificador de reconocimiento facial. Dicho predictor fue construido a partir del *fine tuning* del modelo preentrenado de la arquitectura FaceNet, bajo un conjunto de reentrenamiento con pocas muestras pero con alta similaridad al conjunto original de la arquitectura. Las muestras generadas mediante el uso de los VAEs aportan positivamente a la generalización de los modelos y una estrategia de clasificación ponderada puede mejorar aun más el rendimiento del predictor.

La FaceNet fue la arquitectura que mejor se comportó en la tarea de reconocimiento facial. No obstante, al mismo tiempo se concluyo que las métricas de la arquitectura ResNetFace resaltan por ser bajas aun siendo un modelo bastante robusto en tareas de clasificación de imágenes de grano grueso. Una de las razones de este comportamiento es la combinación de la función de perdida *categorical cross-entropy* con las *skip connections* de la arquitectura. La naturaleza residual de la ResNet50 afecta fuertemente la función de perdida *cross-entropy* generando que esta se sobre ajuste al conjunto de entrenamiento y a las imágenes de buena calidad y fallando en escenarios de perdida de información o poca calidad.

Claramente, las intensiones de realizar transformaciones genéricas son enriquecer a los modelos con pocas muestras y al mismo tiempo evitar el *overfiting*. Sin embargo, en los problemas de clasificación *fine-grained* con pequeños conjuntos de muestras, no son efectivas dichas transformaciones al no modificar las probabilidades conjuntas y condicionales del conjunto de datos. Por esta razón, los VAEs aportan una técnica de aumento de datos capaz de combatir el *overfiting* en comparación a las transformaciones genéricas, para clasificaciones de grano fino.

Se sugiere usar esquemas hibridos y repotenciar las técnicas de *data augmentation* para problemas de clasificación de grano fino, con la agregación de imágenes sintéticas. Al mismo tiempo, los VAEs pueden ser alternativas interesantes y menos demandantes en cuanto a cantidad datos y costo computacional, en comparación a sus homólogos generativos las GANs.

Bibliografía

[1] Zahid Akhtar and Ajita Rattani. A face in any form: new challenges and opportunities for face recognition technology. *Computer*, 50(4):80–90, 2017.

[2] Isha Kalra, Maneet Singh, Shruti Nagpal, Richa Singh, Mayank Vatsa, and PB Sujit. Dronesurf: Benchmark dataset for drone-based face recognition. In *2019 14th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2019)*, pages 1–7. IEEE, 2019.

[3] Gary B Huang, Manjunath Narayana, and Erik Learned-Miller. Towards unconstrained face recognition. In *2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pages 1–8. IEEE, 2008.

[4] Chen Huang, Yining Li, Chen Change Loy, and Xiaoou Tang. Learning deep representation for imbalanced classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5375–5384, 2016.

[5] Shan Li and Weihong Deng. Real world expression recognition: A highly imbalanced detection problem. In *2016 International Conference on Biometrics (ICB)*, pages 1–6. IEEE, 2016.

[6] Manuel Günther, Peiyun Hu, Christian Herrmann, Chi-Ho Chan, Min Jiang, Shufan Yang, Akshay Raj Dhamija, Deva Ramanan, Jürgen Beyerer, Josef Kittler, et al. Unconstrained face detection and open-set face recognition challenge. In *2017 IEEE International Joint Conference on Biometrics (IJCB)*, pages 697–706. IEEE, 2017.

[7] Connor Shorten and Taghi M Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of Big Data*, 6(1):60, 2019.

[8] Xiang Wang, Kai Wang, and Shiguo Lian. A survey on face data augmentation. *arXiv preprint arXiv:1904.11685*, 2019.

[9] Iacopo Masi, Anh Tuan Tran, Tal Hassner, Gozde Sahin, and Gérard Medioni. Face-specific data augmentation for unconstrained face recognition. *International Journal of Computer Vision*, 127(6-7):642–667, 2019.

[10] Luan Tran, Xi Yin, and Xiaoming Liu. Disentangled representation learning gan for pose-invariant face recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1415–1424, 2017.

[11] Daniel Sáez Trigueros, Li Meng, and Margaret Hartnett. Generating photo-realistic training data to improve face recognition accuracy. *arXiv preprint arXiv:1811.00112*, 2018.

[12] Ali Razavi, Aaron van den Oord, and Oriol Vinyals. Generating diverse high-fidelity images with vq-vae-2. *arXiv preprint arXiv:1906.00446*, 2019.

[13] P Chen. Trunk-branch ensemble convolutional neural networks for large scale, few-shot video-to-still face recognition. 2019.

[14] Yandong Guo and Lei Zhang. One-shot face recognition by promoting underrepresented classes. *arXiv preprint arXiv:1707.05574*, 2017.

[15] Shiming Ge, Shengwei Zhao, Xindi Gao, and Jia Li. Fewer-shots and lower-resolutions: Towards ultrafast face recognition in the wild. In *Proceedings of the 27th ACM International Conference on Multimedia*, pages 229–237, 2019.

[16] Xi Yin, Xiang Yu, Kihyuk Sohn, Xiaoming Liu, and Manmohan Chandraker. Feature transfer learning for face recognition with under-represented data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5704–5713, 2019.

[17] Jingxiao Zheng, Rajeev Ranjan, Ching-Hui Chen, Jun-Cheng Chen, Carlos D Castillo, and Rama Chellappa. An automatic system for unconstrained video-based face recognition. *arXiv preprint arXiv:1812.04058*, 2018.

[18] Dieu Linh Tran, Robert Walecki, Stefanos Eleftheriadis, Bjorn Schuller, Maja Pantic, et al. Deepcoder: Semi-parametric variational autoencoders for automatic facial action coding. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3190–3199, 2017.

