INSTITUTO TECNOLÓGICO DE CHIHUAHUA

DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

“DISEÑO DE UN MÉTODO DE AUMENTO DE DATOS BASADO EN REDES GAN Y SU APLICACIÓN EN UN SISTEMA DE VISIÓN PARA LA DETECCIÓN DEL USO DE CUBREBOCAS”

TESIS

QUE PARA OBTENER EL GRADO DE

MAESTRO EN CIENCIAS EN INGENIERÍA ELECTRÓNICA

PRESENTA:

JESÚS ALEJANDRO NAVARRO ACOSTA

DIRECTOR DE LA TESIS:

DR. MARIO IGNACIO CHACÓN MURGUÍA

CHIHUAHUA, CHIH., OCTUBRE 2022

Chihuahua, Chih. 27 de septiembre de 2022

C. JESÚS ALEJANDRO NAVARRO ACOSTA
PRESENTE

En cumplimiento con los requerimientos para la obtención del grado de Maestro en Ciencias en Ingeniería Electrónica y a propuesta de su Comité Tutorial, la División de Estudios de Posgrado e Investigación le concede la autorización para imprimir la tesis titulada ***"Diseño de un método de aumento de datos basado en redes GAN y su aplicación en un sistema de visión para la detección del uso de cubrebocas"*** dirigida por el Dr. Mario Ignacio Chacón Murguía con el siguiente contenido de capítulos:

I Antecedentes
II Diseño de método de aumento de datos GDAM-GAN
III Generación de bases de datos artificiales
IV Entrenamiento de redes profundas con datos artificiales
V Conclusiones y trabajo futuro

A T E N T A M E N T E
Excelencia en Educación Tecnológica®
"La Técnica por el Engrandecimiento de México"

ROGELIO ENRIQUE BARAY ARANA
JEFE DE LA DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

REBA/adcs

EDUCACIÓN
SECRETARÍA DE EDUCACIÓN PÚBLICA
TECNOLÓGICO NACIONAL DE MÉXICO
INSTITUTO TECNOLÓGICO DE CHIHUAHUA
DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

Chihuahua, Chih. 15 de septiembre de 2022

ROGELIO ENRIQUE BARAY ARANA
JEFE DE LA DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN
PRESENTE

En cumplimiento con los requerimientos para la obtención del grado de Maestro en Ciencias en Ingeniería Electrónica, le notificamos que el documento de tesis del alumno **C. JESÚS ALEJANDRO NAVARRO ACOSTA,** titulado ***"Diseño de un método de aumento de datos basado en redes GAN y su aplicación en un sistema de visión para la detección del uso de cubrebocas"*** dirigido por el Dr. Mario Ignacio Chacón Murguía, ha sido aprobado y aceptado para su impresión.

Por lo anterior, proponemos le sea concedida la autorización de impresión correspondiente.

Agradeciendo la atención a la presente, quedamos de usted:

A T E N T A M E N T E
Excelencia en Educación Tecnológica®
"La Técnica por el Engrandecimiento de México"

DR. MARIO IGNACIO CHACÓN MURGUÍA
MIEMBRO DEL COMITÉ TUTORIAL

Juan Ramírez
DR. JUAN ALBERTO RAMÍREZ QUINTANA
MIEMBRO DEL COMITÉ TUTORIAL

MTRA. ALMA DELIA CORRAL SÁENZ
MIEMBRO DEL COMITÉ TUTORIAL

Abimael G
DR. ABIMAEL GUZMÁN PANDO
MIEMBRO DEL COMITÉ TUTORIAL

EDUCACIÓN
SECRETARÍA DE EDUCACIÓN PÚBLICA
TECNOLÓGICO NACIONAL DE MÉXICO
INSTITUTO TECNOLÓGICO DE CHIHUAHUA
DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

CARTA CESIÓN DE DERECHOS

En la ciudad de Chihuahua el día 27 de septiembre de 2022, el que suscribe, C. JESÚS ALEJANDRO NAVARRO ACOSTA con número de control G20061553, de la Maestría en Ciencias en Ingeniería Electrónica, adscrita a la División de Estudios de Posgrado e Investigación del Instituto Tecnológico de Chihuahua, manifiesta que es autor intelectual del presente trabajo de Tesis bajo la dirección del el Dr. Mario Ignacio Chacón Murguía y cede los derechos del trabajo titulado "Diseño de un método de aumento de datos basado en redes GAN y su aplicación en un sistema de visión para la detección del uso de cubrebocas", al Tecnológico Nacional de México y/o Instituto Tecnológico de Chihuahua para su difusión, divulgación, transmisión, reproducción, así como su digitalización con fines académicos y de investigación.

C. JESÚS ALEJANDRO NAVARRO ACOSTA

DECLARACIÓN DE ORIGINALIDAD

En la ciudad de Chihuahua el día 27 de septiembre de 2022, el que suscribe, C. JESÚS ALEJANDRO NAVARRO ACOSTA de la Maestría en Ciencias en Ingeniería Electrónica, con número de control G20061553, adscrito a la División de Estudios de Posgrado e Investigación del Instituto Tecnológico de Chihuahua, manifiesta que es autor intelectual de la tesis titulada "Diseño de un método de aumento de datos basado en redes GAN y su aplicación en un sistema de visión para la detección del uso de cubrebocas" bajo la dirección el Dr. Mario Ignacio Chacón Murguía; que el contenido es original y que las fuentes de información consultadas para su fundamentación están debidamente citadas y referenciadas.

C. JESÚS ALEJANDRO NAVARRO ACOSTA

Chihuahua, Chihuahua, octubre de 2022

Dra. María Elena Álvarez-Buylla Roces
Directora de Conacyt

Sirva la presente para saludarla e informarle que a la fecha he obtenido el grado de **Maestro en Ciencias en Ingeniería Electrónica** en la División de Estudios de Posgrado e Investigación del Instituto Tecnológico de Chihuahua, motivo por el cual agradezco todo el apoyo brindado por este Consejo que usted representa, con el otorgamiento de una beca que permitió dedicarme de manera exclusiva a la realización de mis estudios de posgrado y de esta manera lograr el objetivo principal del convenio establecido.

Sin otro particular quedo a sus órdenes no sin antes reiterar mi agradecimiento.
Atentamente

Jesús Alejandro Navarro Acosta

RESUMEN

DISEÑO DE UN MÉTODO DE AUMENTO DE DATOS BASADO EN REDES GAN Y SU APLICACIÓN EN UN SISTEMA DE VISIÓN PARA LA DETECCIÓN DEL USO DE CUBREBOCAS

Jesús Alejandro Navarro Acosta
Maestría en Ciencias en Ingeniería Electrónica
División de Estudios de Posgrado e Investigación del
Instituto Tecnológico de Chihuahua
Chihuahua, Chih. 2022.
Director de Tesis: Dr. Mario Ignacio Chacón Murguía

En el área de visión por computadora, los modelos de aprendizaje profundo han demostrado ser superiores a otros métodos tradicionales de procesamiento de imágenes. Estos permiten resolver problemas complejos, omitiendo el desarrollo de etapas de extracción de características de manera manual y, generalmente, presentan un mejor desempeño. No obstante, los modelos de redes neuronales profundas, dependiendo de su arquitectura, pueden requerir de una gran cantidad de datos de entrenamiento para que el desempeño del sistema sea óptimo y en algunas ocasiones es difícil obtener muestras para un problema específico. Para afrontar esta situación, se suelen utilizar métodos de aumento de datos, desde transformaciones de muestras existentes, hasta la síntesis de imágenes artificiales con modelos complejos, como redes generativas adversarias (GAN). Este tipo de modelos, han demostrado en los últimos años su potencial de generar imágenes con alta veracidad. Por lo que en este trabajo se desarrolla un método genérico de aumento de datos basado en redes GAN.

El diseño del método genérico de aumento de datos propuesto, el cual se denomina GDAM-GAN consiste en tres elementos principales: un modelo generativo base, un modelo de traducción imagen-a-imagen, y una etapa de depuración de muestras artificiales. Este diseño

posee la capacidad de ser configurable en sus arquitecturas y adaptable, de manera que es potencialmente útil en distintas aplicaciones de visión por computadora. Entre ellas, dos aplicaciones propuestas en este trabajo son: la generación de una base de datos útil para el entrenamiento de un sistema de detección del uso de cubrebocas y para el entrenamiento de un sistema de clasificación de enfermedades en hojas de tomate.

Los experimentos llevados a cabo demostraron que GDAM-GAN se puede utilizar en múltiples aplicaciones bajo el mismo procedimiento general. Se determinó que utilizar estas imágenes para el entrenamiento de modelos de redes profundas, permite reducir el error de clasificación hasta un 5.44%, respecto al entrenamiento sin aumento de datos. Además, el uso de técnicas de aumento de datos, como GDAM-GAN, reduce el tiempo de entrenamiento necesario para que los modelos de redes profundas generalicen.

CONTENIDO

LISTA DE FIGURAS

LISTA DE TABLAS

TABLA DE NOTACIONES

Notación	Descripción
I	Imagen
T	Operador general de transformación
I_{Nueva}	Imagen nueva, resultante de una transformación a una imagen
G	Generador de un modelo GAN
θ_G	Parámetros del generador G
D	Discriminador de un modelo GAN
θ_D	Parámetros del discriminador D
z	Vector con valores aleatorios, información en un espacio latente
$\boldsymbol{z}$	Conjunto de vectores z
$G(z)$	Muestra artificial dada por el generador G con entrada z
$G(\boldsymbol{z})$	Conjunto de muestras artificiales $G(z)$
x	Muestra real
$\boldsymbol{x}$	Conjunto de muestras reales
$p_g(\boldsymbol{x})$	Distribución de probabilidad del conjunto de muestras artificiales generadas por G
$p_z(\boldsymbol{z})$	Distribución de probabilidad de vector z
$p_{data}(\boldsymbol{x})$	Distribución de probabilidad del conjunto de muestras reales
$\hat{y}$	Salida de discriminador para una muestra, probabilidad de que la muestra de entrada sea real.
$\hat{Y}$	Conjunto de varias salidas $\hat{y}$ del discriminador D
y	Salida esperada del discriminador, etiqueta correspondiente a real ($y=1$) o artificial ($y=0$)
Y	Conjunto de salidas esperadas.
i	Iteración de algoritmo de entrenamiento en curso
θ^{i+1}	Parámetros de la red neuronal en la siguiente iteración
θ^{i}	Parámetros de la red neuronal en la iteración actual
α	Razón de aprendizaje
J	Función de costo
∇J	Gradiente de función de costo
J_D	Función de costo para discriminador D
J_G	Función de costo para generador G
l	Variable que representa una clase específica
n_l	Cantidad de clases para un problema específico
x_l	Muestra real de una clase específica l
$G(z\|l)$	Muestra artificial generada a partir de z dada la clase l

$G(\boldsymbol{z}\|l)$	Conjunto de muestras artificiales generadas a partir de $\boldsymbol{z}$ dada la clase l
$p_{data}(\boldsymbol{x}\|l)$	Distribución de probabilidad condicional del conjunto de muestras reales dada una clase l
c	Vector con información adicional utilizado en modelo InfoGAN, sus elementos son variables latentes.
c_i	Corresponde a la i-ésima variable latente del vector c
L	Número variables latentes en c, total de elementos del vector c
$\Gamma(c;G(z,c))$	Información mutua entre c y $G(z,c)$.
$\Gamma(X_1,X_2)$	Información mutua entre dos variables aleatorias X_1 y X_2
Λ	Híper-parámetro de modelo InfoGAN. Regula el efecto de la información mutua en el entrenamiento.
c'	Aproximación a vector c.
$Q(x)$	Transformación para obtener la aproximación c'
D_C	Red crítico en modelo Wasserstein GAN (equivalente a discriminador)
A_{reg}	Factor de regularización
λ_W	Hiperparámetro de entrenamiento en modelo Wasserstein GAN que regula el efecto de A_{reg}
$\ddot{\boldsymbol{x}}$	Conjunto de promedios ponderados entre muestras reales y muestras artificiales
ε	Coeficiente constante para la interpolación $\ddot{\boldsymbol{x}}$
w	Mapeo no lineal de z en modelo StyleGAN
A, B	Dominios de imágenes según su categoría
G_{AB}	Red generador de modelo CycleGAN que realiza la transformación del dominio A al dominio B
G_{BA}	Red generador de modelo CycleGAN que realiza la transformación del dominio B al dominio A
D_A	Red discriminador de modelo CycleGAN que analiza si una red es real respecto al dominio A
D_B	Red discriminador de modelo CycleGAN que analiza si una red es real respecto al dominio B
Ch	Número de canales de una imagen
H	Cantidad de pixeles correspondientes al alto de una imagen
W	Cantidad de pixeles correspondientes al ancho de una imagen
x_A	Muestra real perteneciente al dominio A
x_B	Muestra real perteneciente al dominio B
$\hat{x}_A$	Muestra generada perteneciente al dominio A
$\hat{x}_B$	Muestra generada perteneciente al dominio B
$L_{Gen[1-6]}$	Valores de pérdida para redes generador de modelo CycleGAN

λ_{Cycle}	Hiperparámetro de entrenamiento de modelo CycleGAN para regular el efecto de la pérdida de consistencia cíclica
$\lambda_{Identity}$	Hiperparámetro de entrenamiento de modelo CycleGAN para regular el efecto de la pérdida de identidad
$L_{Disc[1,2]}$	Valores de pérdida para redes discriminador de modelo CycleGAN
$f_{Inception}$	Transformación llevada a cabo por la red InceptionV3
n_l	Número de clases
IS	Valor de métrica *Inception Score*
KL	Divergencia de Kullback-Liebler
$H(p_1,p_2)$	Entropía cruzada entre dos distribuciones de probabilidad p_1 y p_2
$P(l\|x)$	Probabilidad de l dado x. Es decir, la probabilidad de que la muestra x pertenezca a la categoría l.
FID	Valor de métrica FID
m_{real}	Media de vectores de características de muestras reales
m_{fake}	Media de vectores de características de muestras artificiales
C_{real}	Matriz de covarianza de vectores de características de muestras reales
C_{fake}	Matriz de covarianza de vectores de características de muestras artificiales
Tr	Traza de una matriz
$\boldsymbol{x}_T$	Conjunto de muestras reales pertenecientes a la clase deseada
l_T	Clase deseada
$\boldsymbol{x}_S$	Conjunto de muestras reales pertenecientes a la clase base
l_S	Clase base
$\hat{\boldsymbol{x}}_{\boldsymbol{S}}$	Conjunto de muestras artificiales pertenecientes a la clase base
$\hat{\boldsymbol{x}}_{\boldsymbol{T}}$	Conjunto de muestras artificiales pertenecientes a la clase deseada
N	Numero de muestras en un lote
X_S	Dominio de muestras de la clase base
X_T	Dominio de muestras de la clase deseada
$d_{am\text{-}k}(a,b)$	Distancia de alcance mutuo entre los puntos a y b
$core_k(a)$	Radio mínimo que debe tener un círculo, a partir del punto a, de manera que k muestras del conjunto de datos se encuentren dentro de él
$\theta_{G\|minFID}$	Parámetros de la red generador en el punto de entrenamiento donde la métrica FID es mínima

INTRODUCCIÓN

Los modelos de aprendizaje profundo aplicados en sistemas de visión por computadora han demostrado ser superiores a otros métodos de procesamiento de imágenes, tanto en desempeño como en tiempo de diseño [1]. Entre sus cualidades, destaca su capacidad para extraer características abstractas automáticamente, sin la necesidad de intervención humana o el desarrollo de análisis estadísticos. Sin embargo, estos modelos requieren de un conjunto de datos de entrenamiento, con los cuales se ajustan ciertos parámetros del sistema para cumplir con el objetivo deseado. Por lo que, el desempeño de este tipo de sistemas depende de la arquitectura del modelo utilizado, de la cantidad y calidad de los datos de entrenamiento, y de los algoritmos de entrenamiento empleados. No obstante, la adquisición de estos datos, en ocasiones, implica un proceso tardado y costoso, o incluso poco factible, provocando que el conjunto de datos no contenga suficientes muestras para entrenar de manera óptima al modelo de aprendizaje profundo. Para solucionar tal problema, se utilizan distintos métodos de aumento de datos, los cuales, en el área de sistemas de visión, pueden consistir desde métodos simples de procesamiento de imágenes, hasta el uso de metodologías o modelos de aprendizaje profundo. Entre ellos, los modelos basados en redes generativas adversarias (GAN, por sus siglas en inglés) que actualmente plantean un área de oportunidad para el desarrollo eficiente de aumento de datos.

En los últimos años se ha demostrado el potencial de las redes GAN para generar muestras artificiales de alta calidad [2]. Han sido utilizados para el aumento de datos en algunas aplicaciones como sistemas de visión para la agricultura [3] y en imagenología médica [4] - [5]. Sin embargo, tales modelos están limitados a la aplicación para la que fueron diseñados. Por lo que, con el fin de mitigar esta limitante, en este proyecto de investigación se propone un método de aumento de datos basado en redes tipo GAN, denominado GDAM-GAN (por sus siglas en inglés - *Generic Data Augmentation Method with Generative Adversarial Networks*) que posee la capacidad de ser adaptable para solucionar distintas tareas.

El método GDAM-GAN se puede aplicar en problemas donde se cumplan las siguientes consideraciones en el conjunto de datos: se cuenta con una clase deseada, una clase base, y

existe una mayor cantidad de muestras de la clase base con respecto a la clase deseada. La clase deseada se define como la clase cuya cantidad de muestras se busca incrementar, mientras que la clase base se refiere a una categoría que comparte algunas características con la clase deseada, pero que cuenta con un mayor número de muestras. La metodología propuesta de aumento de datos consiste en tres elementos principales: un modelo generativo base, cuyo propósito es generar muestras artificiales de la clase base; un modelo de traducción imagen-a-imagen, que transforma muestras de la clase base a la clase deseada; y una etapa de depuración de muestras artificiales para descartar aquellas muestras que no cumplan con las características de la clase deseada. Esta metodología involucra la capacidad de ser configurable en su arquitectura y adaptable, de manera que es potencialmente útil en distintas aplicaciones de visión por computadora.

En este proyecto de tesis, se aplica GDAM-GAN en la generación de una base de datos útil para el entrenamiento de un sistema para la detección del uso de cubrebocas, los cuales se han convertido en una herramienta útil para el monitoreo de medidas preventivas para evitar el contagio de enfermedades. Adicionalmente, se implementa el método en el problema de clasificación de enfermedades en hojas de tomate.

Los experimentos llevados a cabo demuestran que GDAM-GAN se puede utilizar en múltiples aplicaciones bajo el mismo procedimiento general. Además, el uso de muestras artificiales en el entrenamiento de modelos de redes neuronales profundas mejora el desempeño respecto al entrenamiento sin aumento de datos. También, se reduce el número de épocas de entrenamiento necesarias para que el modelo generalice.

Este documento de tesis se desarrolla en cinco capítulos. El Capítulo I corresponde a los antecedentes, en donde se explican los conceptos fundamentales en el área de aumento de datos y redes generativas adversarias, así como un análisis del estado del arte referente a estas áreas. El Capítulo II describe el diseño del método de aumento de datos GDAM-GAN. El Capítulo III presenta el proceso de implementación de la metodología propuesta para el problema de detección del uso de cubrebocas y clasificación de enfermedades en hojas de tomate. El Capítulo IV describe el uso de las bases de datos artificiales en el entrenamiento de modelos de

redes neuronales profundas. Finalmente, en el Capítulo V se presentan las conclusiones, problemas abiertos y trabajo a futuro de este proyecto de tesis.

CAPÍTULO I. ANTECEDENTES

La visión por computadora es un área de la inteligencia artificial que desarrolla teorías y métodos para el procesamiento y/o interpretación de imágenes utilizando sistemas digitales. Desde hace varios años, los algoritmos de procesamiento de imágenes se han enfocado a la simulación del sistema de visión humano, desarrollando técnicas o modelos especializados en el manejo de imágenes como las redes neuronales profundas [6]. Estos modelos se han convertido recientemente en una de las principales herramientas para el procesamiento de imágenes, gracias a la disponibilidad de recursos de hardware con alta capacidad de procesamiento y bases de datos públicas. Sin embargo, para algunas aplicaciones, la cantidad y calidad de los datos disponibles en estas bases de datos pueden no ser adecuados para el entrenamiento de modelos de redes profundas. Una solución a este problema es utilizar distintos métodos de aumento de datos, los cuales, consisten desde métodos simples de procesamiento de imágenes, hasta modelos de aprendizaje profundo, como redes GAN.

En las siguientes secciones se presentan los conceptos fundamentales relacionados al aumento de datos, enfocado al área de visión por computadora, y modelos GAN, así como un estudio del estado del arte respecto a estos temas. Se recomienda realizar una lectura previa de conceptos fundamentales relacionados al área de aprendizaje profundo, los cuales se pueden consultar en [7].

1.1 Aumento de datos

Para que los algoritmos de procesamiento de imágenes basados en aprendizaje profundo tengan un desempeño óptimo, es necesario que la cantidad de muestras para el entrenamiento del modelo sean suficientes para representar de manera general a todas las clases posibles correspondientes al problema que se aborda. Además, el tamaño del conjunto de entrenamiento deberá ser acorde a la complejidad del modelo de aprendizaje profundo para evitar el problema de sobre entrenamiento u *overfitting*. Al presentarse este problema, el modelo se ajusta completamente a la distribución de los datos entrenamiento. Por lo que, se tiende a memorizar

los datos de entrada. Esto representa un problema al validar el modelo con datos de prueba o desconocidos, provocando que el sistema realice clasificaciones erróneas [8]. En la Figura 1.1 (a) se muestran las gráficas del error en entrenamiento y prueba para visualizar gráficamente el fenómeno de *overfitting* y en la Figura 1.1 (b) se muestra gráficamente el comportamiento deseado.. Se puede observar que el error en entrenamiento tiende a disminuir, pero el error de prueba tiende a incrementarse en las últimas épocas.

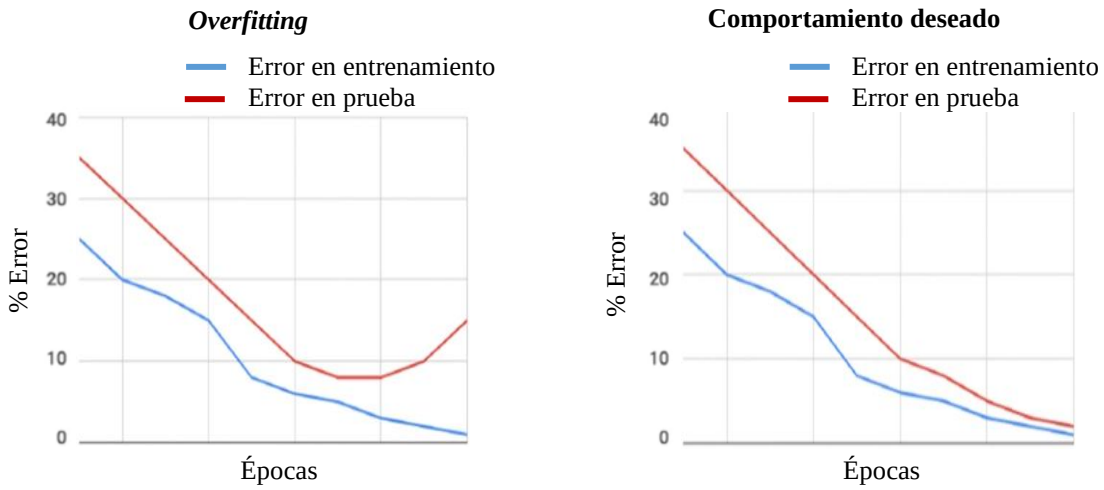


Figura 1.1. Representación gráfica del error de entrenamiento y prueba: a) fenómeno *overfitting* b) comportamiento deseado [8].

Una solución al problema de *overfitting* es incrementar el tamaño y la diversidad del conjunto de datos de entrenamiento. En el área de visión por computadora, existen distintas técnicas para realizar aumento de datos. Estas, se pueden clasificar en dos tipos: técnicas básicas de procesamiento imágenes y enfoques basados en aprendizaje profundo [8].

1.1.1 Técnicas básicas de procesamiento de imágenes

Los métodos de aumento de datos basados en técnicas básicas de procesamiento de imágenes, también conocidos como métodos tradicionales de aumento de datos, consisten en el procesamiento de una muestra del conjunto original, correspondiente a una imagen I, mediante una o varias transformaciones T para generar al menos una muestra nueva I_{Nueva}. Procurando

que los objetos de interés de la imagen nueva conserven la clase a la que estos pertenecen en la imagen original. Como se describe en la Ecuación (1.1).

$$T[I] = I_{Nueva} \tag{1.1}$$

Una transformación geométrica básica es la rotación de imágenes, en la cual la información de la imagen resultante se encuentra girada una cierta cantidad de grados respecto a la imagen original.

Otra técnica geométrica es la reflexión de imágenes de manera horizontal y/o vertical. En este caso, los valores de los pixeles de la imagen original son reflejados respecto a alguno de los ejes de la imagen, similar al efecto de un espejo.

Un método bastante simple es el recorte y escalamiento de imágenes, en donde se toma una porción de la imagen original, y se escala a un tamaño deseado para considerarse como una muestra nueva [9].

Las transformaciones del espacio de color de imágenes son una manera sencilla de crear muestras en donde se simulen cambios de iluminación y de ambiente. Pueden consistir en un aumento de brillo, modificación de contraste, cambio en la saturación del color de la imagen, entre otros.

La adición de ruido a imágenes es una estrategia utilizada para que el modelo sea robusto ante diversos elementos no deseados. Por otro lado, también es posible aplicar diversos filtros espaciales para obtener imágenes nuevas, por ejemplo, utilizar un filtro de suavizado para generar imágenes difuminadas que, en la mayoría de los casos, pertenecen a la misma clase que la muestra original [8].

Otra técnica es el borrado aleatorio de porciones de la imagen. Este método permite generar muestras en donde se simulen oclusiones [10]. Tal porción de la imagen puede ser rellenada con un valor constante o con ruido.

En la Figura 1.2 se muestran varios ejemplos de aumento de datos utilizando técnicas básicas de procesamiento de imágenes. Suponiendo que la imagen original pertenece a la clase *perro*, nótese que para todos los casos la clase no cambia.

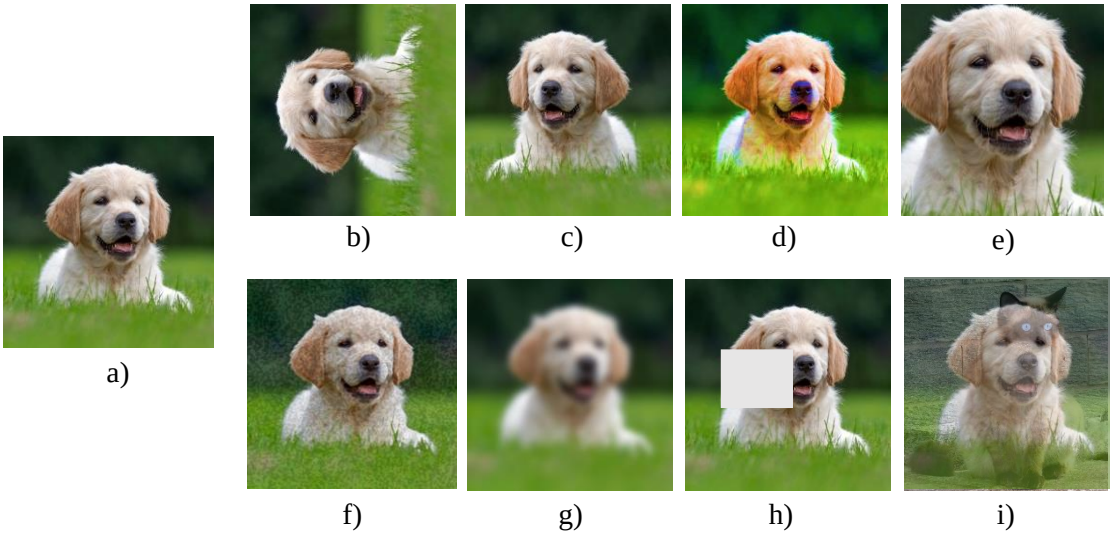


Figura 1.2. Transformaciones básicas de imágenes: a) original, b) rotación, c) reflexión horizontal, d) aumento de saturación de colores, e) recorte y escalamiento, f) adición de ruido, g) filtro de suavizado, h) borrado aleatorio.

De la Figura 1.2b-h se puede inferir el posible beneficio que tienen estas técnicas en el modelo donde se aplican. Por ejemplo, un borrado aleatorio genera muestras en donde se simulan obstáculos en imágenes, de manera que, al entrenar un modelo de aprendizaje de máquina con tales muestras, se esperaría que este fuera robusto ante oclusiones. Sin embargo, existen otros métodos, en donde es complicado deducir el efecto que tienen en el entrenamiento de un modelo de aprendizaje de máquina, como es el caso de la mezcla de imágenes (Figura 1.2i). Esta técnica consiste en la unión de dos o más imágenes para la generación de una muestra nueva. Tal unión se puede realizar a partir de: operaciones lineales, como una suma aritmética; o transformaciones no lineales, como el recorte de ciertas porciones de varias imágenes y su combinación [8].

Es necesario considerar que el uso de múltiples técnicas no siempre es conveniente para prevenir *overfitting* y, para algunos casos, puede que la clase de la imagen generada no sea igual a la clase original. Por ejemplo, si se trata de imágenes con letras y/o números, una reflexión o rotación provocaría que la o las letras y/o números de la imagen no sean iguales a la imagen original. Debido a esto, es necesario analizar cuidadosamente los datos con los que se trabaja

para determinar las técnicas de aumento de datos apropiadas para cada situación en específico o desarrollar un modelo adaptativo.

1.1.2 Enfoques basados en aprendizaje profundo

Los métodos de aumento de datos basados en aprendizaje profundo, generalmente, son más complejos que los métodos mencionados en la sección anterior, debido a que las arquitecturas de redes profundas pueden involucrar una gran cantidad de parámetros, respecto a los métodos tradicionales. A continuación, se describen brevemente los métodos de aumento de datos en el espacio de características, entrenamiento adversario, modelos basados en GAN, y transferencia de estilo.

Las características representan una o varias peculiaridades de una muestra en un espacio distinto, el cual generalmente tiene una dimensionalidad menor a las muestras. Por ejemplo, un sistema que procesa imágenes puede mapear una muestra, que se encuentra en el espacio de color RGB con una resolución espacial fija, a un espacio de menor dimensionalidad que puede ser un vector con los momentos estadísticos de esa muestra. Existe una estrategia de aumento de datos en el espacio de características. Esta consiste en la manipulación de vectores de características para la generación de nuevos vectores, sin haber creado la muestra respectiva en el espacio de entrada, lo cual reduce la carga computacional respecto al procesamiento en el espacio original. Sin embargo, la información representada en el espacio de características, usualmente, es difícil de interpretar [8].

El entrenamiento adversario es un método cuya idea principal es la identificación de puntos débiles de un modelo de aprendizaje profundo. Esto mediante la búsqueda de una perturbación indetectable por el ojo humano, que, al aplicarse a un dato de entrada, provoque una incorrecta clasificación por parte del modelo. En el área de procesamiento de imágenes, una perturbación común es la adición de un elemento de ruido a las muestras originales, con lo cual se producen nuevas muestras. Posteriormente, estas muestras son utilizadas para entrenar al sistema nuevamente y así eliminar el punto débil del modelo [11].

Otro método para el aumento de datos es el uso de modelos basados en GAN para la generación de muestras artificiales a partir de las características de un conjunto conocido. Tales

modelos, debido a su potencial, se han convertido en un tema de investigación muy importante en el área de la inteligencia artificial. Estos modelos y sus variantes se describen a detalle en la siguiente sección.

La transferencia de estilo es una técnica en la cual se reproduce el conjunto de texturas y/o características de una imagen en otra, de manera que el estilo de la muestra generada se asemeja a la original. Originalmente, este método se desarrolló para aplicar características de obras de arte conocidas a imágenes, utilizando la información de capas intermedias de una red neuronal convolucional [12].

1.2 Redes generativas adversarias

Las GAN se refieren a un tipo de modelos de aprendizaje profundo, cuyo objetivo es la generación de datos. Para ello, se utiliza un esquema de entrenamiento, propuesto en 2014 por Goodfellow *et al.* [13]. El esquema básico de un modelo GAN se muestra en la Figura 1.3, éste consiste en un generador G y un discriminador binario D. El generador G es el elemento principal, consiste en una red neuronal artificial, con parámetros θ_G, cuyo propósito es generar una muestra artificial $G(z)$ a partir de información de un vector z, el cual contiene valores aleatorios con una distribución $p_z(z)$ generalmente gaussiana con media cero y desviación estándar unitaria. El discriminador D es otra red neuronal artificial, con parámetros θ_D, que toma como entrada una muestra cualquiera, la cual puede ser una imagen o muestra real x[1] o artificial $G(z)$, y a la salida $\hat{y}$ se obtiene la probabilidad de que tal muestra sea real.

[1] En la literatura es común utilizar el símbolo x para referirse a muestras, las cuales generalmente son imágenes. En esta tesis, a partir de esta sección se utiliza de manera indistinta el término imagen y muestra con el símbolo x.

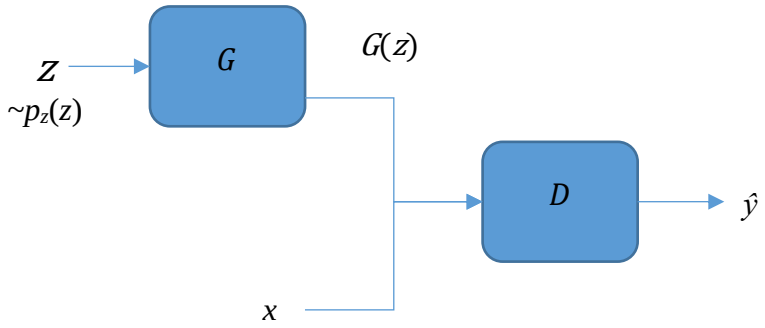


Figura 1.3. Arquitectura básica de un modelo GAN.

En la arquitectura original de GAN [13], el generador se compone por dos capas completamente conectadas (*FCL*) con función de activación lineal rectificada (*ReLU*) y una *FCL* con función de activación sigmoidal. Mientras que el discriminador se compone por dos capas de tipo *maxout* y una *FCL* con función de activación sigmoidal. Las dimensiones de entradas, capas ocultas y salidas varían según sea la base de datos con la que se entrena al modelo. En la Figura 1.4 se muestra el esquema de la arquitectura del generador y discriminador utilizados para trabajar con la base de datos MNIST (el número debajo de cada bloque corresponde a la dimensión de la salida de la capa).

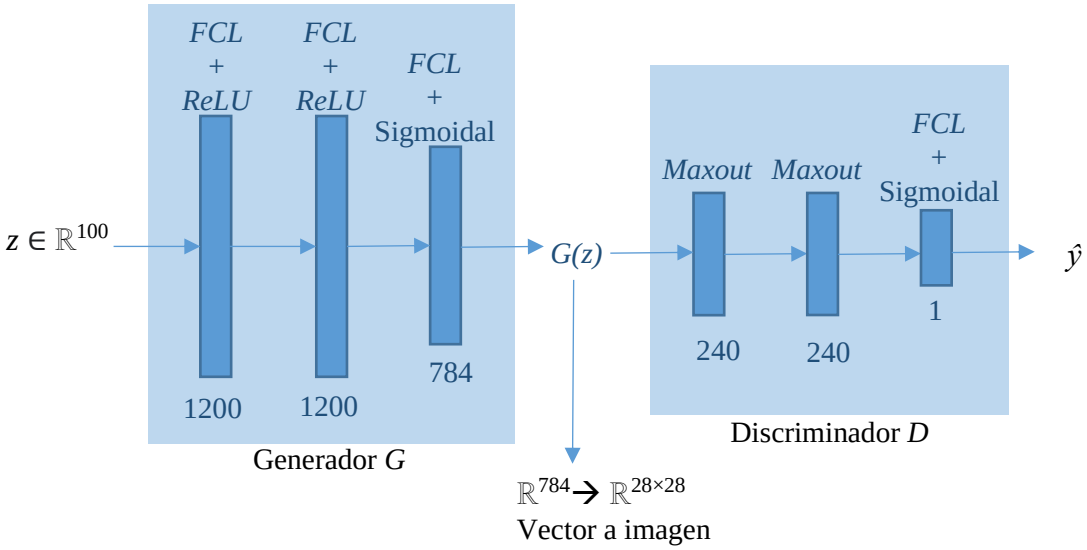


Figura 1.4. Arquitectura original GAN, utilizando la base de datos MNIST [1].

1.2.1 Entrenamiento de GAN

Un modelo GAN implica el entrenamiento de dos redes: la generadora *G* y la discriminadora *D*. El proceso se realiza utilizando el algoritmo de retropropagación (*backpropagation*) y se lleva a cabo de manera alternada pero no simultánea, es decir, se actualizan los parámetros θ_D del discriminador *D* durante una o más épocas y luego se actualizan los parámetros θ_G del generador *G*. De manera general, la actualización de los parámetros θ de una red neuronal se describe con la Ecuación (1.2):

$$\theta^{i+1} = \theta^i - \alpha \nabla J \tag{1.2}$$

En donde *i* se refiere a la iteración del algoritmo de entrenamiento en curso, θ^i es el conjunto de parámetros de la red en la iteración actual, θ^{i+1} es el conjunto de parámetros de la red en la siguiente iteración, α es la razón de aprendizaje y ∇J es el gradiente de la función de pérdida *J* con respecto a los pesos actuales θ^i.

El objetivo de *G* es generar muestras artificiales $G(\boldsymbol{z})$ con una distribución de probabilidad $p_g(\boldsymbol{x})$ similar a la distribución de las muestras reales $p_{data}(\boldsymbol{x})$; mientras que el objetivo de *D* es la correcta clasificación entre las muestras reales y artificiales. El entrenamiento del modelo está basado en el algoritmo *minimax* de la teoría de juegos, cuyo optimizador está dado por la Ecuación (1.3), en donde el argumento del operador min max es una forma simplificada de la función de costo de entropía cruzada binaria (BCE, por sus siglas en inglés).

$$\min_G \max_D \mathbb{E}_{x\sim p_{data}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{z\sim p_z(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))] \tag{1.3}$$

En donde $D(\boldsymbol{x})$ y $D(G(\boldsymbol{z}))$ corresponden a la probabilidad de que $\boldsymbol{x}$ y $G(\boldsymbol{z})$ sean muestras reales; $p_{data}(\boldsymbol{x})$ es la distribución de probabilidad de los datos reales; y $p_z(\boldsymbol{z})$ es la distribución de probabilidad del ruido aleatorio de entrada.

1.2.1.1 Entrenamiento de discriminador

Para el caso del discriminador, el proceso de entrenamiento consiste en maximizar la función de costo. Para ello, en cada iteración, el discriminador toma un conjunto (lote) de muestras generadas $G(\boldsymbol{z})$ y muestras reales $\boldsymbol{x}$, genera las salidas $\hat{Y}$ correspondientes y las compara con los

objetivos esperados *Y* para calcular el valor de la función de costo y actualizar los parámetros θ_D mediante retropropagación. El conjunto de objetivos esperados *Y*, para un conjunto de muestras a la entrada del discriminador poseerán los siguientes valores: *y=1* para cuando la entrada es una muestra real *x*; y *y=0* cuando la entrada es una muestra artificial *G(z)*. En el esquema de la Figura 1.5 se muestra el proceso descrito.

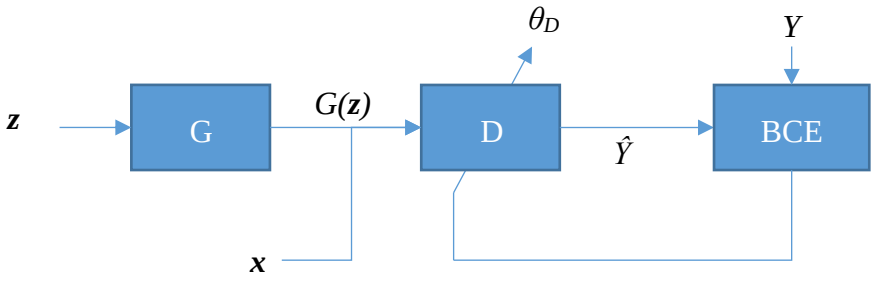


Figura 1.5. Esquema de entrenamiento de discriminador.

Para facilitar el proceso de implementación, al entrenar la red discriminador, es común utilizar el valor negativo de la función de costo, como se muestra en la Ecuación (1.4). Ahora el problema de optimización consiste en encontrar los parámetros θ_D para minimizar tal función.

$$J_D = -\mathbb{E}_{x\sim p_{data}(x)}[\log D(\boldsymbol{x})] - \mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(\boldsymbol{z})))] \tag{1.4}$$

1.2.1.2 Entrenamiento del generador

Por otro lado, el generador utiliza la retroalimentación del discriminador para actualizar los parámetros θ_G. En la Figura 1.6 se muestra el esquema de entrenamiento del generador. Primero, el generador *G* toma como entrada un conjunto de vectores ***z*** y genera muestras artificiales *G(**z**)*, las cuales pasan al discriminador, en donde se determina la probabilidad de que tales muestras sean reales $\hat{Y}$. Luego, se comparan tales resultados con los valores esperados *Y* y se calcula el valor de la función de costo, con la cual se actualizan los parámetros θ_G mediante retropropagación. Para este caso, se espera que el generador sintetice imágenes que parezcan reales, de manera que los valores esperados *Y* sean *y*=1 para todas las muestras *G(**z**)*.

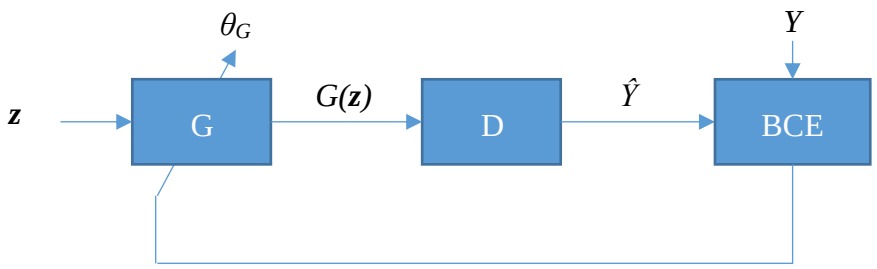


Figura 1.6. Esquema de entrenamiento del generador.

