UNIVERSIDAD DE LA REPÚBLICA URUGUAY

FACULTAD DE INGENIERÍA

Ataques adversarios con algoritmos evolutivos y redes generativas antagónicas

Informe de Proyecto de Grado presentado por

Mateo González, Santiago Guridi, Santiago Parodi y Gonzalo Silva

en cumplimiento parcial de los requerimientos para la graduación de la carrera de Ingeniería en Computación de Facultad de Ingeniería de la Universidad de la República

Supervisor

Sergio Nesmachnow

Montevideo, 22 de octubre de 2024

Ataques adversarios con algoritmos evolutivos y redes generativas antagónicas por Mateo González, Santiago Guridi, Santiago Parodi y Gonzalo Silva tiene licencia CC Atribución 4.0.

Agradecimientos

Agradecemos a nuestro tutor, Sergio Nesmachnow, por su invaluable asistencia y apoyo durante el desarrollo del proyecto de grado.

Agradecemos a todos los profesores que nos han formado durante nuestra carrera, cuyas enseñanzas han inspirado nuestra pasión por el aprendizaje y el crecimiento profesional.

Agradecemos a Clara Ducher por proporcionarnos los límites de StyleGAN2-ADA-PyTorch de su artículo *Improved Attribute Manipulation in the Latent Space of StyleGAN for Semantic Face Editing*, los cuales fueron fundamentales para generar rostros de distintas razas.

Agradecemos a Jamal Toutouh por el conocimiento brindado en el área de este proyecto.

Agradecemos al equipo de ClusterUY por brindarnos acceso a la plataforma.

Finalmente, extendemos nuestra gratitud a nuestras familias y seres queridos por su apoyo constante y aliento incondicional.

Resumen

La clasificación de imágenes enfrenta actualmente desafíos significativos de seguridad debido a los ataques adversarios, que consisten en alteraciones intencionales diseñadas para engañar a los modelos de aprendizaje automático. Abordar los desafíos asociados a los ataques adversarios es fundamental para garantizar la fiabilidad y seguridad de los clasificadores de imágenes, especialmente en aplicaciones críticas como el reconocimiento facial, que es crucial en la autenticación de identidades y la vigilancia. Para que los clasificadores de imágenes permanezcan actualizados y puedan resistir efectivamente las técnicas de ataque emergentes, es esencial seguir desarrollando nuevos ataques adversarios. Este proyecto de grado presenta un enfoque innovador para generar ataques adversarios contra clasificadores de imágenes utilizando una combinación de algoritmos evolutivos y redes generativas antagónicas. El método realiza una exploración en el espacio latente de una red generativa antagónica con un algoritmo genético para encontrar vectores que representen ataques adversarios. El enfoque se evaluó en tres casos de estudio correspondientes a la clasificación de dígitos manuscritos, imágenes de objetos e imágenes de rostros. Los resultados mostraron tasas de éxito de hasta 35 % para dígitos manuscritos, hasta 75 % para imágenes de objetos y hasta 81,91 % para imágenes de rostros de raza blanca. El método demostró ser eficaz en manejar la diversidad de datos, incluso en la generación de ataques adversarios con imágenes de rostros, problema que presentó desafíos adicionales debido a la complejidad y riqueza de información en las imágenes faciales.

Lista de Tablas

5.1. Cantidad de ataques al clasificador c_m agrupados por dígito objetivo de ataque y función de fitness . . . 31
5.2. Número de instancias bien clasificadas en las que las dos probabilidades más altas proporcionadas por el clasificador c_m están a una distancia menor que δ agrupados por dígito objetivo de ataque y función de fitness . . . 31
5.3. Cantidad de ataques al dígito 9 según la función de fitness utilizada . . . 32
5.4. Cantidad de ataques al dígito 3 según la función de fitness utilizada . . . 32
5.5. Ejemplos de ataques al clasificador c_m con el número 9 . . . 33
5.6. Ejemplos de ataques al clasificador c_m con el número 3 . . . 33
5.7. Porcentaje de ataques al clasificador c_m que son efectivos para el clasificador c'_m agrupados por dígito objetivo de ataque y función de fitness . . . 33
5.8. Cantidad de ataques utilizando la función de fitness f_3 agrupados por dígito y clasificador . . . 34
5.9. Cantidad de ataques al clasificador c_c agrupados por clase y función de fitness en CIFAR-10 . . . 37
5.10. Número de instancias bien clasificadas en las que las dos probabilidades más altas proporcionadas por el clasificador c_c están a una distancia menor que δ agrupados por etiqueta objetivo de ataque y función de fitness . . . 37
5.11. Cantidad de ataques al objeto avión según la función de fitness utilizada . . . 38
5.12. Cantidad de ataques al objeto camión según la función de fitness utilizada . . . 39
5.13. Ejemplos de ataques al clasificador c_c con la clase avión . . . 39
5.14. Ejemplos de ataques al clasificador 1 con la clase camión . . . 39
5.15. Porcentaje de ataques al primer clasificador que son igual de efectivos para el segundo clasificador agrupados por dígito objetivo de ataque y función de fitness . . . 40
5.16. Cantidad de ataques al utilizar la función de fitness f_3 agrupados por dígito y clasificador . . . 40
5.17. Tiempo para Generación por Condición . . . 42
5.18. Número de rostros bien clasificados en las que las dos probabilidades más altas proporcionadas por DeepFace c_c están a una distancia menor que δ agrupados raza 52
5.19. Distribución de ataques de raza negra por umbral . . . 53
5.20. Distribución de ataques de raza blanca por umbral . . . 55
5.21. Distribución de ataques de raza asiática por umbral . . . 58

A.1. Ejemplos de ataques al clasificador c_m con el dígito 0 . . . 67
A.2. Ejemplos de ataques al clasificador c_m con el dígito 1 . . . 67
A.3. Ejemplos de ataques al clasificador c_m con el dígito 2 . . . 67
A.4. Ejemplos de ataques al clasificador c_m con el dígito 4 . . . 68
A.5. Ejemplos de ataques al clasificador c_m con el dígito 5 . . . 68
A.6. Ejemplos de ataques al clasificador c_m con el dígito 6 . . . 68
A.7. Ejemplos de ataques al clasificador c_m con el dígito 7 . . . 68
A.8. Ejemplos de ataques al clasificador c_m con el dígito 8 . . . 68

B.1. Ejemplos de ataques al clasificador c_c con la clase avión . . . 71

B.2. Ejemplos de ataques al clasificador c_c con la clase auto 71
B.3. Ejemplos de ataques al clasificador c_c con la clase ave 71
B.4. Ejemplos de ataques al clasificador c_c con la clase gato 72
B.5. Ejemplos de ataques al clasificador c_c con la clase ciervo 72
B.6. Ejemplos de ataques al clasificador c_c con la clase perro 72
B.7. Ejemplos de ataques al clasificador c_c con la clase rana 72
B.8. Ejemplos de ataques al clasificador c_c con la clase caballo 72
B.9. Ejemplos de ataques al clasificador c_c con la clase barco 72
B.10. Ejemplos de ataques al clasificador c_c con la clase camión 73

Lista de Figuras

2.1. Esquema de entrenamiento de una GAN. El Generador transforma el ruido aleatorio en una imagen sintética imitando al dataset de entrenamiento para engañar al discriminador. El discriminador intenta distinguir las imágenes reales de las sintéticas creadas por el generador. . . . 6
2.2. Codificación de soluciones en un algoritmo genético. . . . 9

4.1. Diagrama del método utilizado para generar ejemplos adversarios . . . 20
4.2. Imágenes del conjunto de datos MNIST. . . . 21
4.3. Imágenes del conjunto de datos CIFAR-10. . . . 24
4.4. Imágenes del conjunto de datos FFHQ . . . 25

5.1. Evolución de los valores de fitness según las funciones f_1 y f_2 en MNIST . . . 30
5.2. Representaciones t-SNE de los Ataques al dígito 3 (Atq 3) según la función de fitness utilizada junto con los dígitos que generaron confusión . . . 35
5.3. Evolución de los valores de fitness según las funciones f_1 y f_2 en CIFAR-10 . . . 36
5.4. Representaciones t-SNE de los Ataques al objeto avión (Atq avi) según la función de fitness utilizada junto con los objetos que generaron confusión . . . 41
5.5. Comparación rostro de población inicial con rostro resultado del algoritmo luego de 20 generaciones. . . . 42
5.6. Ejemplo de rostro generado InterfaceGAN. . . . 43
5.7. Uso de distintos multiplicadores en el generador . . . 44
5.8. Ejemplo de rostro asiático generado utilizando los límites de raza negra. . . . 45
5.9. Ejemplo de rostro de raza blanca generado utilizando los límites de raza negra. . . . 46
5.10. Comparación de rostros generados con anomalías y sin anomalías . . . 47
5.11. Ejemplo de rostro asiático generado utilizando los límites de raza blanca. . . . 48
5.12. Rostros de raza distinta a blanca generados utilizando los límites de raza blanca . 49
5.13. Colores de Stone . . . 49
5.14. Ejemplo de rostro de raza blanca generado utilizando los límites de raza asiática. . 50
5.15. Ejemplo de rostro de raza negra generado utilizando los límites de raza asiática. . . 50
5.16. Evolución de las funciones de fitness de raza negra, blanca y asiática en Stylegan2-ADA-Pytorch . . . 52
5.17. Ataques a raza negra (N) por raza clasificada (A: Asiática, B: Blanca, LH: Latino hispano, MO: Medio oriente, I: India) . . . 54
5.18. Representación TSNE de la distribución de los ataques a la raza negra . . . 55
5.19. Ataques a raza blanca (B) por raza clasificada (A: Asiática, N: Negra, LH: Latino hispano, MO: Medio oriente, I: India) . . . 56
5.20. Representación tsne de la distribución de los ataques a la raza blanca. . . . 57
5.21. Ataques a raza asiática (A) por raza clasificada (B: Blanca, N: Negra, LH: Latino hispano, MO: Medio oriente, I: India) . . . 58
5.22. Representación TSNE de la distribución de los ataques a la raza asiática. . . . 59

A.1. Representaciones t-SNE de los Ataques al dígito 9 (Atq 9) según la función de fitness f_1 . . . 69

A.2. Representaciones t-SNE de los Ataques al dígito 9 (Atq 9) según la función de fitness f_2 69

B.1. Representaciones t-SNE de los Ataques al objeto camión (Atq cam) según la función de fitness f_1 junto con los objetos que generaron confusión Función de fitness . . . 73
B.2. Representaciones t-SNE de los Ataques al objeto camión (Atq cam) según la función de fitness f_2 junto con los objetos que generaron confusión Función de fitness . . . 74
B.3. Representaciones t-SNE de los Ataques al objeto camión (Atq cam) según la función de fitness f_3 junto con los objetos que generaron confusión Función de fitness . . . 74

C.1. Rostros de raza negra sin anomalías utilizados para entrenar un clasificador basado en el modelo VGG16 76
C.2. Rostros de raza negra con anomalías utilizados para entrenar un clasificador basado en el modelo VGG16 77
C.3. Rostros de raza negra generados utilizando límite de raza negra 78
C.4. Rostros de otra raza generados utilizando límite de raza negra 79
C.5. Rostros de raza asiática generados utilizando límite de raza asiática 80
C.6. Rostros de otras razas generados utilizando límite de raza asiática 81
C.7. Rostros de raza asiática con gafas generados utilizando límite de raza asiática . . . 82
C.8. Rostros de raza asiática sin gafas generados utilizando límite de raza asiática . . . 83
C.9. Rostros de raza negra con objetos en la cabeza generados utilizando límite de raza negra 84
C.10. Rostros de raza negra sin objetos en la cabeza generados utilizando límite de raza negra 85
C.11. Rostros de raza blanca de edad avanzada generados utilizando límite de raza blanca 86
C.12. Rostros de raza blanca jóvenes generados utilizando límite de raza blanca 87
C.13. Evoluciones sin suavizar de la función de fitness para las razas blanca, asiática y negra. 88
C.14. Ataques a raza negra (N) por raza clasificada y la clasificación con mayor probabilidad (A: Asiática, B: Blanca, LH: Latino hispano, MO: Medio oriente, I: India) . . 89
C.15. Ataques a raza blanca (B) por raza clasificada y la clasificación con mayor probabilidad (A: Asiática, N: Negra, LH: Latino hispano, MO: Medio oriente, I: India) . . 90
C.16. Ataques a raza asiática (A) por raza clasificada y la clasificación con mayor probabilidad (N: Negra, B: Blanca, LH: Latino hispano, MO: Medio oriente, I: India) . . 91

Índice general

1. Introducción **1**

2. Fundamentos teóricos **3**
2.1. Aprendizaje automático . . . 3
2.1.1. Clasificación de modelos de aprendizaje automático . . . 3
2.1.1.1. Aprendizaje supervisado . . . 3
2.1.1.2. Aprendizaje no supervisado . . . 3
2.1.2. Regresión logística . . . 4
2.1.2.1. Función de regresión logística . . . 4
2.1.2.2. Aplicaciones y casos de uso de la regresión logística . . . 4
2.1.3. Redes neuronales . . . 4
2.1.3.1. Estado del arte de las redes neuronales . . . 4
2.1.3.2. Arquitectura de redes neuronales . . . 5
2.1.3.3. Modelos clasificadores . . . 5
2.1.3.4. Redes generativas antagónicas . . . 5
2.1.3.5. Redes generativas antagónicas profundas convolucionales (DCGANs) 7
2.1.3.6. Redes generativas antagónicas profundas convolucionales condicionales (C-DCGANs) . . . 7
2.2. Ataques adversarios . . . 8
2.2.1. Definición de ataque adversario . . . 8
2.2.2. Clasificación de ataques adversarios . . . 8
2.2.3. Seguridad contra ataques adversarios . . . 8
2.3. Algoritmos Genéticos . . . 8
2.3.1. Generación de población inicial . . . 9
2.3.2. Representación de soluciones . . . 9
2.3.3. Operadores de selección . . . 10
2.3.4. Operadores de cruzamiento . . . 10
2.3.5. Operadores de mutación. . . . 10

3. Revisión de antecedentes **13**
3.1. Antecedentes de algoritmos genéticos en ataques adversarios . . . 13
3.1.1. POBA-GA: Perturbation Optimized Black-Box Adversarial Attacks via Genetic Algorithm . . . 13
3.1.2. Genetic Algorithm with Multiple Fitness Functions for Generating Adversarial Examples . . . 14
3.1.3. Generating Adversarial Examples through Latent Space Exploration of Generative Adversarial Networks . . . 14
3.1.4. GenAttack: Practical Black-box Attacks with Gradient-Free Optimization . 15
3.1.5. Practical Black-Box Attacks against Machine Learning . . . 15
3.2. Otros antecedentes relevantes . . . 16
3.2.1. The Space of Transferable Adversarial Examples . . . 16

3.2.2. Improved Attribute Manipulation in the Latent Space of StyleGAN for Semantic Face Editing . . . 16
3.2.3. The Harms of Demographic Bias in Deep Face Recognition Research . . . 17
3.2.4. Studying Bias in GANs through the Lens of Race . . . 17

4. Generación de ejemplos adversarios **19**
4.1. Metodología . . . 19
4.1.1. Estrategia de desarrollo . . . 19
4.1.2. Descripción del método utilizado para generar ejemplos adversarios . . . 19
4.1.3. Requerimientos del método para generar ataques adversarios . . . 20
4.2. Ataques a clasificadores de dígitos manuscritos . . . 20
4.2.1. Conjunto de datos MNIST . . . 20
4.2.2. Selección de clasificadores y de la GAN para MNIST . . . 21
4.2.3. Implementación para generar ataques para MNIST . . . 22
4.2.3.1. Algoritmo genético con PyGAD . . . 22
4.2.3.2. Funciones de fitness . . . 22
4.3. Ataques a clasificadores de imágenes de objetos . . . 23
4.3.1. Conjunto de datos CIFAR-10 . . . 23
4.3.2. Selección de clasificadores y de la GAN para CIFAR-10 . . . 23
4.3.3. Implementación para generar ataques para CIFAR-10 . . . 24
4.4. Ataques a clasificadores de imágenes de rostros . . . 24
4.4.1. Conjunto de datos FFHQ . . . 25
4.4.2. Selección del clasificador y de la GAN para FFHQ . . . 25
4.4.3. Implementación del método para generar ataques para FFHQ . . . 26
4.4.3.1. Funciones de fitness . . . 26
4.4.3.2. Clasificador basado en VGG16 y regresión logística . . . 28
4.4.3.3. Filtro de color basado en Stone . . . 28

5. Experimentación **29**
5.1. Entorno de ejecución . . . 29
5.2. Análisis de ataques a clasificadores de dígitos manuscritos . . . 29
5.2.1. Evaluación de la GAN para MNIST . . . 29
5.2.2. Evaluación del método para generar ataques adversarios para MNIST . . . 30
5.3. Análisis de ataques a clasificadores de imágenes de objetos . . . 35
5.3.1. Evaluación de la GAN para CIFAR-10 . . . 35
5.3.2. Evaluación del método para generar ataques adversarios para CIFAR-10 . . . 36
5.4. Análisis de ataques a clasificadores de imágenes de rostros . . . 41
5.4.1. Evaluación de StyleGAN . . . 41
5.4.1.1. Evaluación de StyleGAN sin extensiones . . . 42
5.4.1.2. Evaluación de InterfaceGAN . . . 43
5.4.1.3. Evaluación de la herramienta basada en StyleGAN 3 . . . 43
5.4.1.4. Evaluación de la herramienta basada en StyleGAN2-ADA-Pytorch 44
5.4.2. Optimización del método para generar ataques con StyleGAN . . . 44
5.4.2.1. Optimización del método para generar ataques de raza negra . . . 45
5.4.2.2. Optimización del método para generar ataques de raza blanca . . . 48
5.4.2.3. Optimización del método para generar ataques de raza asiática . . . 49
5.4.3. Evaluación del método para generar ataques adversarios con StyleGAN 2 . . . 51
5.4.3.1. Evaluación del método para generar ataques adversarios de raza negra en StyleGAN 2 . . . 53
5.4.3.2. Evaluación del método para generar ataques adversarios de raza blanca en StyleGAN 2 . . . 55
5.4.3.3. Evaluación del método para generar ataques adversarios de raza asiática en StyleGAN 2 . . . 57

6. Conclusiones y trabajo futuro **61**
6.1. Conclusiones . . . 61
6.2. Trabajo futuro . . . 62

A. Dígitos manuscritos **67**
A.1. Ejemplos de ataques adversarios con dígitos manuscritos . . . 67
A.2. Representaciones t-SNE de ataques adversarios para MNIST . . . 69

B. Imágenes de objetos **71**
B.1. Ejemplos de ataques adversarios con imágenes de objetos . . . 71
B.2. Representación t-SNE de ataques adversarios para CIFAR-10 . . . 73

C. Imágenes de rostros **75**
C.1. Datos de entrenamiento para rostros de raza negra con anomalías . . . 76
C.1.1. Rostros de raza negra sin anomalías . . . 76
C.1.2. Rostros de raza negra con anomalías . . . 77
C.2. Datos de entrenamiento para modelo clasificador de rostros de raza negra . . . 78
C.2.1. Rostros de raza negra generados utilizando límite de raza negra . . . 78
C.2.2. Rostros de otras razas generados utilizando límite de raza negra . . . 79
C.3. Datos de entrenamiento para modelo clasificador de rostros de raza asiática . . . 80
C.3.1. Rostros de raza asiática generados utilizando límite de raza asiática . . . 80
C.3.2. Rostros de otras razas generados utilizando límite de raza asiática . . . 81
C.4. Datos de entrenamiento para modelo clasificador de rostros con gafas . . . 82
C.4.1. Rostros de raza asiática con gafas generados utilizando límite de raza asiática 82
C.4.2. Rostros de raza asiática sin gafas generados utilizando límite de raza asiática 83
C.5. Datos de entrenamiento para modelo clasificador de rostros con objetos en la cabeza 84
C.5.1. Rostros de raza negra con objetos en la cabeza generados utilizando límite de raza negra . . . 84
C.5.2. Rostros de raza negra sin objetos en la cabeza generados utilizando límite de raza negra . . . 85
C.6. Datos de entrenamiento para modelo clasificador de rostros de edad avanzada . . . 86
C.6.1. Rostros de raza blanca de edad avanzada generados utilizando límite de raza blanca . . . 86
C.6.2. Rostros de raza blanca jóvenes generados utilizando límite de raza blanca . 87
C.7. Evolución de las funciones de fitness para las razas asiática, negra y blanca. . . . 88
C.8. Ejemplos de ataques adversarios de raza negra . . . 89
C.9. Ejemplos de ataques adversarios de raza blanca . . . 90
C.10. Ejemplos de ataques adversarios de raza asiática . . . 91

Capítulo 1

Introducción

En el campo del aprendizaje automático, la clasificación de imágenes ha emergido como una de las aplicaciones más relevantes y con gran impacto en diversas industrias, desde la seguridad hasta el marketing personalizado (Szeliski, 2011). En particular, el reconocimiento facial ha revolucionado significativamente los sistemas de identificación y verificación de identidades al permitir una autenticación precisa mediante el estudio detallado de características faciales únicas. Sin embargo, la robustez de los métodos de clasificación de imágenes frente a ataques adversarios es una preocupación creciente (Vakhshiteh, Nickabadi, and Ramachandra, 2021).

Los ataques adversarios son alteraciones sutiles e intencionales en los datos de entrada de un modelo de aprendizaje automático, con el propósito de engañar al sistema y obtener respuestas equivocadas para poner en riesgo la fiabilidad y precisión del modelo (Goodfellow, Shlens, and Szegedy, 2015). Los ataques adversarios pueden presentarse como imágenes con perturbaciones que inducen a los clasificadores de imágenes a identificar incorrectamente un objeto o una identidad, lo que puede tener implicaciones serias en la seguridad y privacidad de sistemas críticos, por ejemplo en sistemas de vigilancia y seguridad pública. La investigación continua en generar nuevos ataques adversarios es esencial para que los sistemas de clasificación de imágenes puedan mantenerse actualizados y resistentes frente a técnicas de ataque emergentes y se garantice su seguridad, precisión y fiabilidad en entornos cambiantes (Chakraborty, Alam, Dey, Chattopadhyay, and Mukhopadhyay, 2021).

En este proyecto de grado se propuso un enfoque para la generación de ataques adversarios contra clasificadores de imágenes que utiliza una combinación de algoritmos evolutivos y redes generativas antagónicas (GANs). El objetivo general del proyecto fue realizar una búsqueda sobre el espacio latente de una GAN mediante el uso de un algoritmo genético con el fin de encontrar vectores adecuados para generar imágenes que representen ataques adversarios contra ciertos clasificadores de imágenes. Específicamente, los casos de estudio analizados comprendieron la generación de ataques a clasificadores de dígitos manuscritos, imágenes de objetos e imágenes de rostros. El enfoque fue diseñado para generar ejemplos adversarios efectivos, diversos y de alta calidad, con el fin de evaluar la robustez de los clasificadores en distintas variantes del problema.

Los resultados obtenidos en el proyecto mostraron un buen desempeño en la generación de ataques adversarios contra clasificadores de imágenes, con diferentes tasas de éxito en cada problema. Se realizaron ataques exitosos a todos los dígitos manuscritos contra dos clasificadores diferentes y la tasa de éxito más alta observada fue de 35 %. Se realizaron ataques exitosos a todas las imágenes de objetos de baja resolución del conjunto de datos utilizado contra dos clasificadores diferentes y la tasa de éxito más alta registrada fue de 75 %. Finalmente, en las imágenes de rostros, la generación de ataques adversarios resultó ser considerablemente más compleja debido a la mayor complejidad y riqueza de información en las imágenes faciales, lo que requirió el entrenamiento de modelos adicionales para obtener ataques de calidad. A pesar de los desafíos presentados, se lograron producir ataques efectivos contra el sistema Deepface (Serengil and Ozpinar, 2021) con tasas de éxito de 81,91 % para imágenes de rostros de raza blanca, 23,6 % para imágenes de rostros de raza negra y 70,96 % para imágenes de rostros de raza asiática. Los resultados obtenidos mos-

traron la capacidad del método para manejar la diversidad en los datos. Se obtuvieron ataques con características distintivas, por ejemplo accesorios en imágenes de rostros de raza negra o lentes en imágenes de rostros de raza asiática.

El proyecto de grado ha contribuido al campo del aprendizaje automático al presentar un enfoque innovador para generar nuevos ataques adversarios contra distintos modelos de clasificación de imágenes generados utilizando una combinación de algoritmos evolutivos y redes generativas antagónicas.

A continuación se explica la estructura del documento. En el Capítulo 2 se abordan los fundamentos teóricos necesarios para comprender los conceptos clave utilizados en el proyecto de grado. En el Capítulo 3 se presenta una revisión de antecedentes de algoritmos genéticos en ataques adversarios y otros antecedentes relevantes para el proyecto de grado. En el Capítulo 4 se presenta el método desarrollado para la generación de ejemplos adversarios en cada caso de estudio. En el Capítulo 5 se presentan los resultados experimentales y su análisis. Finalmente, en el Capítulo 6 se exponen las conclusiones y se proponen posibles líneas de trabajo futuro.

Capítulo 2

Fundamentos teóricos

En este capítulo se presentan conceptos esenciales que son cruciales para la comprensión de las problemáticas abordadas en nuestro proyecto de grado.

2.1. Aprendizaje automático

El aprendizaje automático (Michie, Spiegelhalter, and Taylor, 1999) es una rama de la inteligencia artificial que se enfoca en el desarrollo de algoritmos y modelos capaces de aprender y tomar decisiones a partir de grandes cantidades de datos. En esta sección se presentan diferentes modelos de aprendizaje automático y su clasificación según su función y características.

2.1.1. Clasificación de modelos de aprendizaje automático

En esta sección se presenta la clasificación de los modelos de aprendizaje automático según su tipo de aprendizaje. Las principales categorías son aprendizaje supervisado y no supervisado.

2.1.1.1. Aprendizaje supervisado

El aprendizaje supervisado consiste en la deducción de información a partir de datos de entrenamiento en los que cada individuo de entrenamiento tiene asociada una etiqueta o resultado deseado. Durante el proceso de aprendizaje, el modelo ajusta sus parámetros para minimizar el error en la predicción de las etiquetas a partir de las características de los datos. Este tipo de aprendizaje es adecuado para tareas como la clasificación y la regresión. Ejemplos comunes de aprendizaje supervisado incluyen las Redes neuronales y la Regresión logística.

2.1.1.2. Aprendizaje no supervisado

El aprendizaje no supervisado se aplica a conjuntos de datos sin etiquetas. El objetivo de este tipo de aprendizaje es identificar patrones o estructuras ocultas en los datos. Los algoritmos de aprendizaje no supervisado buscan agrupar los datos en grupos o reducir la dimensionalidad del conjunto de datos. Este tipo de aprendizaje es útil en exploración de datos, detección de anomalías y reducción de dimensionalidad. Ejemplos comunes de modelos de aprendizaje no supervisado son Clustering y Principal Component Analysis.

2.1.2. Regresión logística

La regresión logística es una técnica de aprendizaje supervisado utilizada principalmente para problemas de clasificación binaria. La regresión logística utiliza un modelo matemático para predecir la probabilidad de ocurrencia de un evento y clasificar las observaciones en una de dos categorías posibles. Esta sección aborda los conceptos clave de la regresión logística y presenta diversas aplicaciones y casos de uso que ilustran su utilidad en el campo del aprendizaje automático.

2.1.2.1. Función de regresión logística

La regresión logística se basa en la función sigmoide, que transforma cualquier valor real a un rango entre 0 y 1. Dada una entrada x, el modelo de regresión logística predice una probabilidad $P(y = 1|x)$ utilizando la función logística definida en la Ecuación 2.1.

$$P(y = 1|x) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x)}} \tag{2.1}$$

donde β_0 y β_1 son los coeficientes que el modelo aprende durante el proceso de entrenamiento. Los coeficientes se ajustan para minimizar una función de pérdida, que mide la discrepancia entre las probabilidades predichas y las etiquetas reales. La función de pérdida más utilizada es la entropía cruzada.

2.1.2.2. Aplicaciones y casos de uso de la regresión logística

La regresión logística se utiliza ampliamente en tareas de clasificación binaria, tales como la detección de fraude en transacciones financieras, la predicción de enfermedades en estudios médicos y la clasificación de correos electrónicos como spam o no spam. La simplicidad, interpretabilidad y eficacia de la regresión logística la convierten en una herramienta popular en muchas aplicaciones prácticas en el campo del aprendizaje automático.

2.1.3. Redes neuronales

Las redes neuronales son modelos computacionales que emulan ciertas funciones del cerebro humano, específicamente su capacidad para reconocer patrones y aprender de los datos de entrenamiento. Con el paso del tiempo, las redes neuronales han avanzado desde configuraciones básicas hacia arquitecturas altamente sofisticadas, permitiendo la resolución de problemas complejos en el campo del aprendizaje automático.

2.1.3.1. Estado del arte de las redes neuronales

El estado del arte en redes neuronales para la clasificación y generación de imágenes ha sido dominado por avances en arquitecturas como las Redes Neuronales Convolucionales (CNNs) y las GANs. Las CNNs, por ejemplo AlexNet, VGG, ResNet y EfficientNet, son la arquitectura estándar para la clasificación de imágenes ya que aprovechan su estructura espacial para extraer características locales y clasificar objetos con alta precisión. Por otro lado, las GANs, por ejemplo StyleGAN y BigGAN, han revolucionado la generación de imágenes realistas al utilizar un sistema de competencia entre un generador y un discriminador. El enfoque actual en redes neuronales para la clasificación y generación de imágenes se centra en mejorar la eficiencia, la robustez y la capacidad de generalización a nuevas tareas con menos datos etiquetados y abordar desafíos como el aprendizaje con pocos ejemplos, la interpretación de modelos y la reducción de sesgos en los datos.

2.1.3.2. Arquitectura de redes neuronales

La arquitectura de una red neuronal se compone de capas de nodos o neuronas conectadas entre sí. Cada capa tiene un conjunto de neuronas que aplican una transformación a los datos de entrada y luego pasan la información a la siguiente capa. Las redes neuronales suelen estar organizadas en capa de entrada, capas ocultas y capa de salida. La capa de entrada recibe los datos iniciales. Las capas ocultas realizan la transformación y extracción de características a través de funciones de activación como ReLU (Rectified Linear Unit), sigmoide o tanh. Entre las capas ocultas, un tipo destacado es la capa convolucional, que se utiliza en las CNNs. Las capas convolucionales aplican filtros a la entrada para extraer características locales, por ejemplo bordes y texturas en imágenes. La capa de salida genera la predicción final, que puede ser una clasificación, regresión u otra salida deseada.

Cada conexión entre las neuronas tiene un peso asociado que se ajusta durante el proceso de entrenamiento para minimizar el error proporcionado por la red. Además, la red puede incluir términos de sesgo que ayudan a mejorar la precisión del modelo. Las redes neuronales profundas, que tienen muchas capas ocultas, han demostrado ser particularmente efectivas en la resolución de problemas complejos que antes eran inabordables.