[19] Yoshua Bengio, Ian Goodfellow, and Aaron Courville. *Deep learning*, volume 1. Citeseer, 2017.

[20] Zhenyu Wang. A digits-recognition convolutional neural network on fpga, 2019.

[21] Henry W Lin, Max Tegmark, and David Rolnick. Why does deep and cheap learning work so well? *Journal of Statistical Physics*, 168(6):1223–1247, 2017.

[22] Cs231n: Convolutional neural networks for visual recognition.

[23] Eric Kauderer-Abrams. Quantifying translation-invariance in convolutional neural networks. *arXiv preprint arXiv:1801.01450*, 2017.

[24] Dominik Scherer, Andreas Müller, and Sven Behnke. Evaluation of pooling operations in convolutional architectures for object recognition. In *International conference on artificial neural networks*, pages 92–101. Springer, 2010.

[25] Junhyuk Hyun, Hongje Seong, and Euntai Kim. Universal pooling–a new pooling method for convolutional neural networks. *arXiv preprint arXiv:1907.11440*, 2019.

[26] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[27] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[28] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[29] Tobias Glasmachers. Limits of end-to-end learning. *arXiv preprint arXiv:1704.08305*, 2017.

[30] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[31] Richard HR Hahnloser, Rahul Sarpeshkar, Misha A Mahowald, Rodney J Douglas, and H Sebastian Seung. Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit. *Nature*, 405(6789):947–951, 2000.

[32] Raimi Karim. Illustrated: 10 cnn architectures, Oct 2019.

[33] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

[34] Xi Cheng, Youhua Zhang, Yiqiong Chen, Yunzhi Wu, and Yi Yue. Pest identification via deep residual learning in complex background. *Computers and Electronics in Agriculture*, 141:351–356, 2017.

[35] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[36] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European conference on computer vision*, pages 630–645. Springer, 2016.

[37] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[38] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Thirty-first AAAI conference on artificial intelligence*, 2017.

[39] Mohit Sewak, Md Rezaul Karim, and Pradeep Pujari. *Practical Convolutional Neural Networks: Implement advanced deep learning models using Python*. Packt Publishing Ltd, 2018.

[40] A. Koul, S. Ganju, and M. Kasam. *Practical Deep Learning for Cloud, Mobile and Edge: Real-World AI and Computer Vision Projects Using Python, Keras and TensorFlow*. O'Reilly Media, Incorporated, 2019.

[41] Dumitru Erhan Christian Szegedy Scott Reed Cheng-Yang Fu Alexander C. Berg Wei Liu, Dragomir Anguelov. Ssd: Single shot multibox detector. *arXiv preprint arXiv:1512.02325*, 2016.

[42] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. June 2016.

[43] Luis Perez and Jason Wang. The effectiveness of data augmentation in image classification using deep learning. *arXiv preprint arXiv:1712.04621*, 2017.

[44] Sebastien C Wong, Adam Gatt, Victor Stamatescu, and Mark D McDonnell. Understanding data augmentation for classification: when to warp? In *2016 international conference on digital image computing: techniques and applications (DICTA)*, pages 1–6. IEEE, 2016.

[45] Barret Zoph, Ekin D Cubuk, Golnaz Ghiasi, Tsung-Yi Lin, Jonathon Shlens, and Quoc V Le. Learning data augmentation strategies for object detection. *arXiv preprint arXiv:1906.11172*, 2019.

[46] Carl Doersch. Tutorial on variational autoencoders. *arXiv preprint arXiv:1606.05908*, 2016.

[47] Peter Sadowski, Julian Collado, Daniel Whiteson, and Pierre Baldi. Deep learning, dark knowledge, and dark matter. In *NIPS 2014 Workshop on High-energy Physics and Machine Learning*, pages 81–87, 2015.

[48] Luke de Oliveira, Michela Paganini, and Benjamin Nachman. Learning particle physics by example: location-aware generative adversarial networks for physics synthesis. *Computing and Software for Big Science*, 1(1):4, 2017.

[49] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4401–4410, 2019.

[50] Jiajun Wu, Yifan Wang, Tianfan Xue, Xingyuan Sun, Bill Freeman, and Josh Tenenbaum. Marrnet: 3d shape reconstruction via 2.5 d sketches. In *Advances in neural information processing systems*, pages 540–550, 2017.

[51] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[52] Hugo Larochelle and Iain Murray. The neural autoregressive distribution estimator. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pages 29–37, 2011.

[53] Ali Razavi, Aaron van den Oord, and Oriol Vinyals. Generating diverse high-fidelity images with vq-vae-2. *arXiv preprint arXiv:1906.00446*, 2019.

[54] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006.

[55] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[56] Diederik P. Kingma and Max Welling. An introduction to variational autoencoders. *Foundations and Trends® in Machine Learning*, 12(4):307–392, 2019.

[57] Ayush Tewari, Michael Zollhofer, Hyeongwoo Kim, Pablo Garrido, Florian Bernard, Patrick Perez, and Christian Theobalt. Mofa: Model-based deep convolutional face autoencoder for unsupervised monocular reconstruction. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 1274–1283, 2017.

[58] Zhilu Zhang and Mert Sabuncu. Generalized cross entropy loss for training deep neural networks with noisy labels. In *Advances in neural information processing systems*, pages 8778–8788, 2018.

[59] Yi Sun, Xiaogang Wang, and Xiaoou Tang. Deeply learned face representations are sparse, selective, and robust. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2892–2900, 2015.

[60] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 815–823, 2015.

[61] Yash Srivastava, Vaishnav Murali, and Shiv Ram Dubey. A performance comparison of loss functions for deep face recognition. *arXiv preprint arXiv:1901.05903*, 2019.