Durante la actualización de pesos del generador *G*, el discriminador *D* solamente evalúa muestras artificiales, de manera que la función de costo, para el generador, se reduce a:

$$J_G = \mathbb{E}_{z\sim p_z(z)}[\log(1-D(G(\boldsymbol{z})))] \tag{1.5}$$

El generador aprende la distribución $p_{data}(\boldsymbol{x})$ de las muestras reales sin conocer tales imágenes, gracias a la retroalimentación dada por el discriminador, la cual indica hacia dónde es necesario modificar los parámetros θ_G, de manera que la salida del discriminador tienda a ser 1.

1.2.2 Variantes de modelos GAN

Para incrementar la capacidad del modelo de GAN original, se han propuesto diversas variantes [14]. A continuación, se mencionan algunas de ellas y sus características principales.

1.2.2.1 GAN con redes convolucionales profundas

Este modelo se denomina *Deep Convolutional Generative Adversarial Networks* (DCGAN) y se trata de una modificación a las arquitecturas de las redes que conforman al modelo GAN, con el propósito de aprovechar las ventajas que presentan las redes tipo convolucional frente a las redes neuronales comunes (*fully connected*) en el procesamiento de imágenes. En este caso, se utilizan redes neuronales convolucionales (CNN) para la arquitectura tanto del generador *G* como del discriminador *D*. Las principales características de tales arquitecturas son [15]:

- Uso de convoluciones transpuestas, en el caso del generador, y se omite el uso de capas tipo *pooling* en ambas redes.
- Uso de normalización por lotes (*batchnorm*) en los valores de las entradas a cada capa.

- Se evita el uso de capas FCL.
- En el generador, se utilizan funciones de activación tipo *ReLU* en todas las capas, excepto en la capa de salida, en donde se utiliza una función *Tanh*.
- En el discriminador se utilizan funciones de activación tipo *LeakyReLU* en todas las capas, excepto a la salida, en donde se utiliza una función sigmoidal.

En la Figura 1.7 y Figura 1.8 se muestra la arquitectura del generador y discriminador de un modelo DCGAN para generar imágenes con una resolución espacial de 64×64 pixeles. Los detalles específicos, como las dimensiones de entrada y el tamaño de kernel de cada capa, pueden variar dependiendo de las características de las imágenes con las que se trabaje. Respecto al entrenamiento, este se realiza mediante el proceso descrito en las secciones 1.2.1.1 y 1.2.1.2.

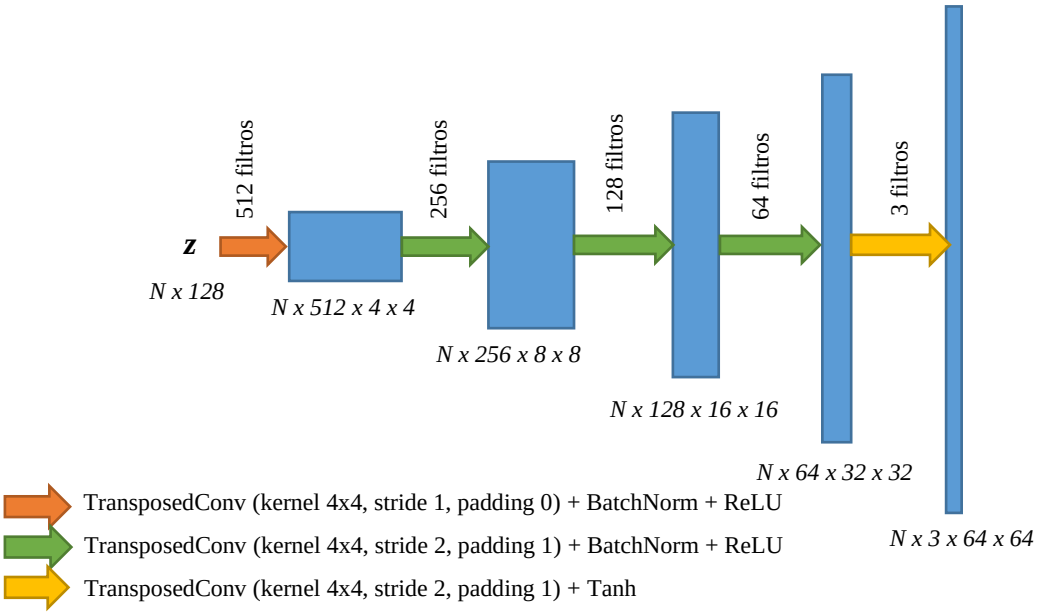


Figura 1.7. Arquitectura de generador en DCGAN.

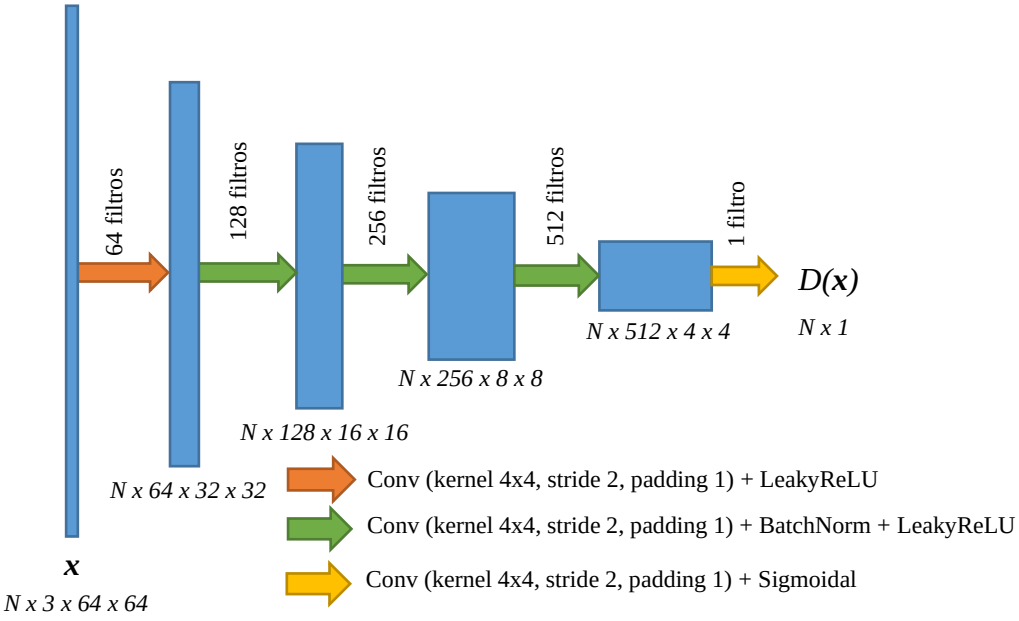


Figura 1.8. Arquitectura de discriminador en modelo DCGAN.

Al analizar la información en el espacio latente, es decir, una extensión abstracta multidimensional en donde se representan los datos. La representación en el espacio latente se obtiene con la salida de algunas capas de la red convolucional, con las cuales es posible interpretar, hasta cierto grado, el proceso que lleva a cabo cada capa de la red, a diferencia de una red neuronal compuesta por únicamente capas tipo FCL, en donde la información latente entre capa y capa tiene un nivel de abstracción más alto y es complejo o incluso imposible interpretarla.

1.2.2.2 GAN condicional (CGAN)

En el modelo original, el generador *G* intenta aproximar la distribución de las muestras reales $p_{data}(\boldsymbol{x})$. Tales muestras reales podrían estar conformadas por diversas clases, provocando que la distribución $p_{data}(\boldsymbol{x})$ tienda a ser multimodal, lo cual incrementa la dificultad de la tarea del generador *G*. En [16] se propone el modelo *Conditional Generative Adversarial Networks* (CGAN), en donde se añade una variable de entrada *l* adicional al generador y discriminador, correspondiente a la etiqueta de alguna clase en específico. En la Figura 1.9 se muestra la estructura genérica del modelo CGAN.

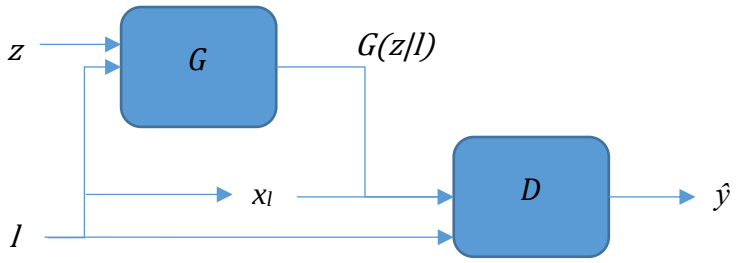


Figura 1.9. Estructura de modelo CGAN.

Utilizando l se delimita el espacio muestral, ahora la tarea del generador G consiste en aproximar la distribución de las muestras reales pertenecientes a una clase l, es decir, $p_{data}(\boldsymbol{x}_l)$ o, de manera general, la probabilidad condicional $p_{data}(\boldsymbol{x}|l)$. El optimizador, ecuación (1.6), correspondiente al modelo CGAN es similar al del modelo GAN original, pero se añade la condición l a la expresión.

$$\min_G \max_D \mathbb{E}_{x\sim p_x(x)}[\log D(\boldsymbol{x}\,|\,l)] + \mathbb{E}_{z\sim p_z(z)}[\log(1-D(G(\boldsymbol{z}\,|\,l)))] \tag{1.6}$$

Durante la implementación, la condición l corresponde a un vector de tipo *one-hot*. Es decir, un vector con valores de cero en la mayoría de sus elementos, excepto en uno de ellos, en donde el valor es 1. Esa posición del vector indica la condición o etiqueta de una clase en específico. Entonces, tal condición l se concatena con z para formar la entrada del generador G. Por otro lado, en el discriminador el proceso es similar, se concatena la condición l a las entradas $\boldsymbol{x}_l$ o $G(\boldsymbol{z}|l)$, a lo largo de la dimensión correspondiente a los canales de la imagen como matrices de ceros o unos. La arquitectura del generador G y discriminador D en el modelo CGAN puede ser la misma que en otros modelos, tomando en cuenta el vector l en las entradas.

1.2.2.3 InfoGAN

Otro modelo condicional se presenta en [17] y se denomina InfoGAN en donde se añade un vector de entrada c con información adicional para la generación de las muestras, los valores de este vector, los cuales son números reales, se denominan variables latentes y se busca que representen alguna característica de las muestras $G(z)$ a generar. Por ejemplo, en generación de rostros, se esperaría que cada elemento del vector c manipule algún atributo facial, como el

color de ojos, forma de nariz, boca, etc. En la ecuación (1.7) se muestra la composición del vector *c*.

$$c = \{c_1, c_2, ..., c_L\} \tag{1.7}$$

En donde c_1, c_2, …, c_L corresponden a las variables latentes y *L* es el total de elementos de *c*. Este modelo, además de aproximar la probabilidad de las muestras reales $p_{data}(\boldsymbol{x})$, busca encontrar una relación entre las variables latentes en *c* y los atributos de las muestras artificiales. De manera que, al modificar el valor de alguna variable latente, se controle alguna característica específica al generar una nueva muestra. Para ello, durante el entrenamiento se generan los vectores *c* y *z* de manera aleatoria, y se añade un nuevo factor a la función de costo, el cual tiene como propósito incrementar la información mutua entre las variables latentes de *c* y las muestras generadas. Esto se describe en la ecuación (1.8).

$$\min_G \max_D \mathbb{E}_{x\sim p_{data}(\boldsymbol{x})}[\log D(\boldsymbol{x})] - \mathbb{E}_{z\sim p_z(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z}, c)))] - \lambda \Gamma(c; G(z, c)) \tag{1.8}$$

En donde $\Gamma(c;G(z,c))$ es la información mutua entre la información adicional *c* y la muestra generada $G(z,c)$; y λ es un híper-parámetro para regular el efecto de la información mutua en el entrenamiento del modelo. La información mutua entre dos variables aleatorias está definida como:

$$\Gamma(X_1, X_2) = \sum_{x_1 \in X_1} \sum_{x_2 \in X_2} p(x_1, x_2) \log\left(\frac{p(x_1, x_2)}{p(x_1)\, p(x_2)} \right) \tag{1.9}$$

Entonces, para calcular $\Gamma(c;G(z,c))$ es necesario conocer la probabilidad conjunta $p(c,G(\boldsymbol{z},c))=p(G(z,c))p(c|G(z,c))$, en donde la probabilidad condicional $p(c|G(\boldsymbol{z},c))$ se desconoce. Por lo cual, se realiza una aproximación *c'* al vector de variables latentes *c* utilizando al discriminador *D*, de manera que $p(c'|G(\boldsymbol{z},c))\approx p(c|G(\boldsymbol{z},c))$. En la Figura 1.10 se muestra la estructura general del modelo InfoGAN.

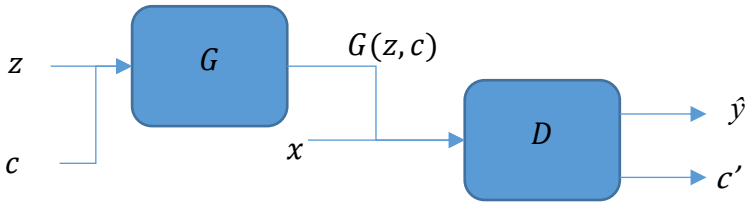


Figura 1.10. Estructura de modelo InfoGAN.

La arquitectura del generador es similar a otros modelos GAN, tomando como entrada una concatenación de los vectores *z* y *c*. Por otro lado, el discriminador añade algunas capas adicionales en el modelo para la aproximación de *c* mediante una transformación *Q(x)=c'*, en la Figura 1.11 se muestra la arquitectura del discriminador para uno de los experimentos llevados a cabo en [17].

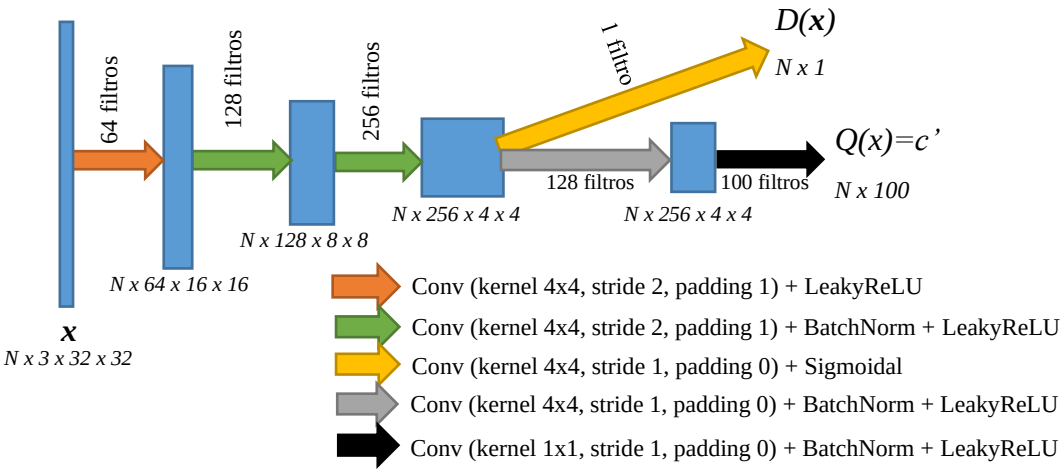


Figura 1.11. Arquitectura de discriminador para modelo InfoGAN.

1.2.2.4 Wasserstein GAN

En el modelo original de GAN, es posible que, durante el entrenamiento, el discriminador tenga un mejor desempeño respecto al generador y se presente el problema del desvanecimiento del gradiente, en donde el valor del gradiente tiende a cero, de manera que el cambio de los pesos del generador sea prácticamente nulo, provocando que el modelo permanezca estático durante el resto del entrenamiento. El modelo *Wasserstein Generative Adversarial Network* (WGAN) [18], propone una solución a este problema. En WGAN se sustituye el discriminador

D por un bloque crítico D_c, el cual se encarga de determinar un valor real, que indica qué tan reales o artificiales son las muestras. Tal cambio en la terminología de la red se atribuye a su función, un discriminador convencional está enfocado a ser un clasificador de muestras reales y artificiales, mientras que el bloque crítico pretende indicar el grado en el que una muestra es real o artificial según lo siguiente: un valor positivo indica que la muestra tiende a ser real; los valores negativos indican que la muestra tiende a ser artificial; mientras que la magnitud de este valor representa el grado en que una muestra es real o artificial. Tal bloque crítico D_c debe representar una función 1-*Lipschitz* continua, es decir, que la norma del gradiente de la función, para cualquier punto, sea menor o igual a 1.

Este modelo utiliza una función de costo basada en la distancia *Earth-Mover's* (EM), que calcula la diferencia entre dos distribuciones de probabilidad en un mismo espacio, y es una analogía a una situación en donde se tienen dos montones de tierra (semejante a dos distribuciones de probabilidad), la distancia EM indicaría el esfuerzo necesario para cambiar la forma y posición de un montón para que sea igual al otro. En la Ecuación (1.10) se muestra el problema de optimización para WGAN.

$$\min_{G}\max_{D_c}\mathbb{E}\left[D_c(\boldsymbol{x})\right]-\mathbb{E}\left[D_c(G(\boldsymbol{z}))\right]+\lambda_W A_{reg} \tag{1.10}$$

En donde $D_c(\boldsymbol{x})$ y $D_c(G(\boldsymbol{z}))$ se refiere a la salida del bloque crítico con muestras reales y muestras generadas, respectivamente; λ_W es un hiperparámetro de entrenamiento de este modelo; y A_{reg} es un factor de regularización cuyo propósito es penalizar al bloque crítico cuando la norma del gradiente es mayor a 1, es decir, incita al crítico a ser una función 1-*Lipschitz* continua [19]. El factor de regularización se determina utilizando las Ecuaciones (1.11) y (1.12).

$$A_{reg}=\mathbb{E}\left[\left\|\nabla D_c(\ddot{\boldsymbol{x}})\right\|_2-1\right]^2 \tag{1.11}$$

$$\ddot{\boldsymbol{x}}=\varepsilon\boldsymbol{x}+(1-\varepsilon)G(\boldsymbol{z}) \tag{1.12}$$

En donde $\ddot{\boldsymbol{x}}$ es un promedio ponderado entre las muestras reales $\boldsymbol{x}$ y las muestras generadas $G(\boldsymbol{z})$; y ε es un coeficiente constante positivo menor a 1.

En la arquitectura del bloque crítico D_c se omite el uso de una función de activación para la capa de salida, a diferencia del modelo GAN, en cuyo discriminador originalmente eran funciones sigmoidales. Entonces, la salida del crítico $(D_c)\epsilon\mathbb{R}$ es una combinación lineal de las salidas de la penúltima capa, y su valor indica qué tan real es la muestra de entrada según las condiciones descritas previamente. Se ha probado que esta estrategia reduce el riesgo de que se presente el problema del desvanecimiento del gradiente y mejora la estabilidad durante el entrenamiento de estos modelos [18].

1.2.2.5 StyleGAN2

El modelo *StyleGAN* se refiere a una variante de la red generador para modelos GAN, propuesta por Karras *et. al* [20], con el propósito de analizar elementos importantes en el proceso de generación de imágenes, como: la composición de atributos generales en la imagen; o el origen de variaciones en los detalles de la muestra generada.

En esta arquitectura, el generador realiza la síntesis de la imagen aumentando paulatinamente la dimensión de la muestra, similar al modelo *ProgressiveGAN* [21]. En otros modelos GAN, la información latente del vector *z* se toma como entrada únicamente en la primera capa de la red, y luego se propaga hasta generar una muestra. En la arquitectura *StyleGAN*, primero se realiza un mapeo no lineal del vector de entrada *z* a un dominio distinto, utilizando una red neuronal con 8 capas completamente conectadas, después, el vector resultante *w* se utiliza como entrada a lo largo de toda la red y se encarga de controlar la síntesis de la imagen. De esta manera, variaciones en el vector *z* permiten controlar el estilo de la imagen, desde atributos generales hasta detalles. Además, se añade una entrada de ruido adicional, como parámetro para incrementar la variedad de las muestras generadas. En la Figura 1.12 se muestra en la arquitectura de esta red.

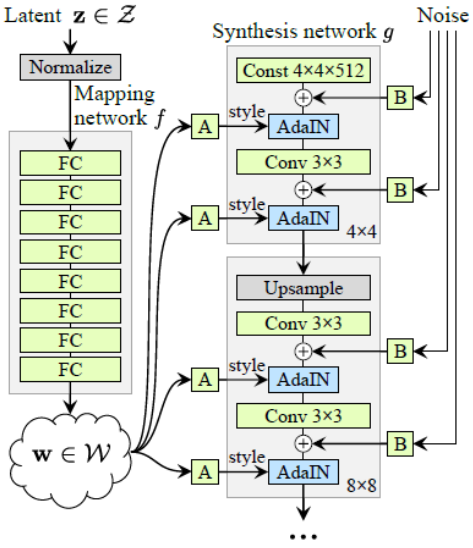


Figura 1.12. Arquitectura de red generador en modelo *StyleGAN* [20].

Posteriormente, en [2] se analizaron algunas estrategias de regularización para mejorar la calidad de las imágenes generadas y se presentó el modelo *StyleGAN2*. En él se realizan modificaciones a la arquitectura de la red generador para evitar la aparición de artefactos en las muestras artificiales (Figura 1.13). Dentro de las modificaciones se incluyen bloques Mod y Demod, los cuales se encargan de modificar los pesos de las capas convolucionales en función de la información del vector *w*. De esta manera, la influencia de este vector se refleja directamente en los parámetros de la red, a diferencia del modelo original, en donde *w* influía en una etapa de normalización.

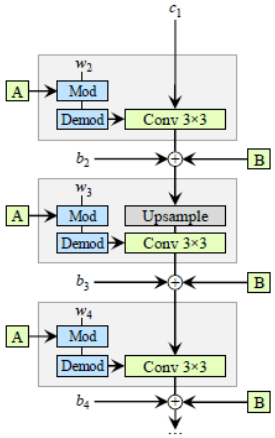


Figura 1.13. Modificaciones de la arquitectura del generador en *StyleGAN2* [2].

1.2.2.6 BigGAN

El modelo BigGAN, propuesto en [22], es un modelo GAN condicional de gran escala, entrenado con la base de datos ImageNet [23], y es capaz de generar imágenes de hasta mil categorías distintas con una resolución de hasta 512x512 pixeles. La arquitectura de la red del generador está compuesta por bloques residuales crecientes (*ResBlock-Up*), los cuales se componen por capas de: normalización de lote (*BatchNorm*); convolucionales (*Conv*); de aumento de dimensionalidad (*Upsample*); completamente conectadas (*linear*); y funciones de activación tipo *ReLU*. Los bloques *ResBlock-Up*, además de procesar la información resultante de bloques anteriores, toman una parte del vector de información latente z y la clase deseada l en forma concatenada. De esta manera, la información latente tiene influencia en todas las capas de la red, incluso en los niveles más altos, similar al modelo *StyleGAN*. Otro elemento importante en el modelo BigGAN es el uso de un bloque de tipo *self-attention* (*non-local*) [54] entre los bloques residuales cercanos a la salida. En la Figura 1.14a se muestra la composición de los bloques *ResBlock-Up* y en la Figura 1.14b se muestra la arquitectura general del generador.

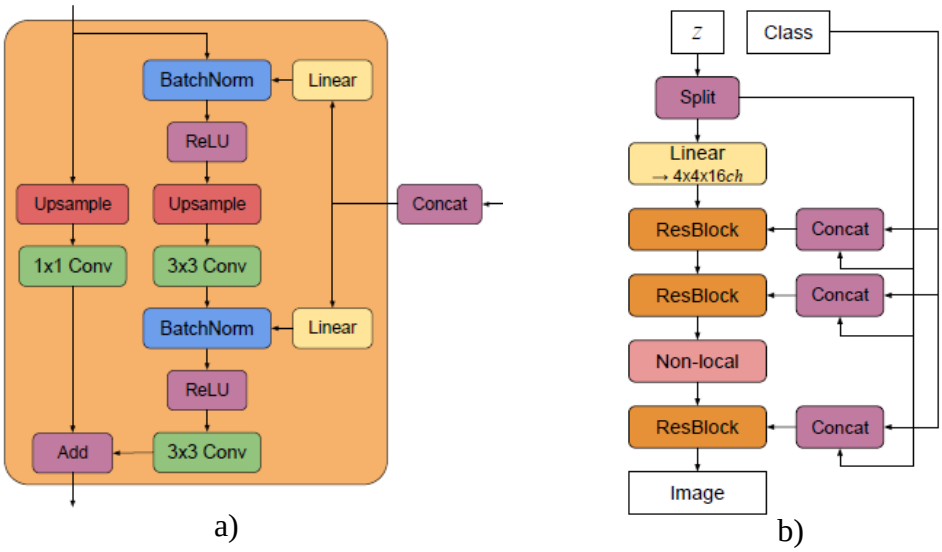


Figura 1.14. Arquitectura de generador en modelo BigGAN [22]. A) Composición de bloque *ResBlock-Up*. B) Esquema general de la arquitectura del generador y conexión de información latente a bloques residuales.

Por otro lado, la arquitectura del discriminador se compone por bloques residuales decrecientes (*ResBlock-Down*), compuestos por capas: convolucionales (*Conv*); agrupación por promedio (*Average Pooling*); y funciones de activación tipo *ReLU*. En la Figura 1.15 se muestra la arquitectura de tales bloques *ResBlock-Down.* Se utiliza un bloque de tipo *self-attention* entre los bloques residuales cercanos a la entrada, a diferencia de la red generador en donde el bloque de tipo *self-attention* se ubica cercano a la salida.

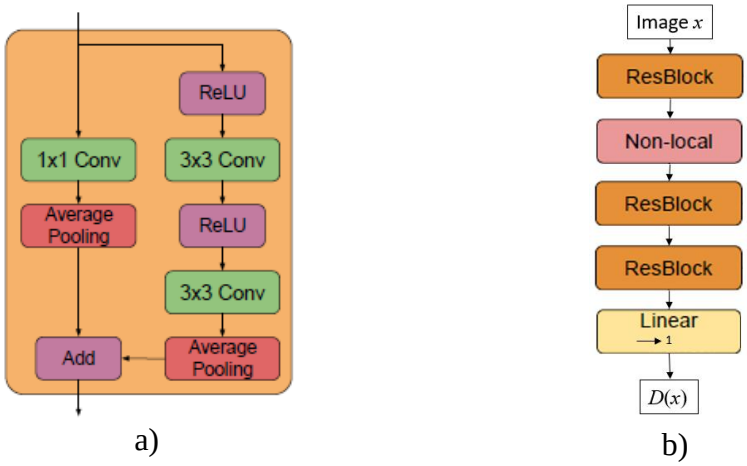


Figura 1.15. Arquitectura de discriminador en modelo BigGAN [22]. a) Composición de bloque *ResBlock-Down*, utilizado en la red discriminador del modelo BigGAN, b) Esquema general de discriminador BigGAN.

1.2.2.7 Modelo CycleGAN

El modelo CycleGAN [25] es una aplicación del esquema de entrenamiento de redes GAN para la traducción imagen-a-imagen, de tipo no emparejada. Este modelo es capaz de transformar imágenes de un dominio *A*, a un dominio *B* y viceversa. Los elementos de este modelo son:

- Generador G_{AB} es una red neuronal que realiza un mapeo de una imagen o muestra x_A en el dominio A con dimensiones $H \times W$ y Ch canales y la transforma a una imagen artificial $\hat{x}_B$ con las mismas dimensiones en un dominio B.

$$G_{AB} : x_A \rightarrow \hat{x}_B \qquad x_A \in A \quad \hat{x}_B \in B \quad (A \cup B) \in \mathbb{R}^{Ch \times H \times W} \tag{1.13}$$

- Generador G_{BA}: similar a la red anterior, transforma una muestra del dominio *B* al dominio *A*.

$$G_{BA} : x_B \rightarrow \hat{x}_A \qquad \hat{x}_A \in A \quad x_B \in B \quad (A \cup B) \in \mathbb{R}^{Ch \times H \times W} \tag{1.14}$$

- Discriminador D_A: se trata de una red neuronal, cuya entrada es una muestra cualquiera x con dimensiones $H \times W$ y Ch canales, y su salida es una matriz de dimensiones H_{Out} W_{Out}, en donde cada elemento está relacionado a la fidelidad de una porción de la imagen, respecto al dominio A.

$$D_A : x \rightarrow D_A(x) \qquad x \in (A \cup B) \quad D_A(x) \in \mathbb{R}^{H_{Out} \times W_{Out}} \quad (A \cup B) \in \mathbb{R}^{Ch \times H \times W} \tag{1.15}$$

- Discriminador D_B: al igual que el discriminador D_A, su entrada es una muestra y su salida es un valor escalar, pero el grado de fidelidad se relaciona con el dominio B.

$$D_B : x \rightarrow D_B(x) \qquad x \in (A \cup B) \quad D_B(x) \in \mathbb{R}^{H_{Out} \times W_{Out}} \quad (A \cup B) \in \mathbb{R}^{Ch \times H \times W} \tag{1.16}$$

Para entrenar este tipo de modelos, es necesario contar con una base de datos con dos conjuntos de imágenes: uno para el dominio A y otro para el dominio B. Para actualizar los parámetros de las redes generador es necesario calcular tres tipos de funciones de pérdida:

- Pérdida adversaria (Figura 1.16a): se refiere a una variante de la función de pérdida de modelos GAN convencionales en donde se utiliza una función de entropía cruzada. En el modelo CycleGAN, este valor se calcula utilizando el error cuadrático medio entre la salida del discriminador (D_A o D_B) y el valor esperado para muestras provenientes de un generador (G_{AB} o G_{BA}), como se muestra en las ecuaciones (1.17) y (1.18):

$$L_{Gen1} = \mathbb{E}\left[\left(D_A\left(G_{AB}\left(x_A\right)\right)-1\right)^2\right] \tag{1.17}$$

$$L_{Gen2} = \mathbb{E}\left[\left(D_B\left(G_{BA}\left(x_B\right)\right)-1\right)^2\right] \tag{1.18}$$

- Pérdida de consistencia cíclica (Figura 1.16b): con este valor de pérdida se pretende mejorar la capacidad de las redes generador para realizar transformaciones inversas entre sí. Utilizando el generador $G_{AB,}$ se procesa una muestra x_A perteneciente al dominio A para obtener a una muestra $\hat{x}_B$ perteneciente al dominio B; después, la muestra $\hat{x}_B$ se procesa utilizando el generador G_{BA} para regresar al dominio A. De manera ideal, la

muestra final debería ser idéntica a la muestra de entrada x_A, el cálculo de la pérdida se lleva a cabo con una diferencia pixel a pixel entre la entrada y la salida final. De manera similar, se repite el proceso utilizando una muestra x_B como entrada. El nombre del modelo CycleGAN se amerita a este tipo de pérdida. Los valores de esta pérdida se calculan con las ecuaciones (1.19) y (1.20):

$$L_{Gen3} = \left\| G_{BA}\left(G_{AB}\left(x_A\right)\right) - x_A \right\|_1 \tag{1.19}$$

$$L_{Gen4} = \left\| G_{AB}\left(G_{BA}\left(x_B\right)\right) - x_B \right\|_1 \tag{1.20}$$

- Pérdida de identidad (Figura 1.16c): este valor está relacionado con la capacidad de la red generador de identificar si la muestra de entrada ya se encuentra en el dominio deseado, es decir, si el generador G_{AB} recibe a la entrada una muestra x_B, de manera ideal la salida debería ser exactamente igual a la entrada ya que pertenece al dominio B de antemano. Para calcular el valor de esta pérdida, se procesan muestras x_B con la red G_{AB} y se calcula la diferencia entre la entrada x_B y la salida con la ecuación (1.21). Lo mismo se lleva a cabo con muestras x_A y la red G_{BA} utilizando la ecuación (1.22):

$$L_{Gen5} = \left\| G_{BA}\left(x_A\right) - x_A \right\|_1 \tag{1.21}$$

$$L_{Gen6} = \left\| G_{AB}\left(x_B\right) - x_B \right\|_1 \tag{1.22}$$

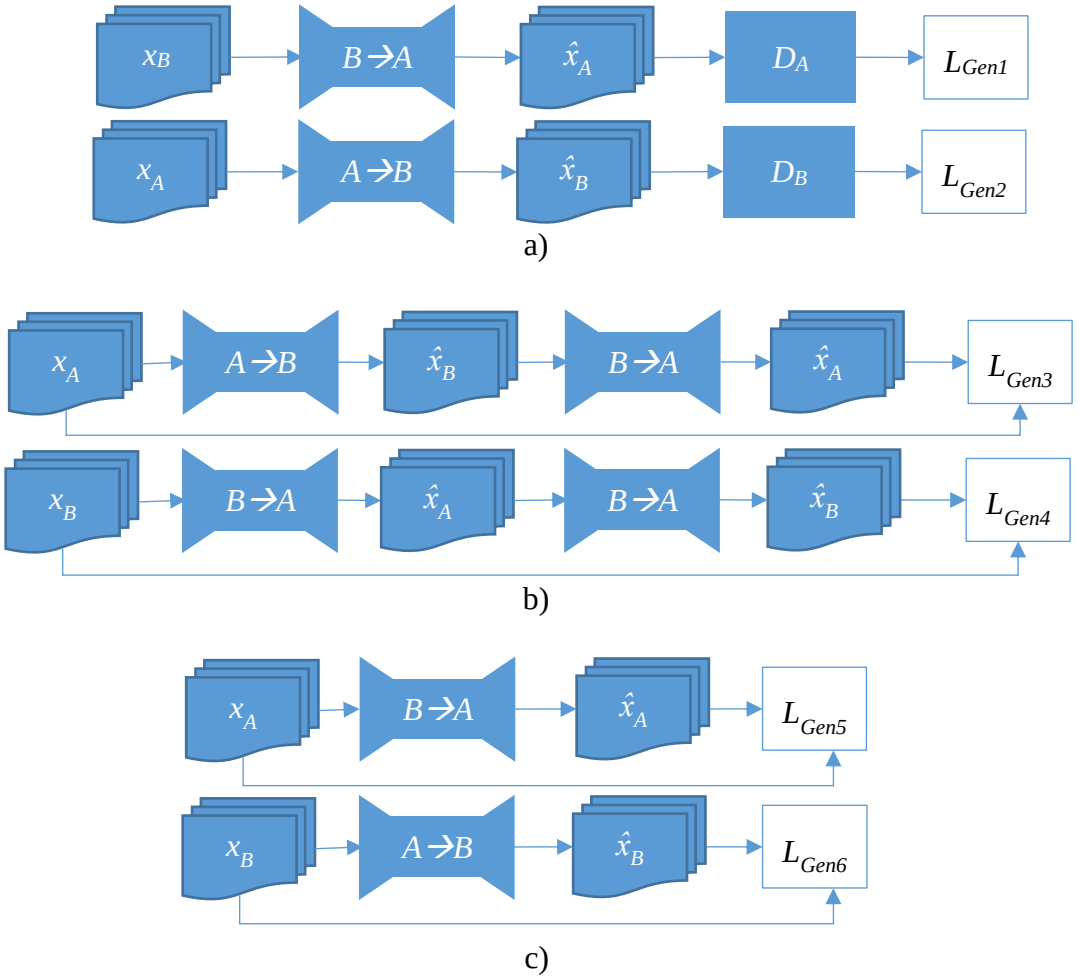


Figura 1.16. Esquema de entrenamiento de redes generador de modelo CycleGAN. a) pérdida adversaria; b) pérdida de consistencia cíclica; c) pérdida de identidad.

En donde: G_{AB} y G_{BA} son las redes generador del modelo; D_A y D_B son las redes discriminador; x_A y x_B son muestras reales de los dominios A y B respectivamente. Luego, los parámetros de las redes generador (G_{AB} y G_{BA}) se actualizan utilizando el algoritmo *backpropagation* con el valor de pérdida dado por la ecuación (1.23):

$$L_{Gen} = L_{Gen1} + L_{Gen2} + \lambda_{cycle}\left(L_{Gen3} + L_{Gen4}\right) + \lambda_{identity}\left(L_{Gen5} + L_{Gen6}\right) \tag{1.23}$$

En donde λ_{cycle} y $\lambda_{identity}$ son hiperparámetros para regular el efecto de los valores de pérdida de consistencia cíclica y de identidad en la actualización de los pesos.

Por otro lado, los parámetros de las redes discriminador (D_A y D_B) se actualizan con base en el valor de pérdida adversaria únicamente. En la Figura 1.17 se muestra el esquema para el cálculo de tal función de pérdida.

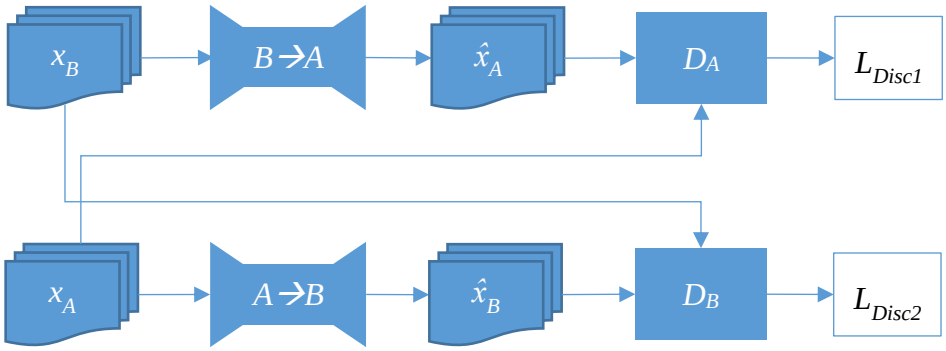


Figura 1.17. Esquema de entrenamiento de redes discriminador de modelo CycleGAN.

Los valores de L_{Disc1} y L_{Disc2}, al igual que para el entrenamiento del generador, se calculan con base en el error cuadrático medio utilizando las ecuaciones (1.24) y (1.25).

$$L_{Disc1} = \mathbb{E}\left[\left(D_A(x_A)-1\right)^2\right] + \mathbb{E}\left[\left(D_A\left(G_{BA}\left(x_B\right)\right)\right)^2\right] \tag{1.24}$$

$$L_{Disc2} = \mathbb{E}\left[\left(D_B(x_B)-1\right)^2\right] + \mathbb{E}\left[\left(D_B\left(G_{AB}\left(x_A\right)\right)\right)^2\right] \tag{1.25}$$

Finalmente, como se muestra en la ecuación (1.26), se utiliza el valor promedio entre L_{Disc1} y L_{Disc2} como función de pérdida en la actualización de los parámetros de las redes discriminador.

$$L_{Disc} = \frac{1}{2}\left(L_{Disc1} + L_{Disc2}\right) \tag{1.26}$$

1.2.3 Evaluación de modelos GAN

Los modelos GAN tienen el potencial de generar imágenes que se asemejan a la realidad. Sin embargo, la evaluación de tales modelos varía según la aplicación, ya que aún no existen métodos oficiales para la prueba de modelos GAN [14]. Una opción es mediante métodos cualitativos, como la inspección manual, en donde se toma cierto número de muestras

representativas de los datos generados para evaluar manualmente su calidad. Las desventajas de esto es que los resultados son subjetivos y el proceso es tardado.

Actualmente existen algunos métodos para llevar a cabo un análisis cuantitativo de la calidad de las muestras obtenidas a partir de un modelo GAN, como: *Inception score* (IS), *Fréchet Inception Distance* (FID), y el método GAN-train/GAN-test, entre otros. A continuación, se describen algunos detalles de los métodos mencionados.

1.2.3.1 Inception Score (IS)

La métrica *Inception score* (IS) fue propuesta en [26]como un método automático para la evaluación de muestras generadas por modelos GAN. Éste consiste en procesar las muestras con un modelo de redes profundas de tipo *Inception* [27], entrenado previamente con la base de datos *ImageNet* [23]. La salida de la red *Inception* corresponde a un vector de aproximadamente mil elementos, cada uno de ellos correspondiente al grado de pertenencia de la muestra de entrada correspondiente a alguna de las mil categorías posibles. En la ecuación (1.27) se muestra el mapeo que realiza la red *Inception*.

$$f_{Inception} : x \rightarrow p(l \mid x) \qquad x \in \mathbb{R}^{Ch \times H \times W}, \quad p(l \mid x) \in \mathbb{R}^{n_l} \tag{1.27}$$

En donde: $\boldsymbol{x}$ es una muestra con dimensiones $H \times W$ y Ch canales; y $p(l|x)$ es un vector de n_l elementos, en donde cada uno de ellos representa la probabilidad de que la muestra x pertenezca a la clase l.

Luego, los vectores de salida se analizan bajo el siguiente criterio: imágenes con mayor veracidad contienen objetos que son fácilmente identificables, de manera que la red *Inception* genera un vector de salida $p(l|x)$en donde la mayoría de los elementos son cercanos a cero y al menos un elemento tiende a ser cercano a 1, indicando la categoría a la que pertenece tal imagen; por otro lado, si la imagen presenta baja veracidad, es decir, los objetos no son cercanos a la realidad, el vector $p(l|x)$presentará valores similares en todos los elementos, indicando que no está claro a qué categoría pertenece tal muestra. De esta manera, la entropía del vector $p(l|x)$ está inversamente relacionada con la veracidad de la imagen x. Entonces, la métrica IS se calcula mediante la ecuación (1.28).

$$IS = \exp\left(\mathbb{E}_x KL\left(p\left(l \mid \boldsymbol{x}\right) \| \, p\left(l\right)\right)\right) \tag{1.28}$$

En donde: *KL* se refiere a la divergencia de Kullback-Leibler (KL); *p*(*l*) es la probabilidad marginal de la distribución de las n_l clases posibles; y $\mathbb{E}_x$ es la media de la divergencia KL a lo largo del conjunto de muestras $\boldsymbol{x}$. La divergencia KL, también conocida como entropía relativa, es una medida que indica la diferencia entre dos distribuciones de probabilidad *p* y *q* con base en la entropía cruzada, como se muestra en la ecuación (1.29).

$$KL\left(p\left(l \mid \boldsymbol{x}\right) \| \, p(l)\right) = H\left(p\left(l \mid \boldsymbol{x}\right), p\left(l\right)\right) - H\left(p\left(l \mid \boldsymbol{x}\right)\right) \tag{1.29}$$

En donde *H* se refiere a la entropía cruzada, entonces, la divergencia KL se calcula utilizando la ecuación (1.30).

$$KL\left(p\left(l \mid \boldsymbol{x}\right) \| \, p(l)\right) = \sum_{i=1}^{n_l} p\left(l_i \mid \boldsymbol{x}\right) \log \frac{1}{p\left(l_i\right)} - \sum_{i=1}^{n_l} p\left(l_i \mid \boldsymbol{x}\right) \log \frac{1}{p\left(l_i \mid \boldsymbol{x}\right)} \tag{1.30}$$

La probabilidad marginal *p(l)*, corresponde a una distribución uniforme con aproximadamente mil elementos, correspondiente a la distribución de la base de datos *ImageNet*.