2.1.3.3. Modelos clasificadores

Los modelos clasificadores son algoritmos de aprendizaje automático que se especializan en diferenciar entre distintas categorías de datos y se enfocan en identificar y clasificar datos en función de sus características. Los modelos clasificadores se pueden expresar como $c : \mathbb{X} \to \mathbb{Y}$, siendo $\mathbb{X}$ un conjunto de datos y $\mathbb{Y}$ el conjunto de categorías. Ejemplos de modelos clasificadores son Regresión logística, Máquinas de Soporte Vectorial (SVM), Bosques Aleatorios (Random Forests) y K-Vecinos Más Cercanos (K-NN). Los modelos clasificadores se emplean en una amplia gama de aplicaciones, como detección de spam en correos electrónicos y la identificación de enfermedades en imágenes médicas.

2.1.3.4. Redes generativas antagónicas

Una GAN es un tipo de sistema de aprendizaje automático ideado por Ian Goodfellow. Una GAN es un modelo generativo que está formado por dos redes neuronales, una red llamada generador y otra red llamada discriminador, que compiten entre sí en un marco de juego de suma cero. Ambas redes neuronales desempeñan roles cruciales y se complementan mutuamente durante el entrenamiento.

El generador es responsable de crear instancias de datos artificiales con el objetivo de engañar al discriminador, utilizando los resultados proporcionados por el discriminador para mejorar la calidad de los datos generados. Durante el entrenamiento, el generador sigue un proceso iterativo que incluye:

- Generación de entrada aleatoria: Se genera una entrada aleatoria, que se utiliza como base para crear una instancia de datos artificial.
- Clasificación por el discriminador: Los datos generados se presentan al discriminador, que determina si son reales o artificiales.
- Retroalimentación al generador: Se proporciona retroalimentación al generador basada en la clasificación del discriminador para ajustar sus parámetros y mejorar la calidad de los datos generados.
- Iteración del proceso de entrenamiento: Este ciclo de generación y retroalimentación se repite varias veces hasta que el generador alcanza un nivel de rendimiento deseado.

El discriminador es responsable de distinguir entre datos reales y datos generados por el generador. Su proceso de entrenamiento implica:

- Etiquetado de datos: Se asignan etiquetas a las instancias de datos reales y generadas para indicar su autenticidad.
- Clasificación de datos: Los datos reales y los generados se introducen en el discriminador, que los clasifica determinando si son reales o generados.
- Retroalimentación al discriminador: Se utiliza el resultado de la clasificación para proporcionar retroalimentación al discriminador, corrigiendo su capacidad para distinguir entre datos reales y generados.
- Iteración del proceso de entrenamiento: Este proceso de clasificación y retroalimentación se repite hasta que el discriminador logra una precisión deseada en la clasificación de datos.

El generador mejora su capacidad para engañar al discriminador a medida que avanza en su entrenamiento. Simultáneamente, el discriminador se perfecciona en su habilidad para detectar falsificaciones, lo que impulsa una competencia continua entre ambas redes neuronales. Este proceso conduce a la creación de datos notablemente realistas utilizados en una amplia variedad de aplicaciones, como la generación de imágenes a partir de descripciones, la conversión entre dominios de imágenes y la mejora de la resolución de imágenes. El proceso de entrenamiento de una GAN se visualiza en la Figura 2.1.

El proceso de generación en una GAN se lleva a cabo en un espacio latente, que es un espacio de dimensiones inferiores al espacio de datos reales y es donde se encuentran los vectores de entrada aleatorios que el generador utiliza para crear nuevas instancias de datos. Los vectores latentes actúan como puntos de partida que se transforman a través de las capas del generador para producir datos artificiales. La estructura del espacio latente es compleja y no tiene una interpretación clara, lo que hace difícil predecir qué tipo de datos serán generados a partir de un punto específico en el espacio latente.

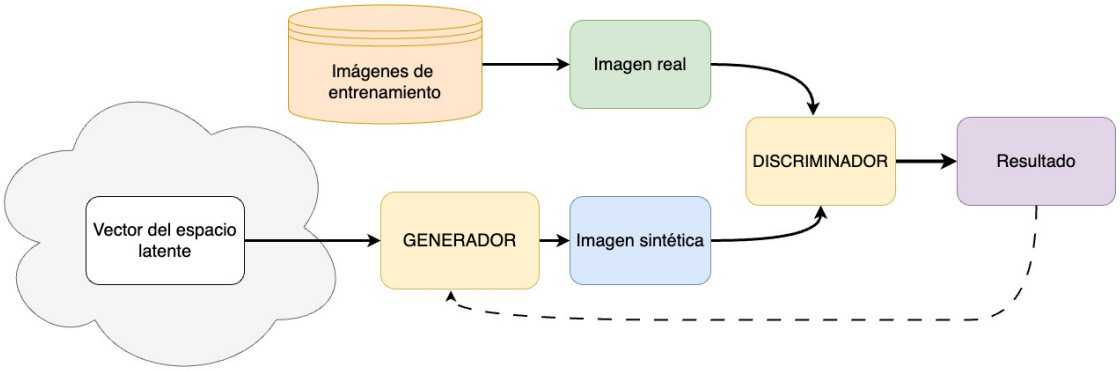


Figura 2.1: Esquema de entrenamiento de una GAN. El Generador transforma el ruido aleatorio en una imagen sintética imitando al dataset de entrenamiento para engañar al discriminador. El discriminador intenta distinguir las imágenes reales de las sintéticas creadas por el generador.

Las GANs han demostrado su versatilidad a través de una amplia gama de aplicaciones en diversas industrias. Las GANs han sido utilizadas con éxito para generar imágenes realistas de personas, paisajes y objetos, incluso a partir de descripciones textuales en el campo de la visión por computadora. La generación de este tipo de imágenes ha encontrado aplicación en la producción de contenido digital, diseño de videojuegos y creación de efectos visuales en cine y televisión. Además, las GANs han revolucionado el campo del diseño de moda al permitir la creación automática de prendas de vestir con estilos únicos y personalizados. En medicina, las GANs se emplean para la generación de imágenes médicas sintéticas que facilitan la investigación y el entrenamiento de modelos de diagnóstico por imagen.

2.1.3.5. Redes generativas antagónicas profundas convolucionales (DCGANs)

Una Red Generativa Antagónica Profunda Convolucional (conocida como Deep Convolutional Generative Adversarial Network en inglés o DCGAN) es una variante avanzada de la GAN que ha demostrado ser una herramienta poderosa en el campo de la generación de imágenes y la síntesis de datos visuales. La característica distintiva de las DCGANs radica en su arquitectura profundamente convolucional, que incorpora capas convolucionales tanto en el generador como en el discriminador.

La arquitectura convolucional, frecuentemente conocida en general como Red Neuronal Convolucional (CNN), proporciona a las DCGANs varias ventajas significativas. En primer lugar, la arquitectura convolucional permite una representación más eficiente de las imágenes y captura patrones locales y globales de manera más efectiva, lo que conduce a una generación de imágenes más realistas y detalladas. Además, las capas convolucionales ayudan a estabilizar el proceso de entrenamiento al mitigar la inestabilidad en la convergencia, lo que mejora la calidad y la consistencia de los resultados generados. Otra ventaja clave de las DCGANs es su capacidad para aprender características jerárquicas de las imágenes durante el proceso de entrenamiento, ya que la red puede aprender automáticamente a representar y generar objetos complejos a partir de características simples. Esta característica facilita la generación de imágenes de alta calidad y la transferencia de estilos entre diferentes dominios visuales.

2.1.3.6. Redes generativas antagónicas profundas convolucionales condicionales (C-DCGANs)

Una variante de las DCGANs son las Redes Generativas Antagónicas Profundas Convolucionales Condicionales (conocidas como Conditional Deep Convolutional Generative Adversarial Networks en inglés o C-DCGAN). La característica distintiva de las C-DCGANs es su capacidad de condicionamiento, ya que generan imágenes específicas según las pautas o restricciones proporcionadas. Esta capacidad se logra mediante la introducción de información adicional, por ejemplo etiquetas de clase o atributos, que guían el proceso de generación. Las C-DCGANs se basan en la arquitectura de las DCGANs y también emplean capas convolucionales profundas para aprender las características esenciales de las imágenes y generar contenido visual altamente realista.

Las C-DCGANs destacan por el control sobre los resultados generados y su notable flexibilidad en la generación de imágenes condicionadas que no se limita únicamente a la inclusión de etiquetas de clase, sino que también abarca otros tipos de información adicional, por ejemplo atributos específicos o incluso características deseadas en las imágenes resultantes. Esta capacidad permite adaptar el proceso de generación según las necesidades específicas de cada aplicación, ya que las C-DCGANs pueden ser utilizadas para una amplia variedad de tareas que requieren imágenes personalizadas o ajustadas a ciertos criterios predefinidos. Las C-DCGANs ofrecen una solución flexible y adaptable que se ajusta a diversos problemas, por ejemplo la generación de imágenes de rostros con diferentes expresiones emocionales, paisajes con variaciones específicas de iluminación o prendas de vestir con diferentes estilos y colores.

Las C-DCGANs tienen una amplia gama de aplicaciones específicas que aprovechan su capacidad de generar imágenes condicionadas. Una de las áreas destacadas es la personalización de productos en el comercio electrónico, ya que las C-DCGANs pueden generar imágenes de productos adaptadas a las preferencias individuales de los usuarios, por ejemplo variaciones de color, diseño

o estilo. Además, en el ámbito del entretenimiento y la publicidad, estas redes pueden utilizarse para crear contenido visual personalizado, como carteles de películas con diferentes actores principales o anuncios de productos con características específicas destacadas. En el campo de la moda y el diseño, las C-DCGANs son útiles para la generación de prendas de vestir personalizadas con ajustes específicos de tamaño, color o estilo según las preferencias del cliente.

2.2. Ataques adversarios

Un ataque adversario (Vakhshiteh, Nickabadi, and Ramachandra, 2021). es el resultado de la manipulación de los datos de entrada de un modelo de aprendizaje automático con la intención de hacer que realice clasificaciones erróneas. Las manipulaciones suelen ser imperceptibles para el ojo humano. Los ataques se realizan de diversas formas, por ejemplo agregar perturbaciones a imágenes, alterar datos de entrada textual o manipular datos de sensores en sistemas físicos. Los ataques adversarios representan desafíos significativos para la seguridad y la confiabilidad de los sistemas de aprendizaje automático, especialmente en aplicaciones donde la robustez y la precisión son críticas.

2.2.1. Definición de ataque adversario

Sea $I = \{i_0, \ldots, i_n\}$ un conjunto de imágenes y $E = \{e_0, \ldots, e_m\}$ un conjunto de categorías tal que para todo $i \in I$ existe un $e \in E$ asociado. Sea $i' \in I$ asociado a la categoría $e' \in E$ y $c : I \to E$ un clasificador que asigna categorías a imágenes. Se define i' como un ataque adversario si y solo si $c(i') = e''$ con $e'' \neq e'$.

2.2.2. Clasificación de ataques adversarios

Los ataques adversarios se clasifican como ataques de caja blanca, de caja negra, de objetivo y de no objetivo. En los ataques de caja blanca el atacante tiene un conocimiento completo del modelo objetivo, incluida su arquitectura, parámetros y datos de entrenamiento. En los ataques de caja negra el atacante no tiene conocimiento de la estructura o parámetros del modelo objetivo. El atacante puede explotar técnicas que involucren consultar al modelo objetivo. En los ataques de objetivo el atacante engaña al modelo objetivo para que la clasificación sea una categoría en particular. En los ataques no objetivo el atacante engaña al modelo objetivo, pero para ninguna categoría en particular.

2.2.3. Seguridad contra ataques adversarios

Los ataques adversarios representan un desafío activo y crítico en el ámbito de la seguridad del aprendizaje automático. La investigación actual está centrada en el desarrollo de clasificadores más resilientes contra ataques adversarios. Para abordar las vulnerabilidades y limitaciones de los modelos de aprendizaje automático se están explorando diversas formas de llevar a cabo nuevos ataques adversarios para comprender a fondo sus mecanismos y desafíos. Los principales aportes incluyen el desarrollo de técnicas de detección de anomalías y la implementación de mecanismos de defensa en los clasificadores, con el objetivo de garantizar la integridad y la confiabilidad de los sistemas de aprendizaje automático en entornos críticos donde la seguridad de los datos es prioritaria.

2.3. Algoritmos Genéticos

La computación evolutiva es un conjunto de técnicas heurísticas y estocásticas que emulan el proceso de evolución natural basándose en los principios de selección natural, reproducción y diversidad genética para resolver problemas de optimización, búsqueda y aprendizaje. Las técnicas de computación evolutiva se dividen en programación genética, estrategias de evolución y algoritmos

genéticos. Los algoritmos genéticos (Mitchell, 1996) imitan la evolución biológica, comenzando con una población de soluciones posibles que se actualizan iterativamente. Los individuos más aptos de la población son seleccionados para cruzarse y mutarse, formando nuevas generaciones. Tras varias iteraciones, la población converge a una solución óptima. En esta sección se detalla el funcionamiento de los algoritmos genéticos y sus componentes principales.

2.3.1. Generación de población inicial

La población inicial es un conjunto de individuos que representan las posibles soluciones del problema. La generación de la población inicial se realiza generalmente de forma aleatoria, lo que garantiza que se abarque un amplio rango del espacio de búsqueda y permite tener una diversidad genética que ayuda a evitar la convergencia prematura hacia soluciones locales subóptimas. El tamaño de la población inicial varía según el problema, pero generalmente incluye desde cientos hasta miles de posibles soluciones. La distribución de la probabilidad de muestreo puede ser ajustada de acuerdo con alguna heurística particular del problema para concentrarse en áreas de mayor interés y asegurar una representación más efectiva de las posibles soluciones.

2.3.2. Representación de soluciones

Los algoritmos genéticos no trabajan sobre el espacio de las soluciones directamente sino que lo hacen sobre una representación de las soluciones. Las representaciones de las soluciones forman el espacio de búsqueda del algoritmo genético. La representación más utilizada es la representación binaria, pero se pueden utilizar otras codificaciones ya que la elección depende de las características del problema a resolver. La Figura 2.2 muestra la relación entre el espacio de las soluciones y el espacio de búsqueda.

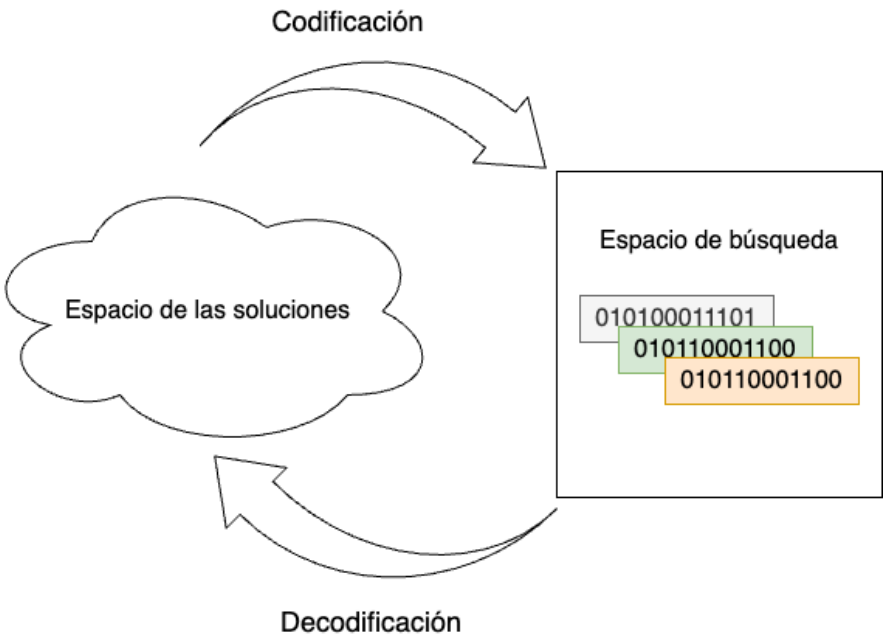


Figura 2.2: Codificación de soluciones en un algoritmo genético.

2.3.3. Operadores de selección

La selección es el proceso mediante el cual se selecciona a los individuos más aptos de una generación. El objetivo de la selección es mantener las características de los individuos más aptos a través de las generaciones. Para evaluar la aptitud de los individuos y seleccionar los más aptos los algoritmos genéticos utilizan una función de fitness. La función de fitness es una medida de ganancia que se quiere maximizar según el problema dado. El operador de selección guía la exploración del algoritmo hacia secciones más prometedoras del espacio de búsqueda. Las técnicas de selección pueden ser deterministas, probabilísticas o mixtas.

Selección por ruleta. En esta técnica la probabilidad de elegir un individuo para la reproducción de la próxima generación es proporcional a su valor de fitness.

Selección por torneo. En esta técnica se seleccionan los individuos con mayor valor de fitness de un subconjunto aleatorio de individuos de la población.

Selección por clasificación. En esta técnica los individuos son ordenados según su valor de fitness y son seleccionados aquellos con mayor fitness.

2.3.4. Operadores de cruzamiento

El cruzamiento es el proceso mediante el cual se combina la información genética de dos individuos progenitores para generar descendencia. El cruzamiento es un proceso estocástico que crea nuevas soluciones a partir de una población existente. En los algoritmos genéticos el operador de cruzamiento es el operador fundamental y es el responsable de la explotación del espacio de búsqueda, que se enfoca en perfeccionar las soluciones en áreas del espacio de búsqueda ya identificadas como prometedoras.

Cruzamiento de un punto. En esta técnica se selecciona un punto aleatorio en los padres. Los bits situados a la derecha de ese punto se intercambian entre los cromosomas de los dos padres, lo que produce dos descendientes con la combinación de información genética de ambos padres. Esta técnica se puede extender con el uso de k puntos.

Cruzamiento uniforme. En esta técnica cada bit es seleccionado aleatoriamente de un padre u otro. En ciertos problemas se le da más importancia a uno de los padres, lo que permite que los descendientes hereden más características de un padre que del otro.

2.3.5. Operadores de mutación.

La mutación es el proceso mediante el cual se introduce diversidad genética. La mutación realiza pequeñas modificaciones a algunos individuos con una cierta probabilidad del orden de 0,001 para cada bit en una representación binaria. Si bien el operador de mutación se trata de un operador secundario en comparación con el de cruzamiento, ayuda a prevenir la convergencia prematura de las soluciones hacia óptimos locales y es el responsable de la exploración del espacio de búsqueda. La exploración busca nuevas áreas del espacio de búsqueda que aún no se han investigado y permite descubrir posibles soluciones mejores que no son evidentes en regiones conocidas.

Mutación de cambio de bit. En esta técnica se selecciona un bit aleatorio de un cromosoma y se invierte su valor. Por ejemplo, si un bit en la posición i tiene un valor de 0, se cambia a 1, y viceversa. Esta técnica introduce diversidad al alterar directamente la información genética de los individuos.

Mutación de intercambio (swap mutation). En esta técnica se seleccionan dos posiciones aleatorias en un cromosoma y se intercambian sus valores. Este operador es útil para problemas de permutación, por ejemplo el problema del viajante (García, Hontoria, and Aleksovski, 2015), en el cual el orden de los elementos es importante.

Mutación de inserción (insertion mutation). En esta técnica se selecciona un gen aleatorio de un cromosoma y se inserta en una posición diferente dentro del mismo cromosoma. Los otros genes se desplazan para acomodar la inserción. Esta técnica también es aplicable a problemas de permutación.

Mutación de desplazamiento (shift mutation). En esta técnica se selecciona un segmento de genes en un cromosoma y se mueve a una nueva posición, desplazando los demás genes para acomodar el cambio. Este operador es similar a la mutación de inserción, pero afecta un segmento de genes en lugar de un solo gen.

Mutación gaussiana. Este operador se utiliza comúnmente en representaciones de valores reales. La técnica agrega un pequeño valor aleatorio, tomado de una distribución gaussiana (normal), a un gen seleccionado aleatoriamente. Este tipo de mutación es útil para problemas donde los genes representan parámetros continuos.

Mutación uniforme. En esta técnica se seleccionan aleatoriamente valores dentro de un rango permitido y se asignan a genes seleccionados. Este tipo de mutación es aplicable a representaciones de valores reales, donde los genes pueden tomar cualquier valor dentro de un intervalo específico.

Mutación de inversión (inversion mutation). Este operador invierte el orden de un segmento seleccionado aleatoriamente en un cromosoma. Es particularmente útil en problemas de orden, ya que mantiene la composición de los genes, pero cambia su orden.

Mutación de permutación (scramble mutation). En esta técnica se selecciona un segmento de genes dentro de un cromosoma y se realiza una permutación aleatoria dentro de ese segmento, lo que mantiene el contenido del segmento, pero cambia su orden de manera aleatoria.

Capítulo 3

Revisión de antecedentes

Este capítulo presenta una revisión de antecedentes relevantes para el desarrollo del proyecto de grado.

3.1. Antecedentes de algoritmos genéticos en ataques adversarios

En esta sección se presentan trabajos de distintos autores que exploran el uso de algoritmos genéticos para la generación de ataques adversarios contra clasificadores de imágenes.

3.1.1. POBA-GA: Perturbation Optimized Black-Box Adversarial Attacks via Genetic Algorithm

En el artículo *Perturbation Optimized Black-box Attacks Against Deep Neural Networks Using Genetic Algorithm* (Su, Chen, Shen, Xiong, and Zheng, 2019) se presentó un nuevo método de ataque antagónico contra redes neuronales basado en un algoritmo genético llamado *Perturbation Optimized Black-Box Attacks (POBA-GA)*. Este método superó a los métodos de ataque de caja negra contemporáneos en términos de tasa de éxito de ataque y control de perturbaciones. La motivación detrás de este trabajo fue la necesidad de desarrollar métodos de ataque adversarios efectivos y prácticos para evaluar y mejorar la robustez de los modelos de aprendizaje en aplicaciones del mundo real, como sistemas de reconocimiento facial.

Se utilizó un algoritmo genético debido a su capacidad para optimizar la generación de ejemplos adversarios, lo que permitió una mayor eficacia en la búsqueda de perturbaciones óptimas. El algoritmo se implementó siguiendo un enfoque estándar de algoritmos genéticos. El algoritmo parte de una población de soluciones candidatas donde cada solución representa una perturbación potencial a una imagen. Luego se evalúa la aptitud de cada individuo y se aplican operadores de selección, cruce y mutación hasta alcanzar la condición de terminación y conseguir una solución aproximadamente óptima para generar ejemplos adversarios de alta calidad.

Se utilizó la base de datos *Labeled Faces in the Wild* (Huang, Mattar, Lee, and Learned-Miller, 2012) para evaluar el método POBA-GA. POBA-GA logró una tasa de éxito de ataque del 96 % al realizar 1 000 consultas, cada una con 50 generaciones y 20 ejemplos por generación. Este resultado fue superior al resultado obtenido al evaluar los métodos ZOO (Chen, Zhang, Sharma, Yi, and Hsieh, 2017) y AutoZOOM (Tu, Ting, Chen, Liu, Zhang, Yi, Hsieh, and Cheng, 2019), lo que demostró la eficacia del método y su potencial utilidad en aplicaciones prácticas del mundo real. La perturbación generada por POBA-GA fue significativamente menor que la producida por otros métodos, lo que hace que el método sea de gran utilidad en entornos con restricciones de tiempo y recursos.

En este artículo se presentó una técnica interesante sobre la generación de ataques adversarios mediante el uso de algoritmos genéticos. No obstante, la metodología de POBA-GA para la generación de ejemplos adversarios difirió de la propuesta en este proyecto de grado, ya que Chen y cols. emplearon un algoritmo evolutivo para generar perturbaciones en lugar de explorar el espacio latente de una GAN.

3.1.2. Genetic Algorithm with Multiple Fitness Functions for Generating Adversarial Examples

En el artículo *Genetic Algorithm with Multiple Fitness Functions for Generating Adversarial Examples* (Wu, Luo, Zhou, Xu, and Zhu, 2021) se presentó un algoritmo genético llamado MF-GA con múltiples funciones de fitness para generar ejemplos adversarios y atacar redes neuronales profundas. La motivación detrás de este trabajo fue explorar y proponer un algoritmo genético con múltiples funciones de fitness para generar ejemplos adversarios, atacar redes neuronales profundas y comparar la efectividad de este enfoque con métodos tradicionales en conjuntos de datos populares como MNIST, FMNIST y CIFAR-10.

El algoritmo MF-GA se basa en el uso de múltiples funciones de fitness para guiar la evolución de la población hacia la generación de ejemplos adversarios efectivos. MF-GA se divide en tres etapas y utiliza diferentes funciones de fitness en cada una para evitar caer en óptimos locales. Este algoritmo incluye procesos como selección, cruce y mutación, y divide el proceso de optimización en etapas de exploración, explotación y estabilidad. Las funciones de fitness fueron diseñadas para reducir la confianza en la categoría correcta, aumentar la confianza en la categoría incorrecta, maximizar la diferencia de confianza entre categorías y maximizar la confianza en la categoría menos probable.

Se utilizaron los conjuntos de datos MNIST, FMNIST y CIFAR-10 para evaluar MF-GA, y se comparó el rendimiento del algoritmo con los métodos de referencia FGSM (Goodfellow, Shlens, and Szegedy, 2015), BIM (Maia, Mêda, and Freitas, 2015), MI-FGSM (Liu, Peng, Zhou, Wu, Zhang, and Zhang, 2021) y SPSA (J. Uesato and B. O'Donoghue and P. Kohli and A. Van den Oord , 2018)), así como un algoritmo genético básico. Los resultados mostraron que MF-GA logró un rendimiento competitivo en entornos de caja negra y caja blanca. En particular, en escenarios de caja negra, MF-GA superó a los métodos de referencia, por lo que demostró su eficacia en la generación de ejemplos adversarios. MF-GA destacó por su capacidad para evadir la detección y causar una clasificación errónea en redes neuronales profundas.

Este artículo presentó un algoritmo genético para generar ejemplos adversarios. No obstante, la metodología para la generación de ejemplos adversarios difirió de la propuesta en el proyecto de grado, ya que el MF-GA se utilizó un algoritmo evolutivo para generar perturbaciones en lugar de explorar el espacio latente de una GAN. Además, MF-GA no fue evaluado sobre conjuntos de datos más complejos, como los de rostros humanos.

3.1.3. Generating Adversarial Examples through Latent Space Exploration of Generative Adversarial Networks

En el artículo *Generating Adversarial Examples through Latent Space Exploration of Generative Adversarial Networks* (Clare and Correia, 2023) se presentó un método llamado GLASSE para generar ataques adversarios de caja negra contra un clasificador de imágenes.

GLASSE consta de una etapa de exploración del espacio latente de una GAN mediante el uso de un algoritmo evolutivo con el objetivo de encontrar potenciales ejemplos adversarios y otra etapa de selección de ataques. El algoritmo parte de una población inicial de vectores latentes que son evaluados por una función de fitness para generar una población de posibles ejemplos adversarios. La función de fitness mide la cercanía de estos individuos a la distribución del conjunto de datos original utilizando un discriminador que evalúa las imágenes generadas a partir de los vectores latentes. En la segunda etapa del método, los posibles ejemplos adversarios son evaluados contra un clasificador de imágenes para definir si son ataques definitivos o no.

Se realizaron 15 ejecuciones independientes del método para generar ataques propuesto con 40 generaciones y una población inicial de 100 individuos sobre el conjunto de datos MNIST. Se tomaron todos los potenciales ejemplos adversarios de cada generación y se clasificaron con dos modelos. Los resultados mostraron que de los potenciales ejemplos adversarios, más de un 82 % fueron ataques para ambos modelos.

El método propuesto para generar los ataques adversarios ha servido como fundamento para la técnica desarrollada en el proyecto de grado. GLASSE, a diferencia de otros enfoques, no incorporó un clasificador en la función de fitness, lo que aumentó la robustez del método al no ser dirigido específicamente contra ningún clasificador en particular. Este método exploró el uso del discriminador como una medida alternativa al FID (Fréchet Inception Distance) para verificar la similitud de distribución entre las imágenes generadas y el conjunto de datos original. No obstante, GLASSE no fue evaluado sobre conjuntos de datos más complejos, como los de rostros humanos.

3.1.4. GenAttack: Practical Black-box Attacks with Gradient-Free Optimization

En el artículo *GenAttack: Practical Black-box Attacks with Gradient-Free Optimization* (Alzantot, Sharma, Chakraborty, Zhang, Hsieh, and Srivastava, 2019) se presentó una técnica de optimización sin gradientes llamada GenAttack que utilizó algoritmos genéticos para generar ejemplos adversarios contra clasificadores de imágenes. La motivación detrás de este trabajo fue explorar la vulnerabilidad de las redes neuronales profundas a los ejemplos adversarios en entornos de caja negra.

GenAttack utiliza una estrategia de optimización basada en poblaciones, ya que evoluciona iterativamente una población de soluciones hasta lograr la generación exitosa de ejemplos adversarios. La evolución de la población se obtuvo al ejecutar un algoritmo genético que tiene una función de fitness que minimiza la probabilidad de la clase objetivo y la probabilidad de otras clases para elegir a los mejores individuos.

Se utilizaron los conjuntos de datos MNIST, CIFAR-10 e ImageNet para evaluar la eficacia de GenAttack al generar ataques contra modelos de reconocimiento de imágenes. Los experimentos se centraron en demostrar la capacidad de GenAttack para generar ejemplos adversarios visualmente imperceptibles con un número significativamente menor de consultas en comparación con métodos anteriores. En comparación con el método ZOO (Chen, Zhang, Sharma, Yi, and Hsieh, 2017), GenAttack logró una tasa de éxito del 95 % con un promedio de 21 966 consultas, en tanto que ZOO solo alcanzó un 6 % de éxito con 3 584 623 consultas al realizar ataques contra el modelo Ens4AdvInceptionV3 de ImageNet. GenAttack mostró una tasa de éxito del 100 % con un promedio de 11 081 consultas al realizar ataques contra el modelo InceptionV3 de ImageNet. GenAttack fue aproximadamente 237 veces más eficiente que ZOO y 9 veces más eficiente que el enfoque GA baseline contra ImageNet. Los resultados mostraron la capacidad de GenAttack para generar ejemplos adversarios de manera efectiva y eficiente en entornos de caja negra y la resistencia del método a las defensas que manipulan o enmascaran los gradientes, ya que no depende de ellos.

Este artículo presentó una técnica interesante sobre la generación de ataques adversarios mediante el uso de algoritmos genéticos. Sin embargo, la metodología de GenAttack para la generación de ejemplos adversarios difiere de la propuesta en el proyecto de grado, ya que Alzantot y cols. emplearon un algoritmo evolutivo para generar perturbaciones en lugar de explorar el espacio latente de una GAN.

3.1.5. Practical Black-Box Attacks against Machine Learning

En el artículo *Practical Black-Box Attacks against Machine Learning* (Papernot, McDaniel, Goodfellow, Jha, Celik, and Swami, 2017) se presentó un enfoque novedoso en la creación de ataques adversarios contra modelos de aprendizaje automático. El método propuesto destacó por generar ejemplos adversarios transferibles a otros clasificadores de imágenes y construir un conjunto de datos sintético para lograr este objetivo.

El método propuesto se basó en entrenar un modelo llamado modelo sustituto con datos sintéticos generados por el adversario y etiquetados por el modelo objetivo, lo que le permitió aprender a generar ejemplos adversarios efectivos. La generación de datos sintéticos se realizó mediante una técnica de aumento de datos basada en la matriz Jacobiana del modelo sustituto que consiste en identificar direcciones en las que la salida del modelo varía alrededor de un conjunto inicial de puntos de entrenamiento.

Se utilizó el conjunto de datos MNIST y modelos de reconocimiento de dígitos manuscritos alojados por Amazon, Google y MetaMind para evaluar el método propuesto para generar ataques adversarios. Al atacar un modelo de aprendizaje profundo alojado por MetaMind, el 84,24 % de los ejemplos adversarios generados por el modelo sustituto fueron clasificados incorrectamente por el modelo objetivo. Al evaluar el modelo sustituto contra modelos alojados por Amazon y Google, se obtuvieron tasas de éxito del 96,19 % y 88,94 % respectivamente en la clasificación errónea de los ejemplos adversarios. Finalmente, los resultados experimentales mostraron que la técnica propuesta logró altas tasas de transferibilidad de ejemplos adversarios.

Este artículo presentó un enfoque novedoso para la creación de ataques adversarios mediante el uso de modelos sustitutos para generar ejemplos adversarios transferibles a otros clasificadores. Sin embargo, la metodología propuesta difiere de la empleada en este proyecto de grado, ya que se enfocó en construir ejemplos adversarios mediante la sustitución de modelos y la manipulación de datos sintéticos, en lugar de explorar el espacio latente de una GAN para encontrar ejemplos adversarios efectivos. Además, la técnica no fue evaluada en conjuntos de datos de mayor complejidad, como los que involucran rostros humanos.

3.2. Otros antecedentes relevantes

En esta sección se presentan estudios de distintos autores que guardan relación con el proyecto de grado.

3.2.1. The Space of Transferable Adversarial Examples

En el artículo *The Space of Transferable Adversarial Examples* (Tramèr, Kurakin, Papernot, Boneh, and McDaniel, 2017) se exploró la transferibilidad de ejemplos adversarios entre diferentes modelos de aprendizaje automático y se propusieron algunas técnicas para estimar la dimensionalidad de los espacios de entrada adversarios, dentro de las cuales se destaca *Gradient Aligned Adversarial Subspace* (GAAS). Además, se exploró la proximidad de los límites entre modelos y se identificaron condiciones para la transferibilidad de ataques y limitaciones del entrenamiento adversario. Los experimentos se realizaron sobre los conjuntos de datos MNIST y DREBIN.

3.2.2. Improved Attribute Manipulation in the Latent Space of StyleGAN for Semantic Face Editing

En el artículo *Improved Attribute Manipulation in the Latent Space of StyleGAN for Semantic Face Editing* (Rai, Ducher, and Cooperstock, 2021) se presentó una extensión de la técnica InterfaceGAN en un modelo pre-entrenado de StyleGAN 2. La técnica se basó en la hipótesis de que existe un hiperplano en el espacio latente que separa un atributo binario, como el género o la edad.

El artículo presentó un método para manipular atributos faciales en el espacio latente de StyleGAN 2 para la edición facial semántica mediante la generación de límites para cada artefacto a modificar. Los resultados experimentales mostraron que al aplicar la técnica propuesta y mejorar la calidad de los límites semánticos se mejoró la calidad de las imágenes generadas y se logró un mayor control sobre la generación de rostros y la edición de atributos complejos.

3.2.3. The Harms of Demographic Bias in Deep Face Recognition Research

En el artículo *The Harms of Demographic Bias in Deep Face Recognition Research* (Garcia, Wandzik, Grabner, and Krueger, 2019) se abordó el problema del sesgo demográfico en los modelos de reconocimiento facial basados en aprendizaje profundo. Los resultados experimentales revelaron la existencia de sesgo demográfico en los modelos de reconocimiento facial basados en aprendizaje profundo, especialmente en ciertos grupos demográficos como las mujeres asiáticas, lo que plantea preocupaciones éticas y de derechos civiles y expone una mala práctica de investigación y desarrollo que puede llevar a una discriminación sistemática de ciertos grupos demográficos en problemas críticos como el control fronterizo automatizado.

3.2.4. Studying Bias in GANs through the Lens of Race

En el artículo *Studying Bias in GANs through the Lens of Race* (Maluleke, Thakkar, Brooks, Weber, Darrell, Efros, Kanazawa, and Guillory, 2022) se analizaron los sesgos raciales en los conjuntos de datos de caras y su influencia en los resultados obtenidos por los modelos generativos. Se investigó cómo el entrenamiento de un generador de imágenes con un conjunto de datos desbalanceado influye en la distribución y la calidad de las imágenes resultantes. Además, se analizó cómo influye el método de truncamiento utilizado en el entrenamiento de StyleGAN para mejorar la calidad de las imágenes generadas.

Las categorías raciales fueron etiquetadas como Negro, Blanco, No-negro y No-blanco, para reducir la ambigüedad perceptual. Se utilizaron los conjuntos de datos FFHQ (Flickr-Faces-HQ) y FairFace para crear 6 conjuntos de datos (100 % Blanco, 100 % Negro, 80 % Negro/20 % blanco (80N-20B), 50 % Negro/50 % blanco (50N-50B), 20 % Negro/80 % blanco (20N-80B) y otra razas. Se entrenaron 6 modelos de StyleGAN2-ADA con los 6 conjuntos de datos y los resultados fueron evaluados por humanos mediante *Amazon Mechanical Turk Annotation* (Sorokin and Forsyth, 2008).

Los resultados de los experimentos mostraron que las GANs preservaron el desbalance presente en el conjunto de datos de entrenamiento. El método de truncamiento utilizado por StyleGAN acentuó el desbalance en la generación de imágenes, lo que provocó que algunas categorías raciales no fueran generadas. Las imágenes de personas de raza negra mantuvieron una calidad de resolución constante independientemente de los datos de entrenamiento y las imágenes generadas de todas las razas tuvieron mayor calidad de resolución cuando fueron producidas por un generador que fue entrenado predominantemente con imágenes de personas blancas.

Este artículo mostró muchas de las dificultades al trabajar con imágenes faciales al evaluar razas lo que aporta valor al desarrollo del proyecto de grado y ayuda a comprender mejor la composición del conjunto de datos FFHQ.

Capítulo 4

Generación de ejemplos adversarios

En este capítulo se presenta la descripción del método utilizado para generar ejemplos adversarios y su implementación para cada caso de estudio. En la Sección 4.1 se detalla el método para generar ejemplos adversarios. En la Sección 4.2 se detalla la implementación para generar ejemplos adversarios contra clasificadores de dígitos manuscritos. En la Sección 4.3 se detalla la implementación para generar ejemplos adversarios contra clasificadores de imágenes de objetos. Finalmente, en la Sección 4.4 se detalla la implementación para generar ejemplos adversarios contra clasificadores de imágenes de rostros.

4.1. Metodología

En esta sección se describe la estrategia de desarrollo de este trabajo y se detalla el método utilizado para la generación de ejemplos adversarios.

4.1.1. Estrategia de desarrollo

En el proyecto de grado se adoptó una estrategia de desarrollo incremental para abordar el problema de generar ejemplos adversarios. Esta estrategia se basó en dividir el problema en fases incrementales, cada una con un objetivo específico y bien definido. Al final de cada fase, se evaluaron los resultados y se seleccionaron las mejores alternativas para avanzar a la siguiente fase.

El objetivo fue generar ejemplos adversarios, con dígitos manuscritos, imágenes de objetos e imágenes de rostros como casos de estudio. Para ello, se llevó a cabo la selección de un clasificador de imágenes apropiado para cada conjunto de datos en consideración, lo que fue crucial para garantizar una correcta evaluación de la efectividad de los ejemplos adversarios generados.

Una vez seleccionado el clasificador, se procedió con la selección de una GAN adecuada para el caso de estudio en cuestión. Se estableció como criterio fundamental que la GAN seleccionada tuviera la capacidad de producir imágenes consistentes con las imágenes del conjunto de datos de evaluación. Luego, se construyó un algoritmo genético para explorar el espacio latente de entrada de la GAN y se configuraron sus parámetros experimentalmente. Este enfoque permitió identificar y generar ejemplos adversarios contra el clasificador.

4.1.2. Descripción del método utilizado para generar ejemplos adversarios

El método que se utilizó para generar ataques adversarios consiste en realizar una búsqueda sobre el espacio latente de una GAN con el fin de encontrar vectores adecuados para generar imágenes que representen ataques adversarios. El método explora el espacio latente mediante el uso

de un algoritmo genético. El algoritmo parte de una población de vectores aleatorios pertenecientes al espacio latente. A partir de este conjunto de vectores se inicia el proceso evolutivo donde se evalúa a los individuos según su valor de fitness, que se calcula a partir del resultado de clasificar la imagen generada por una GAN condicional. Los individuos con mayor valor de fitness se seleccionan para cruzarlos y obtener nuevos descendientes. La nueva población se construye luego de aplicar un operador de mutación a los nuevos descendientes. Los ataques se obtienen al seleccionar aquellos individuos de la nueva población que son adecuados para generar imágenes que engañen a un clasificador. El método propuesto se detalla en la Figura 4.1.

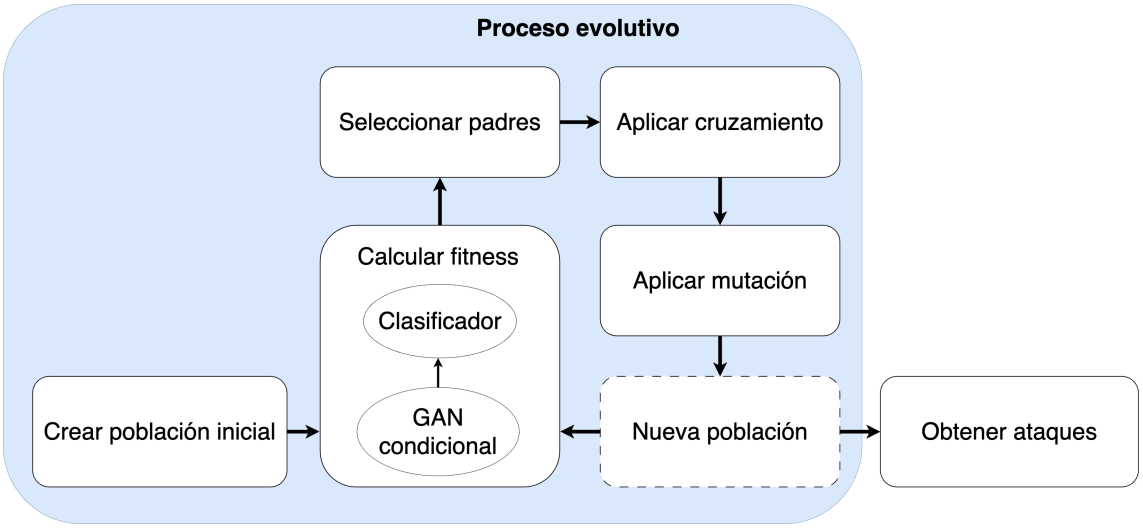


Figura 4.1: Diagrama del método utilizado para generar ejemplos adversarios

4.1.3. Requerimientos del método para generar ataques adversarios

Para desarrollar un método efectivo de generación de ataques adversarios, fue necesario establecer ciertos requerimientos que garanticen la eficiencia y robustez del proceso. El primer requerimiento fue asegurar que las imágenes generadas que representen ataques adversarios mantuvieran una alta calidad visual. El segundo requerimiento fue que los ataques generados debían ser clasificados correctamente por el ojo humano. Finalmente, el tercer requerimiento consistió en la generación de ataques diversos y en garantizar que los ejemplos adversarios incluyeran características variadas.

4.2. Ataques a clasificadores de dígitos manuscritos

En esta sección se detalla la estrategia utilizada para generar ataques efectivos contra clasificadores de dígitos manuscritos. Se presenta el conjunto de datos, los clasificadores, la GAN y la implementación del método detallado en la Sección 4.1.2 aplicado a este caso de estudio.

4.2.1. Conjunto de datos MNIST

MNIST (Li, 2012) es un conjunto de datos que contiene imágenes de dígitos escritos a mano. MNIST consta de un total de 70 000 ejemplos, divididos en 60 000 imágenes para entrenamiento y 10 000 imágenes para evaluación. Las imágenes son monocromáticas y tienen un tamaño de 28×28 píxeles. La Figura 4.2 presenta algunas imágenes pertenecientes al conjunto de datos MNIST.

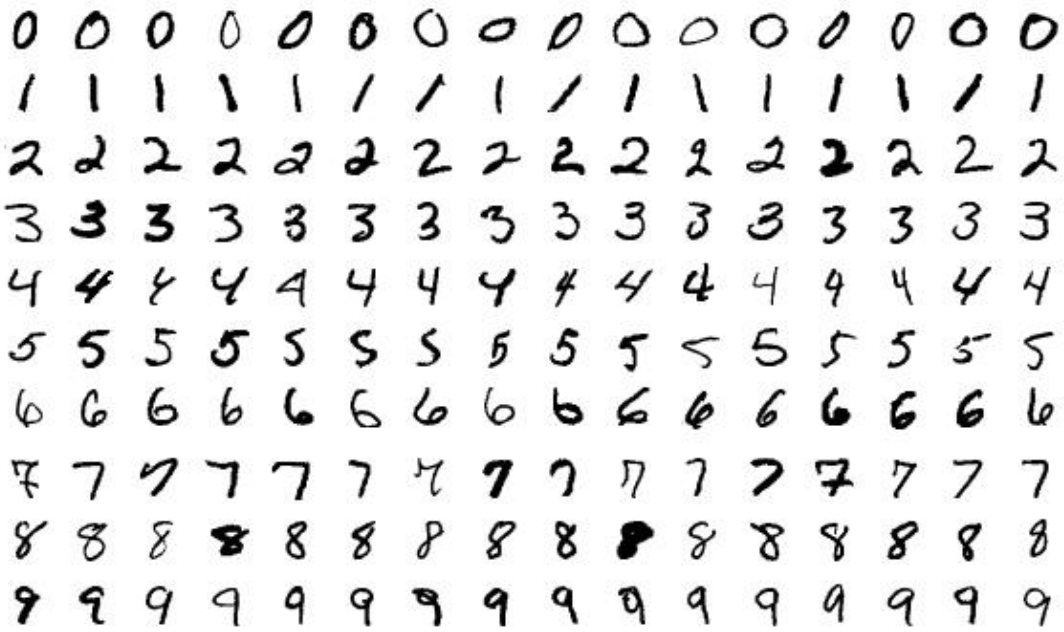


Figura 4.2: Imágenes del conjunto de datos MNIST.

El conjunto de datos MNIST es ampliamente utilizado en el campo de la clasificación de imágenes debido a su simplicidad. Muchas redes neuronales modernas logran alcanzar una precisión cercana al 100 % en la tarea de reconocimiento de dígitos en este conjunto de datos. MNIST es considerado un punto de referencia común para evaluar y comparar diferentes algoritmos y técnicas de aprendizaje automático.

En la fase inicial de este trabajo, se utilizó el conjunto de datos del MNIST debido a su menor complejidad en comparación con conjuntos de datos más avanzados, como conjuntos de datos de imágenes de objetos o rostros. La simplicidad intrínseca de MNIST brinda una base sólida para comprender y perfeccionar los conceptos y técnicas esenciales en la generación y clasificación de imágenes antes de enfrentarse a desafíos más complejos.

4.2.2. Selección de clasificadores y de la GAN para MNIST

Para el reconocimiento de dígitos manuscritos, se empleó un clasificador (c_m) previamente implementado y documentado por el investigador Jamal Toutouh, el cual ha demostrado una precisión (accuracy) superior al 98 % en diversas evaluaciones sobre el conjunto de datos de entrenamiento de MNIST. Además, se trabajó con un segundo clasificador (c'_m) ya implementado (Bose, 2019) que alcanzó 97 % de precisión (accuracy) en las evaluaciones realizadas sobre 1 000 imágenes.

La GAN utilizada fue una Conditional Deep Convolutional GAN (C-DCGAN) pública ya implementada y documentada (Mirza and Osindero, 2014). Esta GAN recibe vectores del espacio latente de 100 dimensiones con valores entre 0 y 1 y retorna imágenes de dígitos representados como matrices de 28×28 donde cada elemento de la matriz representa un píxel. Se seleccionó esta GAN debido a que demostró obtener los mejores resultados en la clasificación realizada por el clasificador al evaluar distintas alternativas durante los experimentos preliminares, tal como se detalla en la Sección 5.2.1.

La utilización de una GAN condicional simplificó la exploración en el espacio latente, ya que permitió la generación de ejemplos variados dentro de un dígito específico con solo modificar la entrada.

4.2.3. Implementación para generar ataques para MNIST

En esta sección se detalla la implementación del método utilizado para generar ataques adversarios para clasificadores del conjunto de datos del MNIST. En la Sección 4.2.3.1 se justifica la elección de la biblioteca PyGAD para la implementación del algoritmo genético. En la Sección 4.2.3.2 se detallan diferentes implementaciones de funciones de fitness.

4.2.3.1. Algoritmo genético con PyGAD

Para comenzar se implementó un algoritmo genético simple en el lenguaje Python sin el uso de bibliotecas auxiliares. Si bien esta implementación fue de utilidad para comprender el funcionamiento de un algoritmo genético, se decidió utilizar la biblioteca PyGAD (`https://pygad.readthedocs.io/en/latest/`) para facilitar la experimentación con el algoritmo genético. Esta biblioteca implementa la estructura de un algoritmo genético que sigue todos los pasos descriptos en la Sección 4.1.2. PyGAD permite personalizar cada uno de estos pasos, lo que posibilitó una experimentación sencilla y facilitó incorporar componentes específicas como el clasificador y la C-DCGAN en este trabajo. La implementación del método para generar ataques adversarios se encuentra disponible en el siguiente repositorio: `https://gitlab.fing.edu.uy/santiago.guridi/ataques-adversarios-con-algoritmos-evolutivos-y-redes-generativas-antagonicas`

4.2.3.2. Funciones de fitness

Se exploraron tres funciones de fitness distintas para evaluar y comparar diferentes soluciones candidatas. Las funciones de fitness utilizadas consideran los siguientes conjuntos y funciones en su definición:

- Un espacio latente de dimensión 100, $\mathbb{Z} \subseteq \mathbb{R}^{100}$
- $\mathbb{I} \subseteq \mathbb{N}^{28\times 28}$ el espacio de imágenes
- $\mathbb{K} = k_0, k_1, \ldots, k_9$ las etiquetas
- $k \in \mathbb{K}$ la etiqueta objetivo de ataque
- $g : \mathbb{Z}, \mathbb{K} \to \mathbb{I}$ la C-DCGAN
- $c_m : \mathbb{I} \to \mathbb{P}$ el primer clasificador, siendo $\mathbb{P} \subseteq [0,1]^{10}$ las probabilidades asignadas a cada etiqueta
- $c'_m : \mathbb{I} \to \mathbb{P}$ un segundo clasificador, siendo $c_m \neq c'_m$
- $c_{mk} : \mathbb{I} \to [0,1]$ la probabilidad asignada por el clasificador c_m a la etiqueta k al clasificar una imagen

Con el objetivo de generar confusión al clasificador entre todas las etiquetas se propuso una primera función de fitness f_1 que minimiza la probabilidad asignada por el clasificador a la etiqueta seleccionada. La Ecuación 4.1 define la función de fitness $f_1 : \mathbb{Z} \to \mathbb{R}$.

$$f_1(z) = 1 - \max_{p \in P} c_m(g(z,k)) \tag{4.1}$$

Con el objetivo de generar confusión al clasificador entre las dos etiquetas más probables se propuso una segunda función de fitness f_2 que minimiza la diferencia entre las dos mayores probabilidades asignadas por el clasificador y la probabilidad asignada a la etiqueta objetivo de ataque. La Ecuación 4.2 define la función de fitness $f_2 : \mathbb{Z} \to \mathbb{R}$.

$$f_2(z) = 1 - \left| \max_{p \in P} c_m(g(z,k)) - \max_{q \in P \setminus \{p\}} c_m(g(z,k)) \right| + 1 - c_{mk}(g(z,k)) \tag{4.2}$$

Se desarrolló una tercera función de fitness f_3 que extiende f_2. Se conserva la lógica de f_2 y se incorpora una componente que utiliza el segundo clasificador c'_m con el propósito de generar ataques efectivos para ambos clasificadores y desarrollar un modelo más robusto en la creación de ejemplos adversarios. La Ecuación 4.4 define la función de fitness $f_3 : \mathbb{Z} \to \mathbb{R}$ con la Ecuación 4.3 como función auxiliar.

$$f'(z, c_m) = f_2(z) \tag{4.3}$$

$$f_3(z) = f'(z, c_m) + f'(z, c'_m) \tag{4.4}$$

4.3. Ataques a clasificadores de imágenes de objetos

En esta sección se detalla la estrategia utilizada para generar ataques efectivos contra clasificadores de imágenes de objetos. Se presenta el conjunto de datos, los clasificadores, la GAN y el algoritmo utilizados para llevar a cabo los ataques.

4.3.1. Conjunto de datos CIFAR-10

CIFAR-10 es un conjunto de datos que contiene imágenes separadas en 10 clases distintas (avión, auto, ave, gato, ciervo, perro, rana, caballo, barco y camión). CIFAR-10 consta de un total de 60 000 ejemplos, divididos en 50 000 imágenes para entrenamiento y 10 000 para evaluación donde cada imagen es a color y tiene un tamaño de 32×32 píxeles. Este conjunto de datos es más complejo que MNIST debido a que las imágenes son a color y presentan mayor diversidad dentro de cada clase. La Figura 4.3 presenta diez imágenes de cada clase perteneciente al conjunto de datos CIFAR-10.

El segundo caso de estudio se centró en CIFAR-10, con el objetivo de experimentar en la generación de ataques contra un clasificador de un conjunto de datos más complejo en comparación con MNIST.

4.3.2. Selección de clasificadores y de la GAN para CIFAR-10

Se seleccionaron dos clasificadores de imágenes de objetos, c_c y c'_c. El primero utiliza el modelo previamente entrenado ResNet56 (Zhao, Wang, Zhang, Bai, and Hu, 2023), que mostró una precisión (accuracy) de 94,37 % en las evaluaciones realizadas sobre el conjunto de datos de entrenamiento de CIFAR-10 por Zhao y cols. El segundo clasificador utiliza el modelo RepVGG_A2 (Ding, Zhang, Ma, Han, Ding, and Sun, 2021) y alcanzó una precisión (accuracy) del 94,98 % en las evaluaciones realizadas sobre el conjunto de datos de entrenamiento de CIFAR-10 por Ding y cols. Las precisiones de los clasificadores seleccionados se consideraron suficientes para la aplicación del método para generar ataques adversarios desarrollado.

La GAN seleccionada para generar imágenes de objetos fue una Energy-based Conditional Generative Adversarial Network (Chen, Li, and Lin, 2021). Esta clase de GAN utiliza una función, llamada función de energía, para evaluar la autenticidad de los datos generados. El discriminador de la GAN actúa como un codificador automático para asignar energía a los datos reales y generados. En tanto, el generador minimiza esta energía para producir resultados convincentes. La GAN seleccionada recibe vectores del espacio latente de dimensión 80 con valores entre 0 y 1 y retorna imágenes de objetos representadas como matrices de 32×32. Se utilizó esta GAN para la generación de imágenes de calidad para cada una de las etiquetas de CIFAR-10. La selección de esta GAN se realizó luego de diversas evaluaciones preliminares detalladas en la Sección 5.3.1.

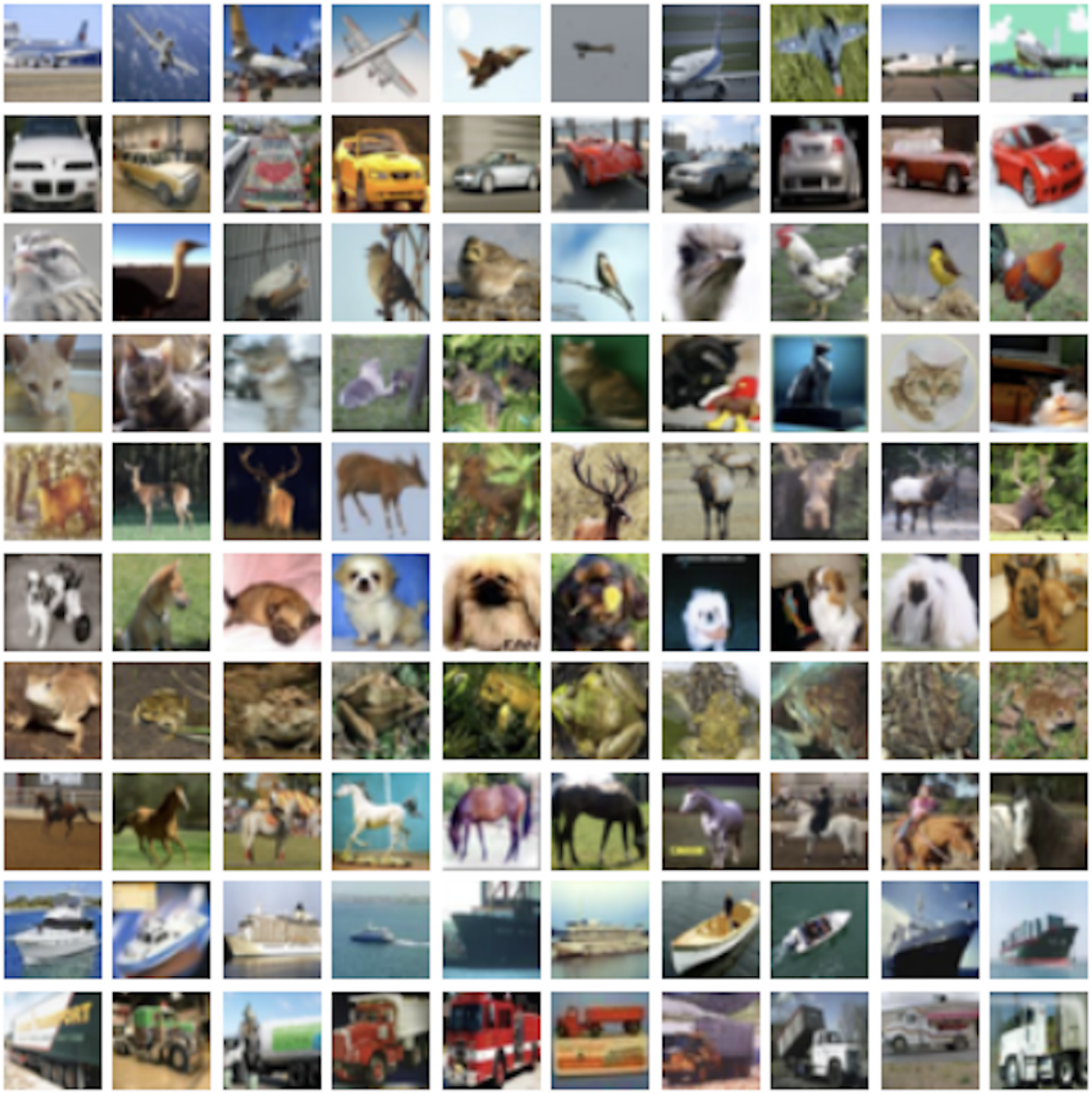


Figura 4.3: Imágenes del conjunto de datos CIFAR-10.

4.3.3. Implementación para generar ataques para CIFAR-10

Se decidió aplicar el método para generar ataques adversarios presentado en la Sección 4.1.2 sobre el conjunto de datos CIFAR-10. Se reutilizó la implementación detallada en la Sección 4.2.3.1. Las funciones de fitness presentadas en la Sección 4.2.3.2 se modificaron para incorporar las nuevas etiquetas (avión, auto, ave, gato, ciervo, perro, rana, caballo, barco y camión) y los nuevos clasificadores (c_c y c'_c). El objetivo de reutilizar la implementación anterior fue evaluar la robustez del algoritmo utilizado para MNIST al aplicarlo en un conjunto de datos diferente.

4.4. Ataques a clasificadores de imágenes de rostros

En esta sección se detalla la estrategia utilizada para generar ataques efectivos contra clasificadores de imágenes de rostros. Se presenta el conjunto de datos, el clasificador, la GAN y el algoritmo utilizados para llevar a cabo los ataques.

4.4.1. Conjunto de datos FFHQ

FFHQ es un conjunto de datos que contiene una colección de imágenes de rostros extraídas de la plataforma Flickr. FFHQ consta de un total de 70 000 rostros seleccionados de 67 646 imágenes de Flickr donde cada imagen tiene un tamaño de 1024×1024 píxeles. Este conjunto de datos fue creado por NVIDIA en 2019 como parte de un proyecto de generación de rostros y tiene una notable diversidad en términos de edades y razas, así como una amplia gama de accesorios, que incluyen gafas, lentes de sol, pendientes y gorros. FFHQ se ha consolidado como un recurso fundamental tanto para aplicaciones comerciales como para investigaciones en el campo de la detección y generación de rostros. La Figura 4.4 presenta una muestra de imágenes del conjunto de datos FFHQ, que abarca una amplia variedad de géneros, razas y edades.

Figura 4.4: Imágenes del conjunto de datos FFHQ

El último caso de estudio estuvo enfocado en el conjunto de datos FFHQ, lo que presentó un desafío mayor a la hora de generar ataques adversarios ya que este conjunto de datos es más complejo que los anteriores.

4.4.2. Selección del clasificador y de la GAN para FFHQ

Se utilizó DeepFace (Serengil and Ozpinar, 2021) como clasificador de imágenes de rostros. DeepFace es un sistema avanzado de reconocimiento facial desarrollado por Facebook que ha alcanzado un alto nivel de precisión al identificar caras en imágenes. Este clasificador utiliza una red neuronal convolucional profunda para extraer características faciales y un modelo de clasificación que compara estas características con una base de datos de rostros conocidos. DeepFace es capaz de reconocer caras en diversas condiciones de iluminación, pose y expresión facial, lo que ha superado los desafíos tradicionales del reconocimiento facial. Este sistema ha sido fundamental en el etiquetado automático de fotos en redes sociales.