Cuando $p(l|\boldsymbol{x})$ presenta baja entropía, lo cual corresponde a muestras con alta fidelidad, la métrica IS tiende a un valor alto, y el límite superior depende del número de muestras con las que se realice el cálculo.

Esta métrica presenta algunas limitaciones. Puesto que la red *Inception* se entrena previamente con *ImageNet*, es posible que esta métrica no sea útil para casos en los que las imágenes artificiales no pertenezcan a ninguna de las categorías predefinidas. También, no existe una comparación entre la distribución de las muestras reales y la distribución de las muestras generadas, la evaluación depende de la capacidad de la red para clasificar a las muestras generadas únicamente.

1.2.3.2 Frechet inception distance (FID)

Similar a IS, la métrica *Frechet Inception Distance* (FID), propuesta en [28], es una manera de analizar cuantitativamente la calidad de muestras generadas por modelos GAN. A diferencia

de IS, la métrica FID compara la distribución de las muestras generadas, con la distribución de las muestras reales. Para llevar esto a cabo, primero se procesan las imágenes, reales y generadas, utilizando una red neuronal de tipo *Inception*, al igual que en la métrica IS. Sin embargo, la información de interés en este caso se extrae de una capa intermedia de la red *Inception*, de manera que se obtienen vectores de características de 2048 elementos. En la Figura 1.18 se muestra la arquitectura de la red *Inception* y se indica la capa intermedia en donde se extrae la información.

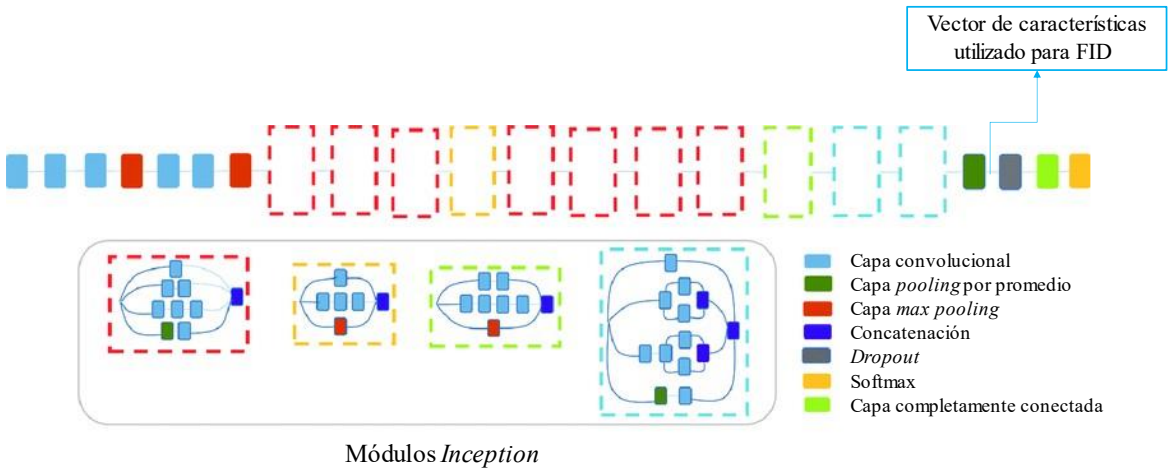


Figura 1.18. Arquitectura de la red *Inception V3*. El vector de características se obtiene de la última capa de *pooling* por promedio.

Una vez obtenidos los vectores de características, se realiza el cálculo de media y matriz de covarianza para el conjunto de muestras reales (m_{real}, C_{real}) y para el conjunto de muestras generadas (m_{fake}, C_{fake}). Finalmente, tomando en cuenta las estadísticas obtenidas, se calcula la distancia entre los dos conjuntos, asumiendo distribuciones gaussianas multivariables. Tal valor se considera como la métrica FID, y se calcula mediante la ecuación (1.31). No existe un número mínimo de muestras con las cuales calcular estos valores, pero se recomienda utilizar una cantidad tan alta como lo permita el equipo de cómputo para obtener resultados precisos.

$$FID = \left\| m_{real} - m_{fake} \right\|_2^2 + Tr\left(C_{real} + C_{fake} - 2\left(C_{real} C_{fake} \right)^{\frac{1}{2}} \right) \tag{1.31}$$

En la Figura 1.19 se muestra un diagrama con el procedimiento para el cálculo de la métrica FID.

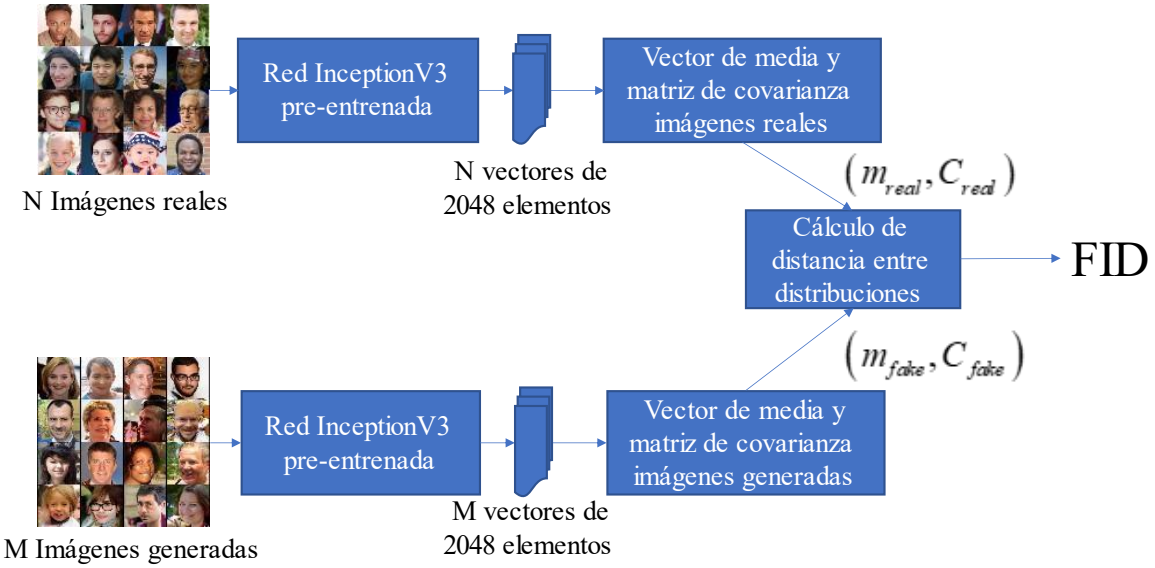


Figura 1.19. Esquema de procedimiento para el cálculo de la métrica FID.

La métrica FID presenta algunas ventajas respecto a IS, entre ellas están: la comparación con muestras reales, lo cual brinda una referencia sólida; y el uso de activaciones en capas intermedias, en lugar de la salida final de la red, permite obtener valores correspondientes a características que describan de manera abstracta a las muestras. Sin embargo, existen algunas limitaciones, como: la suposición de que los grupos cuentan con una distribución gaussiana multivariable, lo cual, en la realidad puede no ser cierto; y que, en ocasiones, es posible obtener un valor de FID favorable a pesar de que las muestras artificiales presenten ciertas perturbaciones espaciales respecto a las muestras reales [28]. A pesar de las limitaciones, IS y FID son las métricas convencionales más populares para la evaluación de imágenes generadas por modelos GAN.

1.2.3.3 GAN-train y GAN-test

En [29] se propone un método de evaluación de imágenes generadas por modelos GAN con arquitecturas condicionales, que supera las limitaciones de las métricas IS y FID. Este método consiste en el cálculo de dos métricas denominadas como *GAN-train* y *GAN-test*.

El cálculo de *GAN-train* consiste en entrenar un clasificador con imágenes generadas por la red GAN, y evaluar tal clasificador utilizando imágenes reales. Es decir, se verifica si el

clasificador es capaz de encontrar características que diferencien las muestras entre las distintas clases a partir de imágenes generadas. Entonces, un desempeño alto indicará que las imágenes generadas cuentan con suficiente diversidad para representar a cada una de las clases de la distribución real.

Por otro lado, *GAN-test* consiste en entrenar un clasificador con imágenes reales y evaluar utilizando imágenes generadas con la red GAN. El desempeño del clasificador estará relacionado con la veracidad de las imágenes generadas.

1.3 Análisis del estado del arte

En esta sección se presenta un estudio del estado del arte respecto al desarrollo reciente de métodos para el aumento de datos utilizando modelos GAN; y el desarrollo de sistemas de visión para la detección del uso de cubrebocas en imágenes.

1.3.1 Aumento de datos con modelos basados en GAN

Recientemente, se ha demostrado que los modelos de GAN son una opción conveniente para el aumento automático de datos. En [3] se utilizó un modelo para la generación de imágenes de hojas de tomate con algún desorden o enfermedad en específico, las cuales se utilizan en el entrenamiento de una red que clasifica tales desórdenes en hojas de tomate. Tal modelo se denomina DCGAN, y añade una red convolucional adicional a la arquitectura del modelo GAN original. Con este método se incrementó el número de muestras del conjunto de entrenamiento en aproximadamente un 300% y se obtuvo un desempeño superior a otros modelos similares. En [30], se utilizó un modelo DCGAN para la generación de imágenes artificiales de lesiones del hígado para entrenar un clasificador de tales tipos de lesiones. De manera cualitativa, se mostró que las muestras artificiales son altamente similares a las imágenes reales, y cuantitativamente, el uso de tales muestras artificiales mejoró el desempeño del modelo en un 7% de sensibilidad y un 4% de especificidad. En [4] se propuso un método para generar imágenes de tomografías computarizadas (CT, por sus siglas en inglés) sin contraste a partir de CT con contraste, las cuales, al utilizarse en el entrenamiento de un algoritmo de segmentación, provocaron una mejora en el valor de la métrica F1 de 0.101 a 0.747. Tal método utiliza el

modelo *CycleGAN*, el cual está basado en GAN y su objetivo es obtener la relación entre dos conjuntos de datos y aplicar tal conocimiento para la generación de nuevas imágenes a partir de muestras reales. En [31] se propone el modelo *Conditional Wasserstein GAN* (CWGAN) para la generación de señales electroencefalográficas (EEG), con el propósito de crear bases de datos para el entrenamiento de modelos de reconocimiento de emociones a partir de señales EEG.

Recientemente se han generado variantes de modelos GAN adaptables, con la capacidad de realizar transformaciones de imágenes a múltiples dominios. En [32] se presenta el modelo *StarGAN* para la traducción imagen-a-imagen múltiple en un mismo modelo. La arquitectura escalable del modelo permite generar muestras artificiales en distintos dominios a partir de una muestra existente. En específico, se aplicó el modelo para la generación de muestras artificiales de expresiones faciales, es decir, a partir de la imagen de un rostro, se generan muestras en donde se modifica la expresión facial de la persona, por ejemplo: felicidad, enojo, sorpresa, indiferencia, etc. En [33] se propone el modelo *Reconfigurable GAN* (RF-GAN) para la traducción bidireccional de imágenes entre dos o más dominios, al igual que el modelo *StarGAN*, pero con una arquitectura hasta un 75% más pequeña. De manera similar, el modelo *InjectionGAN*, propuesto en [34], se encarga de modificar atributos específicos de una imagen, en sus resultados muestran el desempeño del modelo al modificar rasgos físicos de una persona, como el peinado, color de cabello, edad, entre otros. Por otro lado, en [35] se propone un modelo GAN orientado al aumento de datos, en donde se utiliza la arquitectura *DenseNet* [36] como red discriminadora, y una combinación de los modelos *ResNet* [37] y *U-Net* [38] como generador. El trabajo está relacionado a un enfoque de meta-aprendizaje con base en modelos (*model-based meta-learning*), en donde se intenta resolver un problema específico, tomando en cuenta el conocimiento previo de otros problemas similares.

A pesar del potencial que tienen los modelos GAN para generar imágenes con algún grado de realismo, es posible que algunas muestras artificiales no sean útiles para el entrenamiento de modelos de aprendizaje profundo u otros modelos basados en datos. En [39] se analizan las limitantes respecto al uso de muestras artificiales en el entrenamiento de clasificadores, en específico se utiliza la red generativa del modelo *BigGAN* [22] para producir imágenes

artificiales, las cuales, se utilizan en el entrenamiento del modelo *ResNet-50*. El análisis del desempeño del clasificador indica que, utilizando únicamente muestras artificiales durante el entrenamiento, el error del clasificador aumenta de 25.99% a un 57.35%. Posteriormente, en [40], se proponen algunos métodos de uso estratégico de muestras artificiales, generadas con el modelo *BigGAN*, para el entrenamiento de clasificadores. Estas estrategias permiten obtener un desempeño equiparable al de un clasificador entrenado únicamente con muestras reales (88% de exactitud aproximadamente).

Los modelos GAN requieren de una etapa de entrenamiento para que el generador sea capaz de crear los resultados deseados, lo cual implica la necesidad de un conjunto de datos de entrenamiento. Esto se convierte en una desventaja cuando se pretende utilizar estos modelos para el aumento de datos, ya que, de antemano, no se cuenta con una gran cantidad de datos. Para afrontar estas situaciones, es posible utilizar técnicas de transferencia de aprendizaje [41], en donde se utiliza la experiencia de un modelo de aprendizaje de máquina en otro modelo con propósito similar. A partir de lo anterior, surgen otro tipo de métodos para el entrenamiento de modelos de aprendizaje profundo, como *Few Shot Learning* (FSL), cuyo propósito es lograr el aprendizaje a partir de pocas muestras, y conocimiento previo. Algunos enfoques de FSL de acuerdo con [42] son: el aumento de datos; la manipulación del modelo; y el mejoramiento de algoritmos de aprendizaje. Otros métodos similares son *One Shot Learning* (OSL) y *Zero Shot Learning* (ZSL), los cuales se refieren a la cantidad específica de muestras con las que se modela la distribución de los datos. En [43] se propone un método de *ZSL* basado en GAN, denominado GT-GAN para generar muestras artificiales de una clase sin utilizar datos de ésta en el entrenamiento. A pesar de que estas muestras artificiales tienen un impacto positivo en el entrenamiento de clasificadores, los autores mencionan que algunos atributos no son representados correctamente, afectando la veracidad de las imágenes.

1.3.2 Algoritmos para la detección del uso de cubrebocas

En el mes de marzo del año 2020 se declaró un estado de pandemia debido a la enfermedad causada por Covid-19. Desde entonces, el uso de cubrebocas se ha convertido en una de las medidas preventivas más importantes. Sin embargo, el monitoreo de medidas de prevención de

contagio es una ardua tarea para ser llevada a cabo por un ser humano. Debido a esto, se han desarrollado diversos sistemas de visión para la detección de uso de cubrebocas en diversos ambientes. En [44] se propone un modelo que consiste en tres etapas: la primera corresponde a una red profunda para localizar a la persona o personas presentes en la imagen; la segunda etapa consiste en la extracción de características de la región del rostro de la persona, utilizando un análisis de color; finalmente, se realiza la clasificación utilizando un modelo de máquina de vectores de soporte (SVM). En [45] se utiliza una red neuronal pre-entrenada, denominada *InceptionV3*, para identificar personas en la imagen y se añaden capas adicionales para la detección de uso de cubrebocas. En [46] se desarrolló el sistema para la detección de cubrebocas a partir de la CNN denominada *YOLOv3* y se llevó a cabo una recolección de imágenes de manera manual para generar un conjunto de datos de entrenamiento para el modelo. Para incrementar la robustez de la detección de cubrebocas y mejorar el desempeño del sistema para aplicaciones en tiempo real, en [47] se utiliza la CNN *YOLOv4*, con la cual se alcanza una exactitud del 98.3% en reconocimiento de uso de cubrebocas, y es posible procesar hasta 54 cuadros por segundo. Este tipo de modelos se han implementado como parte de sistemas más grandes, en [48] se presenta la integración de un sistema de Internet de las cosas (IoT, por sus siglas en inglés) para el monitoreo de temperatura y detección del uso de cubrebocas. El sistema de visión correspondiente utiliza varios modelos pre-entrenados, como *VGG-16*, *MobileNetV2*, *Inception V3*, etc. Además de detectar el uso correcto e incorrecto de cubrebocas, el sistema es capaz de clasificar el tipo de cubrebocas utilizado, como N-95 o cubrebocas quirúrgico. En [49] se utiliza un modelo de redes convolucionales en cascada [50] para la detección de rostros en una imagen, y posteriormente se analiza cada rostro para identificar el uso de cubrebocas mediante: una red de super resolución, la cual se encarga de ajustar las dimensiones de la imagen de cada rostro manteniendo la mayor cantidad de información posible; y el modelo *MobileNetV2*, cuya arquitectura fue diseñada para su implementación en dispositivos con bajos recursos. En [51] se utilizan los modelos *ResNet50* y *YOLOv2* para la extracción de características y clasificación de uso de cubrebocas, respectivamente. Otros trabajos similares se encuentran en [52]–[54]. Sin embargo, de acuerdo con [55], las bases de datos utilizadas en los trabajos reportados en la literatura varían, lo cual dificulta la comparación del desempeño entre modelos.

1.3.3 Bases de datos enfocadas a detección del uso de cubrebocas

La mayoría de los algoritmos para la detección de uso de cubrebocas se basan en técnicas de aprendizaje de máquina y aprendizaje profundo. Por ello, se ha trabajado en la generación de bases de datos para el entrenamiento de modelos de este tipo. Algunas contienen imágenes reales de personas utilizando cubrebocas. En [57] se documenta la base de datos *Properly Wearing Masked Face Detection Dataset* (PWMFD), la cual contiene un total de 9205 imágenes con al menos un rostro. Cada uno de estos rostros se considera como una muestra, y en total se cuenta con 18532 muestras, las cuales pertenecen a alguna de las tres clases posibles: *WithoutMask*, correspondiente a un rostro sin cubrebocas (10,471 muestras); *WithMask*, correspondiente a un rostro utilizando cubrebocas correctamente (7,695 muestras); e *IncorrectMask*, correspondiente a un rostro utilizando cubrebocas de manera incorrecta, es decir, con nariz y/o boca descubierta (366 muestras). La base de datos *FaceMaskDetection* [57], contiene 853 imágenes con al menos un rostro perteneciente a tres categorías posibles, al igual que PWMFD. Por otro lado, la base de datos *Masked Face Detection In the Wild* [58] cuenta con 16,228 imágenes con al menos un rostro perteneciente a dos categorías: rostro con cubrebocas; y rostro sin cubrebocas. En este caso, no se considera la clase correspondiente a un cubrebocas mal puesto.

Existen otras con imágenes artificiales, como *Simulated Masked Face* [59], en donde se añade un cubrebocas a imágenes de rostros, provenientes de otras bases de datos. En total, esta base de datos contiene aproximadamente 500,000 muestras con dos clases distintas: con cubrebocas y sin cubrebocas. Cabani *et. al* [60] generaron la base de datos *MaskedFace-Net*, la cual contiene imágenes de personas utilizando cubrebocas de manera correcta e incorrecta, las imágenes fueron generadas a partir de la base de datos con imágenes de rostros *Flickr-Faces-HQ3*, con más de 130,000 muestras. Sin embargo, algunas de estas bases de datos solo incluyen imágenes con un mismo estilo de cubrebocas y otras contienen relativamente pocas imágenes para el entrenamiento de una red profunda, especialmente en la clase de uso incorrecto de cubrebocas, la cual, en la base de datos PWMFD corresponde a menos del 2% de las muestras. Además, aún no se han desarrollado bases de datos enfocadas al uso de cubrebocas utilizando métodos basados en GAN, lo cual se aborda en este trabajo de investigación.

1.4 Conclusiones

En las últimas décadas, ha incrementado el uso de modelos de redes neuronales profundas para el procesamiento de imágenes. Conforme aumenta la complejidad de tales modelos, surgen nuevas necesidades, como una mayor capacidad de procesamiento, y bases de datos de entrenamiento con una cantidad suficiente de información. Puesto que estos modelos, generalmente dependen directamente del proceso de entrenamiento, se han desarrollado técnicas para la mejora de datos, como el aumento de datos, en donde se generan nuevas muestras a partir de datos existentes utilizando distintas transformaciones, o incluso se sintetizan muestras artificiales que pertenezcan a la distribución de los datos. Los modelos GAN son herramientas útiles para la generación de imágenes artificiales y recientemente han sido utilizados para el aumento de datos en aplicaciones específicas. Sin embargo, no se ha reportado en la literatura una metodología general para realizar aumento de datos utilizando este tipo de modelos. En los siguientes capítulos se presenta el diseño, implementación, y evaluación de un método para la generación de bases de datos artificiales utilizando redes GAN.

CAPÍTULO II. DISEÑO DE MÉTODO DE AUMENTO DE DATOS GDAM-GAN

Los modelos cuyos parámetros son dependientes de un conjunto de datos, también conocidos como modelos de tipo *data-driven*, en ocasiones requieren de una gran cantidad de muestras de entrenamiento para alcanzar un desempeño óptimo. Sin embargo, dependiendo de la aplicación, los datos pueden ser difíciles de obtener. Una manera de incrementar la cantidad y variedad de muestras de entrenamiento es utilizando técnicas de aumento de datos, las cuales pueden consistir en transformaciones a imágenes existentes, o la generación de nuevas imágenes utilizando modelos de redes neuronales profundas para la síntesis de imágenes, como la red generadora de un modelo basado en GAN. En este capítulo se muestra la principal contribución de este proyecto de investigación, correspondiente al desarrollo de un método de aumento de datos basado en modelos de redes GAN, llamado GDAM-GAN. Se presentan también algunos detalles generales sobre la implementación de tal método en distintas aplicaciones.

2.1 Definición de GDAM-GAN

En la sección 1.3.1 se mencionan algunos modelos basados en GAN para el aumento de datos, los cuales, generalmente, son desarrollados para alguna aplicación específica. El desarrollo de un modelo GAN genérico, es decir, capaz de generar imágenes de cualquier clase, podría implicar la necesidad de conocer la distribución de cada categoría posible. Sin embargo, definir el número de clases de un modelo GAN genérico para el aumento de datos no es una tarea trivial. Al no tener la delimitación de una aplicación específica, el número de clases que un modelo genérico de aumento de datos debería ser capaz de generar tiende al infinito, lo cual es poco factible de implementar debido a las limitaciones de equipo de procesamiento y disponibilidad de datos de entrenamiento. Por esta razón, en lugar de trabajar en un único modelo para la generación de datos, se propone GDAM-GAN, enfocada a ser independiente de la aplicación de los datos, tomando en cuenta las siguientes consideraciones:

- Existe un conjunto de muestras reales $\boldsymbol{x}_T$ pertenecientes a una clase deseada l_T (es decir, la categoría de la cual se desea obtener más muestras). La cantidad de muestras es relativamente baja pero nunca nula.

- Existe una clase base l_S que comparte algunas características generales con la clase deseada l_T, pero que difieren en características propias de cada clase. Por ejemplo, una característica general podría ser la forma de los objetos, y una característica propia podría referirse al color o textura.
- Existe una gran cantidad de muestras $\boldsymbol{x_S}$ de la clase base l_S (respecto a la cantidad de muestras de la clase deseada l_T) y/o existe algún modelo generativo capaz de sintetizar muestras de l_S.

Los detalles respecto al número específico de muestras reales de la clase deseada y clase base dependerán de la aplicación, sin embargo, la idea general de cada consideración deberá cumplirse.

En la Figura 2.1 se muestra el esquema de GDAM-GAN. La idea general es realizar una transformación de muestras de la clase base l_S a muestras con características propias de la clase deseada l_T, pero que además tienden a incorporar en ellas la variedad dada por la gran cantidad disponible de muestras de l_S.

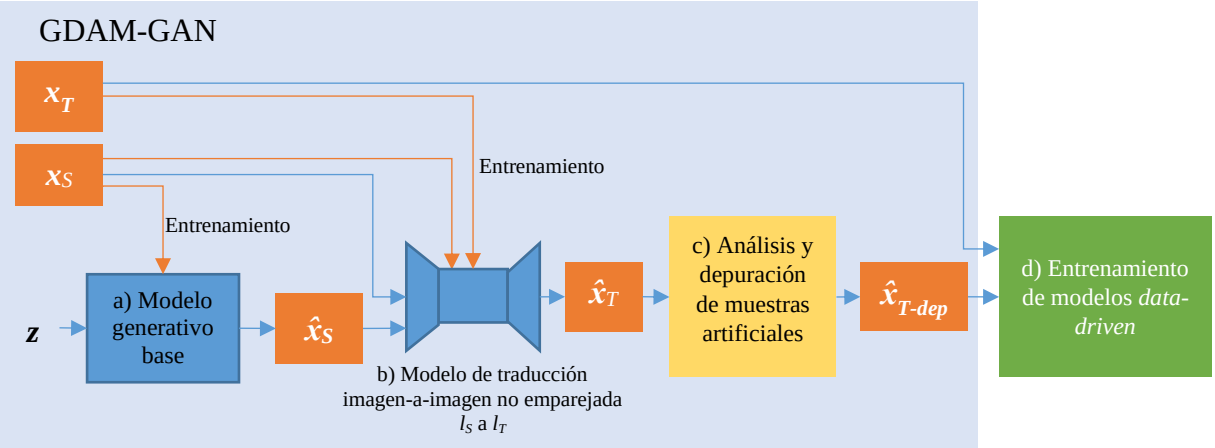


Figura 2.1. Esquema general de GDAM-GAN.

El primer elemento del método (Figura 2.1a) es el uso de una red generador de algún modelo GAN o variante de GAN. Este se entrena utilizando las muestras reales de la clase base $\boldsymbol{x_S}$ con el propósito de generar muestras artificiales $\boldsymbol{\hat{x}_S}$ de la clase l_S. De esta manera, el potencial del método para generar imágenes incrementa considerablemente, ya que, en teoría, este modelo generativo base podría generar un número infinito de imágenes. El segundo elemento (Figura

2.1b) se compone por un modelo de traducción imagen-a-imagen no emparejada, para llevar a cabo la transformación entre clases l_S y l_T. El modelo se entrena con las muestras reales $\boldsymbol{x_T}$ y $\boldsymbol{x_S}$, y se busca identificar aquellos elementos que son comunes entre ambas clases, así como las características propias de cada una de ellas, para llevar a cabo la transformación correspondiente. Este modelo produce un conjunto de muestras artificiales $\hat{\boldsymbol{x}}_{\boldsymbol{T}}$ de la clase deseada l_T, a partir de muestras de la clase base l_S, las cuales pueden ser reales $\boldsymbol{x_S}$ y/o artificiales $\hat{\boldsymbol{x}}_{\boldsymbol{S}}$. Cabe mencionar que, al tratarse de muestras artificiales sintetizadas mediante un esquema de redes GAN, es posible que una porción considerable de las muestras artificiales, tanto de $\hat{\boldsymbol{x}}_{\boldsymbol{T}}$ como de $\hat{\boldsymbol{x}}_{\boldsymbol{S}}$, no sean cualitativamente satisfactorias. Por esta razón, el tercer elemento del método (Figura 2.1c) corresponde al análisis y depuración de las muestras $\hat{\boldsymbol{x}}_{\boldsymbol{T}}$. Finalmente, las muestras artificiales depuradas se utilizan en el entrenamiento de modelos que dependan de datos (Figura 2.1d), como las redes neuronales profundas.

En las siguientes secciones se describen a detalle los elementos principales de GDAM-GAN.

2.2 Modelo generativo base

El elemento inicial de GDAM-GAN es un modelo generativo base, el cual tiene la finalidad de generar imágenes artificiales de la clase base l_S para incrementar la cantidad de muestras en tal dominio. Este modelo generativo base permite obtener imágenes totalmente sintéticas, que son acordes a la distribución de las muestras reales de la clase base. Además, dependiendo de la aplicación, mejora aspectos de privacidad, por ejemplo, si el problema requiere de la generación de rostros, este modelo permite sintetizar imágenes de rostros humanos que no corresponden a un individuo en específico.

El modelo generativo base consiste en alguna red generador de un modelo basado en GAN, que haya sido entrenado con muestras reales de la clase base l_S. La arquitectura de tal red puede ser modificada de acuerdo con el problema que se desee afrontar, es decir, el método no se limita a una resolución específica de imágenes o a una red específica con un cierto grado de complejidad. De manera que es posible utilizar alguna de las diversas arquitecturas GAN existentes en la literatura, que incluso han sido entrenadas con bases de datos de gran escala y que se encuentran disponibles en línea para fines académicos. Algunos de los modelos de

vanguardia más populares son *StyleGAN2* y *BigGAN*, y pueden ser una alternativa útil para el modelo generativo base si se cuenta con una plataforma de hardware capaz de procesar datos utilizando tales arquitecturas.

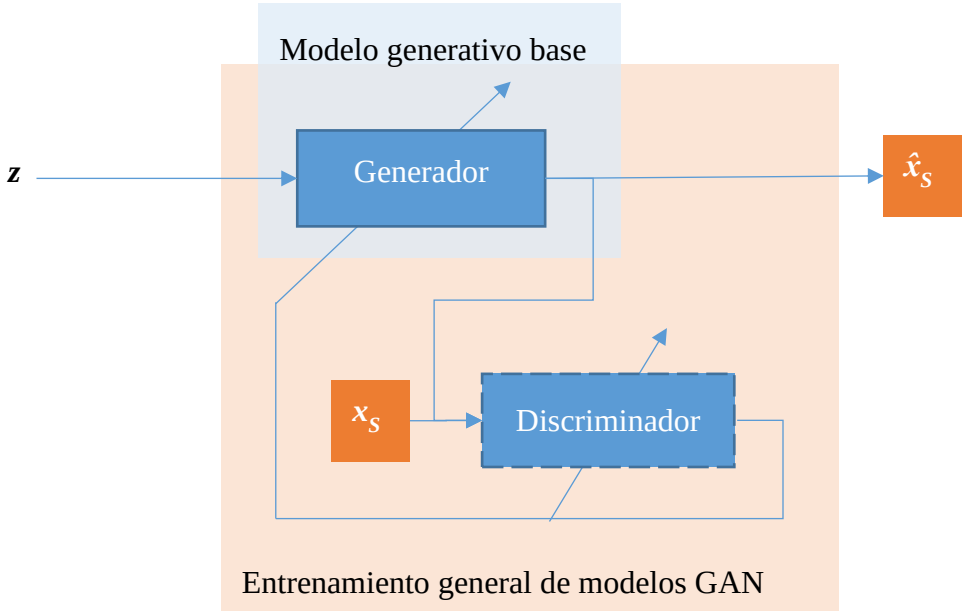


Figura 2.2. Esquema de elementos relacionados a un modelo generativo base.

En este proyecto de investigación se realiza la implementación del modelo generativo base utilizando una arquitectura WGAN, la cual se eligió con base en un análisis preliminar del costo computacional entre distintos modelos (DCGAN, WGAN, ProGAN, y StyleGAN2). Las redes crítico (discriminador) y generador realizan las transformaciones descritas por las ecuaciones (2.1) y (2.2).

$$D_C : \mathbb{R}^{N\times 3\times 128\times 128} \rightarrow \mathbb{R}^{N\times 1} \tag{2.1}$$

$$G : \mathbb{R}^{N\times 128} \rightarrow \mathbb{R}^{N\times 3\times 128\times 128} \tag{2.2}$$

Se modifica ligeramente la arquitectura del modelo WGAN original, añadiendo una capa convolucional adicional, para procesar imágenes con una resolución espacial de 128x128 pixeles. Se eligieron tales dimensiones de imagen con el propósito de tener un balance entre la calidad de las muestras con las que se trabaja y la carga computacional, ya que se ha comprobado que aumentar la resolución espacial de las imágenes artificiales permite generar

muestras con mayor fidelidad y por ende, mayor veracidad [22]. Sin embargo, para lograr lo anterior es necesario incrementar la complejidad del modelo y, por ende, la capacidad de procesamiento requerida. El crítico toma *N* imágenes en el espacio de color RGB, es decir, de tres canales, y con dimensiones de 128x128 pixeles y regresa *N* escalares que indican el grado en que una muestra es real o artificial. Mientras que el generador toma como entrada *N* vectores de 128 elementos con ruido aleatorio bajo una distribución normal $\mathcal{N}(0,1)$, y la salida consiste en *N* imágenes en el espacio de color RGB con dimensiones de 128x128. En la Tabla 2.1 se describe la arquitectura de la red discriminador (o crítico) y en la Tabla 2.2 se presenta la arquitectura de la red generador. En ambos casos se resaltan las capas que se añaden a la arquitectura para trabajar con muestras de la resolución espacial mencionada.

Tabla 2.1. Arquitectura de red discriminador (o crítico) de modelo WGAN implementado. Para las capas convolucionales (conv) se indica, respectivamente, el número de filtros, las dimensiones del kernel de los filtros, el valor de *stride*, y *padding.*

Descripción de capa	**Dimensiones de salida**
Entrada (*N* imágenes)	*N* x 3 x 128 x 128
Conv, 64, 4, 2, 1	
LeakyReLU	*N* x 64 x 64 x 64
Conv, 128, 4, 2, 1	
LayerNorm	
LeakyReLU	*N* x 128 x 32 x 32
Conv, 256, 4, 2, 1	
LayerNorm	
LeakyReLU	*N* x 256 x 16 x 16
Conv, 512, 4, 2, 1	
LayerNorm	
LeakyReLU	*N* x 512 x 8 x 8
Conv, 512, 4, 2, 1	
LayerNorm	
LeakyReLU	*N* x 512 x 4 x 4
Conv, 1, 4, 1, 0	*N* x 1

Para determinar cuantitativamente la veracidad de las imágenes que genera este modelo a lo largo del proceso de entrenamiento se utiliza la métrica FID, cuyos detalles se mencionaron en la sección 1.2.3. Se espera observar un descenso en el valor de la métrica FID conforme se lleva a cabo el entrenamiento y se tomarían en cuenta los pesos de la red generador cuando se alcanza un valor mínimo de FID, el cual puede variar según la aplicación.

Tabla 2.2. Arquitectura de red generador de modelo WGAN implementado. Para las capas de convolución transpuesta (TransposedConv) se indica, respectivamente, el número de filtros, las dimensiones del kernel de los filtros, el valor de *stride*, y *padding*

Descripción de capa	Dimensiones de salida
Entrada: ruido aleatorio $\boldsymbol{z}$	N x 128
TransposedConv, 512, 4, 1, 0	
BatchNorm	
ReLU	N x 1024 x 4 x 4
TransposedConv, 512, 4, 2, 1	
BatchNorm	
ReLU	N x 512 x 8 x 8
TransposedConv, 256, 4, 2, 1	
BatchNorm	
ReLU	N x 256 x 16 x 16
TransposedConv, 128, 4, 2, 1	
BatchNorm	
ReLU	N x 128 x 32 x 32
TransposedConv, 64, 4, 2, 1	
BatchNorm	
ReLU	N x 64 x 64 x 64
TransposedConv, 3, 4, 2, 1	
Tanh	N x 3 x 128 x 128

2.3 Modelos de traducción imagen-a-imagen no emparejada

El siguiente elemento de GDAM-GAN, es un modelo de traducción imagen-a-imagen no emparejada. Los modelos de traducción imagen-a-imagen tienen como propósito realizar una transformación de una imagen en un dominio X_S a un dominio distinto X_T, conservando características generales que son comunes entre ambos dominios, como la forma de algún objeto, y modificando atributos específicos de cada dominio, como el color o la textura [25]. Estos modelos requieren de un conjunto de muestras de cada dominio para su entrenamiento, y se dividen en dos tipos: traducción emparejada y no emparejada. En la traducción emparejada (Figura 2.3a), el conjunto de entrenamiento se compone por n pares de muestras $\{x_S, x_T\}$ en donde $x_T \in X_T$ corresponde a la traducción deseada a partir de la muestra $x_S \in X_S$. Por otra parte, en la traducción no emparejada (Figura 2.3b), los datos de entrenamiento se componen por dos conjuntos, uno compuesto por n muestras $\boldsymbol{x}_S \in X_S$, y el otro contiene m muestras $\boldsymbol{x}_T \in X_T$. En este caso, a diferencia de la traducción emparejada, no existe una asociación entre las muestras de ambos dominios, es decir, no se conoce la traducción deseada para cada una de las muestras en

específico, el modelo infiere la transformación adecuada según las características del conjunto de muestras.

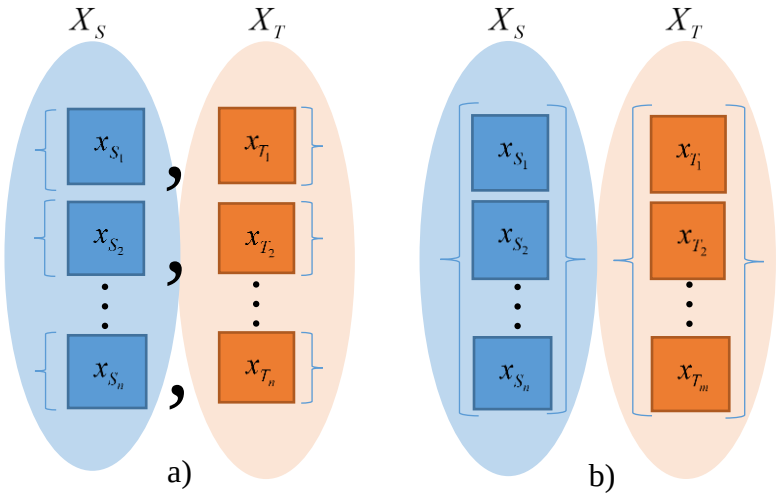


Figura 2.3. Estructura de conjuntos de entrenamiento para modelos de traducción de imágenes. a) Emparejada, b) No emparejada.

Este modelo se utiliza para transformar imágenes de la clase base l_S, a la clase deseada l_T. En esta tesis, se trabaja con traducción de imagen-a-imagen no emparejada, ya que los conjuntos de datos reales de la clase base y la clase deseada, generalmente, no tienen una correspondencia entre las muestras de cada clase. La arquitectura específica para esta etapa se denomina *CycleGAN*, y fue elegida luego de analizar el costo computacional de modelos de traducción no emparejada existentes en la literatura. El esquema de la arquitectura de las redes generador y discriminador de este modelo se muestran en la Figura 2.4 y Figura 2.5, respectivamente.

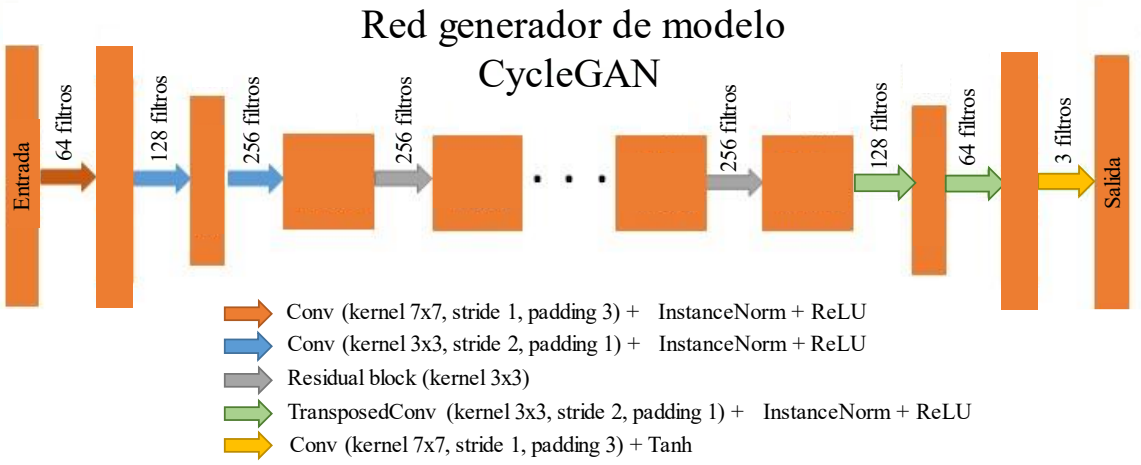


Figura 2.4. Esquema de arquitectura de red generador en modelo CycleGAN.

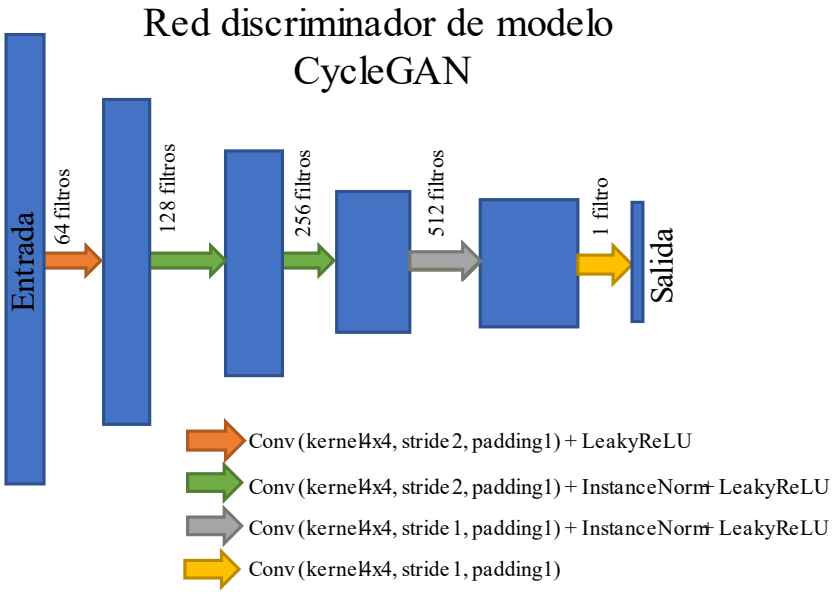


Figura 2.5. Esquema de arquitectura de red discriminador en modelo CycleGAN.