La GAN que se utilizó en este caso de estudio fue StyleGAN (Karras, Laine, and Aila, 2019), en su versión StyleGAN2-ADA-PyTorch (Karras, Aittala, Hellsten, Laine, Lehtinen, and Aila, 2020). StyleGAN fue desarrollada por NVIDIA y surgió como una innovación destacada en la generación de imágenes realistas. StyleGAN destaca por aprender gradualmente a mejorar la resolución de las imágenes durante el proceso de entrenamiento, lo que contribuye a la generación de imágenes de alta calidad. StyleGAN es capaz de generar imágenes de diferentes tamaños y resoluciones según los requisitos específicos del usuario o del conjunto de datos de entrenamiento, ya que la arquitectura

de resolución progresiva implementada en StyleGAN permite la generación de imágenes desde resoluciones bajas hasta altas durante el proceso de entrenamiento. StyleGAN2-ADA-PyTorch es una implementación avanzada del modelo StyleGAN2 que utiliza la librería PyTorch e incorpora la técnica de aumento del discriminador adaptativo (ADA). StyleGAN2-ADA-PyTorch ofrece una gran flexibilidad y facilidad para la creación y el perfeccionamiento de modelos de generación de imágenes en diversos contextos y aplicaciones. En particular, se utilizó una técnica propuesta en el artículo *Improved Attribute Manipulation in the Latent Space of StyleGAN for Semantic Face Editing* (Rai, Ducher, and Cooperstock, 2021) que se basa en StyleGAN2-ADA-Pytorch y utiliza límites que representan distintos atributos como raza negra, raza blanca, raza asiática, entre otros, lo que permite generar rostros con determinada característica de manera condicional al realizar la suma de un vector del espacio latente con su proyección ortogonal en el límite deseado y aplicar un multiplicador que afecta proporcionalmente al nivel de pertenencia al límite indicado. La Ecuación 4.5 define un generador de rostros con una determinada característica c. Esta ecuación considera los siguientes conjuntos y funciones en su definición:

- $\mathbb{Z} \subseteq \mathbb{R}^{512}$, un espacio latente de dimensión 512
- $z \in \mathbb{Z}$, un vector del espacio latente
- $\mathbb{I} \subseteq \mathbb{N}^{1024 \times 1024}$ el espacio de imágenes
- $b \in \mathbb{Z}$ un vector del espacio latente ubicado dentro del límite de la característica c
- $\alpha \in \mathbb{R}$ un multiplicador
- $g : \mathbb{Z}, \mathbb{K} \to \mathbb{I}$ StyleGAN2-ADA-Pytorch
- $g' : \mathbb{Z}, \mathbb{K} \to \mathbb{I}$ el generador de rostros con la característica c

$$g'(z, \alpha, b) = g(z + \alpha \times b) \tag{4.5}$$

En la Sección 5.4.1 se detallan los múltiples experimentos realizados con diferentes versiones e implementaciones de StyleGAN antes de seleccionar la GAN StyleGAN2-ADA-PyTorch para el caso de estudio de imágenes de rostros.

4.4.3. Implementación del método para generar ataques para FFHQ

Se decidió aplicar el método para generar ataques adversarios presentado en la Sección 4.1.2 sobre el conjunto de datos FFHQ. Se reutilizó la implementación detallada en la Sección 4.2.3.1, con el objetivo de evaluar la robustez del algoritmo utilizado para MNIST y CIFAR al aplicarlo en un conjunto de datos más complejo, como imágenes de rostros. A diferencia de los casos de estudio anteriores, se decidió generar ataques enfocados en vulnerar una característica particular de los rostros, específicamente la raza. Se eligió la raza como característica a atacar, ya que es una característica fácilmente distinguible en los rostros, especialmente entre las razas blanca, negra y asiática.

4.4.3.1. Funciones de fitness

Se definieron funciones de fitness distintas según la raza a atacar, con el objetivo de guiar la generación de rostros hacia las secciones del espacio latente que vulneran la clasificación de DeepFace. Las funciones de fitness utilizadas consideran los siguientes conjuntos y funciones en su definición:

- $\mathbb{Z} \subseteq \mathbb{R}^{512}$ el espacio latente
- $\mathbb{I} \subseteq \mathbb{N}^{1024 \times 1024}$ el espacio de imágenes

- $\mathbb{K} = k_b, k_w, k_a, k_m, k_i, k_l$ las etiquetas de raza, siendo:
  - k_b: Raza negra
  - k_w: Raza blanca
  - k_a: Raza asiática
  - k_m: Raza medio oriente
  - k_i: Raza india
  - k_l: Raza latino hispánico
- $k \in \mathbb{K}$ la etiqueta objetivo de ataque
- $g : \mathbb{Z}, \mathbb{K} \to \mathbb{I}$ la GAN StyleGAN2-ADA-PyTorch
- $c : \mathbb{I} \to \mathbb{P}$ el clasificador DeepFace, siendo $\mathbb{P} \subseteq [0,1]^6$ las probabilidades asignadas a cada etiqueta
- $c_k : \mathbb{I} \to [0,1]$ la probabilidad asignada por el clasificador c a una etiqueta k al clasificar una imagen
- $qc : \mathbb{I} \to \{0,1\}$ una función que dada una imagen, retorna 1 si el rostro no presenta anomalías y 0 en otro caso
- $rc_k : \mathbb{I} \to \{0,1\}$ una función que dada una imagen, comprueba si el rostro de la imagen pertenece a la raza k, siendo $k \in \mathbb{K}$
- $scc_k : \mathbb{I} \to \mathbb{R}$ una función que dada una imagen, retorna un multiplicador asociado al color de piel del rostro y la raza k, siendo $k \in \mathbb{K}$

La función de fitness f_b utilizada en el método para generar ataques de raza negra minimiza la probabilidad asignada por el clasificador de que los rostros generados sean de raza negra o asiática (debido a que se generaban rostros de raza asiática), y aplica diversos filtros de calidad, raza y color sobre las imágenes. La Ecuación 4.6 define la función de fitness $f_b : \mathbb{Z} \to \mathbb{R}$.

$$f_b(\mathbf{z}) = \text{máx}\left(0, s_f \times \left(\frac{\text{max_value} - (\text{cond} + \epsilon)}{\text{max_value}}\right)^3\right) \times scc(img) \times qc(img) \times rc_b(img),$$
$$\text{con}$$
$$img = g(\mathbf{z}, k_b),$$
$$\text{cond} = c_b(img) + c_a(img), \tag{4.6}$$
$$\text{max_value} = 100 \text{ la cota superior},$$
$$s_f = 100 \text{ el factor de escala}$$

La función de fitness f_w utilizada en el método para generar ataques de raza blanca minimiza la probabilidad asignada por el clasificador de que los rostros generados sean de raza blanca y aplica diversos filtros de calidad, raza y color sobre las imágenes. La Ecuación 4.7 define la función de fitness $f_w : \mathbb{Z} \to \mathbb{R}$.

$$f_w(\mathbf{z}) = \text{máx}\left(0, s_f \times \left(\frac{\text{max_value} - (\text{cond} + \epsilon)}{\text{max_value}}\right)^3\right) \times scc_w(img) \times (1 - rc_a(img)),$$
$$\text{con}$$
$$\text{cond} = c_w(img),$$
$$\text{img} = g(\mathbf{z}, k_w), \tag{4.7}$$
$$\text{max_value} = 100 \text{ la cota superior},$$
$$s_f = 100 \text{ el factor de escala}$$

La función de fitness f_a utilizada en el método para generar ataques de raza asiática minimiza la probabilidad asignada por el clasificador de que los rostros generados sean de raza asiática. La Ecuación 4.8 define la función de fitness $f_a : \mathbb{Z} \to \mathbb{R}$.

$$f_a(\mathbf{z}) = \text{máx}\left(0, s_f \times \left(\frac{\text{max_value} - (\text{cond} + \epsilon)}{\text{max_value}}\right)^3\right) \times scc_w(img) \times rc_a(img),$$
$$\text{con}$$
$$\text{cond} = c_a(img),$$
$$img = g(\mathbf{z}, a),$$
$$\text{max_value} = 100 \text{ la cota superior},$$
$$s_f = 100 \text{ el factor de escala} \tag{4.8}$$

La construcción y justificación de estas funciones se encuentran en la Sección 5.4.2.

4.4.3.2. Clasificador basado en VGG16 y regresión logística

Las funciones de fitness implementadas utilizan las funciones c_k y qc que están basadas en un modelo que se construyó y se entrenó en el marco del proyecto para clasificar imágenes en dos categorías, por ejemplo imágenes de rostros con anomalías y sin anomalías.

El modelo construido utiliza un conjunto de datos de entrenamiento definido que contiene dos categorías. Este modelo se estructura de la siguiente manera. Inicialmente, se extraen las características de los rostros de ambas categorías del conjunto de datos de entrenamiento creado con la red neuronal VGG16 (Simonyan and Zisserman, 2015) entrenada previamente con el conjunto de datos ImageNet (Deng, Dong, Socher, Li, Li, and Li, 2009). VGG16 retorna como salida un vector de características, que representa información de alto nivel de la imagen. La salida de VGG16 se utiliza como entrada para entrenar un modelo de regresión logística para aprender a clasificar una imagen dada entre las categorías definidas.

4.4.3.3. Filtro de color basado en Stone

Las funciones de fitness implementadas utilizan la función scc_k, que es un filtro de color de piel basado en *Stone* (R. Piña and Chenglong, 2021). *Stone* es una función que asigna un color hexadecimal al color de piel de un rostro dado dentro de un conjunto definido de colores posibles. Se definió un conjunto de colores válidos para los rostros de cada raza con el fin de disminuir el valor de fitness en aquellos rostros que no tuvieran los colores validos. Para cada raza, se creó el correspondiente filtro de color, que dada una imagen de un rostro calcula el color de piel con *Stone* y retorna el multiplicador que se debería aplicar al valor de fitness calculado.

Capítulo 5

Experimentación

Este capítulo describe los experimentos realizados sobre los tres casos de estudio y detalla los resultados obtenidos. En la Sección 5.1 se detalla el entorno de ejecución utilizado para los experimentos. En la Sección 5.2 se detallan los resultados obtenidos al generar ejemplos adversarios para clasificadores de dígitos manuscrito. En la Sección 5.3 se detallan los resultados obtenidos al generar ejemplos adversarios para clasificadores de imágenes de objetos. Finalmente, en la Sección 5.4 se detallan los resultados obtenidos al generar ejemplos adversarios para clasificadores faciales.

5.1. Entorno de ejecución

La ejecución de los experimentos reportados sobre los tres casos de estudio se realizó en Windows 10 sobre un equipo con Procesador Intel Core i5-10400F, 16 GB de RAM y con tarjeta gráfica NVIDIA GeForce RTX 4060 Ti. Los experimentos restantes se llevaron a cabo en la plataforma ClusterUY.

Este entorno proporcionó la potencia de procesamiento y los recursos necesarios para llevar a cabo de manera eficiente y efectiva la ejecución del algoritmo genético, la generación de imágenes, la clasificación y el procesamiento posterior de los resultados obtenidos.

5.2. Análisis de ataques a clasificadores de dígitos manuscritos

En esta sección se detallan los experimentos realizados para generar ataques adversarios sobre el caso de estudio de imágenes de dígitos manuscritos.

5.2.1. Evaluación de la GAN para MNIST

Se llevaron a cabo experimentos iniciales utilizando dos GANs públicas entrenadas sobre el conjunto de datos MNIST (`https://github.com/sssingh/mnist-digit-generation-gan` y `https://github.com/lyeoni/pytorch-mnist-GAN`). Las GANs evaluadas utilizan un discriminador y generador con una arquitectura de cuatro capas lineales totalmente conectadas y una función de activación LeakyReLU. No se consiguió generar números legibles para el ser humano y no se obtuvo una precisión cercana a la obtenida sobre el conjunto de evaluación del clasificador (98 %) al clasificar las imágenes generadas con el clasificador c_m seleccionado en la Sección 4.2.2 por lo que se optó por investigar otras GANs.

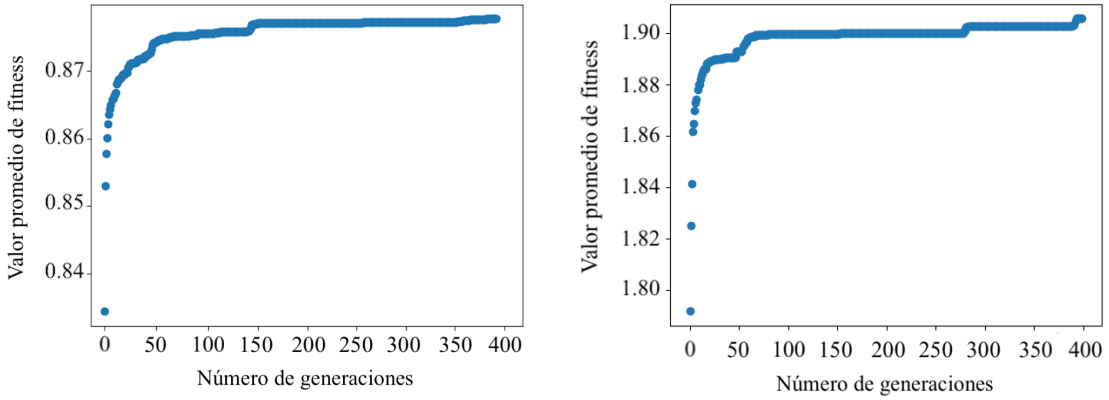
Se evaluó la C-DCGAN seleccionada en la Sección 4.2.2. Esta C-DCGAN inicialmente no cumplió las expectativas debido a que el clasificador c_m asignaba una probabilidad de 50 % a la etiqueta correcta. Esta precisión estaba por debajo de la precisión esperada para este clasificador. Se identificó que el problema residía en que el clasificador esperaba recibir valores en un rango diferente

al de los generados por la C-DCGAN, lo que producía discrepancias debido a la disparidad en los dominios. Se solucionó el problema al normalizar los datos generados por la C-DCGAN para que estuvieran en el mismo dominio que la entrada esperada por el clasificador. Al realizar este cambio, se logró una precisión cercana a 99 % en la clasificación, por lo que se decidió utilizar el modelo C-DCGAN.

5.2.2. Evaluación del método para generar ataques adversarios para MNIST

La evaluación consistió en realizar 20 ejecuciones independientes del método descripto en la Sección 4.2.3 para cada dígito. Se utilizaron las funciones de fitness propuestas en la Sección 4.2.3.2, el operador de selección de estado estacionario, el operador de cruzamiento de un punto, el operador de mutación aleatorio y una población de 50 individuos en cada generación. No se modificó la implementación por defecto proporcionada por PyGAD que aplica el operador de cruzamiento a todos los padres seleccionados y el operador de mutación a todos los descendientes y muta el 10 % de los genes.

Se estableció un total de 150 generaciones como criterio de parada para el experimento. Esta decisión se fundamentó en el análisis del comportamiento de las funciones de fitness f_1 y f_2 que mostraron un estancamiento parcial a partir 150 generaciones. Continuar con más iteraciones resultaría en un aumento significativo en los recursos computacionales sin aumentar más de 0,1 % el valor de fitness de ambas funciones. En la Figura 5.1 se muestra la evolución de los valores de fitness f_1 y f_2 al ejecutar el algoritmo evolutivo con un número máximo de 400 generaciones. Cada punto en la gráfica representa el promedio de los valores de fitness de una generación, calculado sobre 10 iteraciones del algoritmo genético realizadas para cada dígito.



(a) Evolución de los valores de fitness al utilizar f_1 en MNIST

(b) Evolución de los valores de fitness al utilizar f_2 en MNIST

Figura 5.1: Evolución de los valores de fitness según las funciones f_1 y f_2 en MNIST

En la primera fase de experimentación, se exploraron dos enfoques diferentes utilizando las funciones de fitness f_1 y f_2 para generar ataques contra el clasificador c_m. La cantidad de ataques generados para cada dígito objetivo se presenta en la Tabla 5.1. El tiempo de ejecución para obtener todos los ataques con la función de fitness f_1 fue 23 horas y con la función de fitness f_2 fue 21 horas.

Los resultados indicaron que, de todas las imágenes generadas en todas las generaciones para todos los dígitos (1 500 000 imágenes), 24 % fueron ataques contra el clasificador c_m al utilizar la función de fitness f_1 y 35 % al utilizar la función de fitness f_2. Si bien se generó una gran cantidad de ataques por dígito, los dígitos 6 y 9 presentaron menos de 15 000 ataques. En tanto, para los dígitos 3, 4 y 5 se generaron más de 50 000 ataques, lo que sugiere que estos dígitos son más fáciles

fitness	*dígito objetivo de ataque*										**Total**
	0	1	2	3	4	5	6	7	8	9	
f_1	38 098	19 624	37 742	58 512	55 661	57 647	10 409	34 411	43 550	3 918	**359 572**
f_2	46 386	31 555	51 813	108 399	63 309	96 490	9 622	50 141	63 860	1 698	**523 273**

Tabla 5.1: Cantidad de ataques al clasificador c_m agrupados por dígito objetivo de ataque y función de fitness

de atacar que el 6 y el 9. Además, se generó una mayor cantidad de ataques al emplear la función de fitness f_2 para todos los dígitos a excepción de los dígitos 6 y 9.

En los casos en que el clasificador asignó correctamente la etiqueta, un ataque se consideró exitoso si las dos probabilidades más altas estaban a una distancia menor que cierto δ. La Tabla 5.2 presenta el número de instancias generadas que cumplen esta condición. Los resultados están organizados según diferentes valores de δ y las función de fitness empleada. Los valores de δ comienzan en 0,5 y disminuyen en pasos de 0,1.

δ	*dígito objetivo de ataque*										**Total**
	0	1	2	3	4	5	6	7	8	9	
$< 0{,}5$	27 194	39 181	25 482	25 596	24 166	25 959	22 692	31 227	32 935	9 811	**264 243**
$< 0{,}4$	20 927	31 333	19 773	20 553	19 968	20 722	16 959	24 418	26 161	7 022	**207 836**
$< 0{,}3$	15 288	23 721	14 616	15 610	15 793	15 668	11 946	18 179	19 831	4 736	**155 388**
$< 0{,}2$	10 075	16 142	9 884	10 790	11 382	10 772	7 582	12 246	13 568	2 924	**105 365**
$< 0{,}1$	5 278	8 543	5 509	5 907	6 450	5 882	3 751	6 676	7 322	1 478	**56 796**

(a) Función de fitness f_1

δ	*dígito objetivo de ataque*										**Total**
	0	1	2	3	4	5	6	7	8	9	
$< 0{,}5$	22 593	21 311	20 236	9 617	15 687	11 252	25 616	23 886	20 771	11 679	**182 648**
$< 0{,}4$	17 258	15 401	15 383	7 593	12 271	9 010	19 264	18 459	16 434	8 296	**139 369**
$< 0{,}3$	12 581	10 519	11 120	5 707	9 113	6 759	13 860	13 340	12 366	5 396	**100 761**
$< 0{,}2$	8 267	6 396	7 226	3 959	6 103	4 649	8 898	8 698	8 312	3 031	**65 539**
$< 0{,}1$	4 320	2 978	3 739	2 119	3 196	2 494	4 444	4 371	4 349	1 255	**33 265**

(b) Función de fitness f_2

Tabla 5.2: Número de instancias bien clasificadas en las que las dos probabilidades más altas proporcionadas por el clasificador c_m están a una distancia menor que δ agrupados por dígito objetivo de ataque y función de fitness

Los resultados mostraron que se generaron más de 50 000 ejemplos en los que la diferencia entre las dos probabilidades más altas fue menor a 0,1 al utilizar la función de fitness f_1 y más de 30 000 al utilizar la función de fitness f_2. Este hallazgo sugiere que este método para generar ataques adversarios también puede producir ejemplos que sean clasificados correctamente, pero que aún así generen confusión en el clasificador.

Se analizaron los ataques a los dígitos 9 y 3 de forma detallada ya que representan los extremos en la distribución obtenida. Para el dígito 9, se evidenció una mayor dificultad en la generación de ejemplos antagónicos. En tanto, el dígito 3 exhibió una cantidad mayor de ataques. La Tabla 5.3 muestra la cantidad de ataques al dígito 9, organizados según la función de fitness utilizada, diferentes umbrales de probabilidad establecidos para clasificar una predicción como un *ataque* y la etiqueta proporcionada por el clasificador c_m. La Tabla 5.4 presenta los resultados para el dígito 3.

p	*Clasificación*		**Total**
	3	7	
> 0	934	2 984	**3 918**
$> 0{,}5$	672	2 633	**3 305**
$> 0{,}6$	311	1 493	**1 804**
$> 0{,}7$	129	794	**923**
$> 0{,}8$	35	346	**381**
$> 0{,}9$	0	83	**83**

(a) Función de fitness f_1

p	*Clasificación*			**Total**
	3	4	7	
> 0	435	838	425	**1 698**
$> 0{,}5$	435	838	425	**1 698**
$> 0{,}6$	7	130	30	**167**
$> 0{,}7$	0	24	4	**28**
$> 0{,}8$	0	3	0	**0**
$> 0{,}9$	0	0	0	**0**

(b) Función de fitness f_2

Tabla 5.3: Cantidad de ataques al dígito 9 según la función de fitness utilizada

p	*Clasificación*						**Total**
	0	2	5	7	8	9	
> 0	688	5 235	12 985	923	6 960	31 721	**58 512**
$> 0{,}5$	440	4 034	8 987	548	4 344	25 985	**44 338**
$> 0{,}6$	308	2 976	6 133	356	2 897	20 198	**32 868**
$> 0{,}7$	178	2 092	3 907	203	1 816	15 272	**23 468**
$> 0{,}8$	108	1 310	2 212	110	1 034	10 848	**15 622**
$> 0{,}9$	45	635	949	43	433	6 314	**8 419**

(a) Función de fitness f_1

p	*Clasificación*				**Total**
	2	5	8	9	
> 0	3 039	36 660	4 916	63 784	**108 399**
$> 0{,}5$	2 762	34 178	4 577	61 025	**102 542**
$> 0{,}6$	2 035	25 618	3 621	49 768	**81 042**
$> 0{,}7$	1 320	17 870	2 878	39 210	**61 278**
$> 0{,}8$	739	10 755	2 185	28 406	**42 085**
$> 0{,}9$	241	4 450	1 517	16 768	**22 976**

(b) Función de fitness f_2

Tabla 5.4: Cantidad de ataques al dígito 3 según la función de fitness utilizada

El análisis detallado de los dígitos 9 y 3 reveló diferencias significativas en la vulnerabilidad del clasificador c_m a los ataques antagónicos según el dígito objetivo. El clasificador demostró una alta resistencia a ataques antagónicos al clasificar el dígito 9, con una cantidad significativamente menor de clasificaciones erróneas, especialmente cuando se emplean umbrales de probabilidad elevados. Esta resistencia sugiere que las características del dígito 9 dificultan su manipulación a la hora de realizar un ataque. En contraste, para el dígito 3 se evidenció una alta vulnerabilidad en el clasificador, ya que se registró una amplia variedad de ataques exitosos incluso con umbrales de probabilidad elevados, lo que indicó una mayor dificultad del clasificador para distinguir correctamente este dígito en presencia de ejemplos adversarios. El clasificador confundió el dígito 9 con los dígitos 3, 4 y 7, y el dígito 3 con los dígitos 0, 2, 5, 7, 8 y 9. Para el dígito 9, se identificaron 83 casos en los cuales el clasificador asignó una probabilidad mayor a 0,9 al dígito incorrecto al utilizar la función de fitness f_1. En todos estos casos, el clasificador confundió el dígito 9 con el dígito 7. No se encontraron ataques con ese umbral de probabilidad al utilizar la función de fitness f_2. En tanto, para el dígito 3, los ataques donde el clasificador asignó una probabilidad mayor a 0,9 al dígito incorrecto fueron 8 419 al utilizar la función de fitness f_1 y 22 976 al emplear la función de fitness f_2. En más del 70 % de estos ataques el clasificador confundió el dígito 3 con el dígito 9. Las Tablas 5.5 y 5.6 presentan a modo de ejemplo diez ataques generados para los dígitos 9 y 3. El Anexo A.1 muestra más ejemplos de ataques con dígitos manuscritos.

Representación										
Clasificación	7	7	7	7	7	7	7	7	7	7
Probabilidad(p)	0,96	0,97	0,97	0,97	0,97	0,97	0,98	0,98	0,98	0,99

Tabla 5.5: Ejemplos de ataques al clasificador c_m con el número 9

Representación										
Clasificación	9	9	9	9	9	9	9	9	9	9
Probabilidad(p)	0,99	0,99	0,99	0,99	0,99	0,99	0,99	0,99	0,99	0,99

Tabla 5.6: Ejemplos de ataques al clasificador c_m con el número 3

La evaluación de la robustez de los ataques previamente generados se realizó al evaluar su efectividad sobre el segundo clasificador c'_m. Los resultados se reportan en la Tabla 5.7. El porcentaje de ataques al clasificador c_m que resultó efectivo contra el clasificador c'_m fue menor a 50 % para la mayoría de los dígitos con ambas funciones de fitness evaluadas. Esta disparidad impulsó la implementación y el análisis de la función de fitness f_3.

fitness	*dígito objetivo de ataque*									
	0	1	2	3	4	5	6	7	8	9
f_1	44,1 %	16,9 %	40,3 %	33,7 %	97,2 %	17,9 %	11,6 %	5,5 %	10,8 %	1,8 %
f_2	49,2 %	67,6 %	43,9 %	42,1 %	91,5 %	47,1 %	4,1 %	17,6 %	31,5 %	0 %

Tabla 5.7: Porcentaje de ataques al clasificador c_m que son efectivos para el clasificador c'_m agrupados por dígito objetivo de ataque y función de fitness

En la etapa final de la investigación con MNIST se evaluó la función de fitness f_3. El tiempo de ejecución para obtener todos ataques fue 40 horas. Los resultados se reportan en la Tabla 5.8.

clasificador	*dígito objetivo de ataque*										**Total**
	0	1	2	3	4	5	6	7	8	9	
c_m	0	0	13 657	29 447	143 063	58 064	0	1	0	0	**244 232**
c'_m	0	0	29 288	29 579	146 675	57 697	0	0	0	0	**263 239**

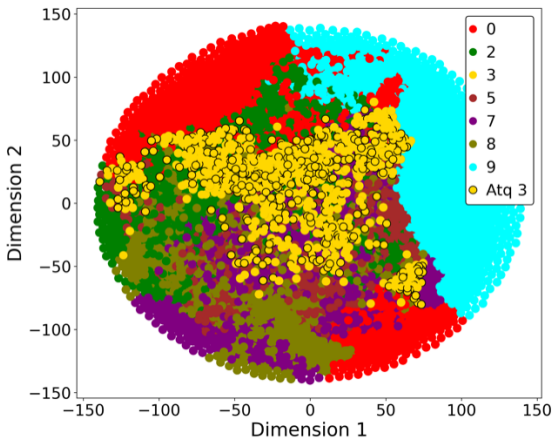
Tabla 5.8: Cantidad de ataques utilizando la función de fitness f_3 agrupados por dígito y clasificador

Los resultados indicaron que, de todas las imágenes generadas en todas las generaciones para todos los dígitos (1 500 000 imágenes), 16 % fueron ataques contra el clasificador c_m y 17 % contra el clasificador c'_m. Estos porcentajes son inferiores a los obtenidos al utilizar las dos primeras funciones de fitness. Para ciertos dígitos no se logró encontrar un ataque exitoso contra ninguno de los dos clasificadores. Este resultado sugiere que el método propuesto no es lo suficientemente robusto para generar ataques que engañen a ambos clasificadores para todos los dígitos. Por otro lado, se identificaron ataques exitosos para ciertos dígitos. En estos casos se generó una cantidad similar de ataques adversarios para ambos clasificadores. Los resultados para los dígitos 9 y 3 evidencian que no se encontraron ataques para el 9, pero si para el 3, aunque en menor medida que utilizando las funciones de fitness f_1 y f_2. En todos los ataques dirigidos al dígito 3, ambos clasificadores confundieron este dígito con el 9. Además, ambos clasificadores asignaron al dígito 9 una probabilidad superior a 0,9 en más del 99 % de los casos.

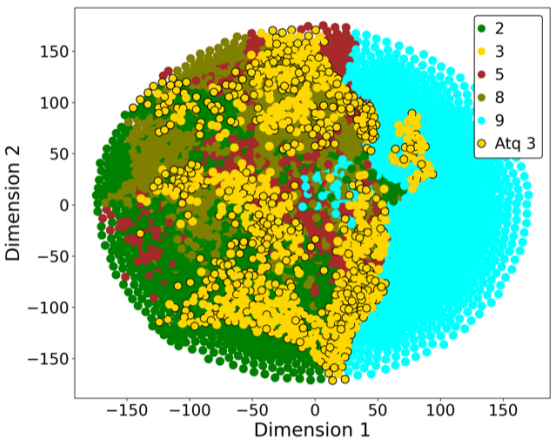
Una forma de representar los ataques es mediante el uso de t-distributed stochastic neighbor embedding (t-SNE), una técnica de reducción de dimensionalidad. t-SNE tiene la capacidad de capturar relaciones no lineales entre los datos y visualizarlos de manera efectiva en un espacio de menor dimensión. En el contexto de los ataques adversarios, la aplicación de t-SNE permite explorar y comprender la estructura subyacente de los datos, revelando patrones o agrupamientos que podrían no ser evidentes en la alta dimensionalidad original. Esta representación visual facilita la interpretación de la distribución de los ataques y contribuye a una comprensión más profunda de la dinámica del conjunto de datos. En la Figura 5.2 se presentan las representaciones t-SNE de los ataques obtenidos para el dígito 3 junto con los dígitos que generaron confusión según la Tabla 5.4 en función de la función de fitness utilizada. Las representaciones t-SNE de los ataques obtenidos para el dígito 9 se encuentran en el Anexo A.2.

El análisis de las representaciones t-SNE reveló que una parte de los ataques se situó en los límites de los conjuntos de puntos que representan un dígito. Sin embargo, también se identificaron ataques que se localizaron en regiones más centrales. Estos últimos ataques se consideran más efectivos, ya que muestran una mayor similitud con el número objetivo.

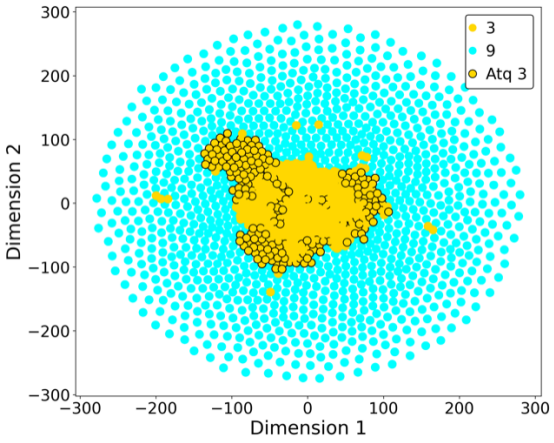
Los resultados obtenidos en la evaluación de MNIST se consideraron positivos y establecieron una base sólida para enfrentar conjuntos de datos más complejos. Se decidió prescindir de pruebas adicionales, como variar los operadores de cruzamiento o mutación en el algoritmo genético y se avanzó directamente al siguiente caso de estudio.



(a) Función de fitness f_1



(b) Función de fitness f_2



(c) Función de fitness f_3

Figura 5.2: Representaciones t-SNE de los Ataques al dígito 3 (Atq 3) según la función de fitness utilizada junto con los dígitos que generaron confusión

5.3. Análisis de ataques a clasificadores de imágenes de objetos

En esta sección se detallan los experimentos realizados para generar ataques adversarios sobre el caso de estudio de imágenes de objetos.

5.3.1. Evaluación de la GAN para CIFAR-10

Inicialmente, se llevaron a cabo experimentos utilizando una GAN condicional pública entrenada sobre el conjunto de datos CIFAR-10 (Mandhana, 2018). La GAN condicional utiliza un generador con capas densas y de convolución transpuesta para generar las imágenes. El discriminador de la GAN procesa las imágenes generadas con capas convolucionales y las clasifica tras añadir ruido Gaussiano. Al utilizar esta GAN, se encontró dificultad al distinguir visualmente la categoría de las imágenes generadas. Se exploró la posibilidad de entrenar la GAN con el conjunto de datos CINIC (Darlow, Crowley, Antoniou, and Storkey, 2018), que tiene 270 000 ejemplos obtenidos de CIFAR-10 e ImageNet, redimensionados a 32×32, de manera similar a CIFAR-10. A pesar de que las imágenes generadas mostraron mejoras en su calidad, no fue suficiente para distinguir la categoría de cada imagen a simple vista, lo que motivó la exploración de otras alternativas.

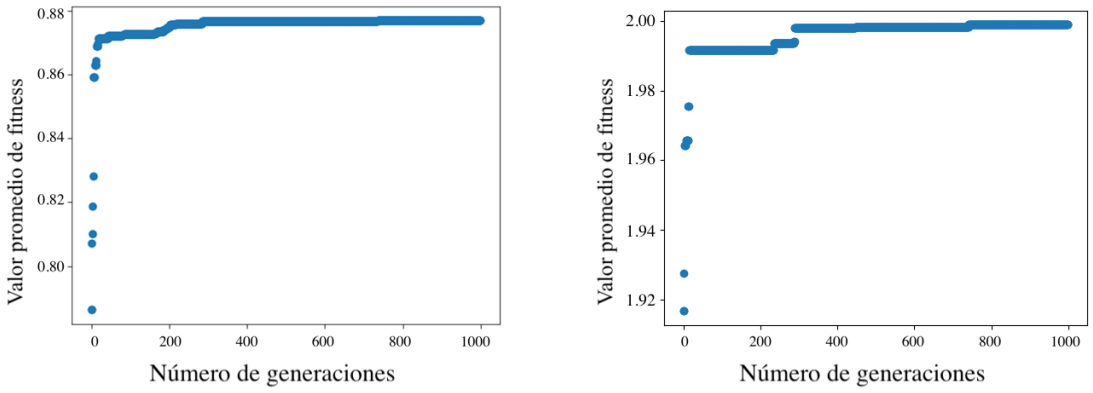
A continuación, se investigó la implementación de una Wasserstein Conditional GAN (WCGAN) (Gulrajani, Ahmed, Arjovsky, Dumoulin, and Courville, 2017) que destaca por tener una función que se basa en la distancia de Wasserstein. Se evidenció una mejora significativa en la calidad de las imágenes al utilizar WCGAN ya que se logró discernir las diferentes categorías de cada imagen. Sin embargo, se encontraron fallos en la condicionalidad, ya que las imágenes generadas en ocasiones pertenecían a una categoría diferente a la esperada.

Finalmente se evaluó la ECGAN presentada en la Sección 4.3.2 utilizando los pesos correspondientes a la iteración 150 000 del pre-entrenamiento de la ECGAN. Las imágenes generadas alcanzaron una calidad aceptable y la condicionalidad funcionó correctamente, lo que confirmó la elección de la ECGAN para este caso de estudio.

5.3.2. Evaluación del método para generar ataques adversarios para CIFAR-10

La evaluación consistió en realizar 20 ejecuciones independientes del método descripto en la Sección 4.3.3 para cada objeto. Se utilizaron las funciones de fitness propuestas en la Sección 4.2.3.2, el operador de selección de estado estacionario, el operador de cruzamiento de un punto, el operador de mutación aleatorio y una población de 50 individuos en cada generación. No se modificó la implementación por defecto proporcionada por PyGAD que aplica el operador de cruzamiento a todos los padres seleccionados y el operador de mutación a todos los descendientes y muta el 10 % de los genes.

Se obtuvo la evolución de la función de fitness para determinar el número máximo de generaciones por iteración del algoritmo genético. Se ejecutaron 1 000 generaciones por iteración del algoritmo planteado y se determinó que 400 generaciones eran suficientes, ya que en ese punto los valores de fitness tendieron a estancarse y no mostraron un crecimiento significativo. En la Figura 5.3 se muestra la evolución de los valores de fitness al ejecutar el algoritmo evolutivo con un número máximo de 1 000 generaciones. Cada punto en la gráfica representa el promedio de los valores de fitness de una generación, calculado sobre 10 iteraciones del algoritmo genético realizadas para cada objeto.



(a) Evolución de los valores de fitness al utilizar f_1 en CIFAR-10

(b) Evolución de los valores de fitness al utilizar f_2 en CIFAR-10

Figura 5.3: Evolución de los valores de fitness según las funciones f_1 y f_2 en CIFAR-10

En la primera fase de experimentación se exploraron dos enfoques diferentes con las funciones de fitness f_1 y f_2 para generar ataques contra el clasificador c_c. La cantidad de ataques generados para cada dígito objetivo se presenta en la Tabla 5.9. El tiempo de ejecución para obtener todos los ataques fue 28 horas para cada una de las funciones de fitness.

	objeto										**Total**
	avión	auto	ave	gato	ciervo	perro	rana	caballo	barco	camión	
f_1	305 578	247 784	216 923	219 213	195 167	268 045	285 000	176 786	251 710	155 062	**2 321 268**
f_2	358 333	314 822	274 273	288 058	273 521	343 726	292 980	257 261	344 986	242 424	**2 990 348**

Tabla 5.9: Cantidad de ataques al clasificador c_c agrupados por clase y función de fitness en CIFAR-10

Los resultados indicaron que, de todas las imágenes generadas en todas las generaciones para todos los objetos, el 58 % fueron ataques contra el clasificador c_c al utilizar la función de fitness f_1 y el 75 % al utilizar la función de fitness f_2. Estos porcentajes son mayores a los obtenidos contra el clasificador c_m para MNIST reportados en la Tabla 5.1. Se generaron más de 150 000 ataques para cada uno de los objetos. Además, se obtuvo una mayor cantidad de ataques al emplear la función de fitness f_2 en comparación con f_1 para cada uno de los objetos.

La Tabla 5.10 presenta el número de instancias bien clasificadas para las que las dos probabilidades más altas tuvieron una diferencia menor que cierto δ. Los resultados están organizados según diferentes valores de δ y la función de fitness empleada. Los valores de δ comienzan en 0,5 y disminuyen en pasos de 0,1.

δ	*Objeto*										**Total**
	avión	auto	ave	gato	ciervo	perro	rana	caballo	barco	camión	
$< 0{,}5$	40 349	43 749	100 378	53 517	48 118	38 989	49 264	39 930	54 978	63 024	**532 296**
$< 0{,}4$	36 139	37 978	97 170	45 548	41 121	33 714	45 936	33 347	47 744	52 232	**470 929**
$< 0{,}3$	31 548	31 893	93 575	37 746	34 092	28 352	42 625	26 599	40 061	41 498	**407 989**
$< 0{,}2$	26 003	25 113	88 941	29 362	26 295	22 605	38 943	19 535	31 256	30 033	**338 086**
$< 0{,}1$	18 557	16 270	78 505	19 370	16 898	15 563	33 518	11 436	19 956	17 561	**247 634**

(a) Función de fitness f_1

δ	*Objeto*										**Total**
	avión	auto	ave	gato	ciervo	perro	rana	caballo	barco	camión	
$< 0{,}5$	10 665	15 233	20 096	21 385	18 522	10 811	16 064	18 874	13 061	31 853	**176 564**
$< 0{,}4$	8 673	12 331	16 232	17 000	14 823	8 692	12 771	14 920	10 517	25 351	**141 310**
$< 0{,}3$	6 672	9 459	12 614	12 761	11 223	6 539	9 668	11 131	8 093	19 160	**107 320**
$< 0{,}2$	4 591	6 512	8 746	8 574	7 696	4 446	6 557	7 499	5 512	12 956	**73 089**
$< 0{,}1$	2 412	3 468	4 714	4 356	3 994	2 326	3 500	3 823	2 846	6 714	**38 153**

(b) Función de fitness f_2

Tabla 5.10: Número de instancias bien clasificadas en las que las dos probabilidades más altas proporcionadas por el clasificador c_c están a una distancia menor que δ agrupados por etiqueta objetivo de ataque y función de fitness

Los resultados mostraron que se generaron más de 240 000 ejemplos en los que la diferencia entre las dos probabilidades más altas fue menor a 0,1 al utilizar la función de fitness f_1 y más de 30 000 al utilizar la función de fitness f_2. Este hallazgo sugiere que este método para generar ataques adversarios también puede producir ejemplos que sean clasificados correctamente, pero que aún así generen confusión en el clasificador, principalmente al utilizar la función de fitness f_1.

Se analizaron los ataques a los objetos avión y camión de forma detallada, ya que representan los extremos en la distribución observada. Si bien se generaron más de 150 000 ataques para ambos objetos, el objeto avión presentó más de 300 000 ataques, lo que sugiere que es más fácil de atacar que el objeto camión. La Tabla 5.11 muestra la cantidad de ataques al objeto avión, organizados según la función de fitness utilizada, diferentes umbrales de probabilidad establecidos para clasificar una predicción como un *ataque* y la etiqueta proporcionada por el clasificador c_c. La Tabla 5.12 presenta los resultados para el objeto camión.

p	*Clasificación*									**Total**
	auto	ave	gato	ciervo	perro	rana	caballo	barco	camión	
> 0	4 835	97 149	93 824	25 691	4 933	33 536	13 242	24 494	7 874	**305 578**
$> 0{,}5$	885	24 973	26 033	13 683	696	9 716	2 825	3 815	669	**83 295**
$> 0{,}6$	590	20 738	19 376	10 971	519	7 779	1 762	2 908	465	**65 108**
$> 0{,}7$	406	17 315	14 521	8 637	376	6 104	1 079	2 158	334	**50 930**
$> 0{,}8$	268	14 050	10 538	6 502	259	4 663	635	1 473	232	**38 620**
$> 0{,}9$	135	10 333	6 863	4 183	163	3 125	276	867	132	**26 077**

(a) Función de fitness f_1

p	*Clasificación*									**Total**
	auto	ave	gato	ciervo	perro	rana	caballo	barco	camión	
> 0	397	88 872	112 568	52 009	4 037	44 359	32 935	20 904	2 252	**358 333**
$> 0{,}5$	250	66 615	84 345	40 044	2 240	31 573	23 107	16 021	1 549	**265744**
$> 0{,}6$	186	57 024	70 743	33 546	1 816	27 122	17 501	13 467	1 159	**222 564**
$> 0{,}7$	131	48 404	58 312	27 643	1 443	23 219	12 746	11 083	851	**183 832**
$> 0{,}8$	80	39 881	46 036	21 974	1 128	19 061	8 677	8 734	579	**146 150**
$> 0{,}9$	39	29 776	32 193	15 572	732	14 041	4 789	6 014	335	**103 491**

(b) Función de fitness f_2

Tabla 5.11: Cantidad de ataques al objeto avión según la función de fitness utilizada

El clasificador confundió las imágenes de aviones y camiones con todos los objetos del conjunto de datos CIFAR-10. Para el objeto avión se obtuvieron 26 077 ataques donde la probabilidad más alta asignada fue mayor a 0,9 al utilizar la función de fitness f_1 y 103 491 ataques al utilizar la función de fitness f_2. En ambos casos, el clasificador confundió al avión generado con un *ave* o un *gato* en más del 60 % de los ataques. Para el objeto camión se obtuvieron 16 444 ataques donde la probabilidad más alta asignada fue mayor a 0,9 al utilizar la función de fitness f_1 y 62 759 ataques al utilizar la función de fitness f_2. En ambos casos, el clasificador confundió al camión generado con un avión, un *auto* o un *gato* en más del 80 % de los ataques. Las Tablas 5.13 y 5.14 presentan diez ataques generados para los objetos avión y camión. El Anexo B.1 muestra más ejemplos de ataques con imágenes de objetos.

La evaluación de la robustez de los ataques previamente generados se realizó al evaluar su efectividad sobre el segundo clasificador c'_c. Los resultados se reportan en la Tabla 5.15. El porcentaje de ataques al clasificador c_c que resultó igualmente efectivo contra el clasificador c'_c al utilizar ambas funciones de fitness fue mayor a 50 % para todos los objetos a excepción de los objetos *ave*, *caballo* y camión al utilizar la función f_1. Estos porcentajes fueron más altos que los reportados en la Tabla 5.7 al evaluar el conjunto de datos MNIST con dos clasificadores, lo que sugiere que los clasificadores de CIFAR-10 son más fáciles de atacar.

p	*Clasificación*									**Total**
	avión	auto	ave	gato	ciervo	perro	rana	caballo	barco	
> 0	22 753	48 630	3 253	37 752	4 649	6 562	16 199	399	14 865	**155 062**
$> 0{,}5$	13 699	30 984	315	19 620	1 057	2 885	6 885	32	7 452	**83 295**
$> 0{,}6$	10 618	21 687	154	14 233	684	2 138	4 597	16	5 566	**82 929**
$> 0{,}7$	7 982	15 237	87	10 292	452	1 563	2 923	10	4 094	**42 640**
$> 0{,}8$	5 789	10 107	42	7 203	251	1 093	1 730	6	2 732	**28 953**
$> 0{,}9$	3 549	5 643	20	4 269	96	652	761	1	1 453	**16 444**

(a) Función de fitness f_1

p	*Clasificación*									**Total**
	avión	auto	ave	gato	ciervo	perro	rana	caballo	barco	
> 0	54 709	76 646	796	60 491	6 667	7 265	21 875	3 254	10 721	**242 424**
$> 0{,}5$	45 873	67 640	382	51 044	4 782	5 431	17 846	2 532	7 642	**203 172**
$> 0{,}6$	38 619	53 513	255	41 975	2 565	4 122	14 075	1 891	5 798	**162 813**
$> 0{,}7$	31 839	41 973	166	34 007	1 322	3015	10 738	1 483	4 228	**128 771**
$> 0{,}8$	25 089	31 371	102	26 091	628	2 007	7 506	1 059	2 777	**96 630**
$> 0{,}9$	17 491	19 867	42	17 635	264	1 021	4 371	634	1 434	**62 759**

(b) Función de fitness f_2

Tabla 5.12: Cantidad de ataques al objeto camión según la función de fitness utilizada

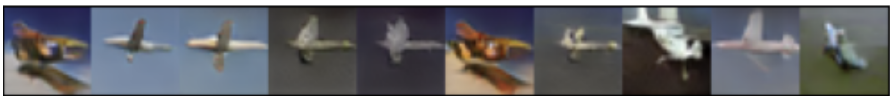
Representación										
Clasificación	auto	ave	ave	gato	ciervo	perro	rana	caballo	barco	camión
Probabilidad	0,99	0,99	0,99	0,99	0,99	0,99	0,99	0,99	0,99	0,99

Tabla 5.13: Ejemplos de ataques al clasificador c_c con la clase avión

Representación										
Clasificación	avión	auto	ave	gato	ciervo	perro	rana	caballo	barco	ciervo
Probabilidad	0,99	0,99	0,93	0,99	0,99	0,99	0,99	0,88	0,99	0,99

Tabla 5.14: Ejemplos de ataques al clasificador 1 con la clase camión

En la etapa final de la investigación con CIFAR-10 se evaluó la función de fitness f_3. El tiempo de ejecución para obtener todos ataques fue 36 horas. Los resultados se reportan en la Tabla 5.16. Los resultados indicaron que, de todas las imágenes generadas en todas las generaciones para todos los dígitos, el 73 % fueron ataques contra el clasificador c_c y el 74 % contra el clasificador c'_c. Estos porcentajes son inferiores a los obtenidos al utilizar la función de fitness f_2 y superiores a los obtenidos al utilizar la función de fitness f_1. En comparación con resultados obtenidos sobre el

fitness	*objeto*									
	avión	auto	ave	gato	ciervo	perro	rana	caballo	barco	camión
f_1	78,1 %	86,3 %	23,8 %	70,6 %	68,7 %	86,6 %	83,7 %	49,7 %	66,8 %	48,3 %
f_2	80,8 %	82,8 %	66,3 %	82,9 %	69 %	89 %	69,9 %	59 %	84,2 %	54,4 %

Tabla 5.15: Porcentaje de ataques al primer clasificador que son igual de efectivos para el segundo clasificador agrupados por dígito objetivo de ataque y función de fitness

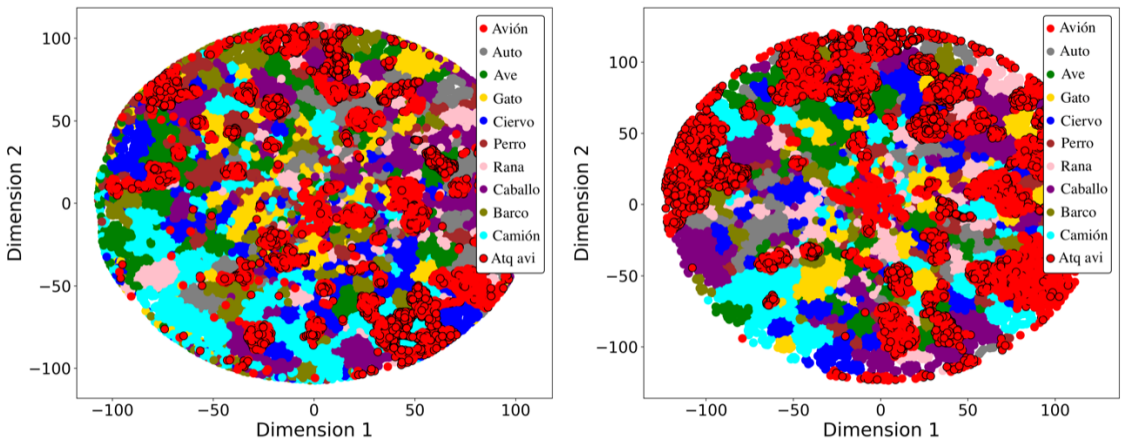
conjunto de datos MNIST, estos porcentajes fueron superiores.

clasificador	*objeto*										**Total**
	avión	auto	ave	gato	ciervo	perro	rana	caballo	barco	camión	
c	346 868	99 991	303 235	337 554	330 327	343 781	227 664	311 302	344 957	262 925	**2 908 604**
c'	349 418	148 629	308 506	338 914	332 770	350 194	214 248	299 679	339 877	272 589	**2 954 824**

Tabla 5.16: Cantidad de ataques al utilizar la función de fitness f_3 agrupados por dígito y clasificador

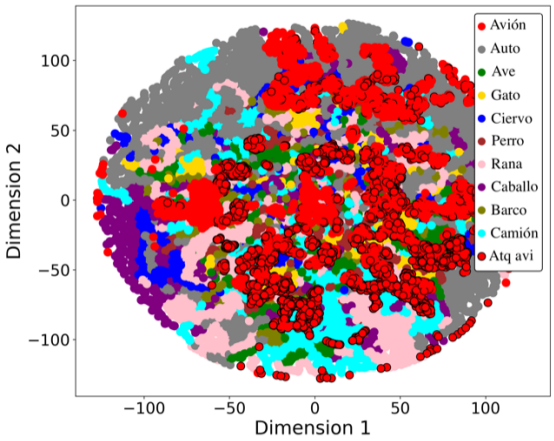
En la Figura 5.4 se presentan las representaciones t-SNE de los ataques obtenidos para el objeto avión junto con los objetos que generaron confusión según la Tabla 5.11 en función de la función de fitness utilizada. Las representaciones t-SNE de los ataques obtenidos para el objeto camión se encuentran en el Anexo B.2. De igual manera que para el caso de estudio de dígitos manuscritos, una parte de los ataques se situó en los límites de los conjuntos de puntos que representan un objeto y otros ataques se situaron en regiones más centrales.

Los resultados obtenidos en la evaluación de CIFAR-10 se consideraron positivos, ya que se logró generar múltiples ataques para todos los objetos del conjunto de datos. Estos hallazgos sugieren que los clasificadores de imágenes de objetos son más vulnerables que los clasificadores de dígitos manuscritos. Esta diferencia podría atribuirse a que los clasificadores de imágenes de objetos necesitan una mayor resolución en las imágenes procesadas para poder distinguir los detalles correctamente. Se decidió prescindir de pruebas adicionales, como variar los operadores de cruzamiento o mutación en el algoritmo genético y se avanzó directamente al siguiente caso de estudio.



(a) Función de fitness f_1

(b) Función de fitness f_2



(c) Función de fitness f_3

Figura 5.4: Representaciones t-SNE de los Ataques al objeto avión (Atq avi) según la función de fitness utilizada junto con los objetos que generaron confusión

5.4. Análisis de ataques a clasificadores de imágenes de rostros

En esta sección se detallan los experimentos realizados para generar ataques adversarios sobre el caso de estudio de imágenes de rostros.

5.4.1. Evaluación de StyleGAN

En esta sección se presentan los experimentos realizados con diferentes versiones e implementaciones de StyleGAN utilizadas en el método para generar ataques adversarios propuesto en la Sección 4.1.2.

5.4.1.1. Evaluación de StyleGAN sin extensiones

Para aplicar el método para generar ataques adversarios sobre el clasificador de imágenes de rostros DeepFace, se buscó una extensión de StyleGAN que proporcionara condicionalidad en función de atributos faciales. Sin embargo, no se encontró ninguna implementación que cumpliera este requisito, por lo que se exploró utilizar StyleGAN sin extensiones en el método para generar ataques.

Se ejecutaron evaluaciones preliminares del método para generar ataques adversarios mediante el uso de StyleGAN 3. Se seleccionó la raza como característica sobre la cual generar ataques. La población inicial fue cuidadosamente seleccionada para incluir individuos de la misma raza, con la intención de que el algoritmo genético mantuviera esta raza a lo largo de las generaciones. Para cada una de las razas, se creó una población inicial de 600 rostros obtenidos tras clasificar con DeepFace rostros aleatorios generados por StyleGAN 3, filtrar aquellos para los que la precisión presentada por DeepFace fuera menor a 90 % y filtrar manualmente imágenes con anomalías no deseadas.

La Tabla 5.17 muestra los tiempos requeridos para generar cada población inicial de las distintas razas, sin incluir el tiempo empleado en el filtrado manual. Las poblaciones iniciales de raza blanca, negra y asiática requirieron menos de 5 horas para ser generadas y las de raza india y latina tardaron más de 170 horas.

	Tiempo para Generación (hs)					
	Blancos	Negros	Asiáticos	Indios	Latinos	Medio oriente
Tiempo	1,3	4,0	2,0	177,0	256,0	47,0

Tabla 5.17: Tiempo para Generación por Condición

El algoritmo genético no mantuvo individuos de la raza inicial luego de algunas generaciones. La Figura 5.5 muestra un ejemplo de un rostro de la población inicial (Figura 5.5a) y uno resultante del algoritmo evolutivo luego de 20 generaciones (Figura 5.5b) de distinta raza al rostro de la población inicial. Para abordar este problema, se ajustaron los parámetros del algoritmo evolutivo y se aplicaron diversas técnicas para suavizar los movimientos en el espacio latente. Sin embargo, ninguna de estas técnicas logró que el algoritmo mantuviera rostros de la misma raza que la población inicial luego de varias generaciones.

(a) Rostro perteneciente a la población inicial

(b) Rostro perteneciente la generación 20

Figura 5.5: Comparación rostro de población inicial con rostro resultado del algoritmo luego de 20 generaciones.

5.4.1.2. Evaluación de InterfaceGAN

Se exploró la técnica InterfaceGAN (Shen, Gu, Tang, and Zhou, 2020). InterfaceGAN es una extensión de StyleGAN que permite generar rostros y determinar atributos faciales, como edad y género mediante la definición de límites en el hiperplano del espacio latente que determinan una división para algunos atributos faciales. Por ejemplo, a un lado del límite se encuentran rostros de mujeres y al otro rostros de hombres.

Con InterfaceGAN se generaron correctamente rostros para una población inicial. Sin embargo, en evaluaciones preliminares con el algoritmo genético luego de algunas generaciones se encontraron múltiples imágenes de rostros con anomalías, como se muestra en la Figura 5.6. Se aplicaron diferentes enfoques para abordar esta problemática, como la utilización del discriminador de StyleGAN para evaluar el realismo de las imágenes generadas. Sin embargo, esta estrategia no solucionó el problema, ya que la salida de la última capa del discriminador de StyleGAN no indica el nivel de realismo en los rostros generados. Se exploró el uso de FID (Heusel, Ramsauer, Unterthiner, Nessler, Klambauer, and Hochreiter, 2017), una métrica que permite medir la similitud entre dos conjuntos de datos. No obstante, esta métrica no resultó útil ya que no está diseñada para comparar una única imagen con un conjunto de datos. Se determinó que el problema residía en la versión de StyleGAN utilizada por InterfaceGAN, StyleGAN 1, por lo que se exploraron versiones más recientes de StyleGAN.

Figura 5.6: Ejemplo de rostro generado InterfaceGAN.

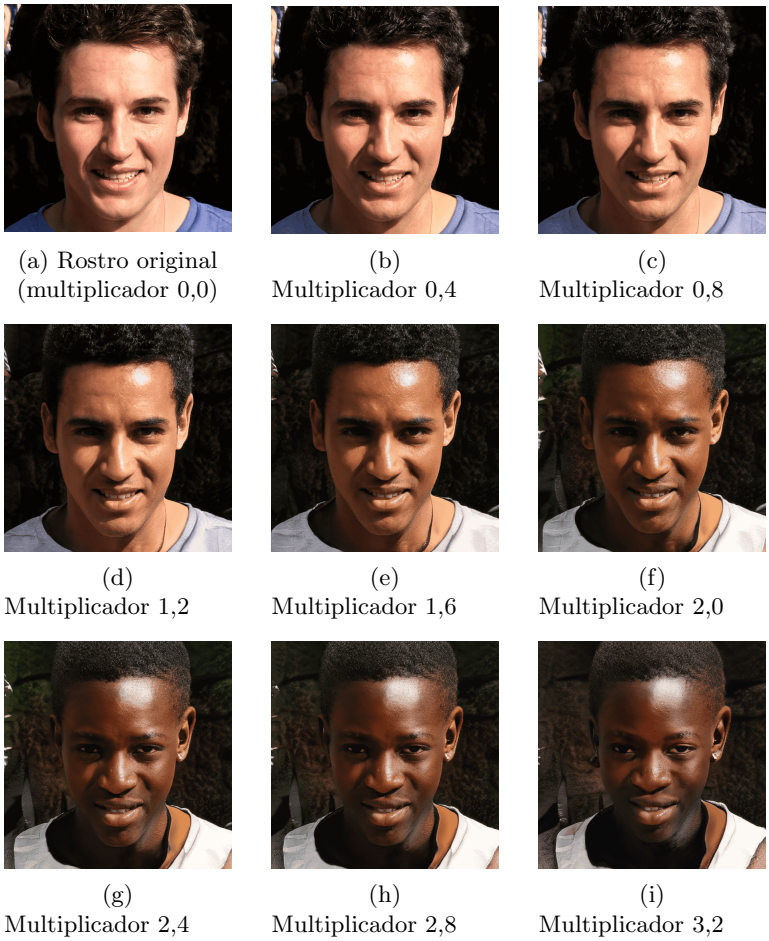
5.4.1.3. Evaluación de la herramienta basada en StyleGAN 3

Se buscó una implementación pública de StyleGAN 3 condicional, sin embargo no se encontró ninguna, por lo que se diseñó una herramienta para generar imágenes de rostros de manera condicional con StyleGAN 3.

Se exploró añadir una capa adicional al generador de StyleGAN 3 para agregar condicionalidad en la generación de rostros según una característica específica, como la raza. Para ello, se comenzó con una población inicial de rostros de la raza deseada y se procedió a calcular hiperplanos sobre esta población mediante SVM (*Support Vector Machine*) (Cortes and Vapnik, 1995) y PCA (*Principal component analysis*) (Mackiewicz and Ratajczak, 1993). Luego, se proyectaron vectores aleatorios del espacio latente sobre estos hiperplanos. Al utilizar esta proyección se generaron rostros de la raza deseada pero demasiado similares entre sí. Se aplicaron diversas mutaciones a la proyección sin aumentar la diversidad, por lo que se descartó esta opción.

5.4.1.4. Evaluación de la herramienta basada en StyleGAN2-ADA-Pytorch

Se exploró la técnica propuesta en el artículo *Improved Attribute Manipulation in the Latent Space of StyleGAN for Semantic Face Editing* (Rai, Ducher, and Cooperstock, 2021) basada en StyleGAN2-ADA-Pytorch, descripta en la Sección 4.4.2. Se generaron rostros de distintas razas, con la aplicación de la Ecuación 4.5. En la Figura 5.7 se muestran diferentes rostros generados luego de aplicar distintos multiplicadores con el límite de raza negra.



(a) Rostro original (multiplicador 0,0) (b) Multiplicador 0,4 (c) Multiplicador 0,8

(d) Multiplicador 1,2 (e) Multiplicador 1,6 (f) Multiplicador 2,0

(g) Multiplicador 2,4 (h) Multiplicador 2,8 (i) Multiplicador 3,2

Figura 5.7: Uso de distintos multiplicadores en el generador

Se optó por utilizar StyleGAN2-ADA-Pytorch con está técinca ya que ofrece condicionalidad por raza y en los experimentos preliminares con el algoritmo genético no generó rostros con anomalías como el de la Figura 5.6 y mantuvo la raza de la población inicial en la mayoría de las generaciones. En la Sección 5.4.2 se detallan las técnicas utilizadas para mantener la condicionalidad en todas las generaciones.

5.4.2. Optimización del método para generar ataques con StyleGAN

En esta sección se detalla la optimización que se realizó sobre el método para generar ataques adversarios de imágenes de rostros para mantener la condicionalidad en todas las generaciones. En particular, se detalla la construcción de las funciones de fitness que se utilizaron para la raza asiática, blanca y negra, definidas en la Sección 4.4.3.1.