Estas arquitecturas pueden procesar imágenes con una resolución espacial mínima de 24x24 pixeles. En los experimentos que se llevan a cabo en esta tesis, para la etapa de traducción imagen-a-imagen, se ajustan las muestras a una resolución espacial de 256x256 pixeles. Estas dimensiones se eligieron acorde con los experimentos que se reportan en [25], en donde se indica que los mejores resultados se obtienen al entrenar el modelo *CycleGAN* con imágenes de mayor resolución. Los detalles de la arquitectura de las redes discriminador y generador del modelo CycleGAN, tomando muestras con una resolución espacial de 256x256 pixeles se

muestran en la Tabla 2.3 y Tabla 2.4, respectivamente. En este caso no se realizaron modificaciones a las arquitecturas.

Tabla 2.3. Arquitectura de red discriminador de modelo CycleGAN implementado (tomando en cuenta una resolución espacial de 256x256 pixeles). Para las capas convolucionales (conv) se indica, respectivamente, el número de filtros, las dimensiones del kernel de los filtros, el valor de *stride*, y de *padding.*

	Descripción de capa	**Dimensiones de salida**
	Entrada (*N* imágenes)	*N* x 3 x 256 x 256
1	Conv, 64, 4, 2, 1	
2	LeakyReLU	*N* x 64 x 128 x 128
3	Conv, 128, 4, 2, 1	
4	InstanceNorm	
5	LeakyReLU	*N* x 128 x 64 x 64
6	Conv, 256, 4, 2, 1	
7	LayerNorm	
8	LeakyReLU	*N* x 256 x 32 x 32
9	Conv, 512, 4, 1, 1	
10	LayerNorm	
11	LeakyReLU	*N* x 512 x 31 x 31
12	Conv, 1, 4, 1, 1	*N* x 1 x 30 x 30

Tabla 2.4. Arquitectura de red generador de modelo CycleGAN implementado. Para las capas convolucionales (*conv* y *transposedconv*) se indica, respectivamente, el número de filtros, las dimensiones del kernel de los filtros, el valor de *stride*, y de *padding.*

		Descripción de capa	**Dimensiones de salida**
		Entrada (*N* imágenes)	*N* x 3 x 256 x 256
1		Conv, 64, 7, 1, 3	
2		InstanceNorm	
3		ReLU	*N* x 64 x 256 x 256
4		Conv, 128, 3, 2, 1	
5		InstanceNorm	
6		ReLU	*N* x 128 x 128 x 128
7		Conv, 256, 3, 2, 1	
8		InstanceNorm	
9		ReLU	*N* x 256 x 64 x 64
.	×9	Bloque residual [37] (3x3)	*N* x 256 x 64 x 64
19		TransposedConv, 128, 3, 2, 1	
20		InstanceNorm	
21		ReLU	*N* x 128 x 128 x 128
22		TransposedConv, 64, 3, 2, 1	
23		InstanceNorm	
24		ReLU	*N* x 64 x 256 x 256
25		Conv, 3, 7, 1, 3	
26		Tanh	*N* x 3 x 256 x 256

Durante el entrenamiento existen cuatro redes en total (Figura 2.6): un discriminador para la clase l_T; un discriminador para la clase l_S; un generador que realiza la transformación de clase l_T a clase l_S; y un generador de clase l_S a clase l_T. Los detalles sobre el proceso de entrenamiento de este modelo se mencionan en la sección 1.2.2.7. La red que realiza la transformación de la clase *ls* a l_T es el elemento de interés para el método de aumento de datos. Y, al igual que para el modelo generativo base, la evaluación de los resultados generados por este modelo se lleva a cabo utilizando la métrica FID. La métrica se calcula respecto al conjunto de muestras reales de la clase deseada $\boldsymbol{x}_T$. Al igual que para el modelo generativo base, se busca observar una disminución en este valor conforme avanza el proceso de entrenamiento y se toman en cuenta los pesos de la arquitectura del generador que realiza la transformación de clase l_S a l_T cuando la métrica FID alcanza un valor mínimo. Adicionalmente y de manera opcional, se podría realizar un análisis cualitativo mediante inspección visual de hasta el 10% de las muestras, para determinar de manera estadística el porcentaje de imágenes generadas que pertenecen a la clase deseada. Para llevar a cabo tal inspección visual, es necesario definir algunos criterios para considerar que las muestras artificiales pertenecen a la clase deseada. Estos criterios dependerán de la aplicación específica y se recomienda que sean definidos por algún experto.

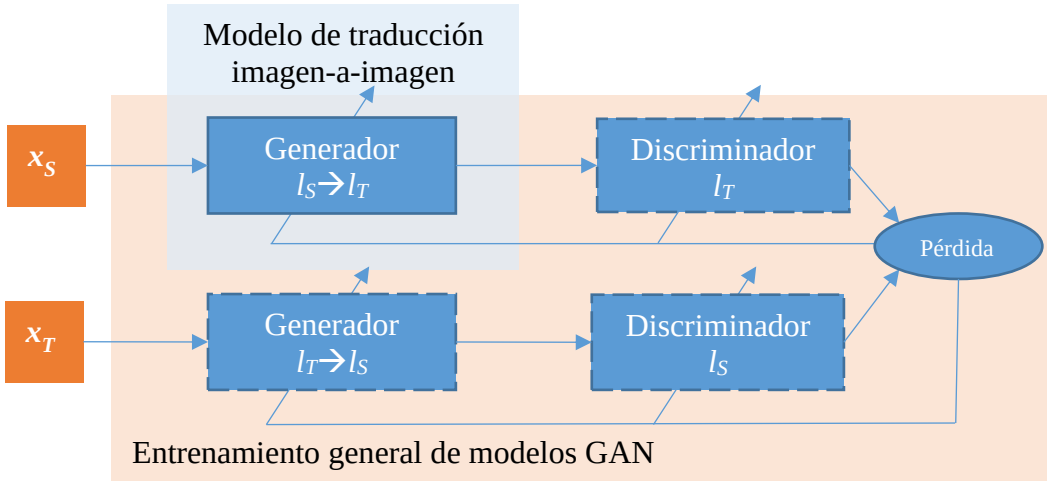


Figura 2.6. Esquema general de entrenamiento de modelo de traducción imagen-a-imagen.

Posteriormente, con el objetivo de eliminar muestras artificiales $\hat{\boldsymbol{x}}_T$ no útiles, se define una etapa de análisis y depuración mediante un algoritmo de agrupamiento. Tal proceso se describe en la siguiente sección.

2.4 Análisis y depuración de muestras artificiales

La tercera etapa de GDAM-GAN consiste en analizar y depurar las muestras artificiales $\hat{\boldsymbol{x}}_{\boldsymbol{T}}$ para descartar aquellas muestras cuya calidad no sea adecuada para construir una base de datos. Determinar si una muestra es útil no es una tarea trivial, puesto que podría depender de criterios subjetivos que varían según la aplicación específica. Además, al contar con un gran número de datos, analizar manualmente la totalidad de las muestras artificiales podría no ser factible, ya que requiere de la intervención de un inspector y podría ser necesario invertir una gran cantidad de tiempo. Por ello es necesario utilizar algún método de análisis automático, el cual sea independiente de la aplicación.

Es posible encontrar estructuras dentro de un conjunto de datos, utilizando distintos algoritmos de agrupamiento, como *k-means* [61]. Mediante estos algoritmos, es posible separar los datos artificiales en distintos grupos que compartan ciertas características y así, facilitar la selección de las muestras útiles para construir una base de datos. En la Figura 2.7 se muestra el proceso propuesto para analizar imágenes artificiales utilizando un algoritmo de agrupamiento. Primero se lleva a cabo una extracción de características (Figura 2.7a), con la finalidad de trabajar con datos en un espacio de menor dimensión y con información que sea más relevante que los valores originales de cada píxel en la imagen. Para obtener los vectores de características, una opción es utilizar algún modelo de red profunda pre-entrenado, como InceptionV3 [27]. Después, se aplica algún método de reducción de dimensiones (Figura 2.7b), como PCA, en caso de ser necesario para reducir carga computacional. Posteriormente, se ejecuta un algoritmo de agrupamiento (Figura 2.7c) para separar las muestras en múltiples grupos (Figura 2.7d). Y finalmente, se seleccionan aquellos grupos cuyas muestras cumplan con los criterios necesarios, según la aplicación, para construir la base de datos.

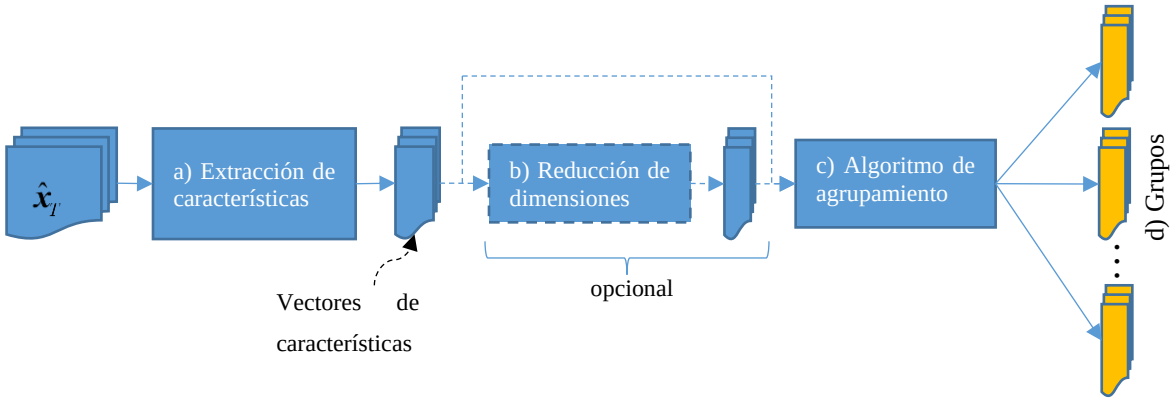


Figura 2.7. Proceso de análisis de muestras artificiales utilizando un algoritmo de agrupamiento.

En esta tesis, se implementa esta etapa utilizando las activaciones de una capa intermedia de la red *InceptionV3* (previamente entrenada) como vectores de características, se seleccionó esta arquitectura por su uso en el cálculo de la métrica FID (véase sección 1.2.3.2). Se eligió el algoritmo de PCA para la reducción de dimensiones, debido a su relativamente bajo costo computacional y, al contar con componentes principales que tienden a tener una correlación baja entre ellos, se tiene información más relevante para su posterior análisis. Y se realiza el agrupamiento mediante el algoritmo HDBSCAN [62], que se eligió por su capacidad para determinar automáticamente el número de grupos, lo cual se lleva a cabo con base en una métrica de distancia entre muestras, tomando en cuenta su densidad. A continuación, se describe de manera breve el algoritmo HDBSCAN.

2.4.1 Algoritmo HDBSCAN

HDBSCAN es una versión mejorada del algoritmo DBSCAN (*density-based spatial clustering of applications with noise,* por sus siglas en inglés) el cual se encarga de encontrar, automáticamente, grupos con alta densidad en un conjunto de datos.

El procedimiento del algoritmo HDBSCAN es el siguiente:

1. Cálculo de las distancias de alcance mutuo (*mutual reachability distance*) entre las muestras del conjunto.

Para dos muestras *a* y *b*, la distancia de alcance mutuo está dada por la ecuación (2.3).

$$d_{am-k}(a,b) = \max\left(core_k(a), core_k(b), d(a,b)\right) \quad (2.3)$$

En donde $d(a,b)$ es la distancia euclidiana entre las muestras o puntos a y b; $core_k(\cdot)$ se refiere al radio mínimo que debe tener un círculo, a partir del punto dado, de manera que k muestras del conjunto de datos se encuentren dentro de tal círculo. De este modo, al escoger el máximo valor, la distancia d_{am-k} involucra la densidad de muestras alrededor de los puntos a y b. Por ejemplo, si la densidad de muestras es baja, es decir, existen pocas muestras alrededor de los puntos a y b, entonces las distancias $core_k(a)$ y $core_k(b)$ serán mayores que la distancia euclidiana $d(a,b)$, ya que el radio de un círculo deberá ser grande para alcanzar a tener k muestras dentro de él. De lo contrario, si la densidad de muestras es alta, las distancias $core_k(a)$ y $core_k(b)$ serán pequeñas, ya que las k muestras estarán dentro de un círculo pequeño alrededor de los puntos a y b, respectivamente. Entonces $d_{am-k}(a,b)$ tenderá a ser la distancia euclidiana $d(a,b)$.

2. Construcción del árbol de expansión de peso mínimo.

Se considera a cada una de las muestras como un vértice, y las conexiones (aristas) entre dos vértices tienen un peso equivalente a la distancia de alcance mutuo entre tales muestras. En este paso se busca el conjunto de aristas que genere un árbol que incluya el total de muestras, pero cuyo peso total sea mínimo. Esto se obtiene mediante el algoritmo de Prim [63] el cual es iterativo. Se parte de una muestra seleccionada de manera aleatoria, se analiza cuál es la arista incidente con menor peso y con ella construye una rama del árbol. Posteriormente, el proceso se repite, buscando la arista, con menor peso, incidente a las muestras pertenecientes al árbol. En la Figura 2.8 se muestra una representación gráfica del resultado de este algoritmo para los datos de ejemplo.

3. Creación de grupos jerárquicos con base en el árbol de expansión de peso mínimo.

Se genera una jerarquía de componentes conectados. Se ordenan los vértices (puntos) del árbol de expansión en orden ascendente y se agrupan las muestras tomando en cuenta una distancia de alcance mutuo en específico a manera de umbral. Esto se puede observar de manera gráfica en la Figura 2.9. Partiendo de un umbral de distancia cero, se comienzan a agrupar

muestras, conforme la distancia umbral incrementa, los puntos se unen en distintos grupos, hasta llegar a un umbral en donde todos los puntos pertenecen a un mismo grupo.

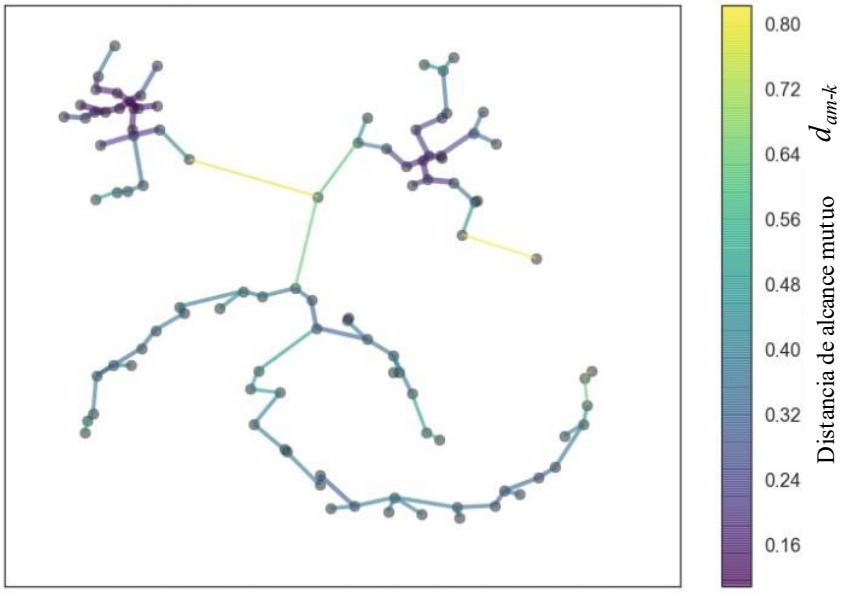


Figura 2.8. Ejemplo de construcción de árbol de expansión de peso mínimo para un conjunto de datos en dos dimensiones. La distancia de alcance mutuo se calcula con un valor de k=5.

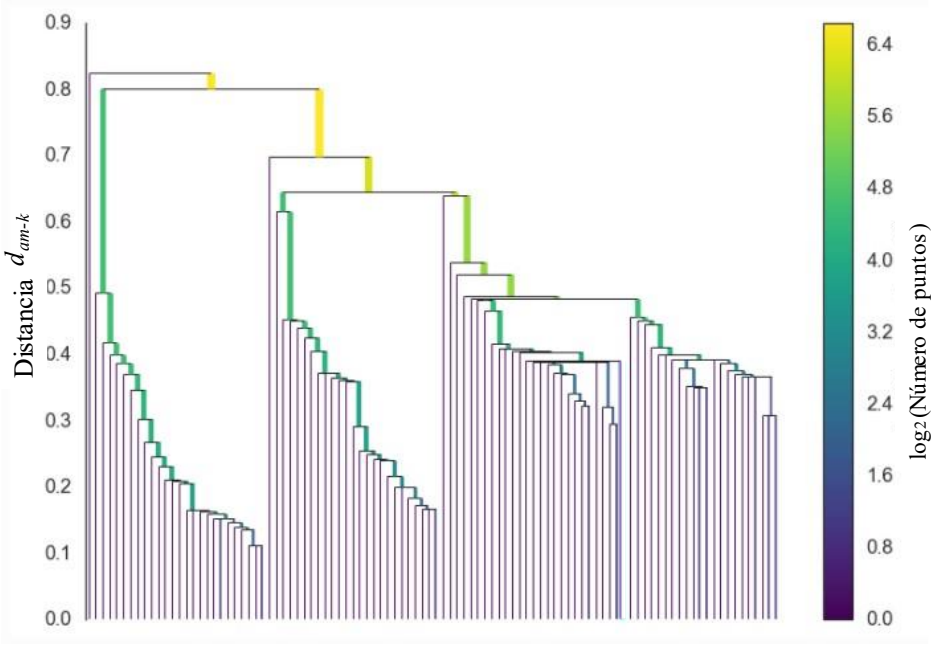


Figura 2.9. Separación jerárquica de datos en grupos, tomando en cuenta la medida de distancia de alcance mutuo.

Estos primeros tres pasos del algoritmo HDBSCAN corresponden al procedimiento del algoritmo DBSCAN, cuyo resultado final consiste en escoger un umbral de distancia, trazar una línea a través de la jerarquía y seleccionar los grupos en tal umbral. Por ejemplo, al seleccionar un umbral de distancia de 0.55, se generarían tres grupos con más de una muestra, como se puede observar en la Figura 2.10. Sin embargo, definir tal umbral es una tarea poco intuitiva. Por esta razón, en el algoritmo HDBSCAN se incluyen dos pasos adicionales para agrupar las muestras sin la necesidad de definir un umbral de distancia, los cuales se describen a continuación en el paso 4 y 5.

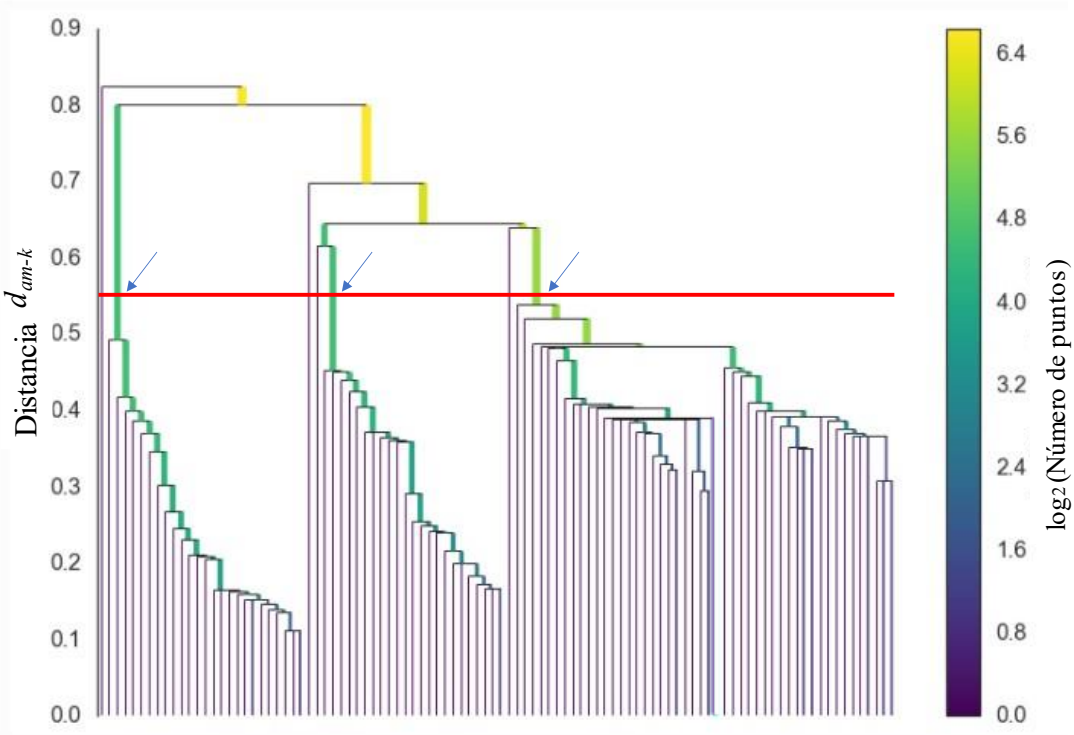


Figura 2.10. Ejemplo de resultados de agrupamiento DBSCAN con un umbral de distancia 0.55.

4. Síntesis de grupos en la jerarquía.

Para simplificar la jerarquía mostrada en la Figura 2.9, se realiza una síntesis de los grupos a lo largo de todo el intervalo de umbrales de distancia. Para ello se define un número mínimo de muestras por grupo y se condensan las separaciones llevadas a cabo al variar el umbral de distancia de forma descendente. Entonces, cuando un grupo A se separa en dos grupos B y C distintos, se verifica si tales grupos contienen el número mínimo de muestras por grupo. Si

ambos contienen suficientes muestras, entonces inician dos grupos nuevos en ese valor de umbral. Si tan solo el grupo B (o grupo C) tiene suficientes muestras, entonces sigue formando parte del grupo A, conservando su presencia, y las muestras del grupo C (o grupo B) son descartadas. Por último, si ningún grupo, B o C, contiene suficientes muestras, entonces ambos son descartados y se dice que el grupo A finaliza en el valor de umbral evaluado en ese momento. Esto genera un esquema de jerarquía condensado como se muestra en la Figura 2.11. En donde $\lambda = 1/d_{am\text{-}k}$.

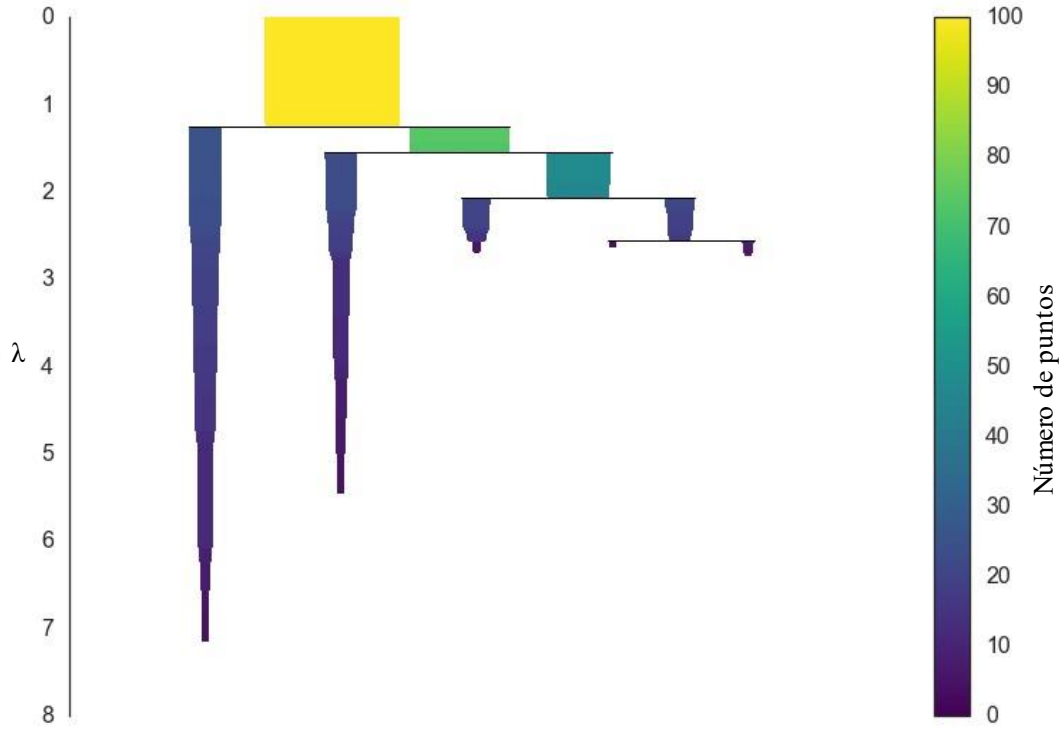


Figura 2.11. Esquema de jerarquía condensada de datos.

5. Seleccionar grupos relevantes.

De manera gráfica, en la mayoría de los casos, los grupos resultantes se pueden apreciar en el esquema de jerarquía condensada como las ramas con el área más grande. Analíticamente, estos grupos se determinan tomando en cuenta: la persistencia del grupo, la cual se relaciona al valor de distancia en donde inicia λ_{inicio} y el valor en donde termina λ_{fin} un grupo; y un análisis para determinar la estabilidad de grupo. Los detalles de este proceso se describen

detalladamente en [62]. En la Figura 2.12 y Figura 2.13 se ilustra el agrupamiento final para el conjunto de datos mostrado en la Figura 2.8.

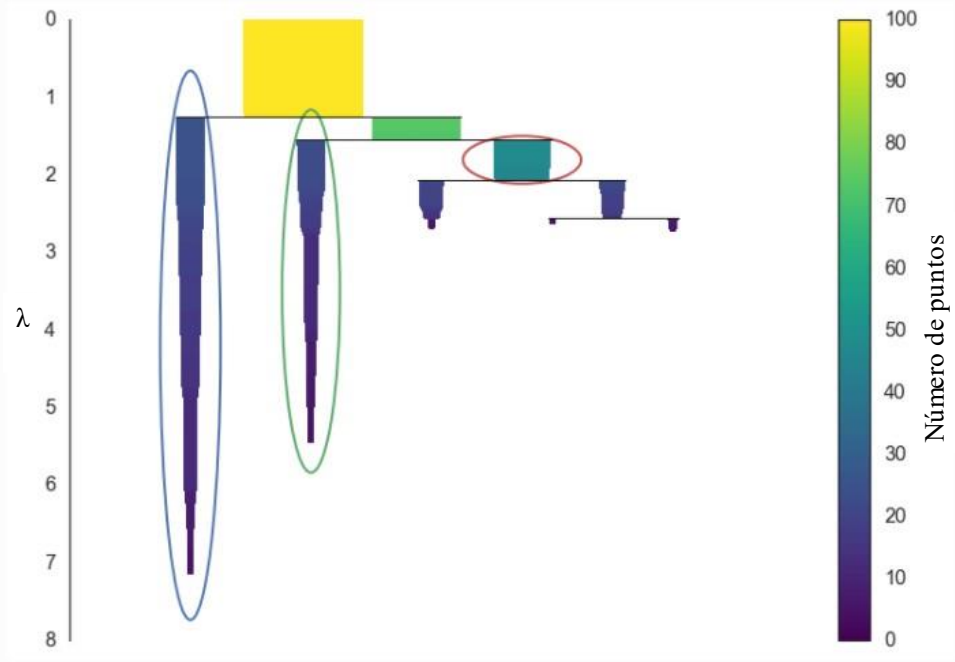


Figura 2.12. Síntesis y selección de grupos en algoritmo HDBSCAN.

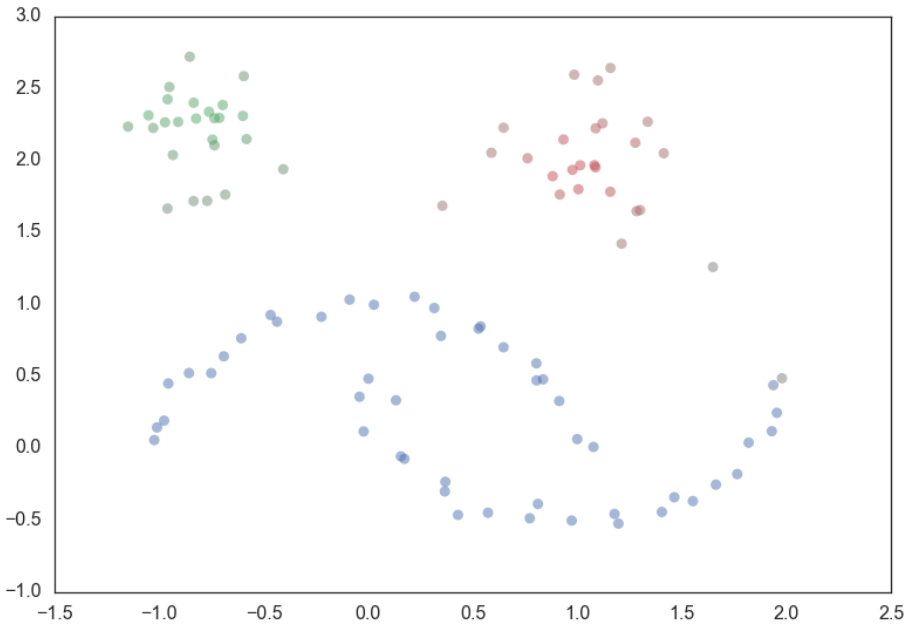


Figura 2.13. Ilustración de agrupamiento de datos mediante algoritmo HDSCAN.

2.5 Conclusiones

La metodología GDAM-GAN planteada en este capítulo está enfocada a ser aplicable para distintos problemas que cumplan con las consideraciones descritas en la sección 2.1. Este método también cuenta con cierto grado de versatilidad, ya que es posible utilizar diversos modelos para cada uno de sus elementos (modelo generativo base, traducción imagen-a-imagen, y depuración). Esto permite que el método pueda ser mejorado con modelos de vanguardia que se presenten en la literatura posteriormente. En los siguientes capítulos se pone en práctica este método en múltiples aplicaciones y se evalúa la calidad y utilidad de las muestras artificiales para el entrenamiento de modelos de redes profundas.

CAPÍTULO III. GENERACIÓN DE BASES DE DATOS ARTIFICIALES

El método GDAM-GAN, propuesto en el capítulo anterior, es una manera de incrementar el número de muestras existentes de alguna categoría en específico. Esta metodología se implementa para generar una base de datos artificiales que potencialmente sea útil para el entrenamiento de modelos de redes profundas destinados a la detección del uso de cubrebocas en imágenes. Adicionalmente, se emplea el mismo procedimiento en otra aplicación para demostrar su potencial como método genérico de aumento de datos. En este capítulo se describen los detalles técnicos de la implementación del método y sus respectivos resultados.

3.1 Entrenamiento de modelo generativo base para la generación de imágenes de rostros

El problema de detección del uso de cubrebocas en imágenes se puede dividir en dos etapas: detección de rostros en una imagen, los cuales se consideran como muestras; y clasificación de tales muestras. En este último problema se definen tres categorías: sin cubrebocas (*WithoutMask*), correspondiente a un rostro con nariz y boca descubierta; uso correcto de cubrebocas (*WithMask*), en donde el rostro tiene cubierta la nariz y la boca; y uso incorrecto de cubrebocas (*IncorrectMask*), en donde la nariz y/o la boca se encuentran parcialmente descubiertas. En las bases de datos disponibles, existe un desbalance en el número de muestras de la clase *IncorrectMask* respecto a las otras dos clases, es decir, generalmente, hay pocas muestras correspondientes al uso incorrecto de cubrebocas (véase sección 1.3.3).

Para este problema, se considera a la clase *IncorrectMask* como la clase deseada l_T y como clase base l_S se utilizan imágenes de rostros humanos, ya que existe una gran cantidad de muestras de tal clase y comparte algunas características con la clase deseada l_T. Adicionalmente, para fines demostrativos, se toma en cuenta una segunda clase deseada l_{T2} correspondiente a la clase *WithMask*.

Se pretende incrementar la cantidad de datos de la clase deseada l_T=*IncorrectMask*, generando muestras a partir de imágenes de la clase l_S=*rostros-humanos*[2], las cuales son abundantes en la base de datos FFHQ [20]. Ésta se compone por 65534 imágenes de rostros, y se eligió debido a la gran variedad de edades, género, etnicidad, y escenarios de fondo. Tales muestras son utilizadas para entrenar a las redes del modelo basado en WGAN, cuya arquitectura se describió en la sección 2.2. El entrenamiento se lleva a cabo utilizando distintas configuraciones de hiperparámetros, los cuales se seleccionan de acuerdo con experimentos reportados en la literatura, relacionados al entrenamiento de modelos WGAN. Estas configuraciones (WGAN_FFHQ_1, WGAN_FFHQ_2, WGAN_FFHQ_3 y WGAN_FFHQ_4) se muestran en la Tabla 3.1 y se presentan los hiperparámetros utilizados en cada una de ellas. Además, para fines de visualización y verificación del proceso de entrenamiento, al inicio del entrenamiento se definen algunos vectores con ruido aleatorio, con los cuales se generan imágenes cada 250 iteraciones y son útiles para analizar cualitativamente la calidad de las imágenes. Por otro lado, para determinar cuantitativamente la veracidad de las muestras artificiales, se calcula el valor de la métrica FID cada 2500 iteraciones. Por lo que, en la Tabla 3.1 se reporta la métrica FID más baja obtenida para cada configuración distinta.

Tabla 3.1. Experimentos de entrenamiento de modelo WGAN para la generación de imágenes de rostros artificiales.

Nombre de experimento	**Datos**	**Hiperparámetros para red discriminador**	**Hiperparámetros para red generador**	**FID**
WGAN_FFHQ_1	FFHQ [20] (65534 imágenes de rostros)	(*lr=0.0004, β_1=0.0, β_1=0.99, minibatch_size=64*)	(*lr=0.0001, β_1=0.0, β_1=0.99, minibatch_size=64*)	26.9461
WGAN_FFHQ_2		(*lr=0.0001, β_1=0.0, β_1=0.99, minibatch_size=64*)	(*lr=0.0001, β_1=0.0, β_1=0.99, minibatch_size=64*)	26.7919
WGAN_FFHQ_3		(*lr=0.0002, β_1=0.0, β_1=0.99, minibatch_size=64*)	(*lr=0.0002, β_1=0.0, β_1=0.99, minibatch_size=64*)	26.0913
WGAN_FFHQ_4		(*lr=0.0004, β_1=0.0, β_1=0.99, minibatch_size=64*)	(*lr=0.0004, β_1=0.0, β_1=0.99, minibatch_size=64*)	25.7683

[2] Para fines prácticos, se utiliza de manera indistinta el nombre de clase *rostros-humanos* y *rostros* en esta tesis.

En la Figura 3.1 se muestran algunas de las imágenes generadas a partir de vectores de entrada fijos, durante la ejecución del entrenamiento WGAN_FFHQ_2. Nótese que las características de la imagen de salida, para un mismo vector de entrada, varían durante las primeras épocas de entrenamiento, y después convergen a un rostro humano. El modelo es capaz de capturar la distribución de los datos de entrenamiento, pero sin generar imágenes exactamente iguales a las muestras de entrenamiento.

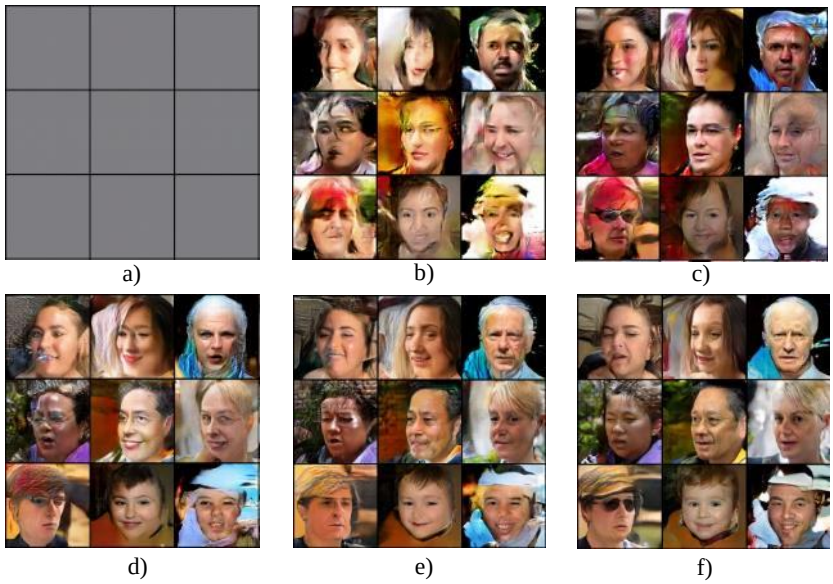


Figura 3.1. Progreso de entrenamiento experimento WGAN_FFHQ_2. a) al inicio de entrenamiento; b) iteración 4000; c) iteración 8000; d) iteración 32000; e) iteración 64000; f) iteración 128000.

Para comparar cualitativamente los resultados al variar los hiperparámetros, para cada experimento se almacenaron los valores de los parámetros de la red generador en el punto de entrenamiento cuando la métrica FID es mínima $\theta_{G|minFID}$, y se generó un conjunto de muestras artificiales con los parámetros $\theta_{G|minFID}$ de cada experimento. Con fines comparativos, se definen 18 vectores de ruido con valores fijos para generar tales muestras artificiales. Éstas se observan en las Figuras 3.2-5. Nótese que los resultados no son repetibles, es decir, en distintos experimentos, un mismo vector de entrada podría generar una muestra artificial con distintas características. Esto se debe a la naturaleza del esquema de entrenamiento, en donde se involucran vectores con valores aleatorios en cada iteración. Sin embargo, en todos los casos se puede observar que las imágenes contienen un rostro, es decir, el entrenamiento converge a

pesar de utilizar distintos valores de hiperparámetros. Después de llevar a cabo el entrenamiento, se elige el modelo con los parámetros $\theta_{G|minFID}$ y se utilizan como modelo generativo base. Para ello, se utiliza como referencia principal el valor de FID, ya que, seleccionar el mejor resultado a partir de una inspección visual puede no ser la mejor opción, ya que sería una evaluación subjetiva. En este caso, el valor más bajo de FID es de 25.7683 y se obtuvo en el experimento WGAN_FFHQ_4.

Figura 3.2. Ejemplos de imágenes de rostros generados en experimento WGAN_FFHQ_1 (FID=26.9461).

Figura 3.3. Ejemplos de imágenes de rostros generados en experimento WGAN_FFHQ_2 (FID=26.7919).

Figura 3.4. Ejemplos de imágenes de rostros generados en experimento WGAN_FFHQ_3 (FID=26.0913).

Figura 3.5. Ejemplos de imágenes de rostros generados en experimento WGAN_FFHQ_4 (FID=25.7683).

Los parámetros de la red generador resultantes en el experimento WGAN_FFHQ_4 son utilizados posteriormente (véase sección 3.3) para generar un conjunto de imágenes artificiales de la clase base $\hat{\boldsymbol{x}}_S$. Esto corresponde al modelo generativo base del método de aumento de datos propuesto. A continuación, se muestran los detalles del entrenamiento del siguiente elemento del método de aumento de datos, correspondiente al modelo de traducción imagen-a-imagen.

3.2 Entrenamiento de modelo de traducción rostros-a-incorrectMask y rostros-a-withMask

El siguiente elemento de GDAM-GAN es un modelo de traducción imagen-a-imagen. Cuyo propósito es transformar muestras de una clase base l_S a una clase deseada l_T. En el problema de detección del uso de cubrebocas, en la sección anterior se definió la clase base l_S=*rostros* y las dos clases deseadas l_{T1}=*incorrectMask* y l_{T2}=*withMask*. Para la implementación de este modelo de traducción imagen-a-imagen, se utiliza la arquitectura CycleGAN (véase sección 1.2.2.7).

Primero se realiza la transformación *rostros-a-incorrectMask*. Para ello, los datos de entrenamiento son: muestras de la clase base l_S=*rostros*, correspondiente a 1000 muestras de la base de datos FFHQ, elegidas de manera aleatoria; y muestras de la clase deseada l_{T1}=*incorrectMask*, que se compone por 346 muestras de la categoría *IncorrectMask* de PWMFD.

Para realizar la evaluación cualitativa, se seleccionan 64 muestras de cada clase, las cuales son procesadas por las redes generadoras correspondientes para cambiar de dominio, es decir, convertir imágenes de rostros a *incorrectMask*, y viceversa. Esto se realiza después de cada época de entrenamiento para visualizar el proceso. Y se definen los criterios para considerar que una muestra artificial pertenece a *incorrectMask*:

- La muestra generada preserva la mayor parte de los rasgos característicos del rostro (ojos, nariz, color de piel, etc.).
- Se añade un artefacto que se asemeja a un cubrebocas.
- El cubrebocas solamente cubre la barbilla o la boca. La nariz deberá mantenerse descubierta.

Por otro lado, para evaluar cuantitativamente los resultados, se calcula la métrica FID, tomando en cuenta la red de interés, es decir, se toman las muestras de salida de la red generador que realiza la conversión rostros-a-*incorrectMask* y estas se comparan con muestras reales de la clase deseada *incorrectMask* para calcular la métrica FID, cuyo procedimiento se describe en la sección 1.2.3.2. Adicionalmente, a manera de referencia, se calcula un valor $FID_{FFHQvsIM}$

que corresponde a aplicar la métrica FID para comparar las muestras reales de la clase base l_S=*rostros* y las muestras reales de la clase deseada l_T=*incorrectMask*. Se realizan dos experimentos, en donde se entrena el modelo *CycleGAN* bajo las configuraciones descritas en la Tabla 3.2.

Tabla 3.2. Configuración de entrenamiento para los experimentos de traducción imagen-a-imagen utilizando las clases rostros-a-*incorrectMask*

Nombre de experimento	**Datos de clase base l_S=*rostros***	**Datos de clase deseada l_{T1}=*incorrectMask***	**Hiperparámetros de entrenamiento**	**FID respecto a l_{T1}=*incorrectMask*** (FID FFHQ vs IM: 209.82)
F-to-IM_1	1000 muestras de FFHQ	346 muestras de clase *incorrectMask* de PWMFD	Optimizador Adam (*lr=0.0002, β_1=0.5, β_1=0.999, minibatch_size=1*) λ_{cycle}=10, $\lambda_{identity}$=0	110.7282
F-to-IM_2	1000 muestras de FFHQ	346 muestras de clase *incorrectMask* de PWMFD	Optimizador Adam (*lr=0.0002, β_1=0.5, β_1=0.999, minibatch_size=1*) λ_{cycle}=10, $\lambda_{identity}$=10	86.85

El primer experimento se denomina F-to-IM_1, y sus hiperparámetros de entrenamiento fueron seleccionados con base en los experimentos descritos en la literatura [25]. La evaluación cuantitativa arrojó un resultado $FID_{F\text{-}to\text{-}IM\_1}$=*110.7282*, el cual indica, respecto al valor $FID_{FFHQvsIM}$=*209.82*, que la distribución de las muestras artificiales generadas $\hat{\boldsymbol{x}}_{\boldsymbol{T}}$ es más cercana a la de las muestras reales de la clase deseada *incorrectMask* que de las muestras reales de la clase base *rostros*. Sin embargo, al analizar cualitativamente los resultados, las muestras artificiales de la clase deseada *incorrectMask* no cumplen con los criterios definidos anteriormente. En la Figura 3.6 se pueden observar las muestras de salida de las redes generador que realizan las conversiones *rostros-a-incorrectMask* e *incorrectMask-a-rostros*. La forma general de los rostros se conserva, sin embargo, algunas características importantes no son sintetizadas de manera correcta, como la posición de ojos, nariz, cubrebocas, o el color de algunos elementos de la imagen.