5.4.2.1. Optimización del método para generar ataques de raza negra

Inicialmente, el método para generar ataques adversarios de raza negra utilizó una función de fitness que minimizaba la probabilidad proporcionada por DeepFace de que el rostro sea de raza negra. La función de fitness inicial está dada por la Ecuación 5.1. Esta ecuación considera los conjuntos y funciones definidos en la Sección 4.4.3.1.

$$f_b(\mathbf{z}) = \frac{1}{c_b(g(\mathbf{z}, k_b))} \tag{5.1}$$

Al aplicar el método para generar ataques adversarios utilizando la función de fitness 5.1 se obtuvieron imágenes de rostros de raza asiática a partir de la generación 15 aproximadamente, como el de la Figura 5.8. Para mitigar este problema, se incorporó a la función de fitness una componente que minimiza la probabilidad asignada por DeepFace de que los rostros sean de raza asiática. La función de fitness modificada está dada por la Ecuación 5.2. Esta modificación en la función de fitness limitó la generación de rostros de raza asiática y la generación de rostros de raza negra que eran clasificados como raza asiática.

$$f_b(\mathbf{z}) = \frac{1}{c_b(g(\mathbf{z}, k_b)) + c_a(g(\mathbf{z}, k_a))} \tag{5.2}$$

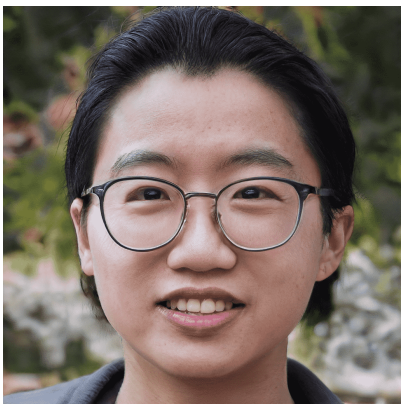


Figura 5.8: Ejemplo de rostro asiático generado utilizando los límites de raza negra.

Al ejecutar el método para generar ataques adversarios con la función de fitness dada por la Ecuación 5.2, se obtuvo una mayor cantidad de rostros de raza negra respecto a utilizar la función dada por la Ecuación 5.1. No obstante, a partir de aproximadamente 20 generaciones se obtuvieron rostros de raza blanca, como el de la Figura 5.9. Para mitigar este problema, sin impedir al método obtener rostros de raza negra clasificados por DeepFace como raza blanca, se incorporó a la función de fitness el filtro de color de piel descripto en la Sección 4.4.3.3. Para el caso de la raza negra, la función retorna 0 si el color de la piel es inválido y retorna 1 si es válido. La función de fitness está definida por la Ecuación 5.3.

$$\text{scc}_b(\text{img}) = \begin{cases} 1 & \text{si Stone(img)} \in \{\#373028, \#422811, \#513B2E, \#6F503C\} \\ 0 & \text{en otro caso} \end{cases} \tag{5.3}$$

La función de fitness modificada que está dada por la Ecuación 5.4 realiza el producto del valor de fitness dado por la Ecuación 5.2 con el resultado de la función scc_b aplicada sobre la imagen. Al ejecutar el método con esta función de fitness no se generaron rostros de raza blanca. Sin embargo, se obtuvieron rostros de raza negra con atributos faciales mal definidos, como los ojos. La Figura 5.10 muestra una comparación entre rostros con anomalías y sin anomalías al utilizar esta función de fitness. Para mitigar el problema se incorporó a la función de fitness el valor de una función qc que retorna 0 si la imagen es clasificada como un rostro con anomalía

Figura 5.9: Ejemplo de rostro de raza blanca generado utilizando los límites de raza negra.

y 1 en caso contrario. La función qc utiliza el modelo clasificador basado en VGG16 y regresión logística definido en la Sección 4.4.3.2. El conjunto de datos de entrenamiento utilizado para el modelo clasificador está formado por imágenes de rostros con anomalías y sin anomalías. En el Anexo C.1 se encuentran más ejemplos de ambas categorías. La función de fitness que incorpora esta modificación está dada por la Ecuación 5.5.

$$f_b(\mathbf{z}) = \frac{1}{c_b(img) + c_a(img)} \times scc_b(img), \text{ con } img = g(\mathbf{z}, k_b) \tag{5.4}$$

$$f_b(\mathbf{z}) = \frac{1}{c_b(img) + c_a(img)} \times scc(img) \times qc(img), \text{ con } img = g(\mathbf{z}, k_b) \tag{5.5}$$

Al ejecutar el método para generar ataques adversarios utilizando la función de fitness dada por la Ecuación 5.5, se obtuvieron ejecuciones que generaron rostros de raza negra sin anomalías en más de 300 generaciones. No obstante, la mayoría de las ejecuciones generaron rostros de raza blanca a partir de cierta generación. Para mitigar este problema, se creó la función rc_b que utiliza el modelo clasificador basado en VGG16 y regresión logística y el filtro de color *Stone*. La función rc_b retorna 1 si el rostro generado pasa el filtro de raza negra creado y 0 en caso contrario. El conjunto de datos de entrenamiento utilizado para este filtro está formado por imágenes de rostros de raza negra y blanca. En el Anexo C.2 se encuentran ejemplos de ambas categorías. Se incorporó el nuevo filtro a la función de fitness de igual manera que para los filtros anteriores. La función de fitness modificada está dada por la Ecuación 5.6.

$$f_b(\mathbf{z}) = \frac{1}{c_b(g(img) + c_a(g(img))} \times scc(img) \times qc(img) \times rc_b(img), \text{ con } img = g(\mathbf{z}, k_b) \tag{5.6}$$

(a) Rostros generados con anomalías

(b) Rostros generados sin anomalías

Figura 5.10: Comparación de rostros generados con anomalías y sin anomalías

Al utilizar la función de fitness dada por la Ecuación 5.6 en el método para generar ataques adversarios, no se encontraron problemas significativos respecto a la raza o la calidad de los rostros generados en ninguna generación. Los valores de fitness obtenidos mostraron fluctuaciones significativas entre cifras de dos dígitos y valores de seis o más dígitos en algunos casos. Los valores más altos se obtuvieron principalmente en casos donde todos los filtros (scc, qc y rc_b) tienen valor uno y las probabilidades calculadas por DeepFace de que el rostro fuera de raza asiática o negra eran cercanas a 0 (aproximadamente 10^{-4}). Se modificó la función de fitness para normalizar los valores y disminuir las fluctuaciones. Se introdujo un término ϵ con valor 1×10^{-3} en el denominador para evitar divisiones por valores demasiado cercanos a cero, lo que estabilizó y suavizó los resultados de la función. Adicionalmente, se utilizó la función máx$(0, \cdot)$ para garantizar que los valores de fitness no fueran negativos y se incorporó un factor de escala s_f para controlar el rango de valores de fitness, que establece una cota superior de 100. Estas modificaciones aseguraron que los valores de fitness se mantuvieran dentro de un rango de 0 a 100. La función de fitness final utilizada para generar ataques de raza negra está dada por la Ecuación 5.7.

$$
\begin{aligned}
f_b(\mathbf{z}) = \text{máx}\left(0, s_f \times \left(\frac{\text{max_value} - (\text{cond} + \epsilon)}{\text{max_value}}\right)^3\right) \times scc(img) \times qc(img) \times rc_b(img), \\
\text{con } img = g(\mathbf{z}, k_b),\ \text{cond} = c_b(img) + c_a(img),\ \text{max_value} = 100 \text{ la cota superior}, \\
s_f = 100 \text{ el factor de escala}
\end{aligned}
\tag{5.7}
$$

5.4.2.2. Optimización del método para generar ataques de raza blanca

Inicialmente, el método para generar ataques adversarios de raza blanca utilizó una función de fitness que minimizaba la probabilidad proporcionada por DeepFace de que el rostro sea de raza blanca para cada rostro de la población. La función de fitness inicial está dada por la Ecuación 5.8. Esta ecuación considera los conjuntos y funciones definidos en la Sección 4.4.3.1.

$$f_w(\mathbf{z}) = \frac{1}{c_w(g(\mathbf{z}, k_w))} \tag{5.8}$$

Al aplicar el método para encontrar ataques adversarios con la función de fitness dada por la Ecuación 5.8, luego de aproximadamente 10 generaciones, se obtuvieron múltiples imágenes de rostros de raza asiática, como el de la Figura 5.11. Para mitigar este problema, se incorporó a la función de fitness el valor opuesto de una función rc_a que retorna 0 si el rostro fue clasificado como asiático por el modelo entrenado y 1 en otro caso. rc_a utiliza el modelo clasificador basado en VGG16 y regresión logística definido en la Sección 4.4.3.2. El conjunto de datos de entrenamiento utilizado para el modelo clasificador constó de imágenes de rostros de raza asiática e imágenes de rostros de raza blanca. En el Anexo C.3 se encuentran más ejemplos de ambas categorías. La función de fitness que incorpora esta modificación está dada por la Ecuación 5.9.

Figura 5.11: Ejemplo de rostro asiático generado utilizando los límites de raza blanca.

$$f_w(\mathbf{z}) = \frac{1}{c_w(img)} \times (1 - rc_a(img)), \text{ con } img = g(\mathbf{z}, k_w) \tag{5.9}$$

Al aplicar el método para encontrar ataques adversarios con la función de fitness dada por la Ecuación 5.9 no se obtuvieron rostros de raza asiática. Sin embargo, se obtuvieron rostros de raza india y latina después de la generación 15 aproximadamente. En la Figura 5.12 se muestran algunos rostros de razas distinta a blanca al utilizar los límites de raza blanca. Para mitigar este problema, se introdujo a la función de fitness el filtro de color de piel descripto en la Sección 4.4.3.3. Para el caso de la raza blanca, la función scc_w, definida en la Ecuación 5.10, asigna un multiplicador distinto según cada color y tiene en cuenta diversos colores con el objetivo de no perder diversidad. En la figura 5.13 se muestran los colores asociados a los valores hexadecimales utilizados. La función de fitness modificada está dada por la Ecuación 5.11.

$$\text{scc}_w(\text{img}) = \begin{cases} 0{,}2 & \text{si Stone(img)} = \#81654F, \\ 0{,}9 & \text{si Stone(img)} = \#9D7A54, \\ 1{,}0 & \text{si Stone(img)} = \#BEA07E, \\ 1{,}4 & \text{si Stone(img)} = \#E5C8A6, \\ 1{,}5 & \text{si Stone(img)} = \#E7C1B8 \text{ o } \#F3DAD6 \text{ o } \#FBF2F3 \end{cases} \tag{5.10}$$

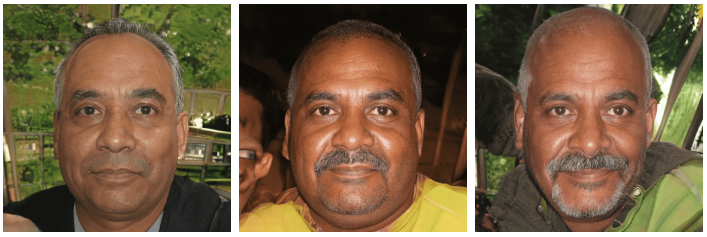


Figura 5.12: Rostros de raza distinta a blanca generados utilizando los límites de raza blanca

Figura 5.13: Colores de Stone

$$f_w(\mathbf{z}) = \frac{1}{c_w(img)} \times (1 - rc_a(img)) \times scc_w(img), \text{ con } img = g(\mathbf{z}, k_w) \tag{5.11}$$

Con la función de fitness dada por la Ecuación 5.11 no se generaron rostros de raza distinta a blanca, ni rostros con anomalías. No obstante, se modificó la función de fitness para normalizar los valores y disminuir las fluctuaciones. Se introdujo un término ϵ con valor 1×10^{-3} en el denominador para evitar divisiones por valores demasiado cercanos a cero, lo que estabilizó y suavizó los resultados de la función. Adicionalmente, se utilizó la función máx$(0, \cdot)$ para garantizar que los valores de fitness no fueran negativos, y se incorporó un factor de escala s_f para controlar el rango de valores de fitness, que establece una cota superior de 150, debido a que el multiplicador de color scc_w puede alcanzar un valor de 1.5. Estas modificaciones aseguraron que los valores de fitness se mantuvieran dentro de un rango de 0 a 150. La función de fitness final utilizada para generar ataques de raza negra está dada por la Ecuación 5.12

$$\begin{gathered} f_w(\mathbf{z}) = \text{máx}\left(0, s_f \times \left(\frac{\text{max_value} - (\text{cond} + \epsilon)}{\text{max_value}}\right)^3\right) \times scc_w(img) \times (1 - rc_a(img)), \\ \text{con cond} = c_w(img), \ \text{img} = g(\mathbf{z}, k_w), \ \text{max_value} = 100 \text{ la cota superior}, \\ s_f = 100 \text{ el factor de escala} \end{gathered} \tag{5.12}$$

5.4.2.3. Optimización del método para generar ataques de raza asiática

Inicialmente, el método para generar ataques adversarios de raza asiática utilizó una función de fitness que minimizaba la probabilidad proporcionada por DeepFace de que el rostro sea de raza asiática para cada rostro de la población. La función de fitness inicial está dada por la Ecuación 5.13. Todas las funciones de fitness definidas en esta sección utilizan los términos, funciones y conjuntos definidos en la Sección 4.4.3.1.

$$f_a(\mathbf{z}) = \frac{1}{c_a(g(\mathbf{z}, k_a))} \tag{5.13}$$

Al aplicar el método para encontrar ataques adversarios con la función de fitness dada por la Ecuación 5.13, a partir de la generación 20 aproximadamente se encontraron rostros de raza blanca. La Figura 5.14 muestra ejemplos de rostros de raza negra al utilizar los límites de raza asiática. Para mitigar este problema, se incorporó a la función de fitness el valor de la función rc_a introducido en la Ecuación 5.9, que retorna 1 si el rostro fue clasificado como asiático por el modelo entrenado y 0 en otro caso. La función de fitness modificada está dada por la Ecuación 5.14.

Figura 5.14: Ejemplo de rostro de raza blanca generado utilizando los límites de raza asiática.

$$f_a(\mathbf{z}) = \frac{1}{c_a(\text{img})} \times rc_a(\text{img}), \text{ con } img = g(\mathbf{z}, k_a) \tag{5.14}$$

Al ejecutar evaluaciones preliminares del método para encontrar ataques adversarios con la función de fitness dada por la Ecuación 5.14, a partir de la generación 30 aproximadamente, se encontraron múltiples rostros de raza negra. La Figura 5.15 muestra ejemplos de rostros de raza negra generados al utilizar los límites de raza asiática. Para mitigar este problema se utilizó la función scc_w, introducida en la Ecuación 5.10. La función de fitness modificada esta dada por la Ecuación 5.15.

Figura 5.15: Ejemplo de rostro de raza negra generado utilizando los límites de raza asiática.

$$f_a(\mathbf{z}) = \frac{1}{c_a(\text{img})} \times rc_a(\text{img}) \times scc_w(\text{img}), \text{ con } img = g(\mathbf{z}, k_a) \tag{5.15}$$

Al aplicar el método para encontrar ataques adversarios con la función de fitness dada por la Ecuación 5.15 únicamente se obtuvieron rostros de raza asiática. No obstante, se modificó la función de fitness para normalizar los valores y disminuir las fluctuaciones. Se introdujo un término

ϵ con valor 1×10^{-3} en el denominador para evitar divisiones por valores demasiado cercanos a cero, lo que estabilizó y suavizó los resultados de la función. Adicionalmente, se utilizó la función máx$(0, \cdot)$ para garantizar que los valores de fitness no fueran negativos y se incorporó un factor de escala s_f para controlar el rango de valores de fitness, que establece una cota superior de 150, debido a que el multiplicador de color scc_w puede alcanzar un valor de 1.5. Estas modificaciones aseguraron que los valores de fitness se mantuvieran dentro de un rango de 0 a 150. La función de fitness final utilizada para generar ataques de raza negra está dada por la Ecuación 5.16.

$$\begin{gathered} f_a(\mathbf{z}) = \text{máx}\left(0, s_f \times \left(\frac{\text{max_value} - (\text{cond} + \epsilon)}{\text{max_value}}\right)^3\right) \times scc_w(img) \times rc_a(img), \\ \text{con cond} = c_a(img), img = g(\mathbf{z}, k_a), \text{max_value} = 100 \text{ la cota superior}, \\ s_f = 100 \text{ el factor de escala.} \end{gathered} \tag{5.16}$$

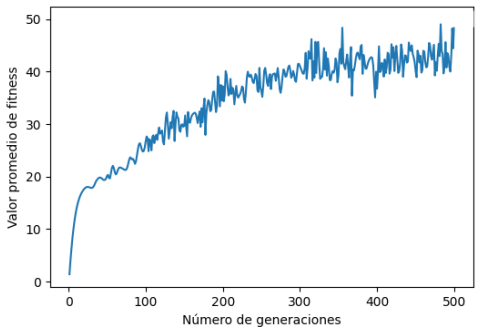
5.4.3. Evaluación del método para generar ataques adversarios con StyleGAN 2

Se evaluó el método para generar ataques adversarios con la GAN seleccionada para las razas negra, asiática y blanca. Para cada una de estas razas, se utilizaron las funciones de fitness definidas en la Sección 4.4.3.1 y los límites correspondientes. El algoritmo genético utilizado para las imágenes de rostros fue configurado con una población de 100 individuos. De cada generación se seleccionaron 55 padres para el proceso de apareamiento, con un cruce de tipo uniforme. Además, se aplicó una política de elitismo que preserva 12 de los mejores individuos de cada generación para asegurar que las soluciones más prometedoras no se pierdan durante el proceso evolutivo. El algoritmo fue configurado para evitar genes duplicados en las soluciones, lo que permitió una mayor diversidad en la población.

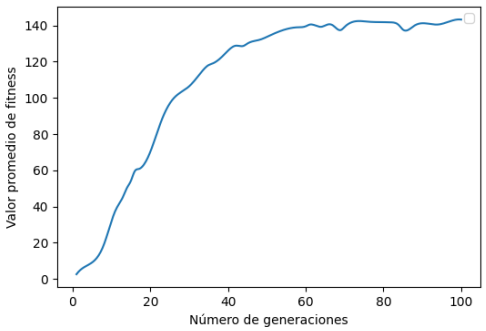
Se registró la evolución de las funciones de fitness del método para encontrar ataques adversarios para las razas blanca, negra y asiática, para determinar el criterio de parada del método. Se aplicó una función de suavizado para mitigar las fluctuaciones de los valores de fitness para cada una de las razas. En el Anexo C.7 se encuentran las gráficas con la evolución de los valores de fitness sin suavizar. Para suavizar los valores de fitness obtenidos se utilizó la función *UnivariateSpline* de SciPy (`https://docs.scipy.org/doc/scipy/reference/generated/scipy.interpolate.UnivariateSpline.html`). La Figura 5.16 presenta la evolución suavizada de los valores de fitness promedio de 5 iteraciones ejecutadas del método para encontrar ataques adversarios para cada una de las 3 razas. Para raza negra se definió un criterio de parada de 300 generaciones por iteración, ya que entre las generaciones 300 y 500 se observaron oscilaciones, pero el valor de fitness no aumentó. Para raza blanca se estableció un criterio de parada de 100 generaciones por iteración, ya que en ese punto se alcanzó un valor de fitness cercano al máximo y luego se estancó. Para la raza asiática se estableció un criterio de parada de 50 generaciones por iteración, ya que se alcanzó un valor de fitness cercano al máximo, y a partir de la generación 50, la calidad de los rostros obtenidos no fue buena en muchos casos.

Se analizaron los rostros generados que no fueron ataques. La Tabla 5.18 presenta el número de rostros bien clasificados donde las dos probabilidades más altas tuvieron una diferencia menor que cierto δ. Los resultados están organizados según diferentes valores de δ y la raza clasificada. Los valores de δ comienzan en 0,5 y disminuyen en pasos de 0,1.

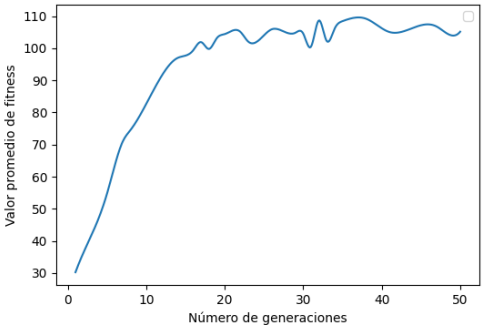
Los resultados mostraron que para la raza blanca, 13,98 % de los rostros clasificados correctamente presentó una diferencia menor a 0,2 entre la probabilidad de la raza más alta (raza blanca) y la segunda. Para los rostros de raza asiática clasificados correctamente, 6,15 % tuvo una diferencia menor a 0,2 entre la probabilidad de la raza más alta (raza asiática) y la segunda. Para los rostros de raza negra clasificados correctamente, 0,72 % tuvo una diferencia menor a 0,2 entre la probabilidad de la raza más alta (raza negra) y la segunda probabilidad más alta. Los resultados indicaron que, en los casos donde se clasificó correctamente los rostros, la raza que confundió más al clasificador fue la raza blanca, seguido por la raza asiática.



(a) Evolución de la función de fitness con rostros de raza negra



(b) Evolución de la función de fitness con rostros de raza blanca



(c) Evolución de la función de fitness con rostros de raza asiática

Figura 5.16: Evolución de las funciones de fitness de raza negra, blanca y asiática en Stylegan2-ADA-Pytorch

δ	*Clasificación*		
	Raza negra	Raza blanca	Raza asiática
$< 0{,}5$	1,36 %	30,52 %	10,25 %
$< 0{,}4$	1,17 %	25,37 %	8,86 %
$< 0{,}3$	0,97 %	19,88 %	7,57 %
$< 0{,}2$	0,72 %	13,98 %	6,15 %
$< 0{,}1$	0,40 %	7,43 %	3,76 %

Tabla 5.18: Número de rostros bien clasificados en las que las dos probabilidades más altas proporcionadas por DeepFace c_c están a una distancia menor que δ agrupados raza

5.4.3.1. Evaluación del método para generar ataques adversarios de raza negra en StyleGAN 2

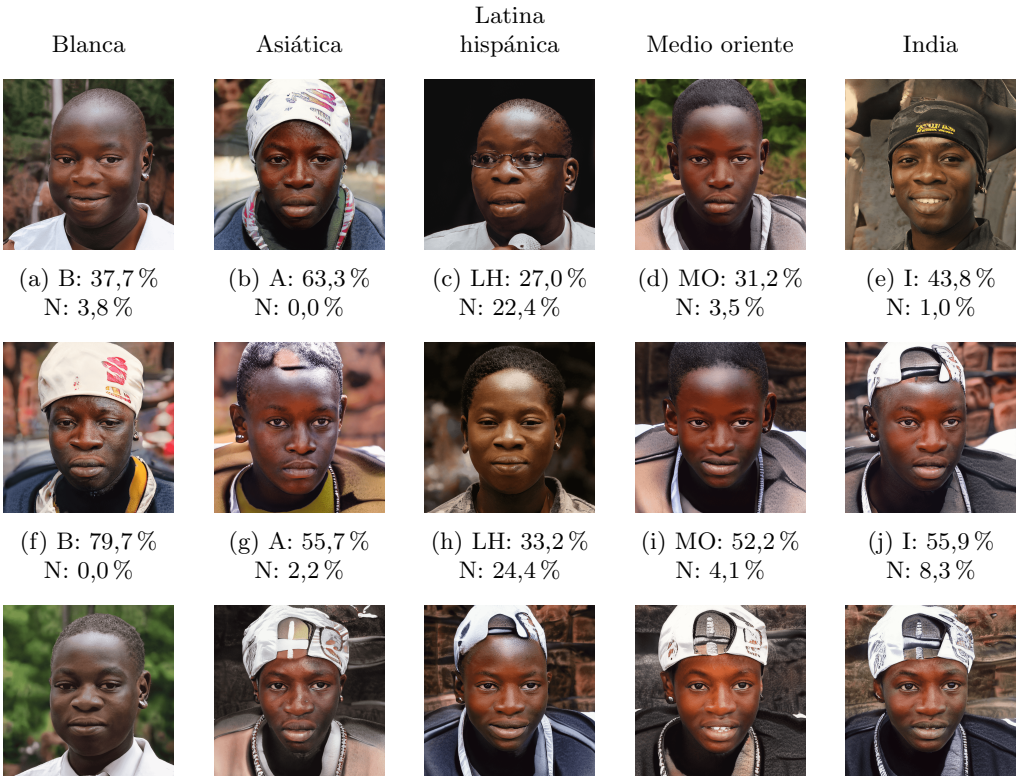
Al aplicar el método para generar ataques adversarios se generaron 220 015 rostros de raza negra, de los cuales 97,03 % pasaron el filtro de calidad definido en la función de fitness y 23,60 % fueron ataques. Se evaluaron los ataques obtenidos al aplicar el método para generar ataques adversarios de raza negra. El tiempo de ejecución fue 60 horas por cada iteración. La Tabla 5.19 muestra la distribución de las clasificaciones de los ataques a la raza negra, organizados según diferentes umbrales de probabilidad establecidos para considerar una clasificación como un ataque.

p	*Clasificación*					**Ataques**
	Blanca	**Asiática**	**Latina Hispana**	**Medio Oriente**	**India**	
> 0	84,58 %	7,08 %	0,24 %	7,45 %	0,65 %	100,0 %
$> 0,1$	84,58 %	7,08 %	0,24 %	7,45 %	0,65 %	100,0 %
$> 0,2$	84,55 %	7,06 %	0,21 %	7,43 %	0,63 %	99,88 %
$> 0,3$	82,59 %	6,49 %	0,07 %	6,52 %	0,40 %	96,07 %
$> 0,4$	78,75 %	5,60 %	0,03 %	5,16 %	0,25 %	89,79 %
$> 0,5$	73,39 %	4,38 %	0,0 %	3,61 %	0,13 %	81,51 %
$> 0,6$	67,01 %	3,08 %	0,0 %	2,14 %	0,06 %	72,29 %
$> 0,7$	60,36 %	2,15 %	0,0 %	1,20 %	0,02 %	63,73 %
$> 0,8$	52,18 %	1,46 %	0,0 %	0,59 %	0,01 %	54,22 %
$> 0,9$	40,81 %	0,78 %	0,0 %	0,18 %	0,0 %	41,77 %

Tabla 5.19: Distribución de ataques de raza negra por umbral

DeepFace confundió los rostros de raza negra que fueron ataques con todas las razas que permite clasificar. De los ataques obtenidos de raza negra, 41,77 % fueron clasificados incorrectamente como otra raza con una probabilidad superior a 0,9, de los cuales 40,81 % fueron clasificados como de raza blanca. La Figura 5.17 muestra ataques para cada una de las razas de DeepFace, agrupados por raza clasificada. El Anexo C.8 muestra más ejemplos de ataques de raza negra obtenidos.

Se evaluaron otras características además de la raza en los ataques obtenidos de raza negra. 91,6 % de los ataques fueron clasificados por Deepface como de género masculino y 8,4 % fueron clasificados como de género femenino. Se construyó un modelo clasificador basado en VGG16 y regresión logística que clasifica si un rostro tiene un accesorio en la cabeza o no, para cuantificar los ataques con esta característica. Para entrenar el modelo VGG16 se definió un conjunto de datos con rostros con accesorios en la cabeza y rostros sin accesorios en la cabeza. Todos los rostros utilizados para el entrenamiento fueron resultado de ejecuciones del método para encontrar ataques adversarios de raza negra. El modelo creado tuvo una precisión de 99 % sobre el conjunto de imágenes de evaluación. El Anexo C.5 muestra rostros pertenecientes al conjunto de datos de entrenamiento. Los resultados mostraron que 82,46 % de los ataques fueron clasificados como rostros con algún accesorio en la cabeza y 17,54 % fueron clasificados como rostros sin accesorios en la cabeza.



(a) B: 37,7 %
N: 3,8 %

(b) A: 63,3 %
N: 0,0 %

(c) LH: 27,0 %
N: 22,4 %

(d) MO: 31,2 %
N: 3,5 %

(e) I: 43,8 %
N: 1,0 %

(f) B: 79,7 %
N: 0,0 %

(g) A: 55,7 %
N: 2,2 %

(h) LH: 33,2 %
N: 24,4 %

(i) MO: 52,2 %
N: 4,1 %

(j) I: 55,9 %
N: 8,3 %

(k) B: 74,5 %
N: 0,6 %

(l) A: 71,0 %
N: 0,9 %

(m) LH: 21,7 %
N: 13,7 %

(n) MO: 45,2 %
N: 0,1 %

(ñ) I: 92,6 %
N: 6,0 %

Figura 5.17: Ataques a raza negra (N) por raza clasificada (A: Asiática, B: Blanca, LH: Latino hispano, MO: Medio oriente, I: India)

En la Figura 5.18 se presentan las representaciones t-SNE de los rostros obtenidos al aplicar el método para raza negra. Los rostros que no fueron ataques están identificados con el color azul y los que fueron ataques se identifican con distintos colores según la raza en la que el rostro que fue clasificado. Los ataques están distribuidos de manera dispersa en toda la figura, sin determinar un patrón específico, lo que podría indicar que los errores de clasificación no son sistemáticos. Se generó una representación t-SNE interactiva de tres dimensiones (https://mateogoncu.github.io/attacks/tsne_visualization_3d_black.html), en la que tampoco se encontraron patrones para identificar los ataques, si no que los ataques también se encontraron dispersos en toda la figura.

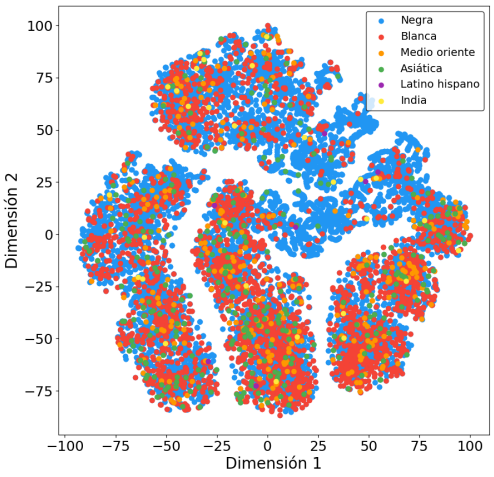


Figura 5.18: Representación TSNE de la distribución de los ataques a la raza negra

5.4.3.2. Evaluación del método para generar ataques adversarios de raza blanca en StyleGAN 2

Al aplicar el método para generar ataques adversarios se generaron 42 916 rostros de raza blanca, de los cuales 99,14 % pasaron el filtro de ser de raza blanca definido en la función de fitness utilizada y 81,91 % fueron ataques. El tiempo de ejecución fue 10 horas por iteración. La Tabla 5.20 muestra la distribución de las clasificaciones de los ataques a la raza blanca, organizados según diferentes umbrales de probabilidad establecidos para considerar una clasificación como un ataque, así como el porcentaje total de los ataques que se encuentran dentro de cada umbral.

p	*Clasificación*					**Ataques**
	Negra	**Asiática**	**Latina Hispana**	**Medio Oriente**	**India**	
> 0	0,05 %	1,19 %	0,99 %	97,74 %	0,03 %	100,0 %
$> 0,1$	0,05 %	1,19 %	0,99 %	97,74 %	0,03 %	100,0 %
$> 0,2$	0,05 %	1,19 %	0,99 %	97,74 %	0,03 %	100,0 %
$> 0,3$	0,03 %	1,07 %	0,75 %	97,60 %	0,02 %	99,47 %
$> 0,4$	0,02 %	0,89 %	0,27 %	96,35 %	0,01 %	97,54 %
$> 0,5$	0,01 %	0,70 %	0,09 %	94,16 %	0,00 %	94,96 %
$> 0,6$	0,01 %	0,60 %	0,04 %	91,57 %	0,00 %	92,22 %
$> 0,7$	0,01 %	0,54 %	0,02 %	88,82 %	0,00 %	89,49 %
$> 0,8$	0,00 %	0,49 %	0,00 %	85,36 %	0,00 %	85,85 %
$> 0,9$	0,00 %	0,43 %	0,00 %	79,52 %	0,00 %	79,95 %

Tabla 5.20: Distribución de ataques de raza blanca por umbral

DeepFace confundió los rostros de raza blanca que fueron ataques con todas las razas que permite clasificar. De los ataques obtenidos de la raza blanca, 79,95 % fueron clasificados incorrectamente como otra raza con una probabilidad superior a 0,9, de los cuales 79,52 % fueron clasificados como de raza medio oriente. La Figura 5.19 muestra ataques para cada una de las razas de DeepFace, agrupados por raza clasificada. El Anexo C.9 muestra más ejemplos de ataques de raza blanca obtenidos.

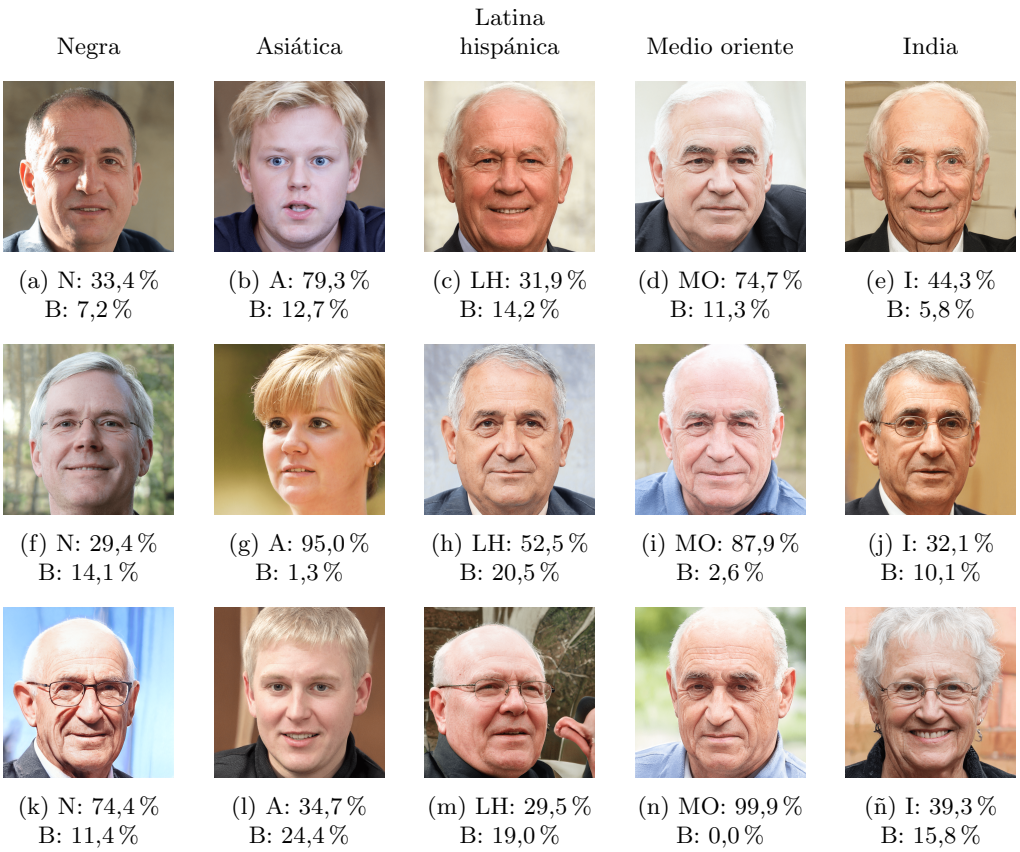


Figura 5.19: Ataques a raza blanca (B) por raza clasificada (A: Asiática, N: Negra, LH: Latino hispano, MO: Medio oriente, I: India)

Se evaluaron otras características además de la raza en los ataques de raza blanca obtenidos. 99,5 % de los ataques fueron clasificados como de género masculino y 0,5 % fueron clasificados como de género femenino. Se utilizó DeepFace para clasificar las imágenes de rostros según la edad, pero se comprobó empíricamente que los resultados no eran consistentes con los rostros analizados, por lo que se construyó un modelo clasificador basado en VGG16 y regresión logística que clasifica si un rostro es joven o de avanzada edad. Para entrenar el modelo se definió un conjunto de datos con dos categorías, rostros de avanzada edad y rostros de jóvenes y adultos. El modelo creado tuvo una precisión de 99 % sobre el conjunto de imágenes de evaluación. Todos los rostros utilizados para el entrenamiento fueron resultado de ejecuciones del método para encontrar ataques adversarios de raza blanca. El Anexo C.6 muestra rostros pertenecientes al conjunto de entrenamiento. Se clasificaron todos los ataques de raza blanca obtenidos al aplicar el método con el modelo construido, de los cuales 98,02 % fue clasificado como de edad avanzada y solamente 1,97 % fue clasificado como rostros de jóvenes o adultos.

En la Figura 5.20 se presentan las representaciones t-SNE de dos dimensiones de los rostros obtenidos al aplicar el método para la raza blanca, tanto ataques como rostros bien clasificados. Los rostros que no fueron ataques están identificados con el color rojo y los que fueron ataques se identifican con distintos colores según la raza en la que el rostro que fue clasificado. Se identifica en la figura una alta densidad de rostros que no fueron ataques (color rojo) distribuidos en la parte central del gráfico, lo que indica una clasificación correcta de DeepFace en estos casos. Los ataques clasificados como raza medio oriente (color naranja) tienden a concentrarse en las regiones periféricas del área central del gráfico, en tanto que los ataques clasificados como otras razas se concentran más cerca del centro de la figura, lo que podría indicar mayor similitud con los rostros clasificados correctamente. Se generó también una representación t-SNE interactiva de tres dimensiones(https://mateogoncu.github.io/attacks/tsne_visualization_3d_white.html) de los rostros obtenidos al aplicar el método para raza blanca.

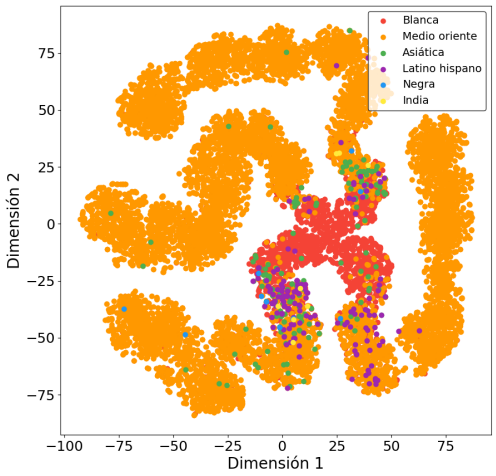


Figura 5.20: Representación tsne de la distribución de los ataques a la raza blanca.

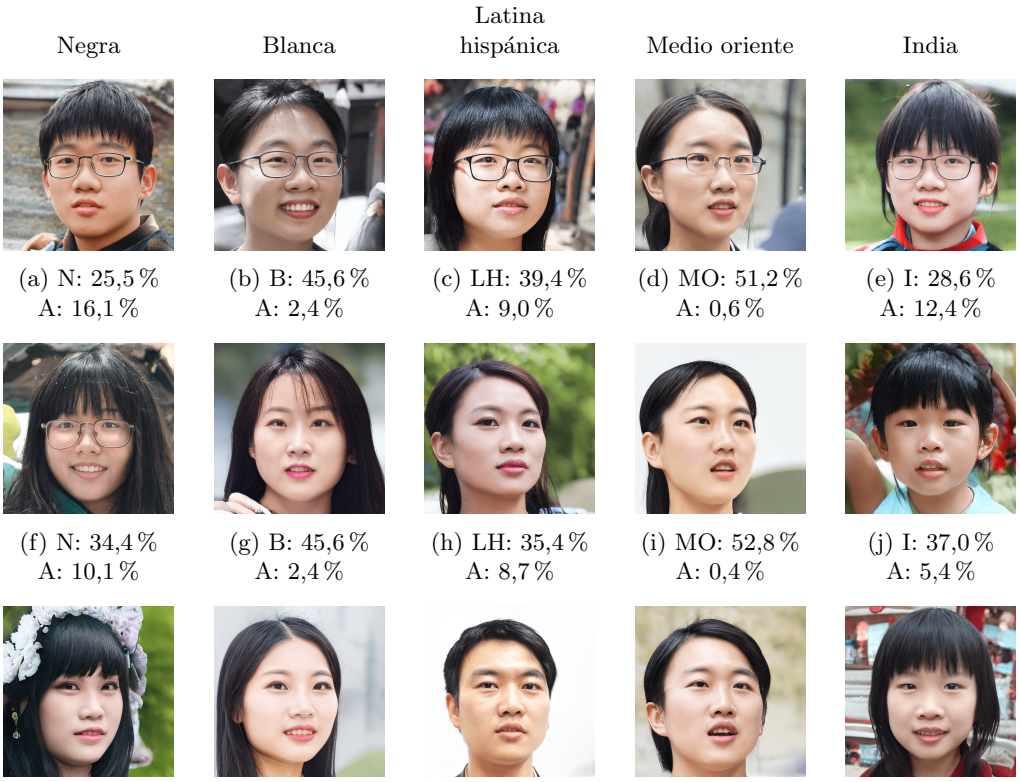
5.4.3.3. Evaluación del método para generar ataques adversarios de raza asiática en StyleGAN 2

Al aplicar el método para generar ataques adversarios se generaron 22 500 rostros de raza asiática, de los cuales 99,28 % pasaron el filtro de ser de raza asiática definido en la función de fitness utilizada y 70,96 % fueron ataques. El tiempo de ejecución fue 5 horas por iteración. La Tabla 5.21 muestra la distribución de las clasificaciones de los ataques a la raza asiática, organizados según diferentes umbrales de probabilidad establecidos para considerar una clasificación como un ataque.

DeepFace confundió los rostros que fueron ataques de raza asiática con todas las razas que permite clasificar. De los ataques a la raza asiática obtenidos, 14,39 % fueron clasificados incorrectamente como otra raza con una probabilidad superior a 0,9, de los cuales 14,29 % fueron clasificados como de raza blanca. La Figura 5.21 muestra rostros clasificados como ataques para cada una de las razas de DeepFace, agrupados por raza clasificada. El Anexo C.10 muestra más ejemplos de ataques de raza asiática obtenidos.

p	*Clasificación*					**Ataques**
	Blanca	**Negra**	**Latina Hispana**	**Medio Oriente**	**India**	
> 0	86.93 %	0.22 %	5.38 %	7.26 %	0.21 %	100.0 %
> 10	86.93 %	0.22 %	5.38 %	7.26 %	0.21 %	100.0 %
> 20	86.93 %	0.22 %	5.38 %	7.26 %	0.21 %	100.0 %
> 30	85.93 %	0.19 %	4.09 %	6.87 %	0.18 %	97.26 %
> 40	75.92 %	0.15 %	0.82 %	4.11 %	0.11 %	81.11 %
> 50	58.62 %	0.14 %	0.09 %	1.31 %	0.05 %	60.21 %
> 60	42.86 %	0.12 %	0.01 %	0.22 %	0.02 %	43.23 %
> 70	30.61 %	0.11 %	0.00 %	0.05 %	0.01 %	30.78 %
> 80	21.54 %	0.09 %	0.00 %	0.01 %	0.01 %	21.65 %
> 90	14.29 %	0.09 %	0.00 %	0.00 %	0.01 %	14.39 %

Tabla 5.21: Distribución de ataques de raza asiática por umbral



(a) N: 25,5 %
A: 16,1 %

(b) B: 45,6 %
A: 2,4 %

(c) LH: 39,4 %
A: 9,0 %

(d) MO: 51,2 %
A: 0,6 %

(e) I: 28,6 %
A: 12,4 %

(f) N: 34,4 %
A: 10,1 %

(g) B: 45,6 %
A: 2,4 %

(h) LH: 35,4 %
A: 8,7 %

(i) MO: 52,8 %
A: 0,4 %

(j) I: 37,0 %
A: 5,4 %

(k) N: 36,8 %
A: 8,3 %

(l) B: 59,8 %
A: 1,0 %

(m) LH: 49,3 %
A: 44,5 %

(n) MO: 46,0 %
A: 0,8 %

(ñ) I: 25,6 %
A: 20,5 %

Figura 5.21: Ataques a raza asiática (A) por raza clasificada (B: Blanca, N: Negra, LH: Latino hispano, MO: Medio oriente, I: India)

Se evaluaron otras características además de la raza en los ataques de raza asiática obtenidos. 42,3 % de los ataques fueron clasificados por DeepFace como de género masculino y 57,7 % fueron clasificados como de género femenino. Se identificó que parte de los ataques obtenidos poseían algún tipo gafas, por lo que se construyó un modelo clasificador basado en VGG16 y regresión logística que clasifica si un rostro posee algún tipo de gafas para cuantificar la cantidad de ataques con esta característica. Para entrenar este modelo se definió un conjunto de datos con dos categorías, rostros con gafas y rostros sin gafas. Todos los rostros utilizados para el entrenamiento fueron resultado de ejecuciones del método para encontrar ataques adversarios de raza asiática. El Anexo C.4 muestra rostros pertenecientes al conjunto de entrenamiento. Se clasificaron todos los ataques de raza asiática obtenidos al aplicar el método con el modelo construido, de los cuales 83,24 % fueron clasificados como rostros con gafas y 16,75 % fueron clasificados como rostros sin gafas.

En la Figura 5.22 se presentan las representaciones t-SNE de los rostros obtenidos para la raza asiática, tanto ataques como rostros bien clasificados. Los rostros que no fueron ataques están identificados con el color verde y los que fueron ataques se identifican con distintos colores según la raza en la que el rostro fue clasificado. Se identifica en la figura una alta densidad de rostros que no fueron ataques distribuidos en la parte central del gráfico, lo que indica una clasificación correcta de DeepFace en estos casos. Los ataques clasificados como raza Latina Hispánica están concentrados en la parte inferior derecha del gráfico. Los ataques clasificados como raza Medio Oriente están concentrados en la parte superior y en la inferior izquierda. Los ataques al resto de las razas están dispersos en el gráfico, con una alta densidad de ataques clasificados como raza blanca. Esto sugiere que DeepFace tiene mayor dificultad para distinguir ciertas características faciales asiáticas de otras razas.

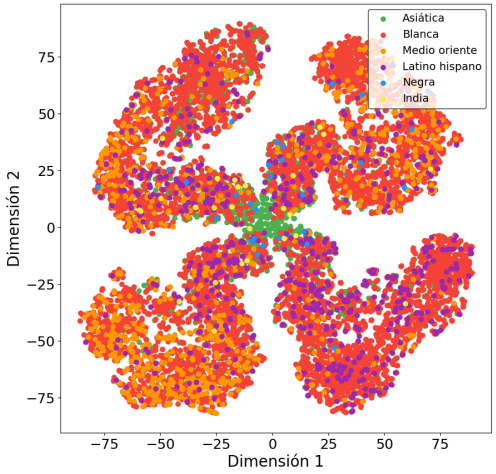


Figura 5.22: Representación TSNE de la distribución de los ataques a la raza asiática.

Capítulo 6

Conclusiones y trabajo futuro

En este capítulo se presentan las conclusiones referentes al proyecto en general, los resultados alcanzados, dificultades encontradas y se plantean mejoras y trabajos a futuro.

6.1. Conclusiones

El método propuesto para generar ataques adversarios ha demostrado ser efectivo contra ciertos clasificadores y categorías de imágenes y ha producido resultados diversos.

Para el caso de los dígitos manuscritos de MNIST, se aplicó el método para generar ataques adversarios de manera exitosa contra dos clasificadores diferentes. Para ciertos dígitos, por ejemplo el 9, los ataques de calidad fueron escasos, lo que indicó una mayor resistencia de los clasificadores ante ciertos dígitos. Se evaluó el método utilizando tres funciones de fitness. La función f_2 alcanzó una tasa de éxito de 35 %, siendo la función con la tasa de éxito más alta. En contraste, la función f_3 logró una tasa de éxito de 16 %, y en algunos casos no se generaron ataques exitosos contra ninguno de los clasificadores, lo que sugiere que el método propuesto no es lo suficientemente robusto para engañar a ambos clasificadores en todos los dígitos. En el análisis a detalle de los dígitos seleccionados, se observó que los clasificadores confundieron mayoritariamente el dígito 3 con el 9, y el 9 con el 7.

Para el caso del conjunto de datos CIFAR-10, se aplicó el método para generar ataques adversarios de manera exitosa contra dos clasificadores diferentes. Los resultados indicaron una mayor vulnerabilidad de los clasificadores de imágenes de CIFAR-10 en comparación con MNIST, con una mayor cantidad de ataques de alta calidad obtenidos. La tasa de éxito del método fue mayor a 58 % al utilizar cada una de las tres funciones de fitness evaluadas. El contraste entre los ataques a dígitos manuscritos e imágenes de objetos puede atribuirse a la mayor complejidad de las imágenes de objetos en CIFAR-10, así como a la baja resolución de las imágenes de este conjunto, lo que las hace más propensas a errores de clasificación. Por otra parte, la baja resolución de las imágenes dificultó su identificación visual por parte del ojo humano, lo que complicó el análisis y la interpretación precisa de los resultados obtenidos. En el análisis a detalle de las imágenes de objetos seleccionadas, se encontró que los clasificadores confundieron el objeto avión con un ave o un gato en más del 60 % de los ataques y al objeto camión con un avión, auto o gato en más del 80 % de los ataques.

Para el caso de los ataques de imágenes de rostros generados contra Deepface se presentaron desafíos adicionales debido a la complejidad de las imágenes faciales y las dificultades inherentes al uso de este tipo de redes generativas. Los ataques fueron creados utilizando StyleGAN2-ADA-PyTorch y ciertos límites que permitieron generar rostros con determinada característica de manera condicional. Sin embargo, el método para generar ataques adversarios inicialmente generaba rostros con características no deseadas al pasar de las generaciones, por ejemplo cambios de raza o anomalías faciales, por lo que se realizaron ajustes significativos en la función de fitness según la raza a atacar y se entrenaron modelos adicionales para mejorar los resultados. Finalmente, se logra-

ron generar ataques adversarios para las razas blanca, negra y asiática. Los resultados mostraron variaciones significativas según la raza objetivo del ataque. Deepface tendió a confundir rostros de raza negra con rostros de raza blanca y los ataques generados fueron en su mayoría personas de género masculino con accesorios en la cabeza. Los rostros de raza blanca fueron comúnmente confundidos con rostros del medio oriente y la mayoría de los ataques fueron hombres de edad avanzada. Los rostros de raza asiática fueron confundidos con rostros de raza blanca y el uso de gafas fue una característica notable en la mayoría de los ataques, con una distribución equitativa entre géneros. Las variaciones en los ataques generados para cada raza podrían estar influenciadas no solo por sesgos en el conjunto de entrenamiento de StyleGAN, sino también por las características y limitaciones propias del clasificador DeepFace, lo que resalta la importancia de considerar tanto los sesgos inherentes en los conjuntos de datos utilizados para entrenar modelos de inteligencia artificial como las particularidades de los clasificadores empleados en estos sistemas.

Los resultados obtenidos demostraron la viabilidad del método propuesto para generar ataques adversarios en diferentes contextos y resaltan la necesidad de realizar ajustes específicos según los tipos de datos y la arquitectura del modelo. Las diferencias observadas en la vulnerabilidad de los clasificadores destacan la importancia de continuar investigando y desarrollando técnicas para mejorar la robustez de los modelos frente a ataques adversarios, considerando además los posibles sesgos que puedan afectar la equidad y fiabilidad de las aplicaciones.

6.2. Trabajo futuro

Los resultados obtenidos en este proyecto proporcionan una base para extender y mejorar los métodos de generación de ataques adversarios, así como para desarrollar nuevas estrategias de defensa más robustas en clasificadores de imágenes.

Una línea de desarrollo interesante sería ampliar el enfoque de los ataques adversarios generados con StyleGAN para incluir una gama más amplia de atributos y características, por ejemplo expresiones faciales, emociones u otras razas. La ampliación de este enfoque permitiría evaluar la eficacia del método en contextos más diversos y complejos, lo que ofrecería una visión más completa de las vulnerabilidades de los clasificadores de imágenes.

Otro aspecto relevante sería evaluar los ataques adversarios contra otros clasificadores de imágenes de rostros para comprobar su eficacia y transferencia en diferentes modelos, lo que permitiría determinar si los ataques adversarios generados son específicos de los clasificadores evaluados o si pueden generalizarse a otros sistemas de reconocimiento de imágenes.

En este proyecto los ataques generados con imágenes de rostros presentaron sesgos relacionados con la raza debido a que el conjunto de datos de entrenamiento utilizado estaba desbalanceado. Otra línea de desarrollo interesante sería trabajar con un conjunto de imágenes balanceado en la generación de imágenes de rostros. Ajustar los conjuntos de entrenamiento de manera que representen equitativamente todas las clases y características relevantes ayudaría a mejorar la equidad y la efectividad de los ataques adversarios.

En este proyecto se utilizaron límites en StyleGAN para generar rostros con características específicas. Si bien el enfoque demostró ser efectivo luego de aplicar ciertas optimizaciones, no se alcanzó el nivel de precisión de una GAN completamente condicional. Dado que no se encontró una GAN condicional de imágenes de rostros, una línea de investigación futura sería desarrollar una GAN con estas características, lo que garantizaría un funcionamiento más eficiente y un rendimiento más consistente al aplicar el método para generar ataques adversarios.

Finalmente, una extensión interesante sería desarrollar y evaluar defensas específicas para los clasificadores de imágenes luego de procesar los nuevos ataques adversarios generados para asegurar un mayor nivel de seguridad y eficacia en aplicaciones prácticas. Las nuevas defensas podrían involucrar la implementación de técnicas de regularización, detección de anomalías en las entradas generadas, re-entrenamiento de los datos y robustecimiento basado en aprendizaje profundo.

Referencias

M. Alzantot, Y. Sharma, S. Chakraborty, H. Zhang, C. Hsieh, and M. Srivastava. GenAttack: practical black-box attacks with gradient-free optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference*, páginas 1111–1119, 2019.

A. Bose. Handwritten Digit Recognition Using PyTorch: Intro to Neural Networks. https://towardsdatascience.com/handwritten-digit-mnist-pytorch-977b5338e627, 2019. Accedido: 2023-09-08.

A. Chakraborty, M. Alam, V. Dey, A. Chattopadhyay, and D. Mukhopadhyay. A survey on adversarial attacks and defences. *CAAI Transactions on Intelligence Technology*, 2021.

P. Chen, H. Zhang, Y. Sharma, J. Yi, and C. Hsieh. ZOO: Zeroth Order Optimization Based Black-box Attacks to Deep Neural Networks without Training Substitute Models. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, páginas 15–26, 2017.

S. Chen, C. Li, and H. Lin. A Unified View of cGANs with and without Classifiers. In *Advances in Neural Information Processing Systems: Proceedings of the 2021 Conference*, páginas 27566–27579, 2021.

L. Clare and J. Correia. Generating Adversarial Examples through Latent Space Exploration of Generative Adversarial Networks. In *Proceedings of the Companion Conference on Genetic and Evolutionary Computation*, páginas 1760–1767, 2023.

C. Cortes and V. Vapnik. Support-vector networks. *Machine learning*, 1995.

L. Darlow, E. Crowley, A. Antoniou, and A. Storkey. CINIC-10 Is Not ImageNet or CIFAR-10. Dataset, University of Edinburgh, 2018. Accedido: 20/11/2023.

J. Deng, W. Dong, R. Socher, L. Li, K. Li, and F. Li. ImageNet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition*, páginas 248–255, 2009.

X. Ding, X. Zhang, N. Ma, J. Han, G. Ding, and J. Sun. RepVGG: Making VGG-style ConvNets Great Again. In *IEEE/Computer Vision Foundation Conference on Computer Vision and Pattern Recognition* , páginas 13728–13737, 2021.

R. Garcia, L. Wandzik, L. Grabner, and J. Krueger. The Harms of Demographic Bias in Deep Face Recognition Research. In *International Conference on Biometrics*, páginas 1–6, 2019.

J. García, E. Hontoria, and D. Aleksovski. El problema del viajante de comercio: Búsqueda de soluciones y herramientas asequibles. *Revista Electrónica de Comunicaciones y Trabajos de Asociación Española de Profesores Universitarios de Matemáticas para la Economía y la Empresa*, 2015.

I. Goodfellow, J. Shlens, and C. Szegedy. Explaining and Harnessing Adversarial Examples. In *3rd International Conference on Learning Representations*, páginas 1–9, 2015.

I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and Aaron Courville. Improved Training of Wasserstein GANs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, páginas 5769–5779, 2017.

M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, G. Klambauer, and S. Hochreiter. GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium. In *Advances in Neural Information Processing Systems*, páginas 1–9, 2017.

G. Huang, M. Mattar, H. Lee, and E. Learned-Miller. Learning to align from scratch. In *Proceedings of the 25th International Conference on Neural Information Processing Systems*, páginas 764–772, 2012.

J. Uesato and B. O'Donoghue and P. Kohli and A. Van den Oord . Adversarial Risk and the Dangers of Evaluating Against Weak Attacks. In *Proceedings of the 35th International Conference on Machine Learning*, páginas 5025–5034, 2018.

T. Karras, S. Laine, and T. Aila. A Style-Based Generator Architecture for Generative Adversarial Networks. In *IEEE/Computer Vision Foundation Conference on Computer Vision and Pattern Recognition*, páginas 4396–4405, 2019.

T. Karras, M. Aittala, J. Hellsten, S. Laine, J. Lehtinen, and T. Aila. Training Generative Adversarial Networks with Limited Data. In *Advances in Neural Information Processing Systems*, páginas 12104–12114, 2020.

D. Li. The MNIST Database of Handwritten Digit Images for Machine Learning Research. *IEEE Signal Processing Magazine*, 2012.

Z. Liu, W. Peng, J. Zhou, Z. Wu, J. Zhang, and Y. Zhang. MI-FGSM on Faster R-CNN Object Detector. In *Proceedings of the 2020 4th International Conference on Video and Image Processing*, páginas 27–32, 2021.

A. Mackiewicz and W. Ratajczak. Principal Components Analysis. *Computers and Geosciences*, 1993.

L. Maia, P. Mêda, and J. Freitas. BIM Methodology, a New Approach - Case Study of Structural Elements Creation. *Procedia Engineering*, 2015.

V. Maluleke, N. Thakkar, T. Brooks, E. Weber, T. Darrell, A. Efros, A. Kanazawa, and D. Guillory. Studying Bias in GANs Through the Lens of Race. In *Computer Vision – European Conference on Computer Vision*, páginas 344–360, 2022.

V. Mandhana. Conditional-GAN-CIFAR-10. https://github.com/mvedang/Conditional-GAN-CIFAR-10, 2018. Accedido: 2023-11-15.

D. Michie, D. Spiegelhalter, and C. Taylor. Machine Learning, Neural and Statistical Classification. *Technometrics*, 1999.

M. Mirza and S. Osindero. Conditional Generative Adversarial Nets. *Computing Research Repository*, 2014.

M. Mitchell. *An Introduction to Genetic Algorithms*. The MIT Press, 1996.

N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. Celik, and A. Swami. Practical Black-Box Attacks against Machine Learning. In *Proceedings of the 2017 ACM on Conference on Computer and Communications Security*, página 506–519, 2017.

R. Alejandro R. Piña and M. Chenglong. Classification Algorithm for Skin Color (CASCo): A new tool to measure skin color in social science research. *Social Science Quarterly*, 2021.

A. Rai, C. Ducher, and J. Cooperstock. Improved Attribute Manipulation in the Latent Space of StyleGAN for Semantic Face Editing. In *20th IEEE International Conference on Machine Learning and Applications*, páginas 38–43, 2021.

S. Serengil and A. Ozpinar. HyperExtended LightFace: A Facial Attribute Analysis Framework. In *International Conference on Engineering and Emerging Technologies*, páginas 1–4, 2021.

Y. Shen, J. Gu, X. Tang, and B. Zhou. Interpreting the Latent Space of GANs for Semantic Face Editing. In *IEEE/Computer Vision Foundation Conference on Computer Vision and Pattern Recognition*, páginas 9240–9249, 2020.

K. Simonyan and A. Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *3rd International Conference on Learning Representations*, páginas 1–14, 2015.

A. Sorokin and D. Forsyth. Utility data annotation with Amazon Mechanical Turk. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, páginas 1–8, 2008.

M. Su, J. Chen, S. Shen, H. Xiong, and H. Zheng. POBA-GA: Perturbation optimized black-box adversarial attacks via genetic algorithm. *Computers and Security*, 2019.

R. Szeliski. *Computer Vision: Algorithms and Applications*. Springer, 2011.

F. Tramèr, A. Kurakin, N. Papernot, D. Boneh, and P. McDaniel. The Space of Transferable Adversarial Examples. In *Proceedings of the 29th Advanced Computing Systems Association Security Symposium*, páginas 1539–1556, 2017.

C. Tu, P. Ting, P. Chen, S. Liu, H. Zhang, J. Yi, C. Hsieh, and S. Cheng. AutoZOOM: Autoencoder-based Zeroth Order Optimization Method for Attacking Black-box Neural Networks. In *Proceedings of the Association for the Advancement of Artificial Intelligence Conference on Artificial Intelligence*, páginas 742–749, 2019.

F. Vakhshiteh, A. Nickabadi, and R. Ramachandra. Adversarial Attacks Against Face Recognition: A Comprehensive Study. *IEEE Access*, 2021.

C. Wu, W. Luo, N. Zhou, P. Xu, and T. Zhu. Genetic Algorithm with Multiple Fitness Functions for Generating Adversarial Examples. In *Congress on Evolutionary Computation*, páginas 1792–1799, 2021.

Z. Zhao, X. Wang, C. Zhang, N. Bai, and X. Hu. SE-ResNet56: Robust Network Model for Deepfake Detection. In *Digital Forensics and Watermarking: 21st International Workshop*, páginas 37–52, 2023.

Anexo A

Dígitos manuscritos

Este Anexo presenta información extra sobre la generación de ataques adversarios a dígitos manuscritos.