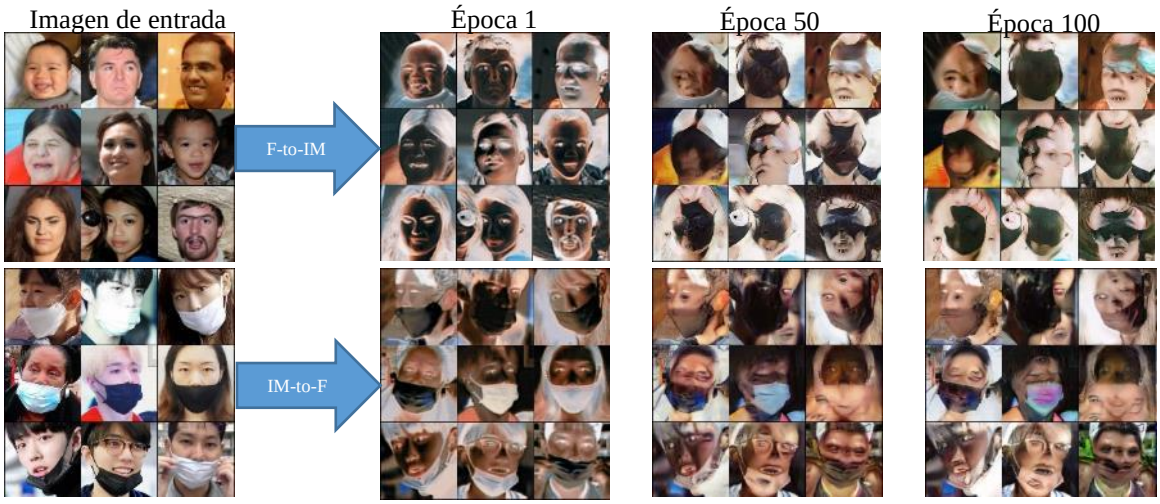


Figura 3.6. Proceso de entrenamiento experimento F-to-IM_1.

El segundo experimento, denominado F-to-IM_2, se realiza con la intención de mejorar la veracidad de los resultados obtenidos en el experimento F-to-IM_1. Para ello, se utilizan los mismos valores de hiperparámetros que en el primer experimento, pero se modifica el valor del hiperparámetro $\lambda_{identity}$=10, el cual tenía un valor de cero en F-to-IM_1. Esta modificación se lleva a cabo con el propósito de tomar en cuenta el efecto de la pérdida de identidad en el entrenamiento de las redes generador, lo cual puede tener un impacto positivo en la veracidad de las imágenes generadas ya que propicia que las características que tienen en común ambas clases se conserven. Se define con el mismo valor por defecto del hiperparámetro λ_{Cycle}. Con esta modificación, el valor de FID se redujo a $FID_{F\text{-}to\text{-}IM\_2}=86.85$. Al tomar en cuenta el valor de la pérdida de identidad, las características relevantes de un rostro se conservan y la red es capaz de añadir artefactos que se asemejan a un cubrebocas en la posición deseada, esto se puede apreciar en la Figura 3.7. Nótese que en algunas muestras no se añade el cubrebocas de manera exitosa. Lo mismo sucede con las imágenes resultantes en el proceso contrario (*incorrectMask*-a-*rostros*), en donde sólo en algunas imágenes se comienzan a observar artefactos correspondientes a una boca, mientras se desvanece el cubrebocas. Los cambios más significativos se observan entre la época 0 y la época 50. Después de 100 épocas no se observaron cambios relevantes.

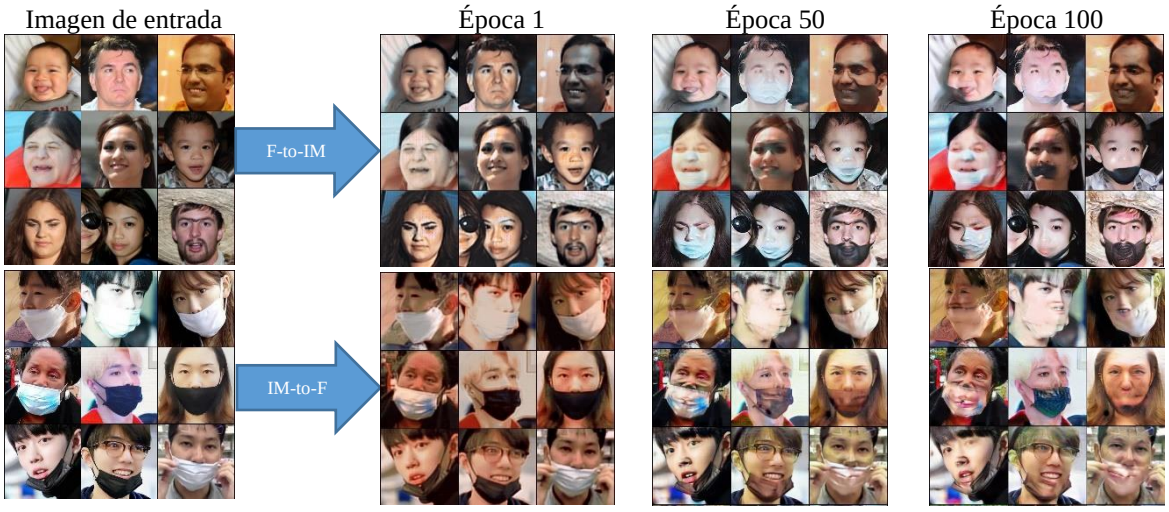


Figura 3.7. Proceso de entrenamiento experimento F-to-IM_2.

Adicionalmente, con el propósito de comparar el proceso de traducción imagen-a-imagen utilizando otra clase deseada, se entrena el modelo CycleGAN para llevar a cabo la traducción rostros-a-*withMask* (cubrebocas colocado correctamente). A pesar de que la base de datos PWMFD cuenta con más de cuatro mil muestras de la clase *withMask*, se utilizan únicamente 350 imágenes para simular las condiciones de la clase *incorrectMask*. El resto de la configuración del experimento se muestra en la Tabla 3.3. En este caso únicamente se lleva a cabo un experimento, denominado F-toWM, utilizando los hiperparámetros de entrenamiento del experimento F-to-IM_2, ya que fue en donde se obtuvieron los mejores resultados para la traducción *rostros*-a-*incorrectMask*.

Tabla 3.3. Configuración de entrenamiento para los experimentos de traducción imagen-a-imagen utilizando las clases rostros-a-*WithMask*.

Nombre de experimento	**Datos de clase base l_S=*rostros***	**Datos de clase deseada l_{T2}=*withMask***	**Hiperparámetros de entrenamiento**	**FID respecto a l_{T2}=*withMask*** (FID FFHQ vs WM: 195.39)
F-to-WM	1000 muestras de FFHQ	350 muestras de clase with*Mask* de PWMFD	Optimizador Adam (*lr=0.0002, β_1=0.5, β_1=0.999, minibatch_size=1*) λ_{cycle}=10, $\lambda_{identity}$=10	69.7561

En la Figura 3.8 se muestran algunas imágenes generadas por las redes generador durante el entrenamiento. En la traducción *rostros*-a-*withMask* (F-to-WM) se conservan las características generales de la muestra original, y la red es capaz de añadir un artefacto similar a un cubrebocas

en algunas de las imágenes. Al igual que en el experimento F-to-IM_2, algunas muestras no presentan un cambio significativo. Por otro lado, en la traducción *withMask*-a-*rostros* (WM-to-F), las muestras resultantes aún conservan la mayor parte de las características propias de la clase *withMask* y no se sintetizan los elementos necesarios para pertenecer a la clase de rostros.

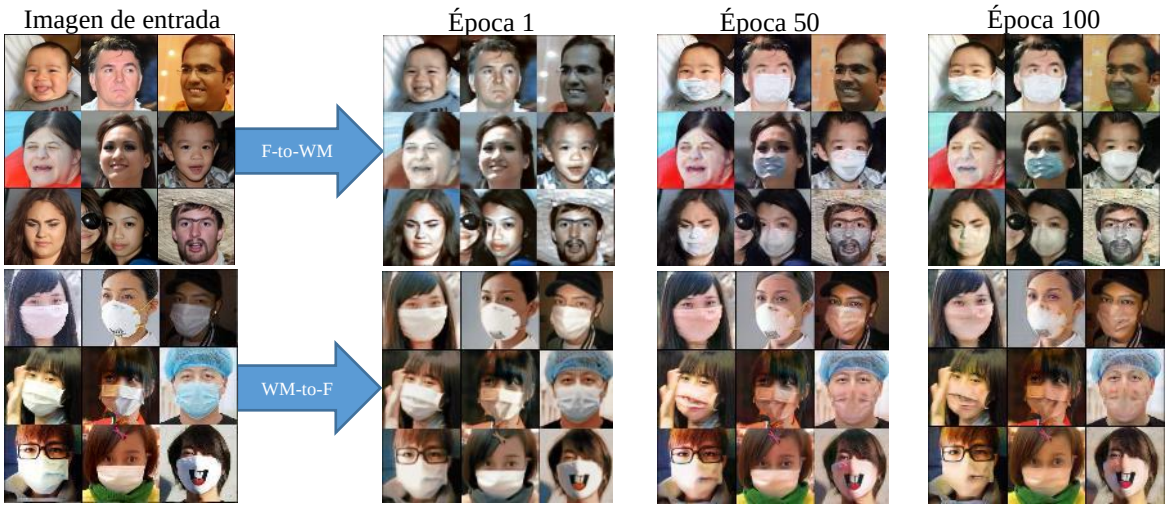


Figura 3.8. Proceso de entrenamiento experimento F-to-WM.

En este experimento, en la evaluación cuantitativa, se comparan las muestras de salida de la red generador que lleva a cabo la traducción rostros-a-*withMask*, con muestras reales de la clase deseada *withMask*, y se calcula la métrica FID obteniendo un valor $FID_{F\text{-}to\text{-}WM}=69.75$. Además, como referencia, se comparan muestras reales de *rostros* con muestras reales de *withMask*, obteniendo un $FID_{FFHQvsWM}=195.39$, es decir, la distribución de las muestras artificiales de la clase deseada *withMask* se aproxima mejor a la distribución de las muestras reales de *withMask* que a la distribución de las muestras reales de la clase *rostros*.

3.3 Generación de muestras artificiales de *incorrectMask* y *withMask*

Una vez entrenado el modelo generativo base y los modelos de traducción imagen-a-imagen, se procede a generar muestras artificiales de las clases deseadas $l_{T1}=incorrectMask$ y $l_{T2}=withMask$.

Primero se genera el conjunto de muestras artificiales de la clase base $\hat{\boldsymbol{x}}_S$, que en este caso corresponden a la clase l_S=*rostros*. Para ello, se toman los parámetros de la red generador de WGAN entrenada durante el experimento WGAN_FFHQ_4. En total se generan 65535 imágenes artificiales de la clase base l_S=*rostros*. Se eligió esta cantidad para contar con una proporción aproximada de 1 muestra artificial por cada muestra real (ya que existen 65534 muestras reales de rostros, correspondientes a la base de datos FFHQ).

Luego, los conjuntos de muestras reales $\boldsymbol{x}_S$ y artificiales $\hat{\boldsymbol{x}}_S$ de la clase base l_S=*rostros* son procesados por el modelo de traducción imagen-a-imagen para transformarlos a la clase deseada l_{T1}=*incorrectMask* y l_{T2}=*withMask*. Esto se lleva a cabo utilizando las redes generador que realizan las transformaciones *rostros-a-incorrectMask* y *rostros-a-withMask*, las cuales fueron entrenadas durante los experimentos F-to-IM_2 y F-to-WM, respectivamente. En la Figura 3.9 se muestran algunos de los mejores resultados de traducciones *rostro*-a-*incorrectMask* y *rostro*-a-*withMask* a partir del conjunto de imágenes de rostros reales $\boldsymbol{x}_S$ de la base de datos FFHQ. Nótese que la red generador tiene la capacidad de añadir el cubrebocas en más de un objeto de interés independientemente de su posición en la imagen, como sucede en el ejemplo encerrado en el recuadro color rojo en la Figura 3.9, en donde se añade cubrebocas al rostro principal y al rostro que aparece en el fondo. Por otro lado, en la Figura 3.10, se muestran algunos de los mejores resultados de traducción de imagen-a-imagen a partir del conjunto de imágenes artificiales $\hat{\boldsymbol{x}}_S$ de rostros. Tales imágenes de entrada, a pesar de no haber sido utilizadas en el entrenamiento del modelo CycleGAN, pueden ser procesadas correctamente por las redes generadoras. Esto indica que la distribución de las imágenes artificiales $\hat{\boldsymbol{x}}_S$ generadas por WGAN es similar a las imágenes reales $\boldsymbol{x}_S$ de FFHQ, lo cual corresponde al comportamiento deseado de un modelo GAN [64].

En resumen, se generan los siguientes conjuntos de muestras:

- RF-to-IM: a partir de rostros reales (RF) se generan muestras artificiales de *incorrectMask* (IM), en total son 65534 muestras.
- FF-to-IM: a partir de rostros reales artificiales (FF) se generan muestras artificiales de *incorrectMask* (IM), en total son 65535 muestras.

- RF-to-WM: a partir de rostros reales (RF) se generan muestras artificiales de *withMask* (WM), en total son 65534 muestras
- FF-to-WM: a partir de rostros reales artificiales (FF) se generan muestras artificiales de *withMask* (WM), en total son 65535 muestras.

Figura 3.9. Muestras artificiales de *withMask* e *incorrectMask* generadas a partir de imágenes de rostros reales.

Figura 3.10. Muestras artificiales de *withMask* e *incorrectMask* generadas a partir de imágenes de rostros artificiales.

Para evaluar cualitativamente las muestras, se definen los siguientes criterios necesarios para considerar que la muestra generada corresponde a la clase deseada (considerando que la imagen de entrada corresponde a un rostro):

- La imagen generada preserva la mayor parte de los rasgos característicos del rostro.
- Se añade un artefacto que se asemeja a un cubrebocas.
- Para la clase deseada *incorrectMask*, el cubrebocas solamente cubre la barbilla o la boca. La nariz deberá mantenerse descubierta.
- Para la clase deseada *withMask*, el cubrebocas se encuentra sobre la nariz y la boca.

A pesar de que las características particulares de la clase deseada sí se presentan en algunos resultados, en una gran parte de las muestras artificiales $\hat{\boldsymbol{x}}_{\boldsymbol{T}}$ no sucede de esta manera, como en los ejemplos mostrados en la Figura 3.11.

Figura 3.11. Muestras artificiales cualitativamente no aceptables.

Al realizar una inspección visual, se determinó que al menos el 19% de las muestras artificiales son aceptables, es decir, cumplen con las características mencionadas para considerar que pertenecen a la clase deseada *incorrectMask* o *withMask*. En la Tabla 3.4 se muestra un resumen de los resultados de la inspección visual a cada uno de los conjuntos de muestras artificiales.

Tabla 3.4. Resultados de inspección visual de una porción de las muestras artificiales.

Conjunto de muestras	**Muestras útiles**	**Muestras no útiles**	**Porcentaje de aprovechamiento**
FF-to-IM	111	539	17.08%
FF-to-WM	108	542	16.61%
RF-to-IM	120	530	18.46%
RF-to-WM	177	473	27.23%
Total	516	2084	19.84%

El porcentaje de muestras aceptables implica la necesidad de llevar a cabo una etapa de depuración para extraer únicamente aquellas muestras que sean útiles para construir una base de datos con imágenes pertenecientes a la clase deseada. Para ello, la inspección visual puede ser una manera de llevarlo a cabo, sin embargo, este método no es factible si existe una gran cantidad de muestras. En la siguiente sección se describe el uso de un algoritmo de agrupamiento como herramienta auxiliar en el proceso de depuración de las muestras artificiales.

3.4 Análisis y depuración de datos artificiales utilizando algoritmo HDBSCAN

El tercer elemento de GDAM-GAN consiste en una etapa de depuración de muestras artificiales. El procedimiento general se muestra en la Figura 2.7. El primer paso es extraer características de las muestras artificiales. Esto se lleva a cabo utilizando el modelo InceptionV3, tomando en cuenta el vector de características que se emplea para el cálculo de la métrica FID (véase sección 1.2.3.2). Opcionalmente, se utiliza un algoritmo para la reducción de dimensionalidad, como PCA. Después, utilizando el algoritmo HDBSCAN, cuyo procedimiento se describe en la sección 2.4.1, se agrupan las muestras en distintos conjuntos con características similares entre sí. De esta manera, un grupo podría estar conformado por muestras que cumplan con las condiciones necesarias para pertenecer a la clase deseada. Sin embargo, también es posible que las características de las muestras, en los grupos generados, no correspondan a la clase deseada. En esta tesis, los grupos generados por el algoritmo HDBSCAN se denominan como *grupo1*, *grupo2, …*, *grupoH,* y *ruido*. En donde *H* es el total de grupos generados, sin tomar en cuenta el grupo *ruido,* tal número de grupos puede variar y es determinado por el mismo algoritmo de manera automática.

Se aplica el procedimiento con cada uno de los conjuntos de muestras artificiales generados (RF-to-IM, FF-to-IM, RF-to-WM, y FF-to-WM). Para el caso del conjunto de entrenamiento FF-to-IM (Tabla 3.5) en la mayoría de los experimentos se obtuvieron dos grupos, además del ruido. En uno de ellos, la mayor parte de los datos, es decir, más del 80% de las muestras, cumplen con las consideraciones para ser parte de la clase *incorrectMask* (Figura 3.12a) y la cantidad de muestras en tales grupos varía entre mil y tres mil. Otro grupo que es consistente a

lo largo de los experimentos llevados a cabo corresponde a aquel con imágenes en donde todos los rostros utilizan lentes oscuros (Figura 3.12b). Por último, el resto de las imágenes son consideradas como parte del *ruido* (Figura 3.12c). Aunque algunas de estas muestras podrían ser consideradas como parte de la clase *incorrectMask*, la mayor parte de las muestras del grupo *ruido* no son útiles. Cabe mencionar que el orden en que aparecen los grupos no está relacionado con las características de sus muestras, por ejemplo, para los experimentos FF2IM_2048, FF2IM_PCA512, y FF2IM_PCA128, coincide que el *grupo1* es aquel en donde la mayoría de las muestras son útiles, a diferencia del experimento FF2IM_PCA32, en donde tales muestras se encuentran en el *grupo2*.

Tabla 3.5. Experimentos de agrupamiento con el conjunto FF-to-IM.

Conjunto FF-to-IM				
Nombre de experimento	**Dimensiones de vector de características**	**Grupos generados con HDBSCAN**	**Cantidad de muestras en grupo**	**¿La mayoría de las muestras (>80%) pertenecen a la clase *incorrectMask*?**
FF2IM_2048	2048	*grupo1*	1,228	Sí
		grupo2	279	No
		ruido	64,029	No
FF2IM_PCA512	512 (PCA)	*grupo1*	1,113	Sí
		grupo2	319	No
		ruido	64,104	No
FF2IM_PCA128	128 (PCA)	*grupo1*	1,711	Sí
		grupo2	570	No
		ruido	63,255	No
FF2IM_PCA32	32 (PCA)	*grupo1*	748	No
		grupo2	2,675	Sí
		ruido	62,113	No

a) b) c)

Figura 3.12. Agrupamiento de conjunto de muestras artificiales FF-to-IM. a) *grupo1*, b) *grupo2*, c) *ruido*.

En los experimentos realizados con el conjunto de muestras RF-to-IM (Tabla 3.6), en los experimentos RF2IM_2048 y RF2IM_PCA128 se obtuvieron resultados similares. Se generan tres grupos: un grupo con imágenes que en su mayoría pertenecen a la clase deseada *incorrectMask*; otro grupo con imágenes de rostros con lentes oscuros; y el grupo de ruido, en donde la mayoría de las muestras no pertenecen a la clase deseada *incorrectMask*. En el caso de los experimentos RF2IM_PCA512 y RF2IM_PCA32, se obtienen cuatro grupos: un grupo con muestras útiles que pertenecen a la clase deseada *incorrectMask*; un grupo con muestras de rostros utilizando lentes oscuros; un grupo con imágenes de personas utilizando toga y birrete; y el grupo de ruido.

Tabla 3.6. Experimentos de agrupamiento con el conjunto RF-to-IM.

Conjunto RF-to-IM				
Nombre de experimento	**Dimensiones de vector de características**	**Grupos generados con HDBSCAN**	**Cantidad de muestras en grupo**	**¿La mayoría de las muestras (>80%) pertenecen a la clase *incorrectMask*?**
RF2IM_2048	2048	*grupo0*	1,790	Sí
		grupo1	279	No
		ruido	63,465	No
RF2IM_PCA512	512 (PCA)	*grupo0*	394	No
		grupo1	1,211	Sí
		grupo2	103	No
		ruido	63,826	No
RF2IM_PCA128	128 (PCA)	*grupo0*	2474	Sí
		grupo1	606	No
		ruido	62,454	No
RF2IM_PCA32	32 (PCA)	*grupo0*	874	No
		grupo1	112	No
		grupo2	3,713	Sí
		ruido	60,835	No

También se llevó a cabo el análisis de las muestras artificiales de la clase *withMask*. Para el conjunto de muestras FF-to-WM (Tabla 3.7), en el experimento FF2WM_2048 todas las muestras fueron consideradas como ruido. Al disminuir las dimensiones de los vectores de características, en los experimentos FF2WM_PCA512, FF2WM_PCA128, y FF2WM_PCA32, se obtuvieron tres grupos: uno en donde la mayor parte de los datos pertenece a la clase *withMask*; otro que se compone por imágenes de rostros con lentes oscuros; y el grupo de ruido, en donde la mayoría de las muestras no son útiles. Este comportamiento se repite en todos los experimentos realizados con el conjunto de muestras RF-to-WM (Tabla 3.8). En el caso de la

clase deseada l_{T2}=*withMask*, la cantidad de muestras, en los grupos marcados como útiles, varía entre 11000 y 15000 imágenes.

Tabla 3.7. Experimentos de agrupamiento con el conjunto FF-to-WM.

Conjunto FF-to-WM				
Nombre de experimento	**Dimensiones de vector de características**	**Grupos generados con HDBSCAN**	**Cantidad de muestras en grupo**	**¿La mayoría de las muestras (>80%) pertenecen a la clase *withMask*?**
FF2WM_2048	2048	Ruido	65,535	No
FF2WM_PCA512	512 (PCA)	Grupo 0	11,669	Sí
		Grupo 1	168	No
		Ruido	53,699	No
FF2WM_PCA128	128 (PCA)	Grupo 0	12,879	Sí
		Grupo 1	402	No
		Ruido	52,255	No
FF2WM_PCA32	32 (PCA)	Grupo 0	702	No
		Grupo 1	14,845	Sí
		Ruido	49,989	No

Tabla 3.8. Experimentos de agrupamiento con el conjunto RF-to-WM.

Conjunto RF-to-WM				
Nombre de experimento	**Dimensiones de vector de características**	**Grupos generados con HDBSCAN**	**Cantidad de muestras en grupo**	**¿La mayoría de las muestras (>80%) pertenecen a la clase *withMask*?**
RF2WM_2048	2048	Grupo 0	12,774	Sí
		Grupo 1	135	No
		Ruido	52,625	No
RF2WM_PCA512	512 (PCA)	Grupo 0	13,534	Sí
		Grupo 1	234	No
		Ruido	51,766	No
RF2WM_PCA128	128 (PCA)	Grupo 0	480	No
		Grupo 1	13,667	Sí
		Ruido	51,387	No
RF2WM_PCA32	32 (PCA)	Grupo 0	15,100	Sí
		Grupo 1	788	No
		Ruido	49,646	No

Con los experimentos llevados a cabo se determinó que es posible realizar la depuración de muestras artificiales de las clases *incorrectMask* y *withMask*, utilizando el algoritmo HDBSCAN como herramienta. En la mayoría de los experimentos se generó un grupo en donde la mayoría de las imágenes pertenecían a una de las clases deseadas. Sin embargo, para otras aplicaciones será necesario realizar experimentos para determinar las dimensiones adecuadas del vector de características, de manera que se realice el agrupamiento adecuado, o incluso se podrían utilizar distintos procedimientos de extracción de características. Además, la evaluación final de estos grupos sigue llevándose a cabo mediante una inspección visual de una

porción de estas muestras (~10%). Aún es necesario encontrar una manera de realizar tal análisis automáticamente. No obstante, el uso de este algoritmo de agrupamiento facilita el proceso de depuración, especialmente cuando se cuenta con una alta cantidad de datos.

Finalmente, las muestras de los grupos marcados como útiles son almacenadas en archivos de imagen y se organizan en carpetas para construir la base de datos. De manera opcional, se realiza una última inspección visual para descartar algunas de las muestras no deseadas. Las bases de datos se componen de la siguiente manera:

- *FakeIncorrectMaskDataset*: corresponde a imágenes artificiales de personas utilizando cubrebocas incorrectamente. Contiene 2300 muestras en total: 911 provienen del conjunto FF-to-IM; y 1389 provienen del conjunto RF-to-IM.
- *FakeWithMaskDataset*: corresponde a imágenes artificiales de personas utilizando cubrebocas correctamente. Se compone por 21669 muestras en total: 12413 provienen de FF-to-WM; y 9256 del conjunto RF-to-WM.

Las muestras del conjunto *FakeIncorrectMaskDataset* se utilizan en el entrenamiento de modelos de redes profundas para la detección del uso de cubrebocas en imágenes, tales experimentos se describen en el CAPÍTULO IV. Por otro lado, el conjunto de muestras *FakeWithMaskDataset* se construye con fines demostrativos y queda disponible para trabajos a futuro. Nótese que existe una diferencia significativa entre la cantidad de muestras útiles entre las bases de datos artificiales de *incorrectMask* y *withMask*. Estas cantidades son dependientes de los distintos elementos del proceso de depuración, es decir, del proceso de extracción de características, las dimensiones de los vectores de características, y del algoritmo HDBSCAN. Un posible trabajo a futuro podría involucrar el análisis de distintos procesos de extracción de características, o incluso utilizar distintos algoritmos de agrupamiento en esta etapa del método de aumento de datos.

En la siguiente sección se muestra la implementación del método propuesto para otra(s) aplicación(es), con el propósito de mostrar el potencial que tiene la metodología para ser utilizado en distintas áreas.

3.5 Implementación de método de aumento de datos en otra aplicación

La metodología GDAM-GAN fue diseñada con la intención de ser utilizada en más de una aplicación. Con el propósito de demostrar lo anterior, se selecciona un problema adicional para implementar aumento de datos con el método propuesto. En [3] se utiliza un modelo DCGAN para aumento de datos en el problema de clasificación de enfermedades en hojas de tomate utilizando un modelo de aprendizaje profundo denominado como GoogLeNet [65]. Se cuenta con una base de datos con múltiples clases, una de ellas correspondiente a hojas de tomate sanas y el resto de las categorías corresponden a varias enfermedades de hojas de tomate. En la Figura 3.13 se muestran ejemplos de cada clase. Se aplica aumento de datos, entrenando un modelo DCGAN utilizando 240 muestras reales de cada categoría.

A continuación, se muestra la implementación de GDAM-GAN, tomando en cuenta las imágenes de hojas de tomate utilizadas en [3], bajo las condiciones definidas en la sección 2.1:

- Existe un conjunto de muestras reales $\boldsymbol{x}_T$ pertenecientes a una clase deseada l_T: en este caso se cuenta con cuatro clases objetivo, correspondientes a las categorías de cada enfermedad de hojas de tomate (*Late Blight, Septoria Leaf Spot, Target Spot Bacteria*, y *Yellow Leaf Curl Virus*).
- Existe una clase base l_S que comparte algunas características generales con la clase deseada l_T, pero que difieren en características propias de cada clase: tal clase similar corresponde a las hojas de tomate sanas (*healthy*).
- Existe una gran cantidad de muestras $\boldsymbol{x}_S$ de la clase base l_S (respecto a la cantidad de muestras de la clase deseada l_T) y/o existe algún modelo generativo capaz de sintetizar muestras de l_S: Se tomarán en cuenta mil muestras de la clase l_S (*healthy*) y 240 muestras por cada una de las clases deseadas (enfermedades de hojas de tomate).

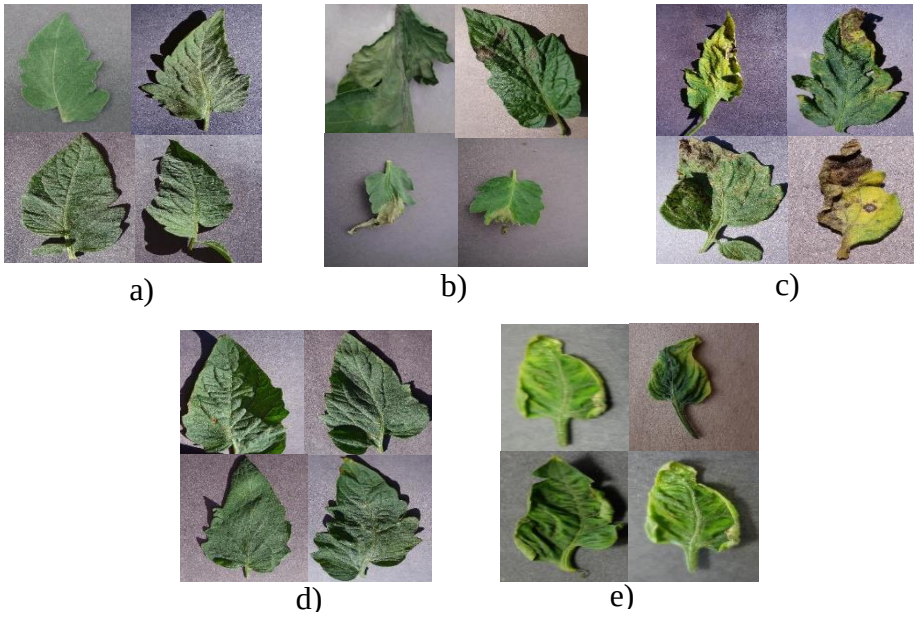


Figura 3.13. Ejemplos de muestras reales de base de datos *Tomato Leaf Disease* [66]. a) Hoja saludable (*healthy)*, b) *Late Blight*, c) *Septoria Leaf Spot*, d) *Target Spot Bacteria*, e) *Yellow Leaf Curl Virus.*

Una vez definidas las clases base l_S=*healthy* y clases deseadas l_{T1}=*LateBlight*, l_{T2}= *SeptoriaLeafSpot*, l_{T3}=*TargetSpotBacteria*, l_{T4}=*YellowLeafCurlVirus*, se ejecuta el proceso para generar muestras artificiales.

Primero se realiza el entrenamiento del modelo WGAN para generar imágenes artificiales de la clase l_S=*healthy*. La configuración de los experimentos y el respectivo valor mínimo de FID obtenido para cada caso se muestra en la Tabla 3.9. Los hiperparámetros de entrenamiento fueron seleccionados acorde a los experimentos llevados a cabo para la generación de rostros artificiales (véase sección 3.1). Para calcular la métrica FID se compara la distribución de las imágenes artificiales de la clase deseada *healthy*, con imágenes reales de tal clase. En la Figura 3.14 se muestra la salida de la red generador durante el proceso de entrenamiento del modelo WGAN. El valor de FID mínimo se obtuvo durante el experimento WGAN_Healthy_3, con una magnitud $FID_{HealthyLeafWGAN}$=87.617.

Tabla 3.9. Configuración de experimentos de entrenamiento de WGAN para la generación de hojas de tomate sanas.

Nombre de experimento	**Datos**	**Hiperparámetros para red discriminador**	**Hiperparámetros para red generador**	**FID**
WGAN_Healthy_1	Clase *Healthy* de *Tomato Leaf Disease dataset* (1000 imágenes de hojas de tomate sanas)	(*lr=0.0001*, β_1*=0.0*, β_1*=0.99*, *minibatch_size=64*)	(*lr=0.0001*, β_1*=0.0*, β_1*=0.99*, *minibatch_size=64*)	124.3935
WGAN_Healthy_2		(*lr=0.0004*, β_1*=0.0*, β_1*=0.99*, *minibatch_size=64*)	(*lr=0.0001*, β_1*=0.0*, β_1*=0.99*, *minibatch_size=64*)	103.0007
WGAN_Healthy_3		(*lr=0.0004*, β_1*=0.5*, β_1*=0.99*, *minibatch_size=64*)	(*lr=0.0001*, β_1*=0.5*, β_1*=0.99*, *minibatch_size=64*)	87.617

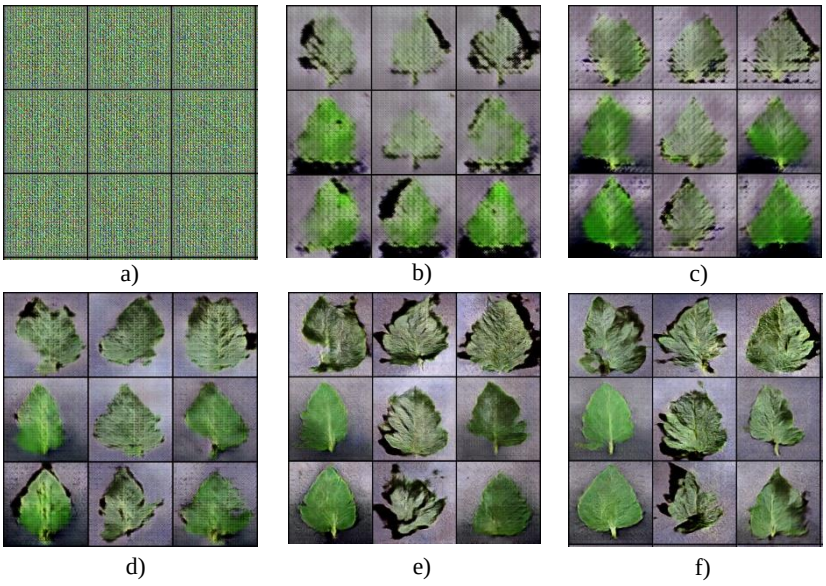


Figura 3.14. Imágenes artificiales de hojas de tomate sanas durante el proceso de entrenamiento de modelo WGAN en experimento WGAN_Healthy_3. a) al inicio de entrenamiento; b) iteración 4309; c) iteración 8618; d) iteración 17236; e) iteración 34503; f) iteración 69006.

Después, se lleva a cabo el entrenamiento del modelo CycleGAN para la traducción imagen-a-imagen de hojas de tomate sanas a hojas de tomate con alguna enfermedad. La configuración de los experimentos llevados a cabo se muestra en la Tabla 3.10. Para todos los casos, se utiliza un total de 240 muestras reales de la clase deseada, es decir, hojas de tomate con alguna enfermedad, y mil muestras reales de la clase base l_S=*healthy*. Para simplificar la notación de cada clase, se define la siguiente nomenclatura:

- H: *healthy*
- D1: *Late Blight*
- D2: *Septoria Leaf Spot*
- D3: *Target Spot Bacteria*
- D4: *Yellow Leaf Curl Virus*

Los hiperparámetros de entrenamiento son definidos con los mismos valores utilizados en el experimento *F-to-IM_2* descrito en la sección 3.2. En la Figura 3.15 se muestran algunas muestras resultantes. En este caso, los criterios para considerar a una muestra como parte de una de las clases deseadas deben ser definidos por algún experto en el área, lo cual está fuera del alcance de este proyecto de investigación. Por esta razón, únicamente se realiza una comparación visual de las muestras reales y muestras artificiales de cada clase deseada, es decir, de hojas de tomate con alguna de las 4 enfermedades.

Tabla 3.10. Configuración de experimentos para traducción hoja de tomate sana a hoja de tomate con alguna enfermedad y métrica FID mínima obtenida.

Nombre de experimento	**Datos de clase base** l_S***=healthy***	**Datos de clase deseada** l_T***={D1,D2,D3,D4}***	**Hiperparámetros de entrenamiento (todas las redes)**	**FID respecto a clase deseada**
H-to-D1	1000 muestras de hojas de tomate sanas	240 muestras de clase *Late Blight*	Optimizador Adam (*lr=0.0002, β_1=0.5, β_2=0.999, minibatch_size=1*) λ_{cycle}=10, $\lambda_{identity}$=10	78.3871
H-to-D2	1000 muestras de hojas de tomate sanas	240 muestras de clase *Septoria Leaf Spot*	Optimizador Adam (*lr=0.0002, β_1=0.5, β_2=0.999, minibatch_size=1*) λ_{cycle}=10, $\lambda_{identity}$=10	55.6738
H-to-D3	1000 muestras de hojas de tomate sanas	240 muestras de clase *Target Spot Bacteria*	Optimizador Adam (*lr=0.0002, β_1=0.5, β_2=0.999, minibatch_size=1*) λ_{cycle}=10, $\lambda_{identity}$=10	36.2361
H-to-D4	1000 muestras de hojas de tomate sanas	240 muestras de clase *Yellow Leaf Curl Virus*	Optimizador Adam (*lr=0.0002, β_1=0.5, β_2=0.999, minibatch_size=1*) λ_{cycle}=10, $\lambda_{identity}$=10	52.8763

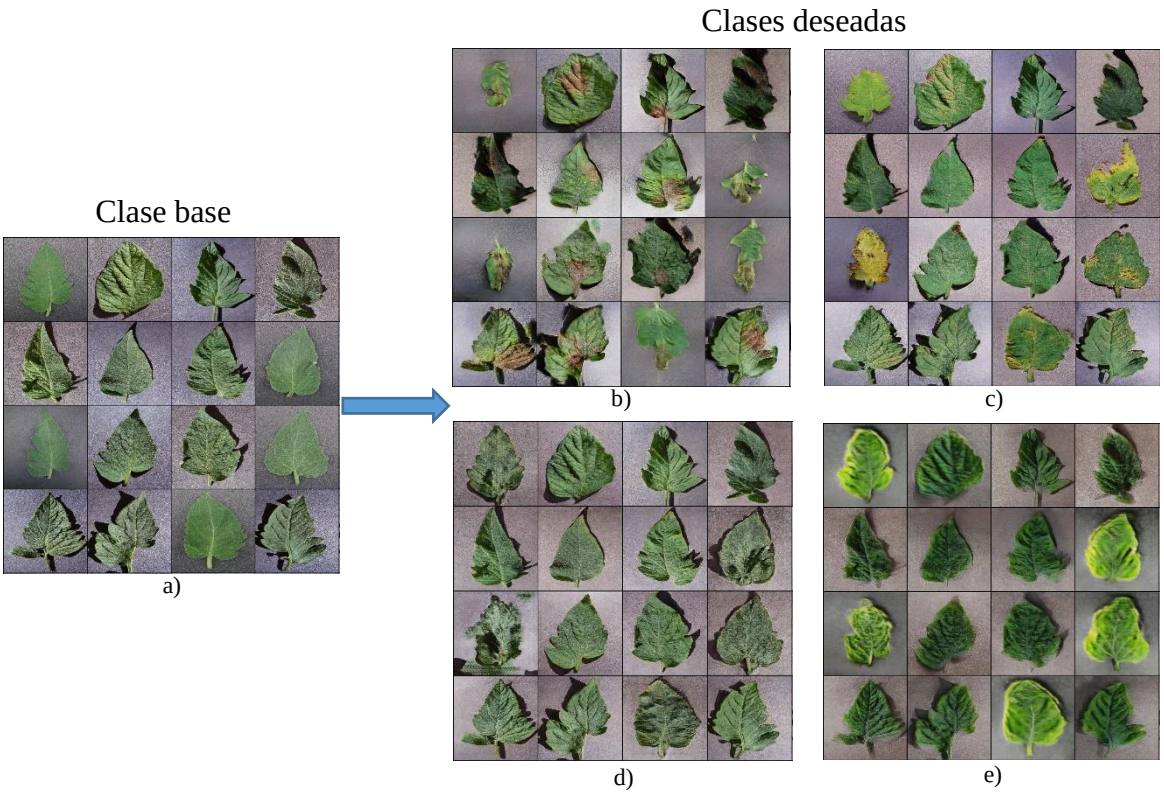


Figura 3.15. Traducción hoja de tomate sana a hoja de tomate con enfermedad. a) Imagen de entrada (*healthy*), b) *Healthy-to-LateBlight*, c) *Healthy-to-SeptoriaLeafSpot*, c) *Healthy-to-TargetSpotBacteria*, e) *Healthy-to-YellowLeafCurlVirus.*

Después, se genera un conjunto de muestras artificiales de cada clase deseada a partir de muestras reales (RH) y artificiales (FH) de la clase base *healthy*, estos conjuntos se denominan:

- RH-to-D1 y FH-to-D1: que corresponden a muestras artificiales de la categoría *Late Blight* (D1) a partir de imágenes de hojas de tomate sanas reales y artificiales, respectivamente.
- RH-to-D2 y FH-to-D2: que corresponden a muestras artificiales de la categoría *Septoria Leaf Spot* (D2) a partir de imágenes de hojas de tomate sanas reales y artificiales, respectivamente.
- RH-to-D3 y FH-to-D3: que corresponden a muestras artificiales de la categoría *Target Spot Bacteria* (D3) a partir de imágenes de hojas de tomate sanas reales y artificiales, respectivamente.

- RH-to-D4 y FH-to-D4: que corresponden a muestras artificiales de la categoría *Yellow Leaf Curl Virus* (D4) a partir de imágenes de hojas de tomate sanas reales y artificiales, respectivamente.

Posteriormente, se lleva a cabo el proceso de depuración de tales conjuntos de muestras artificiales, con la ayuda del algoritmo HDBSCAN. Se redujeron las dimensiones de los vectores de características a ocho elementos. Se llegó a tal valor de manera empírica, ya que con dimensiones mayores, el algoritmo únicamente generaba un grupo, correspondiente al ruido. En estos casos, la mayoría de las muestras artificiales, generadas a partir de muestras reales de la clase *healthy*, fueron consideradas como parte de la respectiva clase deseada (Tabla 3.11). Incluso, en los grupos *ruido* se encontró que las muestras son útiles en estos experimentos. A excepción del *grupo2* en el experimento RH2D2_PCA8 y el *grupo1* en el experimento RH2D3_PCA8.

Tabla 3.11. Resultados de agrupamiento de muestras artificiales de hojas de tomate con alguna enfermedad. Conjuntos generados a partir de imágenes reales de hojas sanas.

Agrupamiento de conjuntos de muestras artificiales					
Nombre de experimento	**Conjunto de muestras**	**Dimensiones de vector de características**	**Grupos**	**Cantidad de muestras en grupo**	**¿La mayoría de las muestras (>80%) pertenecen a la clase deseada (D1, D2, D3, D4)?**
RH2D1_PCA8	RH-to-D1	8 (PCA)	*grupo1*	455	Sí
			grupo2	198	Sí
			ruido	347	Sí
RH2D2_PCA8	RH-to-D2	8 (PCA)	*grupo1*	37	Sí
			grupo2	197	No
			ruido	766	Sí
RH2D3_PCA8	RH-to-D3	8 (PCA)	*grupo1*	818	Sí
			grupo2	15	No
			ruido	167	Sí
RH2D4_PCA8	RH-to-D4	8 (PCA)	*grupo1*	267	Sí
			grupo2	265	Sí
			ruido	468	Sí

Por otro lado, la mayoría de las muestras artificiales generadas a partir de muestras artificiales de la clase *healthy* (Tabla 3.12) fueron consideradas como no útiles. Esto indica que la distribución de las muestras artificiales de la clase *healthy* difiere de la distribución de las muestras reales de tal clase, provocando que el proceso de traducción no se lleve a cabo de manera correcta.

Tabla 3.12. Resultados de agrupamiento de muestras artificiales de hojas de tomate con alguna enfermedad. Conjuntos generados a partir de imágenes artificiales de hojas sanas.

	Agrupamiento de conjuntos de muestras artificiales				
Nombre de experimento	**Conjunto de muestras**	**Dimensiones de vector de características**	**Grupos**	**Cantidad de muestras en grupo**	**¿La mayoría de las muestras (>80%) pertenecen a la clase deseada (D1, D2, D3, D4)?**
FH2D1_PCA8	FH-to-D1	8 (PCA)	*grupo1*	107	No
			grupo2	461	No
			ruido	1,480	No
FH2D2_PCA8	FH-to-D2	8 (PCA)	*grupo1*	1,038	No
			grupo2	13	No
			grupo3	243	Sí
			ruido	754	No
FH2D3_PCA8	FH-to-D3	8 (PCA)	*grupo1*	1,120	Sí
			grupo2	254	No
			ruido	674	No
FH2D4_PCA8	FH-to-D4	8 (PCA)	*grupo1*	273	Sí
			grupo2	586	No
			ruido	1,189	No

Finalmente, las muestras de cada grupo considerado como útil se organizan en archivos de imagen, se realiza una última inspección visual para eliminar muestras manualmente, y se construyen cuatro bases de datos con muestras artificiales de cada clase deseada.

- *FakeD1*: corresponde a imágenes artificiales de la clase *Late Blight*. Contiene 804 muestras en total que provienen de los conjuntos *grupo1*, *grupo2* y *ruido*, del agrupamiento de RH-to-D1.
- *FakeD2*: corresponde a imágenes artificiales de la clase *Septoria Leaf Spot*. Contiene 786 muestras en total: 530 provienen de los conjuntos *grupo1* y *ruido*, del agrupamiento de RH-to-D2; y 256 provienen del conjunto *grupo3*, del agrupamiento de FH-to-D2.
- *FakeD3*: corresponde a imágenes artificiales de la clase *Target Spot Bacteria*. Contiene 818 muestras en total, las cuales provienen de los conjuntos *grupo1* y *ruido* del agrupamiento de RH-to-D3.
- *FakeD4*: corresponde a imágenes artificiales de la clase *Yellow Leaf Curl Virus*. Contiene 805 muestras en total: 532 provienen de los conjuntos *grupo1*, *grupo2* y *ruido* del agrupamiento de RH-to-D4; y 273 provienen del conjunto *grupo1* del agrupamiento de FH-to-D4.

Estas bases de datos son utilizadas en el entrenamiento de un modelo de redes profundas para la clasificación de hojas de tomate, con el objetivo de analizar el efecto que tiene en el desempeño del modelo. Esto se describe a detalle en la sección 4.2.

Como trabajo a futuro, es necesario probar esta metodología en más aplicaciones. A la fecha de conclusión de este proyecto de investigación, se trabaja en la clasificación de imágenes de leucocitos, en donde existe el potencial de aplicar este método de aumento de datos para la generación de una clase de leucocitos, de la cual no se cuenta con suficientes muestras, respecto a otras clases. En el apéndice A8 de esta tesis se muestran algunos avances en la implementación del método para tal aplicación.

3.6 Conclusiones

En este capítulo se presentó la implementación de GDAM-GAN para más de una aplicación, generando múltiples conjuntos de muestras artificiales. Durante la experimentación se determina que es necesario realizar una sintonización de hiperparámetros en algunas de las etapas del método (entrenamiento de modelo generativo base y modelo de traducción imagen-a-imagen) al aplicarlo en distintos problemas, no obstante, el procedimiento general se mantiene constante.

Se generaron bases de datos para varias aplicaciones distintas. La principal corresponde a la clasificación de uso de cubrebocas, para la cual se generó una base de datos con imágenes de la clase *IncorrectMask* y otra con imágenes pertenecientes a la clase *WithMask*. Otra aplicación en donde se implementó el método propuesto es en la clasificación de enfermedades de hojas de tomate, para la cual se generaron cuatro bases de datos, cada una de ellas con imágenes de una enfermedad de hoja de tomate. También se comenzó a implementar el método en el problema de clasificación de leucocitos (apéndice A8).

Durante la ejecución del método, se llevan a cabo evaluaciones cuantitativas de la veracidad de las muestras artificiales mediante la métrica FID, con el propósito de encontrar el punto de entrenamiento óptimo de los distintos elementos del método.

En el siguiente capítulo se realiza una evaluación de la calidad y utilidad de estos datos artificiales, al utilizarlos en el entrenamiento de modelos de redes profundas para dos aplicaciones: clasificación de imágenes de rostros utilizando cubrebocas; y clasificación de imágenes de hojas de tomate con alguna enfermedad.

CAPÍTULO IV. ENTRENAMIENTO DE REDES PROFUNDAS CON DATOS ARTIFICIALES

Con el propósito de analizar el impacto que tiene el uso de datos artificiales en el desempeño de distintos modelos de aprendizaje profundo, se llevan a cabo múltiples experimentos, en los cuales, el elemento variable corresponde al conjunto de datos de entrenamiento. Tal análisis se realiza, principalmente, en el problema de la detección del uso de cubrebocas en imágenes. Para esto, se realiza el entrenamiento de múltiples modelos utilizando distintas estructuras de datos con muestras reales y combinaciones de muestras reales y artificiales generadas mediante el método GDAM-GAN, así como otras técnicas tradicionales para incrementar el número de muestras. Adicionalmente, se analiza el problema de clasificación de enfermedades en imágenes de hojas de tomate.

La ejecución de los algoritmos se llevó a cabo en un equipo de cómputo con: un procesador AMD Ryzen™ 5600G @3.9 GHz; 16 GB de memoria RAM; y GPU Nvidia GeForce RTX™ 2070 Super con 8 GB de memoria. Para la implementación de los modelos de redes profundas se utilizó el lenguaje de programación Python 3, el framework PyTorch, y otras librerías para la gestión de imágenes y herramientas matemáticas (NumPy, OpenCV, Scikit-learn, etc).

4.1 Análisis de modelos de redes neuronales profundas para la detección del uso de cubrebocas en imágenes

En la sección 1.3.2 se menciona el estado del arte respecto al desarrollo de sistemas de visión para la detección del uso de cubrebocas. En estos trabajos, el enfoque más común es el uso de modelos de redes neuronales profundas, generalmente en dos etapas (Figura 4.1): la primera para la identificación de personas o rostros en la imagen; y la segunda para la clasificación de tal muestra, a tal etapa se le denomina en esta tesis como clasificador de uso de cubrebocas. Sin embargo, las bases de datos utilizadas varían entre los distintos trabajos, lo cual dificulta la comparación del desempeño entre los distintos modelos.

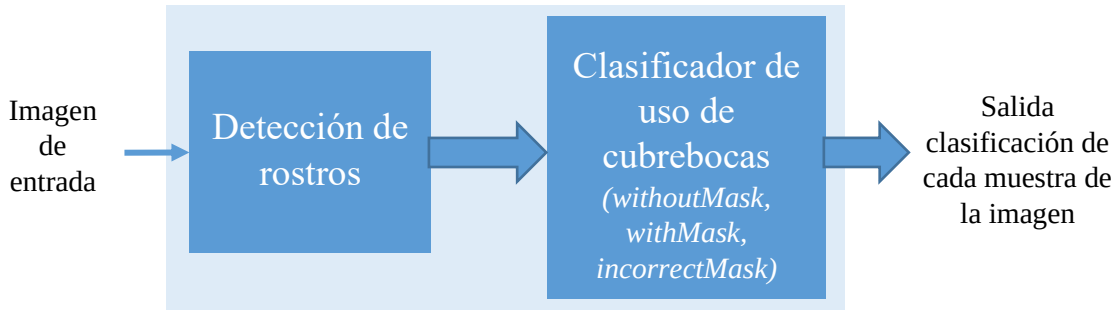


Figura 4.1. Esquema común de sistemas para la detección de cubrebocas en dos etapas: detección de rostros y clasificador de uso de cubrebocas.

En esta sección se presenta el análisis de seis modelos de aprendizaje profundo para la segunda etapa de un sistema común para la detección de uso de cubrebocas en imágenes, es decir, el clasificador de uso de cubrebocas, en donde se tienen las siguientes tres clases posibles: *withoutMask*, correspondiente a rostros sin cubrebocas; *withMask*, correspondiente a rostros utilizando cubrebocas correctamente; e *incorrectMask*, que corresponde a rostros utilizando cubrebocas incorrectamente, es decir, con la nariz y/o parte de la boca descubierta. Los modelos de redes neuronales profundas se seleccionan con base en las arquitecturas más comunes para este problema, según la literatura. Estas son:

- Net1-CNN: consiste en un modelo con tres capas convolucionales, cada una de ellas seguida por una capa de normalización de lote y función de activación tipo *LeakyReLU*, y dos capas de neuronas artificiales completamente conectadas, de las cuales, la primera utiliza una función de activación tipo *LeakyReLU* y la segunda corresponde a la capa de salida con activación tipo *Softmax* (véase apéndice A1). Este modelo está basado en una arquitectura similar a *AlexNET* [67] con una cantidad de parámetros reducida. El propósito de esta arquitectura es contar con una red de menor complejidad a comparación con el resto de los modelos utilizados.
- ResNet18 [37]: se trata de un modelo de red neuronal convolucional, compuesta por múltiples bloques en cascada, cada uno de estos bloques consiste en una capa convolucional (kernel con dimensiones 3x3, el número de filtros varia a lo largo de la arquitectura) con una función de activación tipo *ReLU*, seguida por otra capa de convolución (kernel con dimensiones 3x3, el número de filtros varia a lo largo de la

arquitectura). Cada bloque incluye una conexión residual, la cual consiste en sumar la entrada del bloque al resultado de la segunda capa de convolución. Finalmente se aplica una función de activación *ReLU* a la salida del bloque (véase apéndice A2). En este caso, la red cuenta con un total de 18 capas.

- ResNet34 [37]: este modelo es similar a *ResNet18*, pero con un total de 34 capas. Se compone también por bloques con conexión residual (véase apéndice A3).
- ResNet50 [37]: en este modelo, el bloque fundamental consiste en tres capas convolucionales: la primera con kernels de dimensiones 1x1, seguida por una función de activación tipo *ReLU*; la segunda capa contiene kernels de dimensión 3x3, seguida por una función de activación tipo *ReLU*; y la tercera capa se conforma por kernels de 1x1. Todos los bloques incluyen conexión residual al igual que en *ResNet18* y *ResNet34*. En total, la red cuenta con 50 capas (véase apéndice A3).
- MobileNetV2 [68]: este modelo está enfocado a la reducción de carga computacional conservando el desempeño respecto a otros modelos más complejos. Su arquitectura está basada en *ResNet*, y sus bloques fundamentales son similares a los utilizados en *ResNet50*, pero se sustituye la capa convolucional intermedia (kernels 3x3) por una capa de tipo *Depthwise Separable Convolution* (*dwise*) con kernels de la misma dimensión. Este tipo de capas se componen por filtros de un solo canal, los cuales se aplican únicamente a un canal de la información de entrada a la capa, de manera que el número de filtros en la capa *dwise* es igual al número de canales de la información de entrada de la capa, a diferencia de las capas convolucionales convencionales, en donde el número de filtros puede variar, pero cada filtro debe contar con el mismo número de canales que la información de entrada, lo cual implica un mayor número de parámetros a comparación con las capas *dwise*. (véase apéndice A3)
- InceptionV3 [27]: en este modelo se utilizan bloques tipo *Inception* en donde se aplican múltiples capas convolucionales, con una configuración en paralelo, y a la salida del bloque se concatena el resultado de cada capa. (véase apéndice A4)

En la siguiente sección se mencionan las características de la base de datos que se utiliza, para el entrenamiento de estas arquitecturas de redes profundas.

4.1.1 Base de datos PWMFD

En la sección 1.3.3 se mencionan algunas bases de datos enfocadas a la detección de cubrebocas, entre ellas se encuentra PWMFD. La distribución de las muestras, en las distintas clases para los conjuntos de entrenamiento y prueba de la base de datos PWMFD, se muestra en la Tabla 4.1.

Tabla 4.1. Distribución original de las muestras de PWMFD.

Clase	**Número de muestras**			**Porcentaje**
	Train	***Val***	**Total**	
WithoutMask	9680	791	10471	56.5%
WithMask	6702	993	7695	41.52%
IncorrectMask	320	46	366	1.97%
Todas las clases	16702	1830	18532	100%

Las muestras de esta base de datos contienen varios escenarios, orientaciones de rostro, tamaño de imagen, etc. En la Figura 4.2 se muestran algunas muestras. Aunque las muestras incluyen distintos estilos de cubrebocas, predominan los cubrebocas convencionales de color azul y blanco.

Figura 4.2. Ejemplos de muestras de base de datos PWMFD.

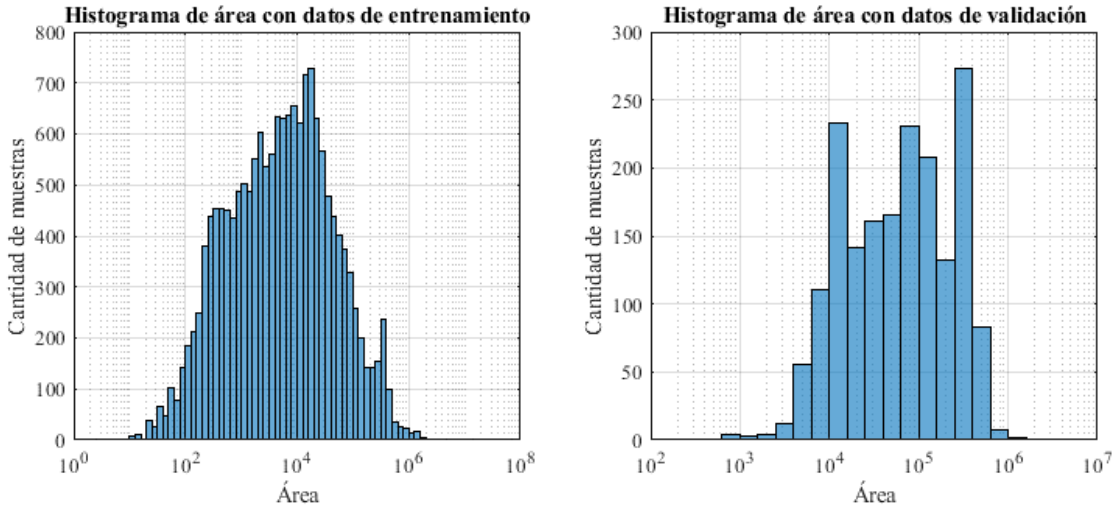
Luego de un análisis preliminar, mediante inspección visual, se determinó que aproximadamente el 2% de las muestras se encontraban etiquetadas incorrectamente y se realizaron las correcciones respectivas. La clase con la mayor cantidad de muestras corregidas es *WithMask*. En la Tabla 4.2 se muestra el resumen de estas correcciones, el número de

muestras corregidas se refiere a la cantidad de muestras cuya etiqueta marcaba una clase distinta a la clase real.

Tabla 4.2. Resumen de corrección de etiquetas en base de datos PWMFD.

Clase real	**Número de muestras corregidas**	
	Datos de entrenamiento	**Datos de validación**
WithoutMask	12	1
WithMask	249	68
IncorrectMask	81	6
Total	342 de 16702 (2.04%)	75 de 1830 (4.1%)

También se analizó el área las muestras para descartar aquellas imágenes que no contengan información útil para representar a alguna de las tres clases posibles y, por ende, sean poco útiles para el entrenamiento del clasificador de uso de cubrebocas. El área corresponde al producto de las dimensiones ancho por alto, en pixeles, del recuadro que delimita a una muestra. En la Figura 4.3 se muestra el histograma de los valores de área para las muestras de los conjuntos de entrenamiento y validación.



a) b)

Figura 4.3. Histograma de área. a) datos de conjunto de entrenamiento, b) datos de conjunto de validación.

Se determinó que las muestras con área pequeña, es decir, con un área menor a 1000 pixeles^2, tienden a tener más ruido que información relevante sobre la clase a la que pertenecen, incluso, a simple vista puede ser complicado distinguir su categoría, como se muestra en la Figura 4.4. Tales muestras fueron descartadas. En la Tabla 4.3 se muestra la distribución de las

muestras de PWMFD después de llevar a cabo la corrección de las etiquetas y descarte de muestras con área pequeña. En las siguientes secciones, al mencionar la base de datos PWMFD, se hace referencia al conjunto de datos después de realizar las correcciones y descarte de muestras.

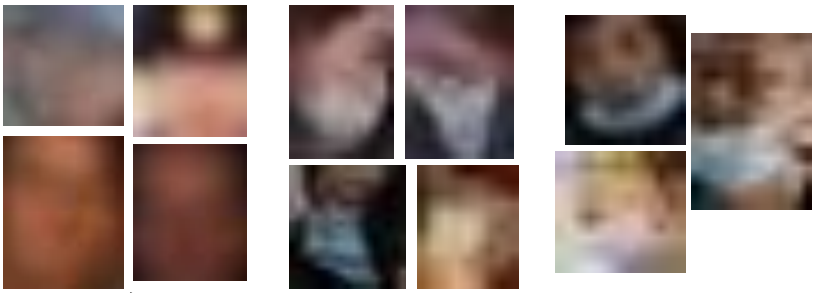


Figura 4.4. Ejemplos de muestras con áreas menores a 1000 pixeles, se muestran con acercamiento. a) clase *WithoutMask*, b) clase *WithMask*, c) clase *IncorrectMask*.

Tabla 4.3. Distribución de las muestras de PWMFD después de la corrección de etiquetas y descarte de muestras con área pequeña.

Clase	**Número de muestras**			**Porcentaje**
	Train	***Val***	**Total**	
WithoutMask	7343	718	8061	56.56%
WithMask	4742	1050	5792	40.64%
IncorrectMask	346	52	398	2.79%
Todas las clases	12431	1820	14251	100%

En las siguientes secciones se describe el uso de la base de datos PWMFD después de realizar correcciones de etiquetas y de eliminar muestras con área pequeña, así como de los conjuntos de muestras artificiales generados con el método de aumento de datos propuesto, en el entrenamiento de modelos de redes neuronales profundas para la clasificación de imágenes de rostros en las categorías *withoutMask*, *withMask*, e *incorrectMask*.

4.1.2 Configuración de entrenamiento de modelos

Cada experimento consiste en el entrenamiento de las seis arquitecturas de redes profundas mencionadas en la sección 4.1. En un mismo experimento, se utiliza un determinado conjunto de entrenamiento para todas las arquitecturas y tal proceso se realiza tomando en cuenta los siguientes hiperparámetros de entrenamiento en todos los casos:

- Optimizador tipo Adam (β_1=0.9, β_2=0.999)
- Razón de aprendizaje: 0.0001
- Se lleva a cabo el entrenamiento por 150 épocas.
- Se utiliza un entrenamiento por lotes (*mini-batch*) de 128 muestras en todos los modelos, excepto en *InceptionV3*, en el cual se utilizan 64 muestras debido a una restricción de memoria.
- Entrada: imagen RGB con dimensiones 128x128. Normalizada al intervalo [-1, 1].
- Para cada modelo, se inicializan los parámetros con los mismos valores a lo largo de todos los experimentos y así partir de un mismo punto de entrenamiento.

La configuración de entrenamiento mencionada se mantiene constante y lo que varía en cada uno de los experimentos, son los conjuntos de datos de entrenamiento. Se definen cuatro tipos de experimentos distintos, respecto a los datos de entrenamiento:

- Solo muestras reales: se refiere al experimento de referencia. Únicamente se utilizan las muestras existentes en la base de datos PWMFD.
- Reales + técnicas tradicionales de aumento de datos (TDA): se utilizan las muestras existentes de la base de datos PWMFD y se aplican técnicas de aumento de datos tradicional en la categoría deseada, la cual, en este caso, corresponde a la categoría *IncorrectMask*. Las TDA utilizadas son las siguientes: rotaciones, reflexiones, cambio de saturación, ecualización de histograma, y adición de ruido gaussiano.
- Reales + GDAM: se utilizan las muestras de PWMFD y se añaden las muestras artificiales generadas con GDAM-GAN (estas muestras se identifican como GDAM para fines prácticos).
- Reales + GDAM + TDA: se utiliza una combinación de muestras reales de PWMFD, muestras generadas con GDAM-GAN, y se aplican técnicas tradicionales de aumento de datos.

En la Tabla 4.4 se definen los experimentos que se realizan y se muestra la configuración de los datos de entrenamiento para cada uno de ellos.

Tabla 4.4. Estructura de datos de entrenamiento para distintos experimentos.

Estructura de datos de entrenamiento por clase										
Tipo de experimento	Nombre de experimento	*withoutMask*			*WithMask*			*IncorrectMask*		
		Real	GDAM	TDA	Real	GDAM	TDA	Real	GDAM	TDA
Solo reales	PWMFD-ref	7343	0	0	4742	0	0	346	0	0
Reales + TDA	PWMFD-TDA	7343	0	0	4742	0	0	346	0	5654
Reales + GDAM	PWMFD-GDAM-0	7343	0	0	4742	0	0	346	346	0
	PWMFD-GDAM-1	7343	0	0	4742	0	0	346	692	0
	PWMFD-GDAM-2	7343	0	0	4742	0	0	346	1384	0
	PWMFD-GDAM-3	7343	0	0	4742	0	0	346	2076	0
	PWMFD-GDAM-4	7343	0	0	4742	0	0	346	2300	0
Reales + GDAM + TDA	PWMFD-GDAM-TDA-0	7343	0	0	4742	0	0	346	346	5308
	PWMFD-GDAM-TDA-1	7343	0	0	4742	0	0	346	692	4962
	PWMFD-GDAM-TDA-2	7343	0	0	4742	0	0	346	1384	4270
	PWMFD-GDAM-TDA-3	7343	0	0	4742	0	0	346	2076	3578
	PWMFD-GDAM-TDA-4	7343	0	0	4742	0	0	346	2300	3354

En el experimento denominado PWMFD-ref, se utilizan las imágenes de la base de datos PWMFD después de llevar a cabo una corrección de etiquetas y de eliminar muestras con área pequeña (véase sección 4.1.1), tal experimento se toma como referencia para comparar el resto de los experimentos. Al utilizar TDA, se incrementa el número de muestras de manera que exista un balance entre las tres clases, para ello, se toma el promedio de muestras entre las clases *withoutMask* y *withMask*, el cual es de aproximadamente seis mil muestras. Respecto al uso de GDAM, se definen múltiples casos, empíricamente, según la proporción de muestras reales contra GDAM: uno a uno; uno a dos; uno a cuatro; uno a seis; y tomando todas las muestras GDAM disponibles. Estos experimentos se llevan a cabo para analizar el efecto de utilizar distintas proporciones de muestras reales y muestras artificiales, lo cual no está reportado claramente en la literatura. Finalmente, cuando se utilizan GDAM y TDA, se utiliza la misma proporción de muestras reales contra GDAM y luego se incrementa el número de muestras utilizando TDA hasta llegar a un máximo de seis mil muestras. Los experimentos PWMFD-TDA y PWMFD-GDAM-x se llevan a cabo con el propósito de comparar el efecto que pueden tener las muestras artificiales, generadas con técnicas tradicionales de aumento de datos y con la metodología GDAM-GAN, en el desempeño de los modelos que se entrenan.

Nótese que no se incluye un experimento en donde únicamente se utilizan muestras artificiales generadas con modelos GAN en el entrenamiento, ya que se ha demostrado que esto

puede provocar un efecto negativo en el desempeño de los modelos [39], por lo cual se omite tal configuración de datos de entrenamiento.

4.1.3 Análisis de resultados

A continuación, se muestra el desempeño de los seis modelos de redes profundas mencionados en la sección 4.1. Como métrica de desempeño se utiliza el error de clasificación, el cual se calcula utilizando la ecuación (4.1).

$$e = \frac{n_{error}}{n_{Total}} \tag{4.1}$$

En donde n_{error} es el número de muestras clasificadas incorrectamente y n_{Total} es el número total de muestras del conjunto de datos. También, se calcula el error de clasificación para cada categoría (que en este caso son tres: *withoutMask*, *withMask*, *incorrectMask*) mediante:

$$e_i = \frac{n_{i\,error}}{n_i} \quad i = 1, 2, ... \tag{4.2}$$

En donde $n_{i\ error}$ se refiere a las muestras, de la *i*-ésima categoría, que fueron clasificadas incorrectamente y n_i es el total de muestra de la *i*-ésima categoría.

En la Tabla 4.5 se muestra un resumen de los resultados obtenidos utilizando la media y desviación estándar del error mínimo de clasificación, con datos de validación, a lo largo de todos los modelos. Se muestra el error de cada una de las clases, así como el error total. La columna resaltada con color azul corresponde a la categoría *incorrectMask*, en la cual se aplica aumento de datos tradicional y/o aumento de datos con el método propuesto, y los resultados del experimento PWMFD-ref, en donde solamente se utilizan muestras reales, se toman como punto de referencia. En tal experimento, el error de clasificación en la clase *incorrectMask* es de 33.97%. En la mayoría del resto de los experimentos, en donde se aplica algún método de aumento de datos para la categoría *incorrectMask*, se obtiene un error de clasificación menor para tal clase, con respecto a la referencia en PWMFD-ref.

Tabla 4.5. Desempeño de clasificador de uso de cubrebocas. Media y desviación estándar de error mínimo de clasificación a lo largo de todos los modelos con datos de validación.

Tipo de experimento	Nombre de experimento	Media μ y desviación estándar σ de error mínimo de clasificación de todos los modelos con datos de validación							
		WithoutMask		*WithMask*		*IncorrectMask*		Error total	
		μ	σ	μ	σ	μ	σ	μ	σ
Solo reales	**PWMFD-ref**	**3.92%**	**1.25%**	**2.44%**	**0.81%**	**33.97%**	**20.96%**	**3.92%**	**1.39%**
Reales + TDA	PWMFD-TDA	4.06%	2.11%	3.71%	2.72%	17.95%	12.10%	4.25%	2.71%
Reales + GDAM	PWMFD-GDAM-0	3.99%	1.19%	3.03%	1.01%	33.65%	23.19%	4.28%	3.99%
	PWMFD-GDAM-1	3.74%	0.59%	3.03%	1.30%	34.29%	20.65%	4.20%	3.74%
	PWMFD-GDAM-2	3.57%	1.00%	3.43%	1.42%	30.77%	19.69%	4.26%	3.57%
	PWMFD-GDAM-3	4.46%	1.66%	3.44%	1.52%	30.45%	23.02%	4.61%	4.46%
	PWMFD-GDAM-4	3.81%	0.89%	4.34%	2.64%	28.53%	18.74%	4.82%	3.81%
Reales + GDAM + TDA	PWMFD-GDAM-TDA-0	3.83%	0.92%	3.88%	2.17%	23.08%	19.61%	4.41%	3.83%
	PWMFD-GDAM-TDA-1	4.64%	1.60%	3.79%	2.63%	28.53%	16.97%	4.83%	4.64%
	PWMFD-GDAM-TDA-2	4.76%	1.84%	4.65%	3.15%	16.35%	10.52%	5.03%	4.76%
	PWMFD-GDAM-TDA-3	3.97%	1.31%	5.14%	3.25%	19.55%	14.15%	5.09%	3.97%
	PWMFD-GDAM-TDA-4	4.16%	1.84%	4.31%	2.15%	24.04%	19.09%	4.81%	4.16%

El experimento en donde se obtuvo el error de clasificación de *incorrectMask* más bajo fue PWMFD-GDAM-TDA-2 en donde se utiliza una combinación de muestras GDAM y TDA, seguido por el experimento PWMFD-TDA en donde únicamente se aplican técnicas tradicionales de aumento de datos.

Respecto al error de clasificación con los datos de entrenamiento, los resultados se muestran en la Tabla 4.6. Tomando en cuenta la categoría más relevante, la cual es *incorrectMask*, el error promedio es de 9.87% con una desviación estándar de 20.32%, a lo largo de todas las arquitecturas. Tales valores disminuyen en todos los experimentos en donde se aplica algún método de aumento de datos. En el experimento PWMFD-GDAM-TDA-2, en donde se utilizan muestras reales + GDAM + TDA, se obtienen los mejores resultados, con un error promedio de 0.53% y una desviación estándar de 1.09% para la categoría *incorrectMask*.

Tabla 4.6. Desempeño de clasificador de uso de cubrebocas. Media y desviación estándar de error mínimo de clasificación a lo largo de todos los modelos con datos de entrenamiento.

Tipo de experimento	Nombre de experimento	Media μ y desviación estándar σ de error mínimo de clasificación de todos los modelos con datos de entrenamiento							
		WithoutMask		*WithMask*		*IncorrectMask*		Error total	
		μ	σ	μ	σ	μ	σ	μ	σ
Solo reales	**PWMFD-ref**	**0.64%**	**1.30%**	**0.69%**	**1.38%**	**9.87%**	**20.32%**	**0.93%**	**1.88%**
Reales + TDA	PWMFD-TDA	1.14%	2.33%	2.87%	5.86%	2.23%	4.58%	2.17%	4.44%
Reales + GDAM	PWMFD-GDAM-0	0.83%	1.69%	0.85%	1.77%	7.35%	14.97%	1.03%	2.11%
	PWMFD-GDAM-1	0.76%	1.51%	1.14%	2.34%	5.44%	10.95%	1.11%	2.26%
	PWMFD-GDAM-2	0.92%	1.78%	1.47%	3.03%	3.23%	6.61%	1.30%	2.64%
	PWMFD-GDAM-3	1.17%	2.26%	1.32%	2.61%	2.98%	5.88%	1.31%	2.57%
	PWMFD-GDAM-4	0.71%	1.39%	1.10%	2.22%	1.38%	2.79%	0.95%	1.91%
Reales + GDAM + TDA	PWMFD-GDAM-TDA-0	0.45%	0.93%	1.25%	2.60%	1.00%	2.00%	0.93%	1.93%
	PWMFD-GDAM-TDA-1	1.06%	2.09%	2.09%	4.28%	1.84%	3.44%	1.68%	3.39%
	PWMFD-GDAM-TDA-2	0.73%	1.48%	1.10%	2.21%	0.53%	1.09%	0.94%	1.89%
	PWMFD-GDAM-TDA-3	0.72%	1.45%	1.08%	2.23%	0.68%	1.35%	0.93%	1.90%
	PWMFD-GDAM-TDA-4	0.81%	1.64%	1.07%	2.18%	1.03%	1.98%	0.97%	1.97%

En las Tablas 4.5 y 4.6 se observa, de manera general, que los mejores resultados se obtienen al realizar una combinación de muestras reales + GDAM + TDA en el conjunto de entrenamiento. Sin embargo, al analizar los resultados con el conjunto de datos de validación (Tabla 4.5), en algunos casos se presenta, en la clase de interés (*incorrectMask*), una desviación estándar considerablemente mayor a comparación con los resultados al evaluar utilizando el conjunto de datos de entrenamiento. Por esta razón, se realiza un análisis del error de clasificación de *incorrectMask* con cada uno de los modelos entrenados, en la Tabla 4.7 se muestran los valores de error mínimo de cada una de las arquitecturas con datos de validación y en la Tabla 4.8 se muestra para los datos de entrenamiento. Se indica con colores rojo y verde los valores de error máximos y mínimos, respectivamente, para cada arquitectura a lo largo de todos los experimentos.

Tabla 4.7. Desempeño de clasificador de uso de cubrebocas. Error mínimo de clasificación para la categoría *IncorrectMask* con datos de validación.

	Nombre de experimento	Error mínimo de clasificación en categoría *IncorrectMask*					
		Net1-CNN	MobileNetV2	ResNet18	ResNet34	ResNet50	InceptionV3
Solo reales	**PWMFD-ref**	78.8462%	44.2308%	15.3846%	28.8462%	25.0000%	11.5385%
Reales + TDA	PWMFD-TDA	46.1538%	13.4615%	9.6154%	19.2308%	9.6154%	9.6154%
Reales + GDAM	PWMFD-GDAM-0	82.6923%	50.0000%	19.2308%	17.3077%	19.2308%	13.4615%
	PWMFD-GDAM-1	80.7692%	38.4615%	17.3077%	28.8462%	26.9231%	13.4615%
	PWMFD-GDAM-2	75.0000%	38.4615%	15.3846%	19.2308%	21.1538%	15.3846%
	PWMFD-GDAM-3	82.6923%	36.5385%	17.3077%	11.5385%	23.0769%	11.5385%
	PWMFD-GDAM-4	73.0769%	26.9231%	15.3846%	19.2308%	19.2308%	17.3077%
Reales + GDAM + TDA	PWMFD-GDAM-TDA-0	69.2308%	21.1538%	13.4615%	11.5385%	17.3077%	5.7692%
	PWMFD-GDAM-TDA-1	67.3077%	32.6923%	17.3077%	21.1538%	19.2308%	13.4615%
	PWMFD-GDAM-TDA-2	38.4615%	23.0769%	7.6923%	5.7692%	13.4615%	9.6154%
	PWMFD-GDAM-TDA-3	51.9231%	19.2308%	5.7692%	11.5385%	19.2308%	9.6154%
	PWMFD-GDAM-TDA-4	69.2308%	21.1538%	9.6154%	19.2308%	11.5385%	13.4615%

En la mayoría de los modelos, el mejor caso, al evaluar con datos de validación, se obtuvo en los experimentos donde se utiliza una combinación de muestras reales + GDAM + TDA durante el entrenamiento, a excepción de los modelos MobileNetV2 y ResNet50, los cuales alcanzan el menor valor de error al utilizar muestras reales + TDA.

El error de clasificación con datos de entrenamiento en la categoría *IncorrectMask* (Tabla 4.8) para la mayoría de los modelos tiende a ser cero a lo largo de todos los experimentos, a excepción del modelo con menor complejidad *Net1-CNN*. Además, al comparar los resultados mostrados en las Tablas 4.7 y 4.8, se determina que existe una diferencia entre las distribuciones de los datos de entrenamiento y los datos de validación; y podría estar presente el fenómeno de *overfitting*. Sin embargo, se determinó que tal fenómeno se presenta únicamente durante el entrenamiento de la red *Net1-CNN*, lo cual se puede apreciar en las gráficas de error de entrenamiento y validación durante la ejecución de los experimentos (las cuales se pueden

consultar en el Apéndice A5), en donde se observa que el error de validación converge a un valor aproximadamente constante para cada clase, en la mayoría de los modelos, excepto para la red *Net1-CNN* en donde se observa que el error de validación comienza a incrementar mientras que el error de entrenamiento continua disminuyendo hasta llegar a cero.

Tabla 4.8. Desempeño de clasificador de uso de cubrebocas. Error mínimo de clasificación para la categoría *IncorrectMask* con datos de entrenamiento.

	Nombre de experimento	Error mínimo de clasificación en categoría *IncorrectMask*					
		Net1-CNN	MobileNetV2	ResNet18	ResNet34	ResNet50	InceptionV3
Solo reales	**PWMFD-ref**	58.9595%	0.0000%	0.2890%	0.0000%	0.0000%	0.0000%
Reales + TDA	PWMFD-TDA	13.2833%	0.0000%	0.0000%	0.0000%	0.0333%	0.0833%
Reales + GDAM	PWMFD-GDAM-0	43.4971%	0.5780%	0.0000%	0.0000%	0.0000%	0.0000%
	PWMFD-GDAM-1	31.8882%	0.7707%	0.0000%	0.0000%	0.0000%	0.0000%
	PWMFD-GDAM-2	19.1908%	0.0000%	0.0000%	0.1734%	0.0000%	0.0000%
	PWMFD-GDAM-3	17.1759%	0.0413%	0.0000%	0.5367%	0.1239%	0.0000%
	PWMFD-GDAM-4	8.1255%	0.0000%	0.0378%	0.0000%	0.0000%	0.1134%
Reales + GDAM + TDA	PWMFD-GDAM-TDA-0	5.8333%	0.1833%	0.0000%	0.0000%	0.0000%	0.0000%
	PWMFD-GDAM-TDA-1	10.1333%	0.2833%	0.3000%	0.2500%	0.0500%	0.0000%
	PWMFD-GDAM-TDA-2	3.1667%	0.0167%	0.0000%	0.0000%	0.0000%	0.0167%
	PWMFD-GDAM-TDA-3	3.9500%	0.0500%	0.0000%	0.0500%	0.0000%	0.0167%
	PWMFD-GDAM-TDA-4	5.8000%	0.3833%	0.0000%	0.0167%	0.0000%	0.0000%

Al continuar con el análisis de las gráficas de error, se determinó que el uso de técnicas de aumento de datos reduce el número de épocas de entrenamiento necesarias para que el modelo generalice. En la Figura 4.5 se muestra una comparación entre las gráficas de error de clasificación de *incorrectMask*, con datos de entrenamiento, de cuatro experimentos: PWMFD-ref, que corresponde al experimento de referencia, en donde únicamente se utilizan muestras reales en el entrenamiento; PWMFD-TDA, en donde se utilizan muestras reales y aumento de datos tradicional (TDA); PWMFD-GDAM-2, en donde se utilizan muestras reales y GDAM; y PWMFD-GDAM-TDA-2, en donde se utilizan muestras reales, GDAM y TDA.

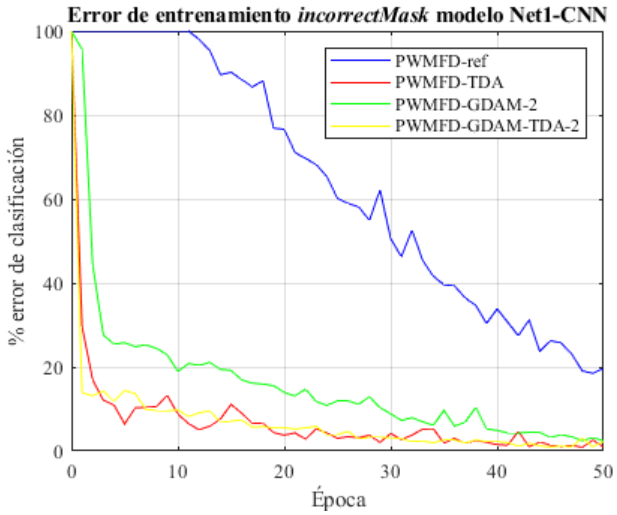


Figura 4.5. Gráfica de error de clasificación con datos de entrenamiento para los experimentos PWMFD-ref, PWMFD-TDA, PWMFD-GDAM-2, y PWMFD-GDAM-TDA-2.

En la Figura 4.5 se muestran los resultados al utilizar la arquitectura Net1-CNN, los resultados con el resto de las arquitecturas se pueden consultar en el Apéndice A6, en donde se puede observar que el comportamiento se repite a lo largo de todos los modelos. Esto indica que las técnicas de aumento de datos, además de que tienden a mejorar el desempeño de clasificación, reducen el tiempo de entrenamiento. Cabe mencionar que el impacto más significativo se obtiene en los modelos con menor complejidad (*Net1-CNN* y *MobileNetV2*).

En la siguiente sección se describen los experimentos llevados a cabo para otra aplicación y sus respectivos resultados.

4.2 Clasificación de imágenes de hojas de tomate con alguna enfermedad utilizando un modelo de redes neuronales profundas

En esta sección se presentan los experimentos llevados a cabo para analizar el efecto que tiene el uso de muestras artificiales en el entrenamiento de un modelo de redes profundas para la clasificación de enfermedades en imágenes de hojas de tomate. Tal problema se aborda en [3], en donde se utiliza una parte de la base de datos *Tomato Leaf Disease* [69] y se realiza aumento de datos mediante un modelo DCGAN, posteriormente, las muestras artificiales se emplean en el entrenamiento de la red GoogLeNet [65] para analizar la utilidad de tales muestras. La arquitectura de tal modelo incluye bloques denominados como módulos *Inception*,

los cuales se componen por múltiples capas convolucionales en una configuración tipo paralelo, tal como se muestra en la Figura 4.6.