A.1. Ejemplos de ataques adversarios con dígitos manuscritos

Representación										
Clasificación	9	9	9	9	9	9	8	9	9	9
Probabilidad(p)	0,99	0,99	0,99	0,99	0,99	0,99	0,99	0,99	0,99	0,99

Tabla A.1: Ejemplos de ataques al clasificador c_m con el dígito 0

Representación										
Clasificación	6	6	6	6	6	6	6	6	6	6
Probabilidad(p)	0,97	0,97	0,97	0,97	0,98	0,98	0,98	0,98	0,98	0,99

Tabla A.2: Ejemplos de ataques al clasificador c_m con el dígito 1

Representación										
Clasificación	9	9	9	9	9	9	8	8	9	9
Probabilidad(p)	0,99	0,99	0,99	0,99	0,99	0,99	0,99	0,99	0,99	0,99

Tabla A.3: Ejemplos de ataques al clasificador c_m con el dígito 2

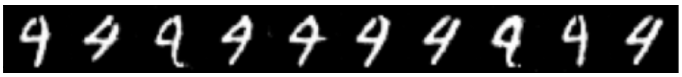
Representación										
Clasificación	9	9	9	9	9	9	9	9	9	9
Probabilidad(p)	0,99	0,99	0,99	0,99	0,99	0,99	0,99	0,99	0,99	0,99

Tabla A.4: Ejemplos de ataques al clasificador c_m con el dígito 4

Representación										
Clasificación	6	6	6	6	6	6	6	6	6	6
Probabilidad(p)	0,99	0,99	0,99	0,99	0,99	0,99	0,99	0,99	0,99	0,99

Tabla A.5: Ejemplos de ataques al clasificador c_m con el dígito 5

Representación										
Clasificación	5	5	5	5	5	5	5	5	5	5
Probabilidad(p)	0,91	0,92	0,93	0,93	0,93	0,94	0,94	0,94	0,95	0,95

Tabla A.6: Ejemplos de ataques al clasificador c_m con el dígito 6

Representación										
Clasificación	9	9	9	9	9	9	9	9	9	9
Probabilidad(p)	0,99	0,99	0,99	0,99	0,99	0,99	0,99	0,99	0,99	0,99

Tabla A.7: Ejemplos de ataques al clasificador c_m con el dígito 7

Representación										
Clasificación	9	9	9	9	0	0	3	0	9	0
Probabilidad(p)	0,99	0,99	0,99	0,99	0,99	0,99	0,99	0,99	0,99	0,99

Tabla A.8: Ejemplos de ataques al clasificador c_m con el dígito 8

A.2. Representaciones t-SNE de ataques adversarios para MNIST

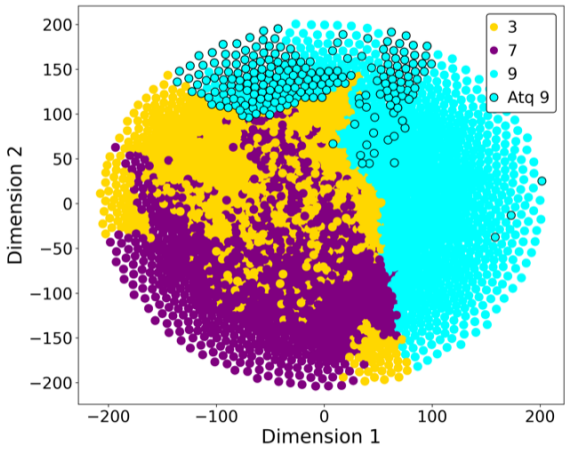


Figura A.1: Representaciones t-SNE de los Ataques al dígito 9 (Atq 9) según la función de fitness f_1

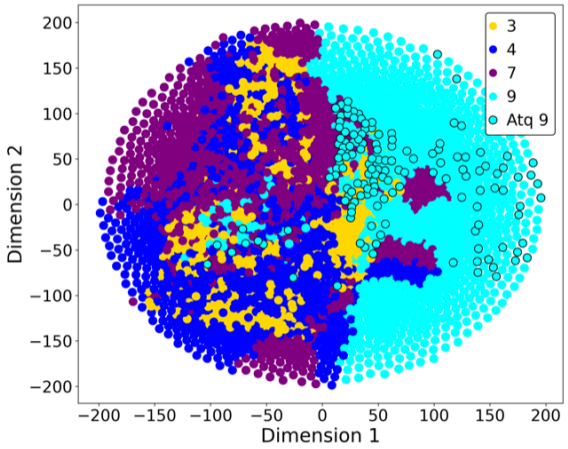


Figura A.2: Representaciones t-SNE de los Ataques al dígito 9 (Atq 9) según la función de fitness f_2

Anexo B

Imágenes de objetos

Este Anexo presenta información extra sobre la generación de ataques adversarios a imágenes de objetos.

B.1. Ejemplos de ataques adversarios con imágenes de objetos

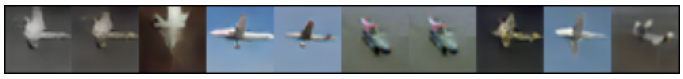
Representación										
Clasificación	gato	rana	ave	ave	ave	ave	ave	gato	ave	gato
Probabilidad(p)	0,99	0,99	0,99	0,99	0,99	0,99	0,99	0,99	0,99	0,99

Tabla B.1: Ejemplos de ataques al clasificador c_c con la clase avión

Representación										
Clasificación	gato	rana	camión	oveja	camión	perro	gato	perro	oveja	perro
Probabilidad(p)	0,98	0,98	0,99	0,99	0,99	0,99	0,99	0,99	0,99	0,99

Tabla B.2: Ejemplos de ataques al clasificador c_c con la clase auto

Representación										
Clasificación	avión	ciervo	rana	avión	ciervo	ciervo	gato	gato	rana	ciervo
Probabilidad(p)	0,99	0,99	0,99	0,99	0,99	0,99	0,99	0,99	0,99	0,99

Tabla B.3: Ejemplos de ataques al clasificador c_c con la clase ave

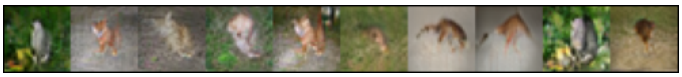
Representación										
Clasificación	rana	rana	rana	perro	ciervo	ciervo	ciervo	ciervo	rana	ave
Probabilidad(p)	0,99	0,99	0,99	0,99	0,99	0,99	0,99	0,99	0,99	0,99

Tabla B.4: Ejemplos de ataques al clasificador c_c con la clase gato

Representación										
Clasificación	ave	ave	ave	gato	ave	rana	ave	ave	ave	ave
Probabilidad(p)	0,99	0,99	0,99	0,99	0,99	0,99	0,99	0,99	0,99	0,99

Tabla B.5: Ejemplos de ataques al clasificador c_c con la clase ciervo

Representación										
Clasificación	gato	rana	ave	gato	gato	gato	ciervo	rana	rana	gato
Probabilidad(p)	0,99	0,99	0,99	0,99	0,99	0,99	0,99	0,99	0,99	0,99

Tabla B.6: Ejemplos de ataques al clasificador c_c con la clase perro

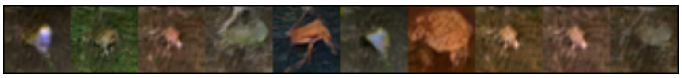
Representación										
Clasificación	ave	ciervo	ciervo	gato	gato	gato	gato	ciervo	perro	gato
Probabilidad(p)	0,99	0,99	0,98	0,99	0,97	0,98	0,97	0,97	0,99	0,97

Tabla B.7: Ejemplos de ataques al clasificador c_c con la clase rana

Representación										
Clasificación	ave	perro	gato	perro	ciervo	ciervo	ave	gato	perro	ave
Probabilidad(p)	0,99	0,99	0,99	0,99	0,99	0,99	0,99	0,99	0,99	0,99

Tabla B.8: Ejemplos de ataques al clasificador c_c con la clase caballo

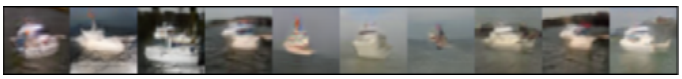
Representación										
Clasificación	gato	gato	gato	gato	ave	rana	ave	gato	perro	perro
Probabilidad(p)	0,99	0,99	0,99	0,99	0,99	0,99	0,99	0,99	0,99	0,99

Tabla B.9: Ejemplos de ataques al clasificador c_c con la clase barco

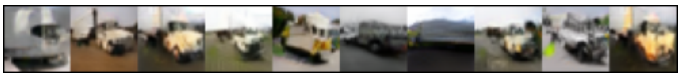
Representación										
Clasificación	auto	avión	ciervo	avión	avión	gato	oveja	oveja	auto	oveja
Probabilidad(p)	0,99	0,98	0,99	0,99	0,98	0,99	0,98	0,99	0,99	0,98

Tabla B.10: Ejemplos de ataques al clasificador c_c con la clase camión

B.2. Representación t-SNE de ataques adversarios para CIFAR-10

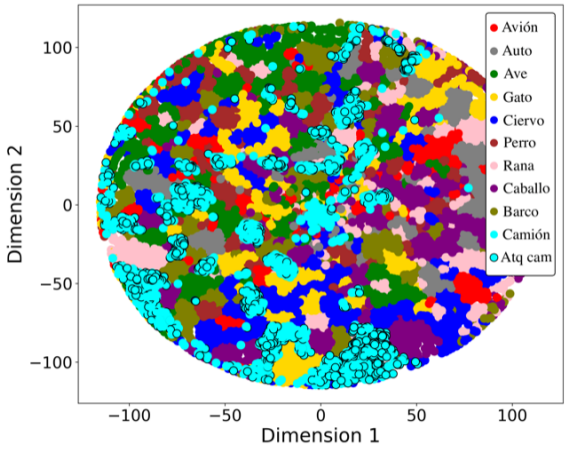


Figura B.1: Representaciones t-SNE de los Ataques al objeto camión (Atq cam) según la función de fitness f_1 junto con los objetos que generaron confusión Función de fitness

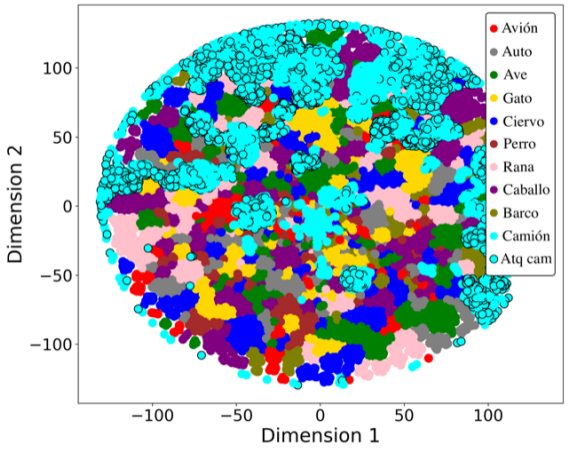


Figura B.2: Representaciones t-SNE de los Ataques al objeto camión (Atq cam) según la función de fitness f_2 junto con los objetos que generaron confusión Función de fitness

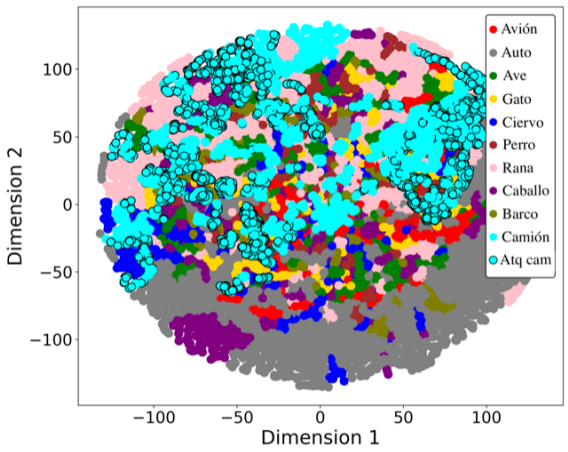


Figura B.3: Representaciones t-SNE de los Ataques al objeto camión (Atq cam) según la función de fitness f_3 junto con los objetos que generaron confusión Función de fitness

Anexo C

Imágenes de rostros

Este Anexo presenta información extra sobre la generación de ataques adversarios a imágenes de rostros.

C.1. Datos de entrenamiento para rostros de raza negra con anomalías

C.1.1. Rostros de raza negra sin anomalías

Figura C.1: Rostros de raza negra sin anomalías utilizados para entrenar un clasificador basado en el modelo VGG16

C.1.2. Rostros de raza negra con anomalías

Figura C.2: Rostros de raza negra con anomalías utilizados para entrenar un clasificador basado en el modelo VGG16

C.2. Datos de entrenamiento para modelo clasificador de rostros de raza negra

C.2.1. Rostros de raza negra generados utilizando límite de raza negra

Figura C.3: Rostros de raza negra generados utilizando límite de raza negra

C.2.2. Rostros de otras razas generados utilizando límite de raza negra

Figura C.4: Rostros de otra raza generados utilizando límite de raza negra

C.3. Datos de entrenamiento para modelo clasificador de rostros de raza asiática

C.3.1. Rostros de raza asiática generados utilizando límite de raza asiática

Figura C.5: Rostros de raza asiática generados utilizando límite de raza asiática

C.3.2. Rostros de otras razas generados utilizando límite de raza asiática

Figura C.6: Rostros de otras razas generados utilizando límite de raza asiática

C.4. Datos de entrenamiento para modelo clasificador de rostros con gafas

C.4.1. Rostros de raza asiática con gafas generados utilizando límite de raza asiática

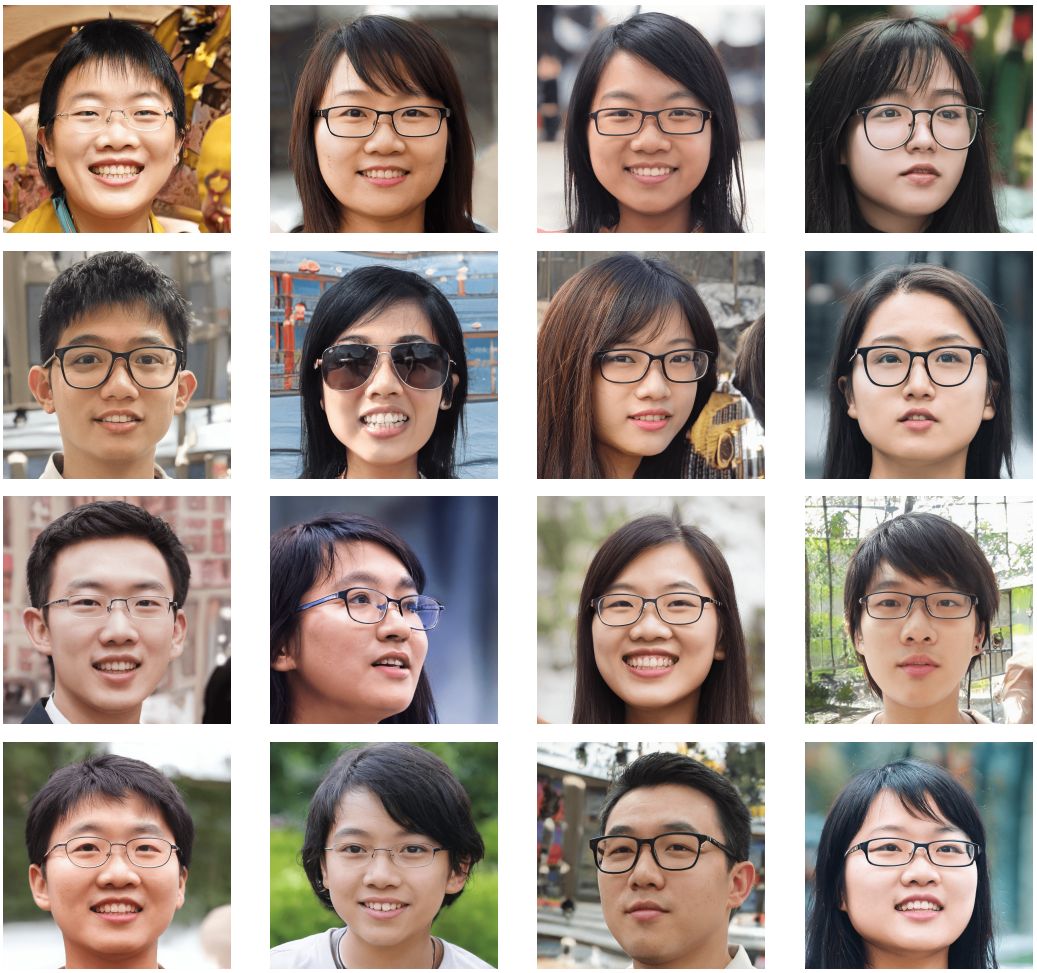


Figura C.7: Rostros de raza asiática con gafas generados utilizando límite de raza asiática

C.4.2. Rostros de raza asiática sin gafas generados utilizando límite de raza asiática

Figura C.8: Rostros de raza asiática sin gafas generados utilizando límite de raza asiática

C.5. Datos de entrenamiento para modelo clasificador de rostros con objetos en la cabeza

C.5.1. Rostros de raza negra con objetos en la cabeza generados utilizando límite de raza negra

Figura C.9: Rostros de raza negra con objetos en la cabeza generados utilizando límite de raza negra

C.5.2. Rostros de raza negra sin objetos en la cabeza generados utilizando límite de raza negra

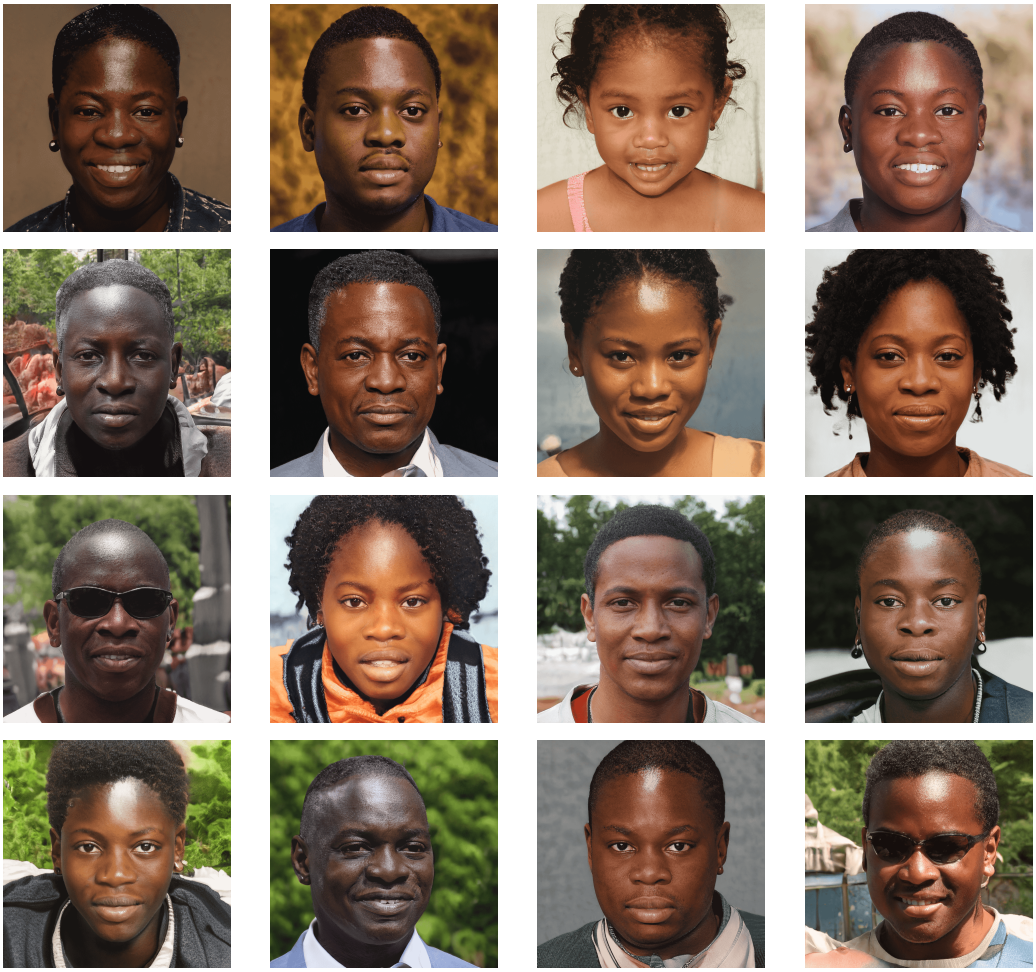


Figura C.10: Rostros de raza negra sin objetos en la cabeza generados utilizando límite de raza negra

C.6. Datos de entrenamiento para modelo clasificador de rostros de edad avanzada

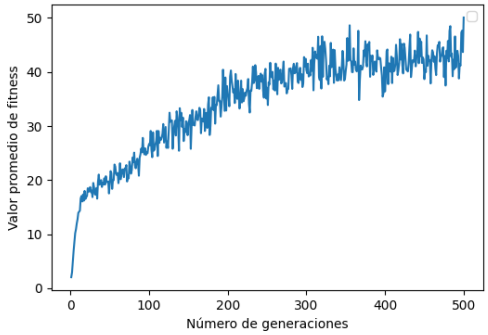
C.6.1. Rostros de raza blanca de edad avanzada generados utilizando límite de raza blanca

Figura C.11: Rostros de raza blanca de edad avanzada generados utilizando límite de raza blanca

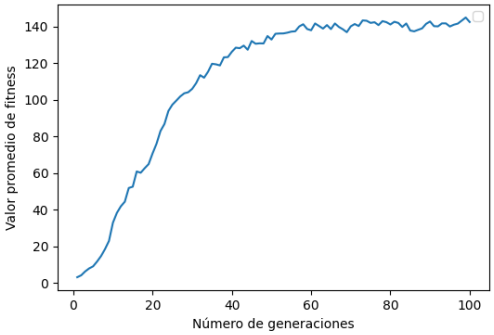
C.6.2. Rostros de raza blanca jóvenes generados utilizando límite de raza blanca

Figura C.12: Rostros de raza blanca jóvenes generados utilizando límite de raza blanca

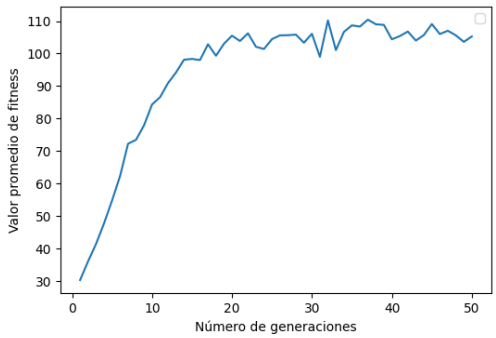
C.7. Evolución de las funciones de fitness para las razas asiática, negra y blanca.



(a) Evolución sin suavizar de la función de fitness para la raza negra.



(b) Evolución sin suavizar de la función de fitness para la raza blanca.



(c) Evolución sin suavizar de la función de fitness para la raza asiática.

Figura C.13: Evoluciones sin suavizar de la función de fitness para las razas blanca, asiática y negra.

C.8. Ejemplos de ataques adversarios de raza negra

(a) B: 97,0 %
N: 0,0 %

(b) B: 73,9 %
N: 0,0 %

(c) B: 99,3 %
N: 0,0 %

(d) B: 96,5 %
N: 2,3 %

(e) B: 54,3 %
N: 0,3 %

(f) B: 64,9 %
N: 0,0 %

(g) B: 84,6 %
N: 0,0 %

(h) B: 50,8 %
N: 0,0 %

(i) B: 99,5 %
N: 1,1 %

(j) B: 78,0 %
N: 0,4 %

(k) B: 89,8 %
N: 0,0 %

(l) B: 63,2 %
N: 0,1 %

(m) B: 70,6 %
N: 0,0 %

(n) B: 65,5 %
N: 0,1 %

(ñ) B: 86,0 %
N: 0,0 %

(o) B: 75,7 %
N: 0,2 %

(p) B: 58,6 %
N: 0,4 %

(q) B: 94,1 %
N: 0,0 %

(r) B: 49,0 %
N: 10,6 %

(s) B: 78,8 %
N: 1,8 %

(t) B: 58,9 %
N: 0,0 %

(u) B: 27,9 %
N: 10,8 %

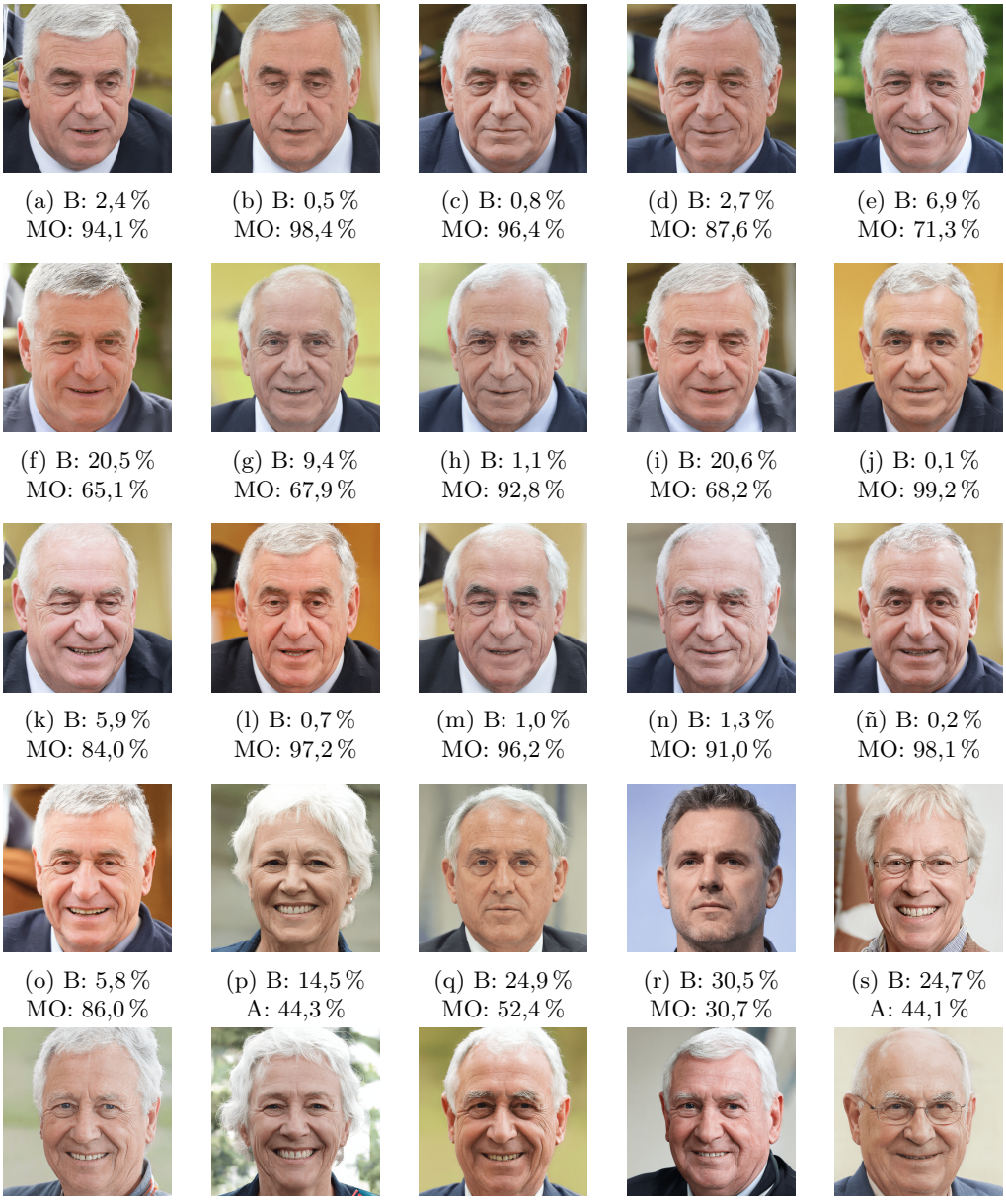
(v) B: 87,0 %
N: 0,0 %

(w) B: 95,5 %
N: 0,0 %

(x) B: 47,7 %
N: 0,1 %

Figura C.14: Ataques a raza negra (N) por raza clasificada y la clasificación con mayor probabilidad (A: Asiática, B: Blanca, LH: Latino hispano, MO: Medio oriente, I: India)

C.9. Ejemplos de ataques adversarios de raza blanca



(a) B: 2,4 %
MO: 94,1 %

(b) B: 0,5 %
MO: 98,4 %

(c) B: 0,8 %
MO: 96,4 %

(d) B: 2,7 %
MO: 87,6 %

(e) B: 6,9 %
MO: 71,3 %

(f) B: 20,5 %
MO: 65,1 %

(g) B: 9,4 %
MO: 67,9 %

(h) B: 1,1 %
MO: 92,8 %

(i) B: 20,6 %
MO: 68,2 %

(j) B: 0,1 %
MO: 99,2 %

(k) B: 5,9 %
MO: 84,0 %

(l) B: 0,7 %
MO: 97,2 %

(m) B: 1,0 %
MO: 96,2 %

(n) B: 1,3 %
MO: 91,0 %

(ñ) B: 0,2 %
MO: 98,1 %

(o) B: 5,8 %
MO: 86,0 %

(p) B: 14,5 %
A: 44,3 %

(q) B: 24,9 %
MO: 52,4 %

(r) B: 30,5 %
MO: 30,7 %

(s) B: 24,7 %
A: 44,1 %

(t) B: 20,0 %
L: 26,7 %

(u) B: 13,7 %
A: 46,5 %

(v) B: 7,3 %
MO: 63,6 %

(w) B: 13,0 %
MO: 33,1 %

(x) B: 31,6 %
MO: 48,7 %

Figura C.15: Ataques a raza blanca (B) por raza clasificada y la clasificación con mayor probabilidad (A: Asiática, N: Negra, LH: Latino hispano, MO: Medio oriente, I: India)

C.10. Ejemplos de ataques adversarios de raza asiática

(a) A: 2,3 %
B: 60,9 %

(b) A: 12,0 %
B: 37,4 %

(c) A: 3,8 %
B: 55,0 %

(d) A: 8,4 %
MO: 32,1 %

(e) A: 3,3 %
B: 85,3 %

(f) A: 2,8 %
B: 73,0 %

(g) A: 21,8 %
B: 49,4 %

(h) A: 1,5 %
B: 48,7 %

(i) A: 8,2 %
B: 44,1 %

(j) A: 6,5 %
B: 83,1 %

(k) A: 0,8 %
B: 97,8 %

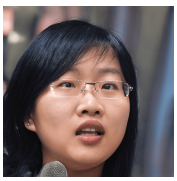
(l) A: 6,5 %
B: 61,1 %

(m) A: 3,1 %
B: 68,0 %

(n) A: 0,0 %
B: 99,8 %

(ñ) A: 17,1 %
B: 76,8 %

(o) A: 7,0 %
B: 36,8 %



(p) A: 27,5 %
B: 50,5 %

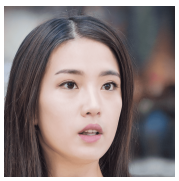
(q) A: 25,4 %
A: 25,4 %

(r) A: 7,2 %
L: 36,6 %

(s) A: 12,4 %
B: 50,4 %

(t) A: 6,9 %
B: 60,0 %

(u) A: 0,2 %
B: 92,6 %



(v) A: 0,7 %
B: 66,0 %

(w) A: 0,8 %
B: 59,8 %

(x) A: 1,6 %
B: 52,1 %

Figura C.16: Ataques a raza asiática (A) por raza clasificada y la clasificación con mayor probabilidad (N: Negra, B: Blanca, LH: Latino hispano, MO: Medio oriente, I: India)