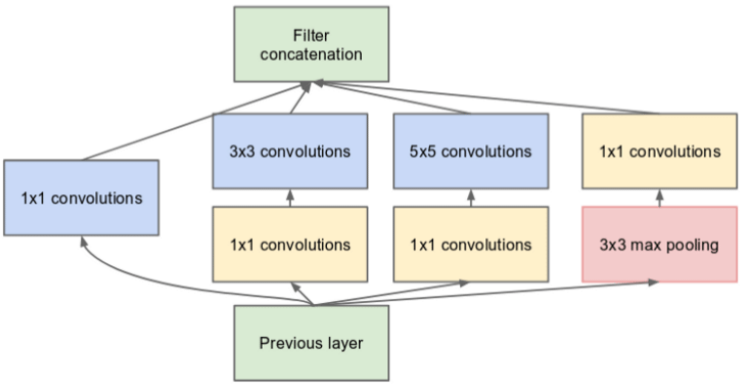


Figura 4.6. Módulo *Inception* utilizado en modelo *GoogLeNet*.

Y la estructura completa de la red GoogLeNet se describe en la Tabla 4.9, mencionando los detalles sobre el tamaño de *kernels*, *stride*, *padding*, y dimensiones de salida de cada una de las capas de la red.

Tabla 4.9. Arquitectura de red GoogLeNet

	Descripción de capa/bloque	**Dimensiones de salida**
	Entrada (*N* imágenes)	*N* x 3 x 224 x 224
1	Conv, 64, 7, 2, 1	*N* x 64 x 112 x 112
2	Max pooling, -, 3, 2	*N* x 64 x 56 x 56
3	Conv, 192, 3, 1, 1	*N* x 192 x 56 x 56
4	Max pooling, -, 3, 2	*N* x 192 x 28 x 28
5	Módulo inception	*N* x 256 x 28 x 28
6	Módulo inception	*N* x 480 x 28 x 28
7	Max pooling, -, 3, 2	*N* x 480 x 14 x 14
8	Módulo inception	*N* x 512 x 14 x 14
9	Módulo inception	*N* x 512 x 14 x 14
10	Módulo inception	*N* x 512 x 14 x 14
11	Módulo inception	*N* x 528 x 14 x 14
12	Módulo inception	*N* x 832 x 14 x 14
13	Max pooling, -, 3, 2	*N* x 832 x 7 x 7
14	Módulo inception	*N* x 832 x 7 x 7
15	Módulo inception	*N* x 1024 x 7 x 7
16	Average pooling, -, 7, 1	*N* x 1024 x 1 x 1
17	Dropout (40%)	*N* x 1024 x 1 x 1
18	Capa completamente conectada	*N* x 1000 x 1 x 1
19	Softmax	*N* x 1000 x 1 x 1

Los experimentos que se muestran en esta sección tienen el propósito de comparar el método GDAM-GAN, con la metodología utilizada en [3]. Tomando en cuenta las siguientes consideraciones:

- En [3] se aplica aumento de datos a cinco clases (*Healthy, LateBlight, SeptoriaLeafSpot, TargetSpot, YellowLeafCurlVirus*) mediante un modelo basado en DCGAN. Mientras que, en esta tesis, se aplica el método propuesto, considerando como clase base a la clase *Healthy*, y se tienen cuatro clases deseadas, correspondientes a *LateBlight, SeptoriaLeafSpot, TargetSpot,* y *YellowLeafCurlVirus*. El proceso de generación de muestras artificiales se describe en la sección 3.5.
- En [3] se utilizan 240 muestras reales de cada una de las cinco clases y, posteriormente, se añaden 760 muestras artificiales a cada clase y con tales conjuntos, se entrena un clasificador con una arquitectura *GoogLeNET*. Mientras que, en esta tesis, también se realiza el entrenamiento de tal arquitectura, pero los datos de entrenamiento cambian: se toman 1000 muestras reales de la clase base l_S=*Healthy*, no se añaden muestras artificiales a esta clase base; y 240 muestras de cada una de las clases deseadas l_T=*{LateBlight, SeptoriaLeafSpot, TargetSpot, YellowLeafCurlVirus}*, y se añaden 760 muestras artificiales a cada una de estas cuatro clases deseadas.
- En esta tesis, al igual que en [3], se utilizan 60 muestras reales de cada clase para el conjunto de prueba.

En la siguiente sección se mencionan las características de la base de datos utilizada para esta aplicación.

4.2.1 Base de datos *Tomato Leaf*

La base de datos *Tomato Leaf* se compone por más de catorce mil imágenes de hojas de tomate, divididas en diez categorías: nueve enfermedades, y una categoría correspondiente a hojas de tomate sin enfermedad. En [3] se realiza el análisis con cinco categorías: *Healthy* (H), *LateBlight* (D1), *SeptoriaLeafSpot* (D2), *TargetSpot* (D3), *YellowLeafCurlVirus* (D4). Y se

simula un entorno con un número reducido de muestras, seleccionando, de manera aleatoria, 300 imágenes por cada una de las clases. Estas muestras reales se dividen en dos conjuntos: 240 muestras de entrenamiento, las cuales se utilizan para entrenar un modelo DCGAN y generar muestras artificiales; y 60 muestras de prueba. La distribución de muestras utilizadas en [3] se muestra en la Tabla 4.10.

Tabla 4.10. Distribución de muestras de datos utilizada en [3].

Clase	**Entrenamiento**		**Prueba**
	Muestras reales	**Muestras DCGAN**	**Muestras reales**
Healthy	240	760	60
LateBlight	240	760	60
SeptoriaLeafSpot	240	760	60
TargetSpot	240	760	60
YellowLeafCurlVirus	240	760	60

En la Figura 3.13 se muestran algunos ejemplos de muestras reales de esta base de datos.

Por otro lado, en este proyecto de investigación, se definen múltiples conjuntos de entrenamiento:

- Solo muestras reales: se refiere al experimento de referencia. Únicamente se utilizan muestras reales de la base de datos *Tomato Leaf*. Se toman 1000 muestras de la clase *Healthy*; y 240 muestras de cada una de las clases *LateBlight, SeptoriaLeafSpot, TargetSpot, YellowLeafCurlVirus*.
- Reales + técnicas tradicionales de aumento de datos (TDA): se utilizan muestras reales de la base de datos *Tomato Leaf* y se aplican técnicas de aumento de datos tradicional en las clases deseadas l_T*={LateBlight, SeptoriaLeafSpot, TargetSpot, YellowLeafCurlVirus}*. Como TDA se utilizan las siguientes transformaciones de manera aleatoria: rotaciones, reflexiones, cambio de saturación, ecualización de histograma, y adición de ruido gaussiano.
- Reales + GDAM: se utilizan las muestras reales de *Tomato Leaf* y se añaden las muestras artificiales generadas con GDAM-GAN para las clases deseadas.
- Reales + GDAM + TDA: se utiliza una combinación de muestras reales de *Tomato Leaf*, muestras artificiales de las clases deseadas generadas con GDAM-GAN, y se aplican técnicas tradicionales de aumento de datos a tales clases deseadas.

En la Tabla 4.11 se muestra la distribución de los conjuntos de datos para los distintos experimentos que se llevan a cabo.

Tabla 4.11. Distribución de datos de entrenamiento para el problema de clasificación de hojas de tomate.

Tipo de experimento	**Nombre de experimento**	**Clase base l_S=*Healthy***			**Clases deseadas l_T={*D1,D2,D3,D4*} (cantidades por clase)**		
		Real	**GDAM**	**TDA**	**Real**	**GDAM**	**TDA**
Solo reales	Tomato-ref	1000	0	0	240	0	0
Reales + TDA	Tomato-TDA	1000	0	0	240	0	760
Reales + GDAM	Tomato-GDAM	1000	0	0	240	760	0
Reales + GDAM + TDA	Tomato-GDAM-TDA-0	1000	0	0	240	240	520
	Tomato-GDAM-TDA-1	1000	0	0	240	480	280

Cada uno de los experimentos mostrados en la Tabla 4.11, consiste, en el entrenamiento de un clasificador con arquitectura *GoogLeNet*. En la siguiente sección, se describe la configuración de tales experimentos y sus respectivos resultados.

4.2.2 Entrenamiento de red GoogLeNet para clasificación de hojas de tomate

En la sección 3.5 se reporta la generación de imágenes artificiales de hojas de tomate con alguna enfermedad. Tales muestras se emplean para entrenar una red GoogLeNet con el propósito de validar la veracidad de las imágenes artificiales. Para ello, se utiliza una configuración de hiperparámetros de entrenamiento similar a la utilizada en [3]:

- Optimizador gradiente descendiente con momento (μ=0.5)
- Razón de aprendizaje: 0.02 (se reduce a la mitad cada 512 iteraciones)
- Se lleva a cabo el entrenamiento por 50 épocas.
- Se utiliza un entrenamiento por lotes (*mini-batch*) de 16 muestras.
- Dato de entrada: imagen en el espacio de color RGB con dimensiones 224x224 pixeles de una hoja de tomate sana o con alguna enfermedad. Se aplica una normalización para ajustar el rango dinámico de valores de cada píxel al intervalo [-1, 1].

Después de cada época de entrenamiento se calcula el error de clasificación por cada clase y el error total de clasificación a lo largo de todas las clases, utilizando las ecuaciones (4.1) y (4.2). En la Tabla 4.12 se muestran los resultados de los experimentos llevados a cabo y se

incluye el desempeño reportado en [3]. Se indica con color verde el valor de error total más bajo, y el más alto se indica con color rojo.

Tabla 4.12. Resultados en clasificación de imágenes de hojas de tomate utilizando la arquitectura GoogLeNet. Error de clasificación por categoría.

Tipo de experimento	**Nombre de experimento**	**Error de clasificación mínimo por clase con datos de prueba**					**Error total**
		0 Healthy	**1 Late Blight**	**2 Septoria Leaf Spot**	**3 Target Spot**	**4 Yellow Leaf Curl Virus**	
Reportado en [3]	DCGAN	N/A	N/A	N/A	N/A	N/A	5.6666%
Solo reales	Tomato-ref	0.0000%	8.3333%	1.6667%	3.3333%	3.3333%	3.3333%
Reales + TDA	Tomato-TDA	0.0000%	3.3333%	8.3333%	3.3333%	0.0000%	3.0000%
Reales + GDAM	Tomato-GDAM	0.0000%	3.3333%	3.3333%	5.0000%	1.6667%	2.6666%
Reales + GDAM + TDA	Tomato-GDAM-TDA-0	0.0000%	1.6667%	1.6667%	5.0000%	0.0000%	1.6666%
	Tomato-GDAM-TDA-1	0.0000%	1.6667%	1.6667%	3.3333%	1.6667%	1.6666%

Los mejores resultados también se obtienen al utilizar muestras reales, GDAM, y técnicas tradicionales de aumento de datos en el conjunto de entrenamiento. Asimismo, se analizan las gráficas de error durante el proceso de entrenamiento. En este caso, al igual que en los experimentos de la sección 4.1, se reduce el número de épocas necesarias para que el modelo sea capaz de generalizar. Para el experimento Tomato-ref, esto sucede después de la época 10, por otro lado, en los experimentos en donde se aplican técnicas de aumento de datos, la generalización se alcanza después de la época 5 aproximadamente. Esto se puede observar en la Figura 4.7.

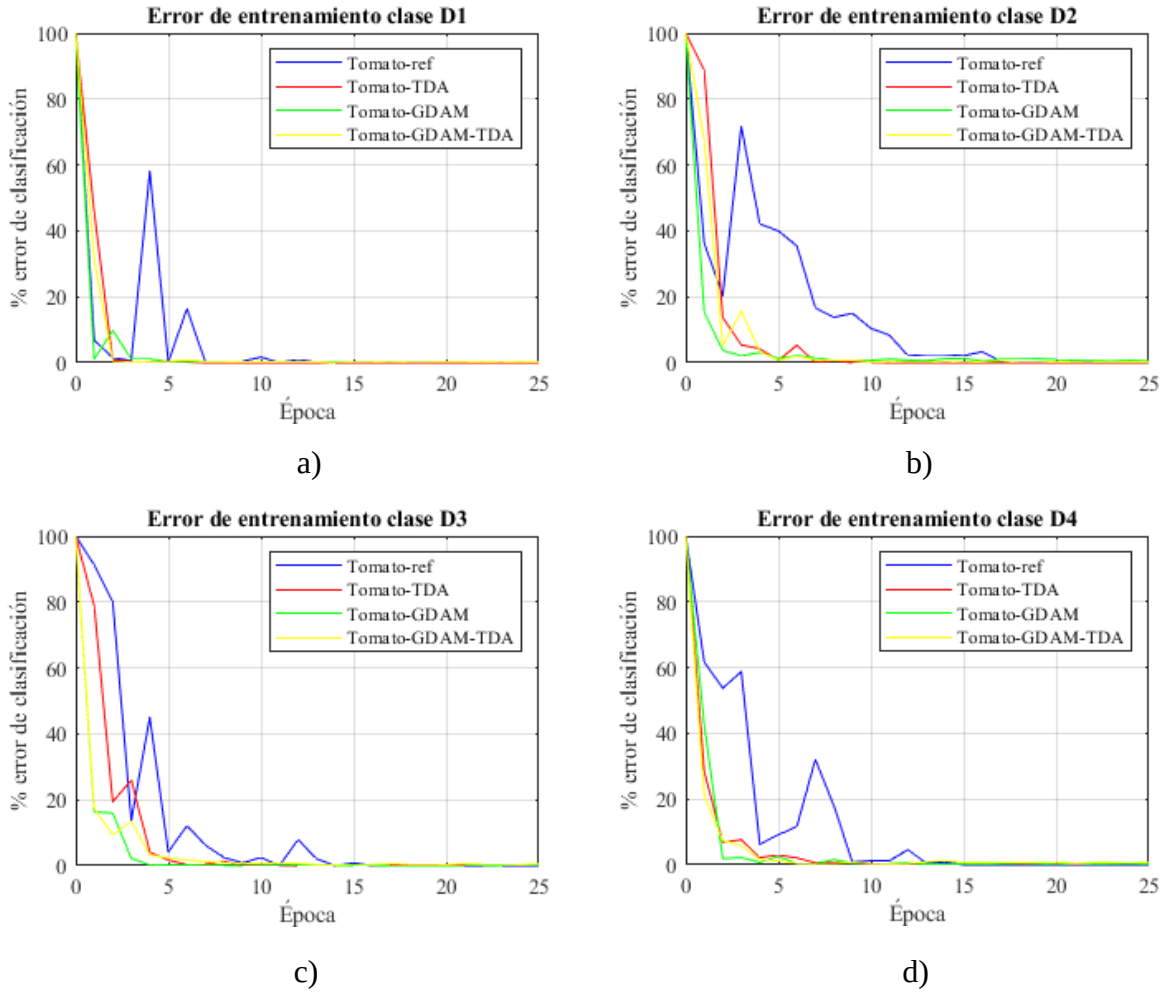


Figura 4.7. Comparación de gráficas de error de clasificación con datos de entrenamiento durante los distintos experimentos. a) Clase deseada *Late Blight* (D1), b) clase deseada *Septoria Leaf Spot* (D2), c) clase deseada *Target Spot Bacteria* (D3), d) clase deseada *Yellow Leaf Curl Virus* (D4).

4.3 Conclusiones

Los experimentos llevados a cabo en este capítulo tienen el propósito de evaluar el impacto que tiene el uso de muestras artificiales en el entrenamiento de modelos de redes profundas, frente al uso de muestras reales únicamente. Tal evaluación se llevó a cabo para dos aplicaciones distintas: clasificación de imágenes para la detección del uso de cubrebocas; y clasificación de imágenes de hojas de tomate con alguna enfermedad. En ambos casos, se generaron imágenes artificiales utilizando GDAM-GAN. En la clasificación de imágenes para la detección del uso de cubrebocas se analizaron seis arquitecturas con distinta complejidad; y

en la clasificación de imágenes de hojas de tomate se evaluó con una arquitectura específica para realizar la comparación con un trabajo existente en la literatura. Al analizar los experimentos realizados se determinaron dos puntos importantes respecto al uso de muestras artificiales en el entrenamiento de modelos de redes profundas:

- El uso de muestras artificiales generadas con GDAM-GAN, en conjunto con técnicas tradicionales de aumento de datos, reduce el error de clasificación con respecto al entrenamiento utilizando únicamente datos reales.
- El número de épocas de entrenamiento necesarias para alcanzar un desempeño de clasificación óptimo se reduce al utilizar muestras artificiales GDAM+TDA en el conjunto de datos de entrenamiento.

En el siguiente capítulo se mencionan las conclusiones de este proyecto de investigación, así como algunos puntos relevantes para trabajo a futuro.

CAPÍTULO V. CONCLUSIONES Y TRABAJO A FUTURO

En este proyecto de tesis se diseñó la metodología GDAM-GAN con tres elementos principales: modelo generativo base, modelo de traducción imagen-a-imagen, y una etapa de depuración. Para los primeros dos elementos, se implementan modelos basados en redes GAN, y la tercera etapa se realiza con la ayuda de un algoritmo de agrupamiento. Se definieron también las condiciones necesarias para aplicar este método:

- Existe un conjunto de muestras reales $\boldsymbol{x}_T$ pertenecientes a una clase deseada l_T.
- Existe una clase base l_S que comparte algunas características generales con la clase deseada l_T, pero que difieren en características propias de cada clase.
- Existe una gran cantidad de muestras $\boldsymbol{x}_S$ de la clase base l_S (respecto a la clase deseada l_T) y/o existe algún modelo generativo capaz de sintetizar muestras de l_S.

La experimentación llevada a cabo estuvo enfocada principalmente en el problema de detección del uso de cubrebocas en imágenes, con el objetivo de mejorar el desempeño de al menos un modelo de redes profundas que clasifica a una muestra como parte de alguna de las tres categorías posibles: *WithoutMask*, *WithMask*, e *IncorrectMask*. Esta última categoría se selecciona como la clase deseada, es decir, la clase en donde se aplica el aumento de datos l_T=*incorrectMask*, y como clase base se utilizan imágenes de rostros l_S=*rostros*. También se implementó el método propuesto en otra aplicación para demostrar que la metodología puede ser utilizada en múltiples problemas sin modificar el procedimiento general. Tal aplicación corresponde a la clasificación de imágenes de hojas de tomate con alguna enfermedad, contando con cinco categorías en total: hojas de tomate sanas, las cuales se seleccionan como clase base l_S=*healthy*; y cuatro enfermedades distintas, las cuales, en este caso, se seleccionan como clases deseadas en donde se aplica aumento de datos l_T=*{LateBlight, SeptoriaLeafSpot, TargetSpot, YellowLeafCurlVirus}*. Para ambas aplicaciones, el procedimiento general para la generación de conjuntos de muestras artificiales se mantuvo constante. Únicamente fue necesario llevar a cabo una sintonización de hiperparámetros de entrenamiento con base en lo reportado en la literatura para cada problema específico. Los conjuntos de muestras artificiales generados

fueron utilizados en el entrenamiento de redes neuronales profundas y se determinó que estas muestras permiten obtener las siguientes mejoras:

- Reduce el error de clasificación, respecto a únicamente utilizar las muestras reales disponibles. Y el mejor resultado se obtiene al combinar: muestras reales, muestras generadas con GDAM-GAN, y uso de técnicas tradicionales de aumento de datos. Esto indica que las muestras artificiales pertenecen a la distribución de la clase deseada y proveen una mayor variedad, lo cual permite que el modelo generalice de una mejor manera, respecto al entrenamiento con únicamente muestras reales. En el caso de la clasificación de uso de cubrebocas, de manera general se reduce el error de clasificación (de la clase deseada) de un 33.97% a un 16.34%. Y en el caso de la clasificación de hojas de tomate, se alcanza un error de clasificación de 1.66%, frente a un error de 5.66% reportado en la literatura al utilizar DCGAN.
- Reduce el número de épocas de entrenamiento necesarias para que la red neuronal profunda sea capaz de generalizar. Esto es observable en las gráficas de error durante el entrenamiento, en donde se aprecia que el valor converge en menos épocas cuando se utiliza aumento de datos. Para los experimentos de clasificación de uso de cubrebocas, dependiendo de la arquitectura, tal reducción se encuentra entre 5 y 50 épocas (véase Apéndice A6). Y para la clasificación de imágenes de hojas de tomate, la reducción es de 5 épocas aproximadamente (véase Figura 4.7).

Además, el método propuesto tiene la ventaja de ser un procedimiento semiautomático e independiente de la aplicación, a diferencia de los métodos tradicionales de aumento de datos, en donde es necesario analizar el problema para determinar las transformaciones adecuadas para generar nuevas muestras. No obstante, durante el proceso de diseño e implementación se identificaron algunas limitantes importantes respecto al método GDAM-GAN:

- El proceso de entrenamiento de algunas arquitecturas, para el modelo generativo base y la traducción imagen-a-imagen, puede ser tardado (más de 20 horas con el equipo de cómputo utilizado). Y es necesario sintonizar los hiperparámetros de entrenamiento de estas arquitecturas, especialmente los tamaños de lote y las razones de aprendizaje, lo cual es común en el área de aprendizaje profundo.

- La etapa de depuración, aunque ayuda a descartar muestras no útiles, no garantiza la veracidad de las muestras con respecto a la clase deseada y es posible que algunas muestras útiles no sean identificadas y/o que muestras poco útiles sean consideradas como parte de la clase deseada, lo cual podría tener un efecto negativo si se utilizan estas muestras en el entrenamiento de algún modelo de redes neuronales profundas. Al analizar visualmente las imágenes artificiales de *incorrectMask*, se estimó que aproximadamente un 18% de las muestras son útiles, sin embargo, tomando como base los resultados del algoritmo de agrupamiento se seleccionó únicamente el 1.75% de las muestras. Por otro lado, en el problema de clasificación de hojas de tomate, al realizar la etapa de depuración con el algoritmo HDBSCAN, en ocasiones no se generaban grupos o estos no contenían información relevante. Algunos factores que podrían afectar a esta etapa son: la arquitectura utilizada para la extracción de características de las muestras artificiales; las dimensiones de los vectores de características; el algoritmo de agrupamiento, etc.

A partir de tales limitantes se determinó el trabajo a futuro. Principalmente, es necesario implementar GDAM-GAN en más aplicaciones para validar el método genérico e identificar aquellas situaciones en donde este procedimiento funciona de manera óptima y en qué problemas no es apto. También es preciso analizar la relación de cantidad de muestras de la clase base respecto a la clase deseada, con el fin de establecer un requisito de cantidad mínima de muestras para obtener resultados aceptables. Otro punto relevante para tomar en cuenta es que los distintos elementos del método pueden ser implementados con distintas arquitecturas. Si se cuenta con el hardware adecuado, se podrían utilizar arquitecturas con mayor complejidad, como StyleGAN, que genera imágenes de alta fidelidad y veracidad, alcanzando valores de FID≈4 [2]. También, para mitigar la carga computacional del proceso de entrenamiento, es posible utilizar transferencia de aprendizaje (*transfer learning*) [41], en donde se toman como base los parámetros de un modelo entrenado previamente para resolver una problemática similar. Un elemento importante es la evaluación de las muestras artificiales generadas por los modelos GAN, a pesar de contar con algunas herramientas, como la métrica FID y los algoritmos de agrupamiento, es necesario incluir una evaluación visual, la cual es subjetiva y

depende de la aplicación específica. Para esta etapa, una alternativa podría ser utilizar métodos basados en meta-aprendizaje [70], tal paradigma permite aprovechar un conocimiento previo en el modelado de otras distribuciones que permita evaluar las muestras artificiales.

En este caso, GDAM-GAN se utilizó para generar imágenes artificiales, sin embargo, el procedimiento podría ser aplicable en otro tipo de señales, utilizando arquitecturas adecuadas para el modelo generativo base y generalizando la traducción imagen-a-imagen como traducción señal-a-señal. Por ejemplo, en el área de procesamiento de señales EEG [71].

GDAM-GAN es un método con enfoque en datos (*data-centric*) para la mejora del desempeño de modelos de aprendizaje profundo. Sin embargo, existen otros enfoques que pueden ser útiles en ciertas aplicaciones donde no se cumplan las consideraciones que requiere el método propuesto, por ejemplo, en el problema de *zero-shot learning*, en donde no se cuenta con muestras de entrenamiento, en estos casos se podrían utilizar métodos con enfoque en el modelo (*model-centric*).

Por último, se mencionan algunas consideraciones importantes y recomendaciones para el trabajo a futuro en el área de aumento de datos:

- La calidad de los datos es de suma importancia. Contar con grandes cantidades de muestras no garantiza un mejor entrenamiento. Es necesario tomar en cuenta el universo del problema que se desea resolver para encontrar una metodología de mejora de datos adecuada.
- Los modelos basados en GAN son una herramienta con un alto potencial para la generación de datos. Sin embargo, estos pueden generar datos no deseados y es necesario encontrar una manera automática para descartar tales muestras, con base en las características de la distribución de datos deseada.
- Es conveniente analizar la relación costo-beneficio. El método propuesto requiere de una mayor capacidad de procesamiento debido al tipo de modelos que se implementan, y el beneficio podría ser similar al obtenido cuando se emplean métodos de aumento de datos con menor complejidad.

REFERENCIAS

[1] A. Voulodimos, N. Doulamis, A. Doulamis, and E. Protopapadakis, "Deep Learning for Computer Vision: A Brief Review," *Comput Intell Neurosci*, vol. 2018, 2018, doi: 10.1155/2018/7068349.

[2] T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, and T. Aila, "Analyzing and Improving the Image Quality of StyleGAN," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 8107–8116, Dec. 2019, doi: 10.48550/arxiv.1912.04958.

[3] Q. Wu, Y. Chen, and J. Meng, "Dcgan-based data augmentation for tomato leaf disease identification," *IEEE Access*, vol. 8, pp. 98716–98728, 2020, doi: 10.1109/ACCESS.2020.2997001.

[4] V. Sandfort, K. Yan, P. J. Pickhardt, and R. M. Summers, "Data augmentation using generative adversarial networks (CycleGAN) to improve generalizability in CT segmentation tasks," *Scientific Reports 2019 9:1*, vol. 9, no. 1, pp. 1–9, Nov. 2019, doi: 10.1038/s41598-019-52737-x.

[5] H. C. Shin *et al.*, "Medical Image Synthesis for Data Augmentation and Anonymization Using Generative Adversarial Networks," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 11037 LNCS, pp. 1–11, Sep. 2018, doi: 10.1007/978-3-030-00536-8_1.

[6] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, 2nd ed. Nueva Jersey: Prentice Hall, 2002.

[7] L. Alzubaidi *et al.*, "Review of deep learning: concepts, CNN architectures, challenges, applications, future directions," *Journal of Big Data 2021 8:1*, vol. 8, no. 1, pp. 1–74, Mar. 2021, doi: 10.1186/S40537-021-00444-8.

[8] C. Shorten and T. M. Khoshgoftaar, “A survey on Image Data Augmentation for Deep Learning,” *J Big Data*, vol. 6, no. 1, pp. 1–48, Dec. 2019, doi: 10.1186/S40537-019-0197-0/FIGURES/33.

[9] R. Takahashi, T. Matsubara, and K. Uehara, “Data Augmentation Using Random Image Cropping and Patching for Deep CNNs,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 30, no. 9, pp. 2917–2931, Sep. 2020, doi: 10.1109/TCSVT.2019.2935128.

[10] Z. Zhong, L. Zheng, G. Kang, S. Li, and Y. Yang, “Random Erasing Data Augmentation,” *AAAI 2020 - 34th AAAI Conference on Artificial Intelligence*, pp. 13001–13008, Aug. 2017, doi: 10.48550/arxiv.1708.04896.

[11] C. Xie, M. Tan, B. Gong, J. Wang, A. L. Yuille, and Q. v. Le, “Adversarial examples improve image recognition,” *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 816–825, 2020, doi: 10.1109/CVPR42600.2020.00090.

[12] L. Gatys, A. Ecker, and M. Bethge, “A Neural Algorithm of Artistic Style,” *J Vis*, vol. 16, no. 12, pp. 326–326, Aug. 2016, doi: 10.1167/16.12.326.

[13] I. Goodfellow *et al.*, “Generative Adversarial Networks,” *Commun ACM*, vol. 63, no. 11, pp. 139–144, Jun. 2014, doi: 10.48550/arxiv.1406.2661.

[14] Z. Pan, W. Yu, X. Yi, A. Khan, F. Yuan, and Y. Zheng, “Recent Progress on Generative Adversarial Networks (GANs): A Survey,” *IEEE Access*, vol. 7, pp. 36322–36333, 2019, doi: 10.1109/ACCESS.2019.2905015.

[15] A. Radford, L. Metz, and S. Chintala, “Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks,” *4th International Conference on Learning Representations, ICLR 2016 - Conference Track Proceedings*, Nov. 2015, doi: 10.48550/arxiv.1511.06434.

[16] M. Mirza and S. Osindero, “Conditional Generative Adversarial Nets,” Nov. 2014, doi: 10.48550/arxiv.1411.1784.

[17] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel, “InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets,” *Adv Neural Inf Process Syst*, pp. 2180–2188, Jun. 2016, doi: 10.48550/arxiv.1606.03657.

[18] M. Arjovsky, S. Chintala, and L. Bottou, “Wasserstein Generative Adversarial Networks,” in *Proceedings of the 34th International Conference on Machine Learning*, 2017, pp. 214–223.

[19] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. Courville, “Improved Training of Wasserstein GANs,” *Adv Neural Inf Process Syst*, vol. 30, pp. 5768–5778, Mar. 2017, doi: 10.48550/arxiv.1704.00028.

[20] T. Karras, S. Laine, and T. Aila, “A Style-Based Generator Architecture for Generative Adversarial Networks,” *IEEE Trans Pattern Anal Mach Intell*, vol. 43, no. 12, pp. 4217–4228, Dec. 2018, doi: 10.48550/arxiv.1812.04948.

[21] T. Karras, T. Aila, S. Laine, and J. Lehtinen, “Progressive Growing of GANs for Improved Quality, Stability, and Variation,” *6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings*, Oct. 2017, doi: 10.48550/arxiv.1710.10196.

[22] A. Brock, J. Donahue, and K. Simonyan, “Large Scale GAN Training for High Fidelity Natural Image Synthesis,” *7th International Conference on Learning Representations, ICLR 2019*, Sep. 2018, doi: 10.48550/arxiv.1809.11096.

[23] J. Deng, W. Dong, R. Socher, L.-J. Li, Kai Li, and Li Fei-Fei, “ImageNet: A large-scale hierarchical image database,” pp. 248–255, Mar. 2010, doi: 10.1109/CVPR.2009.5206848.

[24] H. Zhang, I. Goodfellow, D. Metaxas, and A. Odena, “Self-Attention Generative Adversarial Networks,” in *36th International Conference on Machine Learning, ICML 2019*, May 2018, vol. 2019-June, pp. 12744–12753. doi: 10.48550/arxiv.1805.08318.

[25] J. Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks," in *Proceedings of the IEEE International Conference on Computer Vision*, Dec. 2017, vol. 2017-October, pp. 2242–2251. doi: 10.1109/ICCV.2017.244.

[26] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved Techniques for Training GANs," *Adv Neural Inf Process Syst*, pp. 2234–2242, Jun. 2016, doi: 10.48550/arxiv.1606.03498.

[27] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the Inception Architecture for Computer Vision," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Dec. 2015, vol. 2016-December, pp. 2818–2826. doi: 10.48550/arxiv.1512.00567.

[28] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium," *Adv Neural Inf Process Syst*, vol. 30, pp. 6627–6638, Jun. 2017, doi: 10.48550/arxiv.1706.08500.

[29] K. Shmelkov, C. Schmid, and K. Alahari, "How Good Is My GAN?," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 11206 LNCS, pp. 218–234, 2018, doi: 10.1007/978-3-030-01216-8_14.

[30] M. Frid-Adar, E. Klang, M. Amitai, J. Goldberger, and H. Greenspan, "Synthetic data augmentation using GAN for improved liver lesion classification," in *Proceedings - International Symposium on Biomedical Imaging*, May 2018, vol. 2018-April, pp. 289–293. doi: 10.1109/ISBI.2018.8363576.

[31] Y. Luo, "EEG Data Augmentation for Emotion Recognition Using a Conditional Wasserstein GAN," in *Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBS*, Oct. 2018, vol. 2018-July, pp. 2535–2538. doi: 10.1109/EMBC.2018.8512865.

[32] Y. Choi, M. Choi, M. Kim, J. W. Ha, S. Kim, and J. Choo, "StarGAN: Unified Generative Adversarial Networks for Multi-domain Image-to-Image Translation,"

Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pp. 8789–8797, Dec. 2018, doi: 10.1109/CVPR.2018.00916.

[33] A. Köksal and S. Lu, "RF-GAN: A Light and Reconfigurable Network for Unpaired Image-to-Image Translation," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 12625 LNCS, pp. 542–559, Nov. 2020, doi: 10.1007/978-3-030-69538-5_33.

[34] C. Ding, W. Kang, J. Zhu, and S. Du, "InjectionGAN: Unified Generative Adversarial Networks for Arbitrary Image Attribute Editing," *IEEE Access*, vol. 8, pp. 117726–117735, 2020, doi: 10.1109/ACCESS.2020.3003139.

[35] A. Antoniou, A. Storkey, and H. Edwards, "Augmenting Image Classifiers Using Data Augmentation Generative Adversarial Networks," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 11141 LNCS, pp. 594–603, Oct. 2018, doi: 10.1007/978-3-030-01424-7_58.

[36] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, "Densely Connected Convolutional Networks," in *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, Aug. 2016, vol. 2017-January, pp. 2261–2269. doi: 10.48550/arxiv.1608.06993.

[37] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Dec. 2016, vol. 2016-December, pp. 770–778. doi: 10.1109/CVPR.2016.90.

[38] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 9351, pp. 234–241, 2015, doi: 10.1007/978-3-319-24574-4_28.

[39] S. Ravuri and O. Vinyals, “Seeing is Not Necessarily Believing: Limitations of BigGANs for Data Augmentation | OpenReview,” 2019. https://openreview.net/forum?id=rJMw747l_4 (accessed Mar. 13, 2022).

[40] V. Besnier, H. Jain, A. Bursuc, M. Cord, and P. Pérez, “This Dataset Does Not Exist: Training Models from Generated Images,” in *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, pp. 1–5. doi: 10.1109/ICASSP40776.2020.9053146.

[41] K. Weiss, T. M. Khoshgoftaar, and D. D. Wang, “A survey of transfer learning,” *J Big Data*, vol. 3, no. 1, pp. 1–40, Dec. 2016, doi: 10.1186/S40537-016-0043-6/TABLES/6.

[42] Y. Wang, Q. Yao, J. T. Kwok, and L. M. Ni, “Generalizing from a Few Examples,” *ACM Computing Surveys (CSUR)*, vol. 53, no. 3, Jun. 2020, doi: 10.1145/3386252.

[43] J. Dong, B. Xiao, B. Ding, and H. Wang, “GT-GAN: A General Transductive Zero-Shot Learning Method Based on GAN,” *IEEE Access*, vol. 8, pp. 147173–147184, 2020, doi: 10.1109/ACCESS.2020.3015334.

[44] I. Buciu, “Color quotient based mask detection,” *2020 14th International Symposium on Electronics and Telecommunications, ISETC 2020 - Conference Proceedings*, Nov. 2020, doi: 10.1109/ISETC50328.2020.9301079.

[45] G. Jignesh Chowdary, N. S. Punn, S. K. Sonbhadra, and S. Agarwal, “Face Mask Detection Using Transfer Learning of InceptionV3,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 12581 LNCS, pp. 81–90, Dec. 2020, doi: 10.1007/978-3-030-66665-1_6.

[46] M. R. Bhuiyan, S. A. Khushbu, and M. S. Islam, “A Deep Learning Based Assistive System to Classify COVID-19 Face Mask for Human Safety with YOLOv3,” *2020 11th International Conference on Computing, Communication and Networking Technologies, ICCCNT 2020*, Jul. 2020, doi: 10.1109/ICCCNT49239.2020.9225384.

[47] J. Yu and W. Zhang, “Face Mask Wearing Detection Algorithm Based on Improved YOLO-v4,” *Sensors*, vol. 21, no. 9, p. 3263, May 2021, doi: 10.3390/S21093263.

[48] S. Hussain *et al.*, “IoT and Deep Learning Based Approach for Rapid Screening and Face Mask Detection for Infection Spread Control of COVID-19,” *Applied Sciences*, vol. 11, no. 8, p. 3495, Apr. 2021, doi: 10.3390/APP11083495.

[49] B. Qin and D. Li, “Identifying Facemask-Wearing Condition Using Image Super-Resolution with Classification Network to Prevent COVID-19,” *Sensors 2020, Vol. 20, Page 5236*, vol. 20, no. 18, p. 5236, Sep. 2020, doi: 10.3390/S20185236.

[50] K. Zhang, Z. Zhang, Z. Li, and Y. Qiao, “Joint Face Detection and Alignment Using Multitask Cascaded Convolutional Networks,” *IEEE Signal Process Lett*, vol. 23, no. 10, pp. 1499–1503, Oct. 2016, doi: 10.1109/LSP.2016.2603342.

[51] M. Loey, G. Manogaran, M. H. N. Taha, and N. E. M. Khalifa, “Fighting against COVID-19: A novel deep learning model based on YOLO-v2 with ResNet-50 for medical face mask detection,” *Sustain Cities Soc*, vol. 65, p. 102600, Feb. 2021, doi: 10.1016/J.SCS.2020.102600.

[52] W. Han, Z. Huang, A. Kuerban, M. Yan, and H. Fu, “A Mask Detection Method for Shoppers under the Threat of COVID-19 Coronavirus,” *Proceedings - 2020 International Conference on Computer Vision, Image and Deep Learning, CVIDL 2020*, pp. 442–447, Jul. 2020, doi: 10.1109/CVIDL51233.2020.00-54.

[53] A. S. Joshi, S. S. Joshi, G. Kanahasabai, R. Kapil, and S. Gupta, “Deep Learning Framework to Detect Face Masks from Video Footage,” *Proceedings - 2020 12th International Conference on Computational Intelligence and Communication Networks, CICN 2020*, pp. 435–440, Sep. 2020, doi: 10.1109/CICN49253.2020.9242625.

[54] G. T. S. Draughon, P. Sun, and J. P. Lynch, “Implementation of a Computer Vision Framework for Tracking and Visualizing Face Mask Usage in Urban Environments,” *2020 IEEE International Smart Cities Conference, ISC2 2020*, Sep. 2020, doi: 10.1109/ISC251055.2020.9239012.

[55] E. Mbunge, S. Simelane, S. G. Fashoto, B. Akinnuwesi, and A. S. Metfula, "Application of deep learning and machine learning models to detect COVID-19 face masks - A review," *Sustainable Operations and Computers*, vol. 2, pp. 235–245, Jan. 2021, doi: 10.1016/J.SUSOC.2021.08.001.

[56] X. Jiang, T. Gao, Z. Zhu, and Y. Zhao, "Real-Time Face Mask Detection Method Based on YOLOv3," *Electronics (Basel)*, vol. 10, no. 7, p. 837, Apr. 2021, doi: 10.3390/ELECTRONICS10070837.

[57] "Face Mask Detection," 2020. https://www.kaggle.com/datasets/andrewmvd/face-mask-detection (accessed Sep. 29, 2021).

[58] "Masked Face Detection In the Wild dataset," 2020. https://www.kaggle.com/datasets/sigmind/masked-face-detection-wider-dataset (accessed Sep. 29, 2021).

[59] Z. Wang *et al.*, "Masked Face Recognition Dataset and Application," Mar. 2020, doi: 10.48550/arxiv.2003.09093.

[60] A. Cabani, K. Hammoudi, H. Benhabiles, and M. Melkemi, "MaskedFace-Net – A dataset of correctly/incorrectly masked face images in the context of COVID-19," *Smart Health*, vol. 19, p. 100144, Mar. 2021, doi: 10.1016/J.SMHL.2020.100144.

[61] S. Theodoridis and K. Koutroumbas, *Pattern recognition*. Elsevier/Academic Press, 2006.

[62] R. J. G. B. Campello, D. Moulavi, and J. Sander, "Density-based clustering based on hierarchical density estimates," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 7819 LNAI, no. 2, pp. 160–172, 2013, doi: 10.1007/978-3-642-37456-2_14/COVER/.

[63] Vojtěch Jarník, "On a certain problem of minimization," *Práce moravské přirodovědecké společnosti*, vol. 6, pp. 57–63, 1930.

[64] A. Creswell, T. White, V. Dumoulin, K. Arulkumaran, B. Sengupta, and A. A. Bharath, "Generative Adversarial Networks: An Overview," *IEEE Signal Process Mag*, vol. 35, no. 1, pp. 53–65, Jan. 2018, doi: 10.1109/MSP.2017.2765202.

[65] C. Szegedy *et al.*, "Going deeper with convolutions," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 07-12-June-2015, pp. 1–9, Oct. 2015, doi: 10.1109/CVPR.2015.7298594.

[66] David. P. Hughes and M. Salathe, "An open access repository of images on plant health to enable the development of mobile disease diagnostics," Nov. 2015, doi: 10.48550/arxiv.1511.08060.

[67] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in *Advances in Neural Information Processing Systems*, 2012, vol. 25. [Online]. Available: https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf

[68] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L. C. Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 4510–4520, Jan. 2018, doi: 10.48550/arxiv.1801.04381.

[69] "Tomato leaf disease detection | Kaggle." https://www.kaggle.com/datasets/kaustubhb999/tomatoleaf (accessed Apr. 05, 2022).

[70] E. Rivas-Posada and M. I. Chacon-Murguia, "General meta-learning paradigm based on prior-models, meta-model, meta-algorithm, and few-shot-base-model," in *Proceedings of the International Joint Conference on Neural Networks*, Jul. 2021, vol. 2021-July, pp. 1–8. doi: 10.1109/IJCNN52387.2021.9533374.

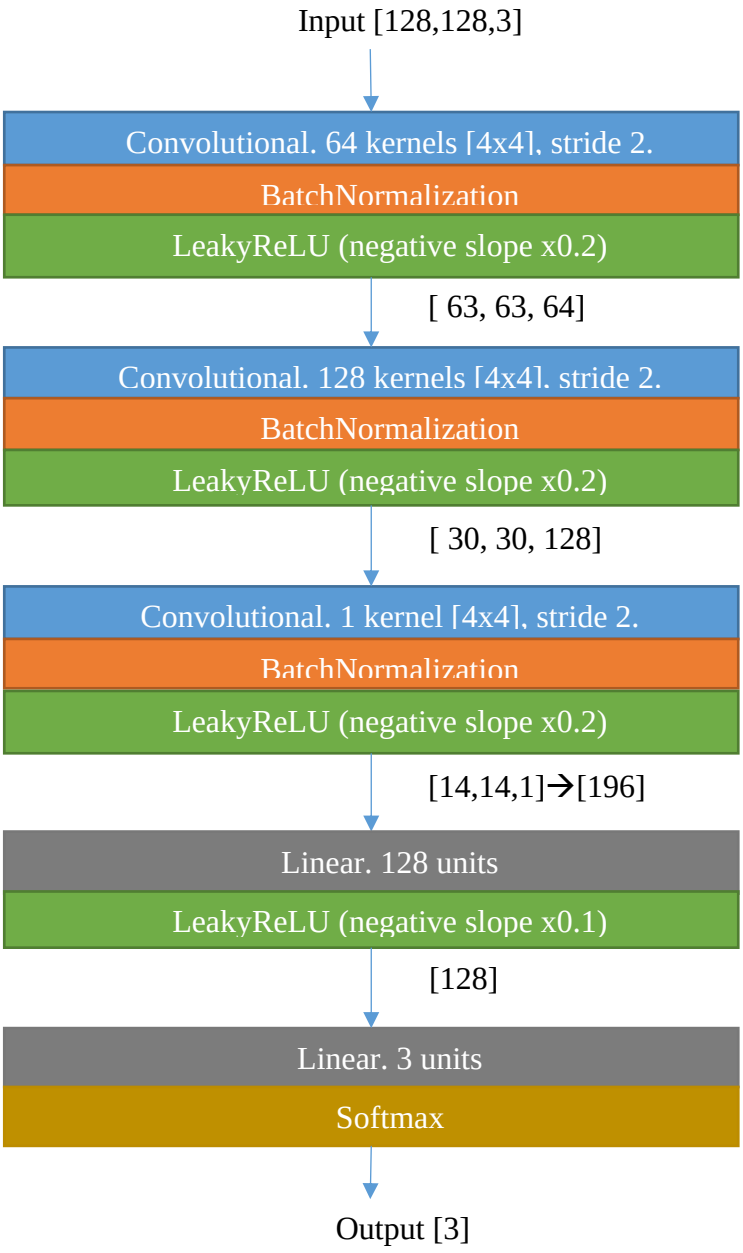
[71] J. M. Macias-Macias, J. Alberto Ramirez-Quintana, J. Salvador, A. Méndez-Aguirre, M. Ignaciochacon-Murguia, and A. D. Corral-Saenz, "Procesamiento Embebido de P300 Basado en Red Neuronal Convolucional para Interfaz Cerebro-Computadora Ubicua,"

ReCIBE, Revista electrónica de Computación, Informática, Biomédica y Electrónica, vol. 9, no. 2, pp. 1–24, Feb. 2020, doi: 10.32870/RECIBE.V9I2.153.

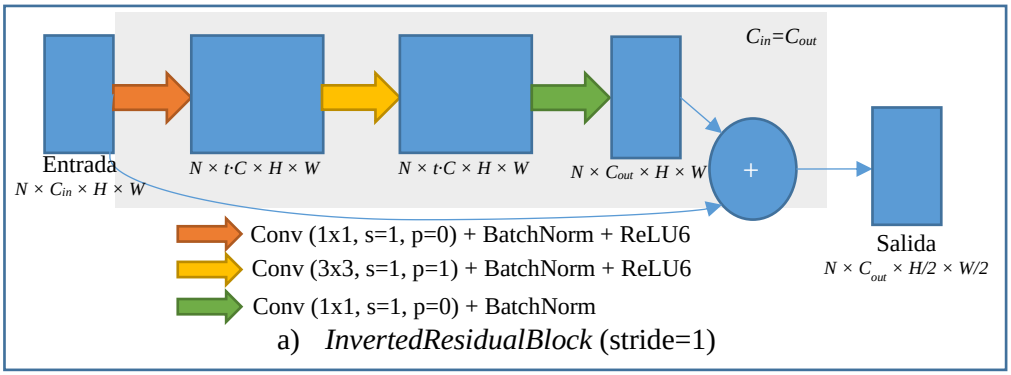
[72] "Guía avanzada de Inception v3 | Cloud TPU | Google Cloud." https://cloud.google.com/tpu/docs/inception-v3-advanced?hl=es-419 (accessed May 15, 2022).

APÉNDICES

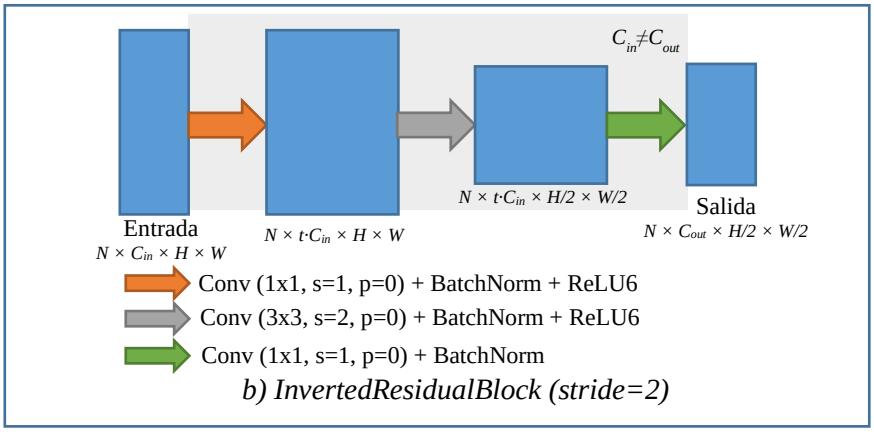
A1. Arquitectura de red Net1-CNN



A2. Arquitectura de red MobileNetV2



a) *InvertedResidualBlock* (stride=1)

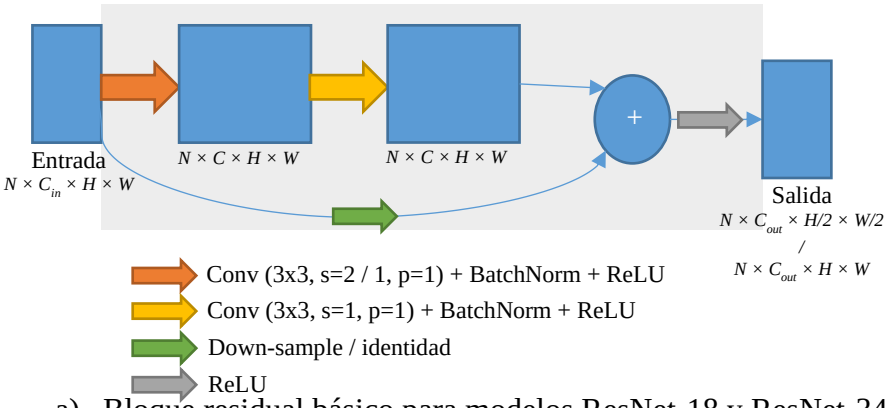


b) InvertedResidualBlock (stride=2)

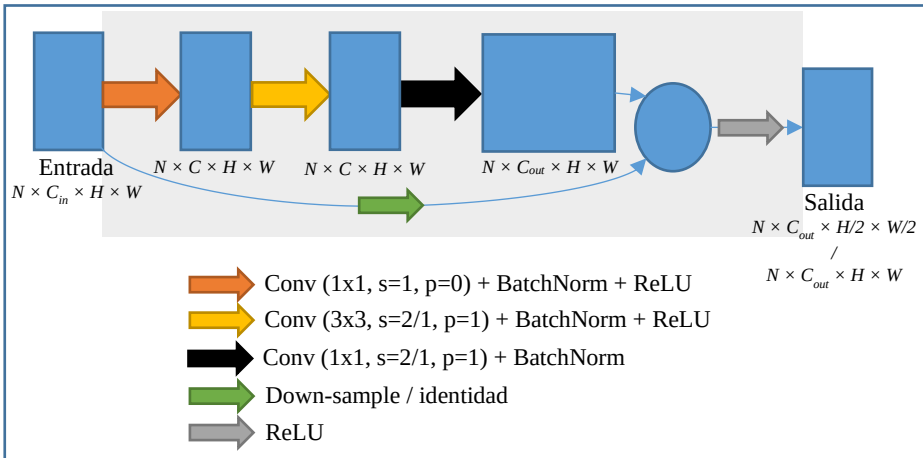
Entrada	**Tipo de capa o bloque**	**t**	**C_{out}**	**s**
$224^2 \times 3$	Conv + BatchNorm + ReLU6	-	32	2
$112^2 \times 32$	InvertedResidual (stride=1)	1	16	1
$112^2 \times 16$	InvertedResidual (stride=2)	6	24	2
$56^2 \times 24$	InvertedResidual (stride=1)	6	24	1
$56^2 \times 24$	InvertedResidual (stride=2)	6	32	2
$28^2 \times 32$	InvertedResidual (stride=1)	6	32	1
$28^2 \times 32$	InvertedResidual (stride=1)	6	32	1
$28^2 \times 32$	InvertedResidual (stride=2)	6	64	2
$14^2 \times 64$	InvertedResidual (stride=1)	6	64	1
$14^2 \times 64$	InvertedResidual (stride=1)	6	64	1
$14^2 \times 64$	InvertedResidual (stride=1)	6	64	1
$14^2 \times 64$	InvertedResidual (stride=1)	6	96	1
$14^2 \times 96$	InvertedResidual (stride=1)	6	96	1
$14^2 \times 96$	InvertedResidual (stride=1)	6	96	1
$14^2 \times 96$	InvertedResidual (stride=2)	6	160	2
$7^2 \times 160$	InvertedResidual (stride=1)	6	160	1
$7^2 \times 160$	InvertedResidual (stride=1)	6	160	1
$7^2 \times 160$	InvertedResidual (stride=1)	6	320	1
$7^2 \times 320$	Conv + BatchNorm + ReLU6	-	1280	1
$7^2 \times 1280$	AveragePooling	-	-	-
1×1280	Linear (Fully connected layer)	-	# clases	-

c) Arquitectura de modelo MobileNetV2 [68]

A3. Arquitectura general de modelos ResNet



a) Bloque residual básico para modelos ResNet-18 y ResNet-34

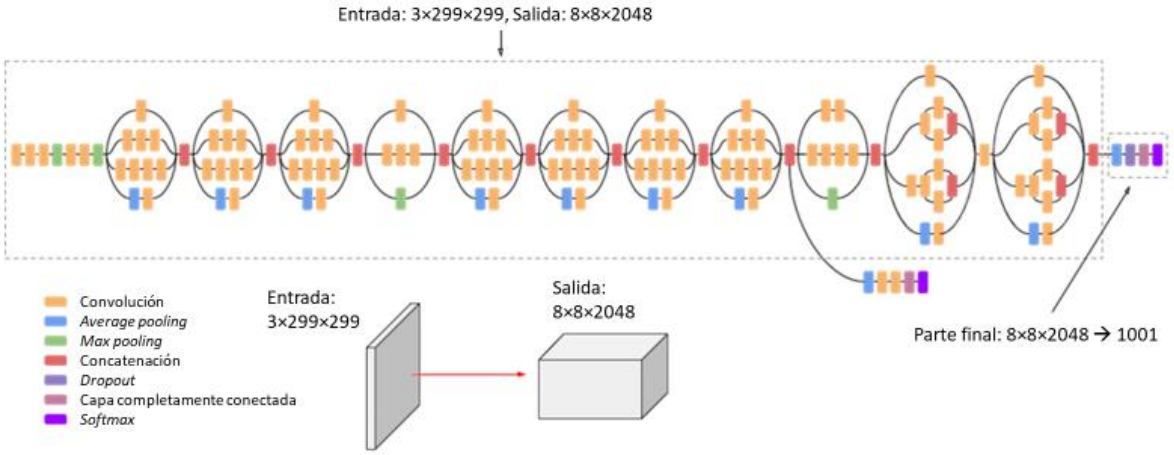


b) Bloque residual básico para modelos ResNet-50, ResNet-101 y ResNet-152

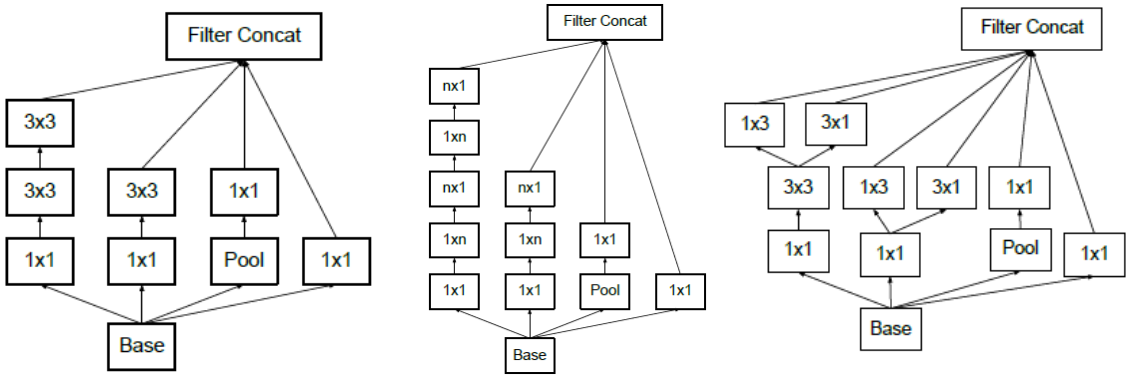
layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix} \times 2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix} \times 3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix} \times 2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix} \times 4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix} \times 8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix} \times 2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix} \times 6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix} \times 23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix} \times 36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix} \times 2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix} \times 3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

c) Arquitecturas de modelos basados en ResNet [37]

A4. Arquitectura de modelo InceptionV3

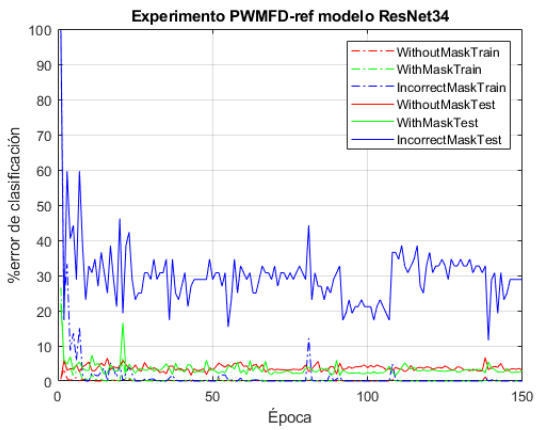


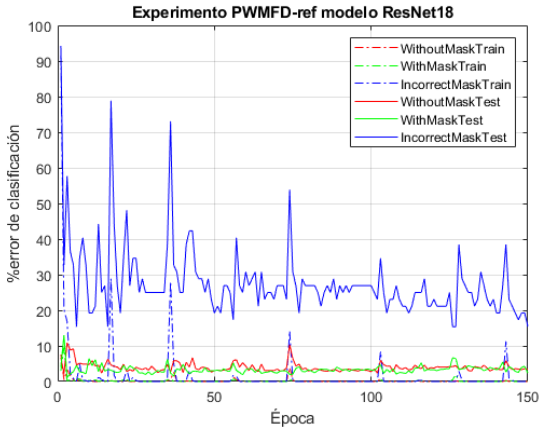
a) Estructura general de modelo InceptionV3 [72].

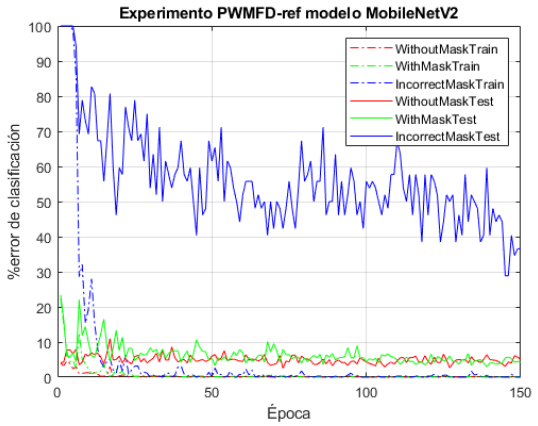


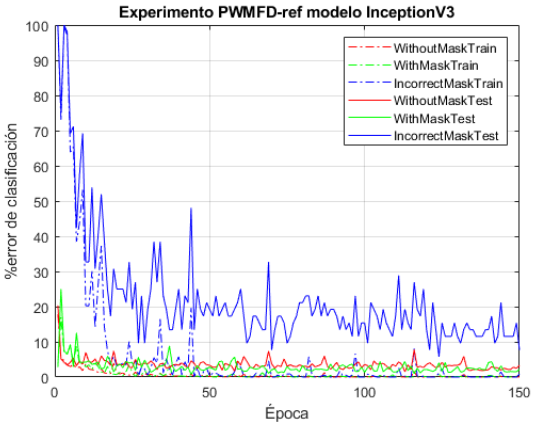
b) Estructuras de módulos *Inception* utilizados en modelo InceptionV3.

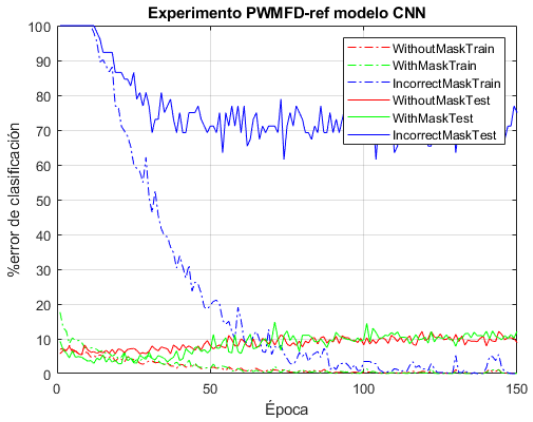
A5. Gráficas de error durante el entrenamiento de distintas arquitecturas para el clasificador de uso de cubrebocas.

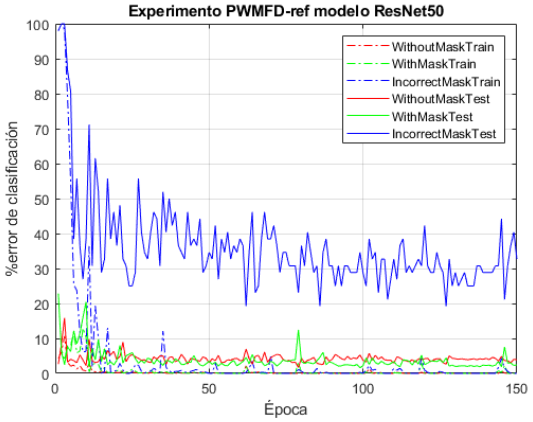


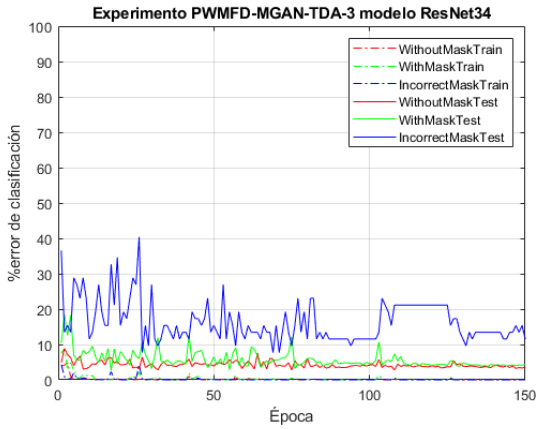


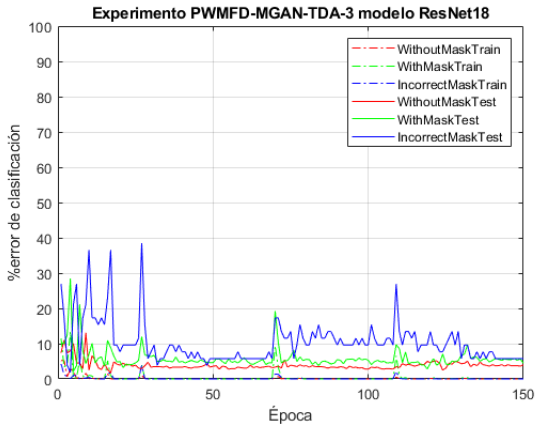


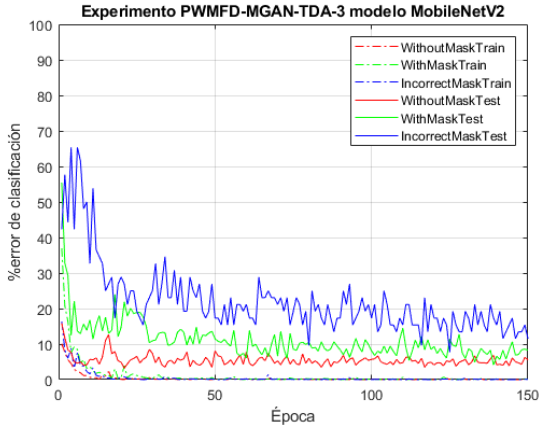


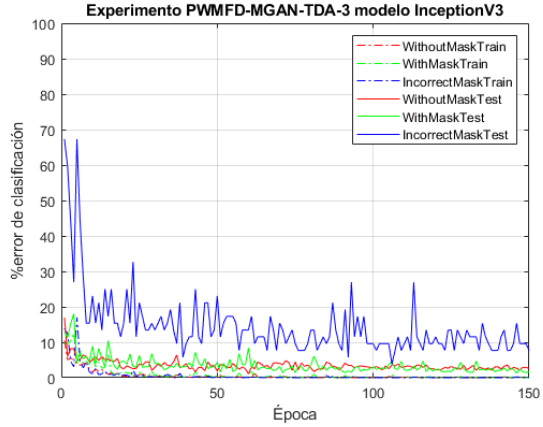


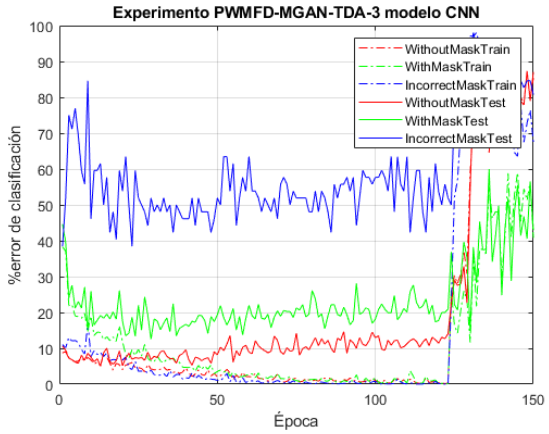


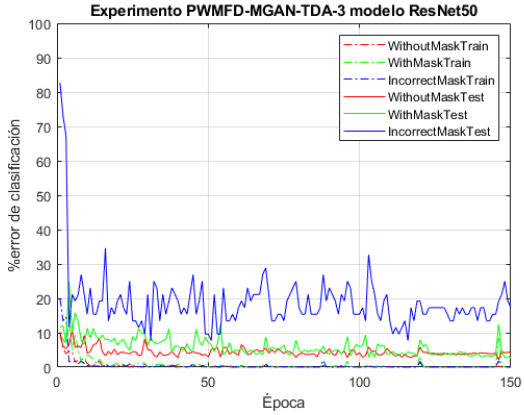

Experimento PWMFD-MGAN-TDA-3 modelo ResNet34
%error de clasificación
Época
WithoutMaskTrain
WithMaskTrain
IncorrectMaskTrain
WithoutMaskTest
WithMaskTest
IncorrectMaskTest


Experimento PWMFD-MGAN-TDA-3 modelo ResNet18
%error de clasificación
Época
WithoutMaskTrain
WithMaskTrain
IncorrectMaskTrain
WithoutMaskTest
WithMaskTest
IncorrectMaskTest

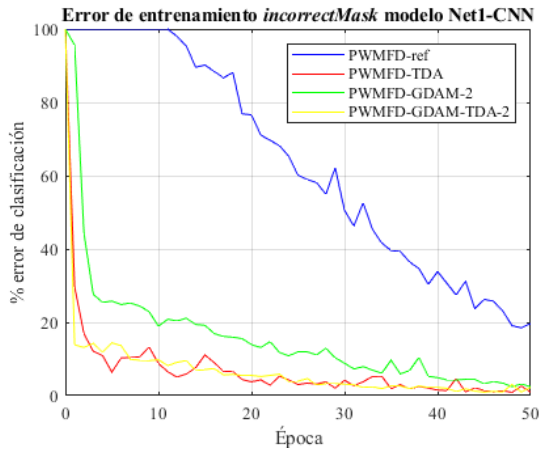

Experimento PWMFD-MGAN-TDA-3 modelo MobileNetV2
%error de clasificación
Época
WithoutMaskTrain
WithMaskTrain
IncorrectMaskTrain
WithoutMaskTest
WithMaskTest
IncorrectMaskTest

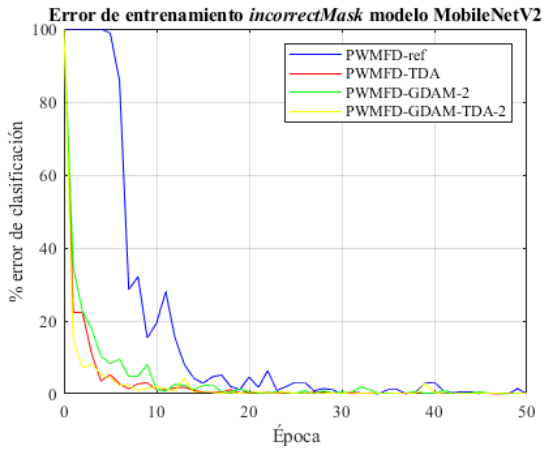

Experimento PWMFD-MGAN-TDA-3 modelo InceptionV3
%error de clasificación
Época
WithoutMaskTrain
WithMaskTrain
IncorrectMaskTrain
WithoutMaskTest
WithMaskTest
IncorrectMaskTest

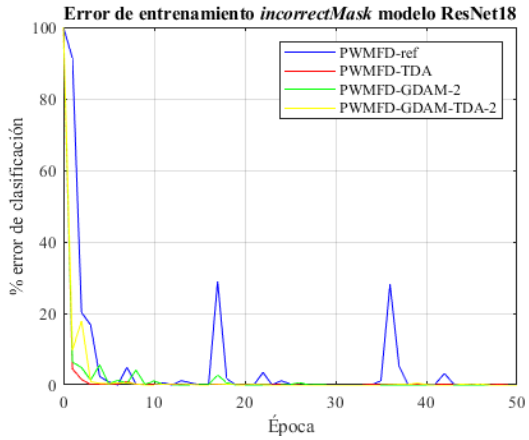

Experimento PWMFD-MGAN-TDA-3 modelo CNN
%error de clasificación
Época
WithoutMaskTrain
WithMaskTrain
IncorrectMaskTrain
WithoutMaskTest
WithMaskTest
IncorrectMaskTest

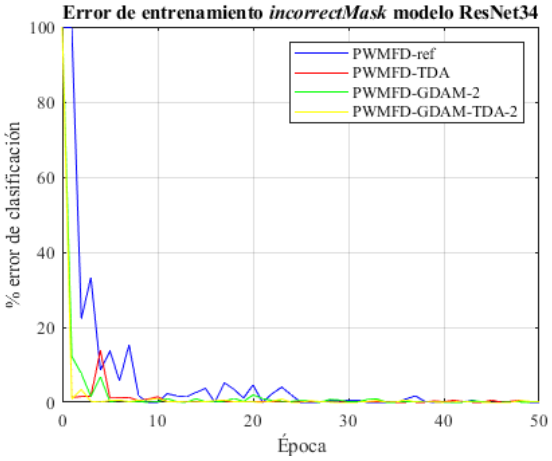

Experimento PWMFD-MGAN-TDA-3 modelo ResNet50
%error de clasificación
Época
WithoutMaskTrain
WithMaskTrain
IncorrectMaskTrain
WithoutMaskTest
WithMaskTest
IncorrectMaskTest

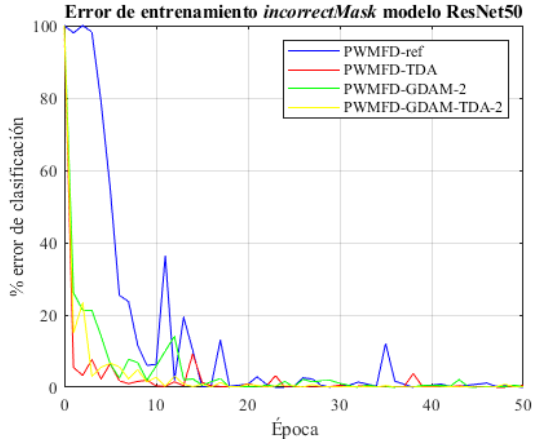
A6. Gráficas de error de clasificación de *incorrectMask* durante el entrenamiento de distintas arquitecturas para el clasificador de uso de cubrebocas.

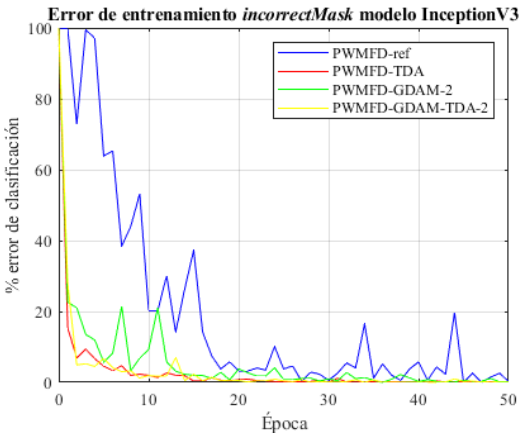




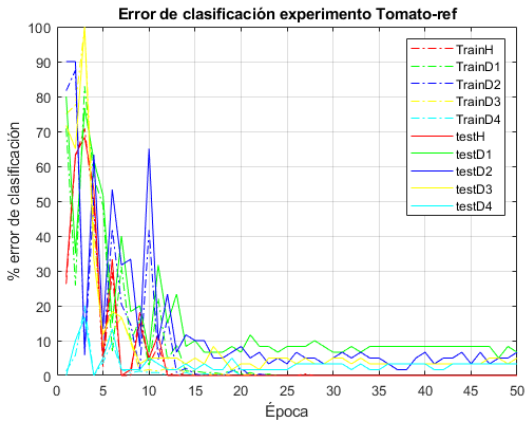


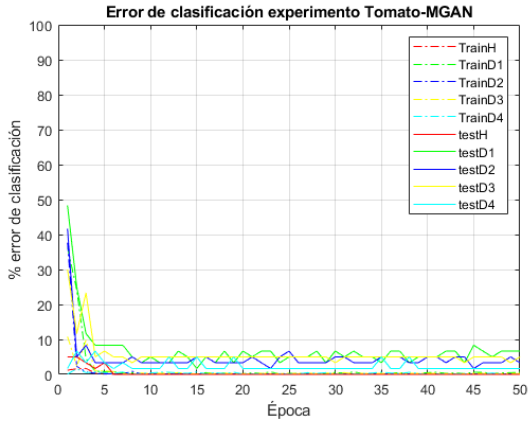


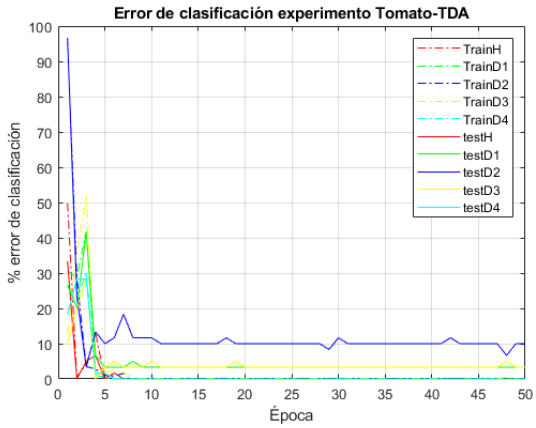


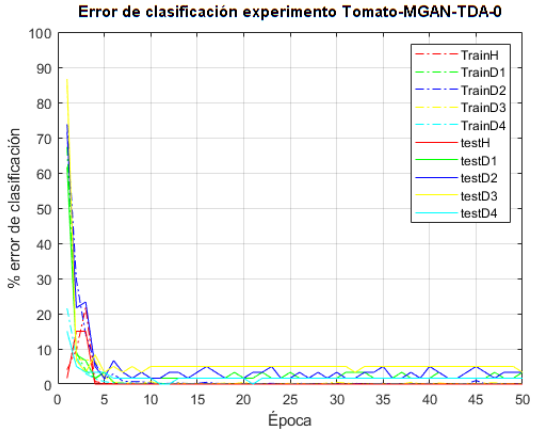


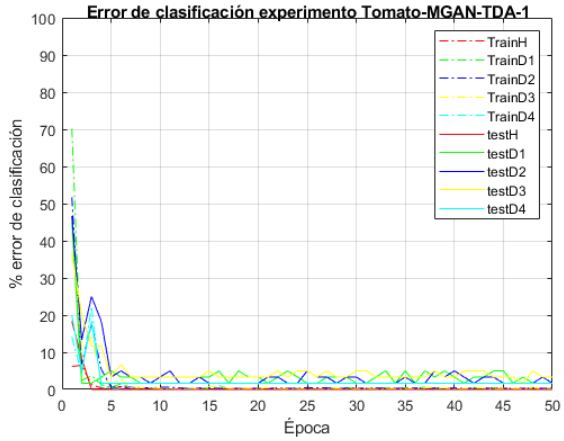
A7. Gráficas de error durante el entrenamiento en problema de clasificación de imágenes de hojas de tomate con alguna enfermedad





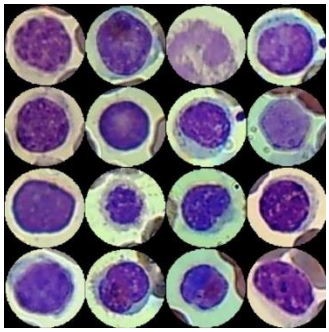




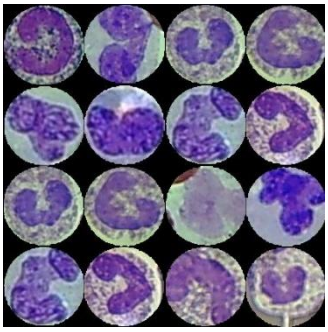


A8. Avances en implementación de GDAM-GAN para el problema de clasificación de imágenes de leucocitos

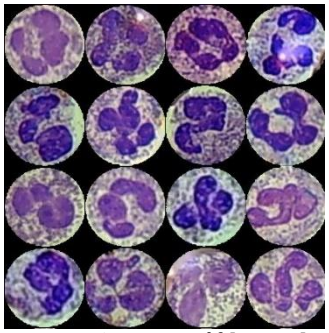
Se utilizan muestras de una base de datos privada con imágenes de leucocitos, los cuales se pueden dividir en cinco clases. Para los siguientes experimentos se toman en cuenta únicamente tres clases: *linfocitos* (149 muestras), *monocitos* (14 muestras), y *neutrófilos* (91 muestras):



a) Linfocitos reales



b) Monocitos reales

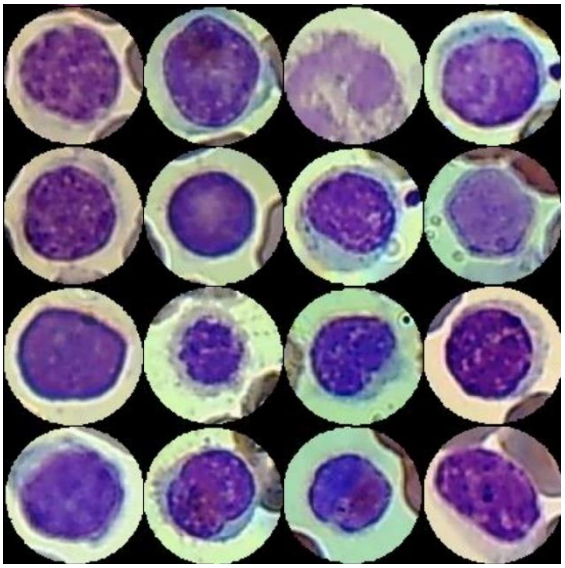


a) Neutrófilos reales

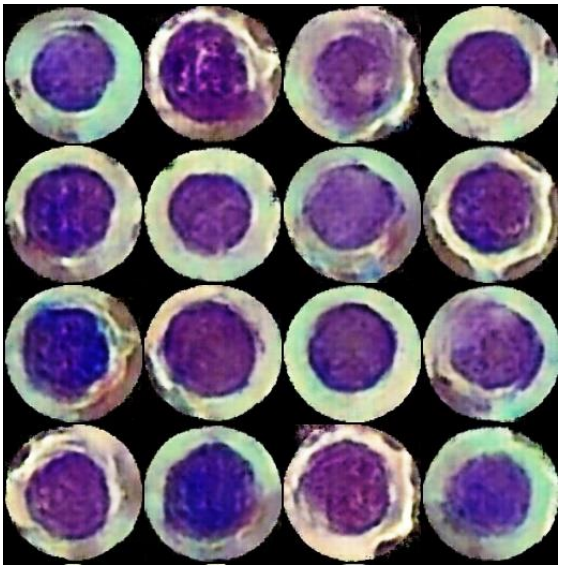
Se define como clase base l_S=*linfocitos*, ya que es aquella categoría de la cual se tienen más muestras, y se comparten algunas características con el resto de las clases. Y se definen dos clases deseadas l_{T1}=*monocitos* y l_{T2}=*neutrófilos*.

Se entrenan los elementos de GDAM-GAN conforme a la configuración establecida en el CAPÍTULO III de esta tesis.

Resultados de entrenamiento de modelo generativo base:



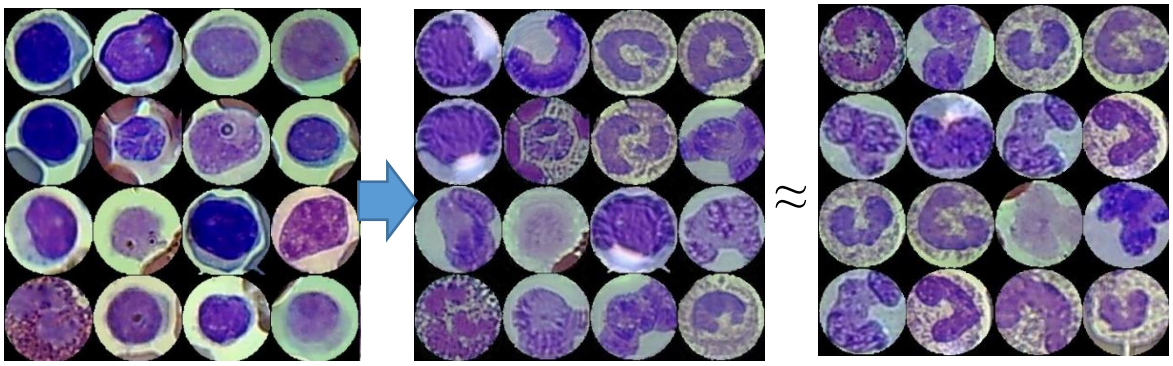
A) Imágenes reales de clase base *linfocitos*



B) Imágenes artificiales de clase base *linfocitos* (FID=156.9815)

Resultados de entrenamiento de modelo de traducción imagen-a-imagen:

Linfocitos-a-monocitos

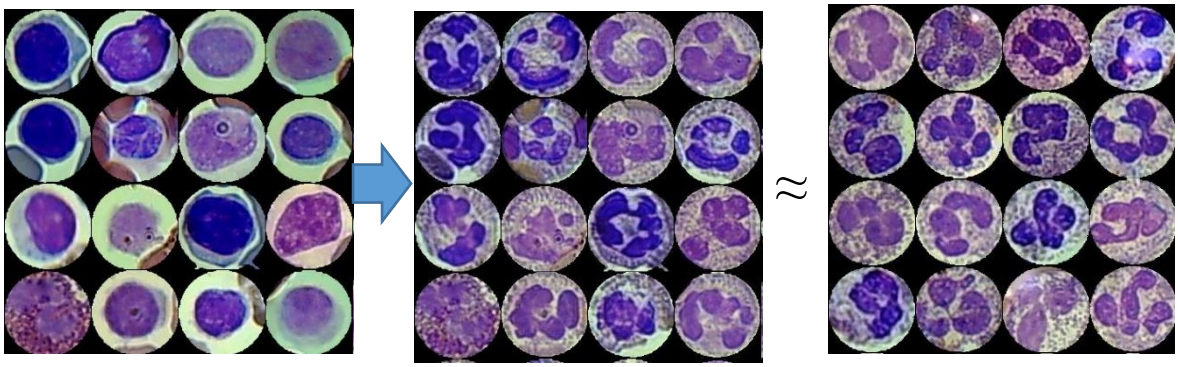


A) Imágenes reales de clase base *linfocitos*

B) Imágenes artificiales de clase deseada *monocitos* (FID=112.7164)

C) Imágenes reales de clase deseada *monocitos*

Linfocitos-a-neutrófilos



B) Imágenes reales de clase base *linfocitos*

B) Imágenes artificiales de clase deseada *neutrófilos* (FID=69.0746)

C) Imágenes reales de clase deseada *neutrófilos*

Se crearon los siguientes conjuntos de muestras artificiales:

- RL-to-M: a partir de imágenes reales de la clase *linfocitos* (RL), se generan muestras artificiales de la clase deseada *monocitos*. En total son 149 muestras en este conjunto.
- FL-to-M: a partir de imágenes artificiales de la clase *linfocitos* (FL), se generan muestras artificiales de la clase deseada *monocitos*. En total son 1024 muestras en este conjunto.

- RL-to-N: a partir de imágenes reales de la clase *linfocitos* (RL), se generan muestras artificiales de la clase deseada *neutrófilos*. En total son 149 muestras en este conjunto.
- FL-to-N: a partir de imágenes artificiales de la clase *linfocitos* (FL), se generan muestras artificiales de la clase deseada *neutrófilos*. En total son 1024 muestras en este conjunto.

Se aplicó el proceso de depuración descrito en la sección 2.4. En este caso, el algoritmo de agrupamiento no generó resultados favorables para seleccionar muestras útiles, ya que, en algunos casos, solamente se genera el grupo *ruido*. Y cuando se producen otros grupos, la mayoría de las muestras de tales grupos no pertenecen a la clase deseada (el criterio utilizado para determinar si las muestras pertenecen a la clase deseada fue una comparación visual con las muestras reales de tal clase).

A partir de estos experimentos, se determina que, es necesario realizar mejoras en la etapa de depuración del método de aumento de datos propuesto. No obstante, tal metodología tiene el potencial para generar imágenes artificiales, independientemente de la aplicación.