INSTITUTO TECNOLÓGICO DE BUENOS AIRES – ITBA

ESCUELA DE INGENIERÍA Y GESTIÓN

AUTOENCODERS Y ANÁLISIS DE COMPONENTES PRINCIPALES: Propuesta de generación de ejemplos adversarios en el contexto de sistemas de reconocimiento facial

AUTOR/ES: **Fuster, Marina (Leg. Nº 57613)**

Vidaurreta, Ignacio Matías (Leg. Nº 57250)

DOCENTE/S TITULAR/ES O TUTOR/ES: Pierri, Alan

TRABAJO FINAL PRESENTADO PARA LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN INFORMÁTICA

Lugar: Buenos Aires, Argentina
Fecha: 11/10/2021

Resumen

En el presente trabajo se propone una nueva metodología de generación de ejemplos adversarios, datos manipulados para confundir algoritmos de clasificación. El objetivo en este caso es vulnerar sistemas de reconocimiento facial con una estrategia que se basa en el uso de *autoencoders*, un tipo de redes neuronales, y análisis de componentes principales.

Para la construcción de estos ejemplos se debió seleccionar las características que debía tener el *autoencoder* y, luego, una vez seleccionada la arquitectura a utilizar, se aplicó análisis de componentes principales sobre las representaciones de los datos en el espacio latente de dicho *autoencoder*.

Se realizaron numerosos experimentos para estudiar la idoneidad de la metodología propuesta como forma plausible de generación de ejemplos adversarios. Se concluye que la propuesta no conduce a un esquema de ataque realista y se analiza en profundidad el contexto teórico y la toma de decisiones que llevó a esta conclusión.

1. Introducción

La llegada del COVID-19 tomó al mundo por sorpresa y provocó que la industria informática alcanzará nuevos niveles de relevancia. Un aspecto de la vida cotidiana que resultó particularmente impactado por la pandemia fue la modalidad de trabajo. Las empresas tuvieron que modificar su metodología de operación para preservar la salud de los trabajadores. Una de las medidas principales fue la llegada del trabajo 100% remoto para empleados no esenciales.

Este cambio implicó una modificación muy grande en la infraestructura tecnológica de las empresas. Según una encuesta realizada por Forbes [34], algunas de las preocupaciones de los ejecutivos alrededor del mundo sobre la forma en que la pandemia modificó sus recursos informáticos son:

- El 85% de los ejecutivos encuestados cree que las inversiones en seguridad de identidad son críticas.
- El 55% invirtió en nuevas capacidades en este área desde el comienzo de la pandemia.
- El 60% incrementó los gastos relacionados a la identidad debido al trabajo remoto y el 69% de los ejecutivos espera que las inversiones en administración de identidad y acceso (IAM[1] por sus siglas en inglés) aumenten en el próximo año.

Como se puede observar, durante la pandemia la autenticación tomó un rol muy importante como consecuencia de la adopción acelerada de trabajo remoto. Un estudio del *McKinsey Global Institute* indica que, en un escenario post-pandemia, el trabajo remoto mantendrá una adopción entre cuatro o cinco veces mayor respecto a los niveles pre-pandemia [36]. De aquí surge la pregunta ¿cómo se mejora la seguridad de autenticación? La respuesta, generalmente recomendada por la industria, es la autenticación de múltiples factores (o MFA por sus siglas en inglés).

Tradicionalmente, MFA toma dos formas [35]: “algo que sabés” (contraseñas) y “algo que tenés” (celular, yubikey). Sin embargo, en los últimos años estuvo ganando popularidad otro método: la autenticación biométrica. Este tipo de autenticación responde a la pregunta “algo que sos” y puede tener la forma de detección de iris, huella dactilar o, como es el caso de interés en el presente trabajo, de rostro. A modo de ejemplo, LastPass, un reconocido administrador de contraseñas, ofrece una solución de MFA utilizando reconocimiento facial para autenticar a los empleados en sus aplicaciones de trabajo [37]. Si bien estos sistemas son fáciles de utilizar y difíciles de engañar, la crítica más fuerte que se les hace es la imposibilidad de modificar la información complementaria del sistema de autenticación: si la misma se encontrase comprometida, no sería posible utilizar el sistema.

El área de sistemas biométricos y, en particular, de sistemas de reconocimiento facial, muestran un gran avance en este nuevo paradigma. Según la empresa de investigación *MarketsandMarkets*, se estima que el mercado de reconocimiento facial global crecerá de USD 3.8 billones en 2020 a USD 8.5 billones para 2025 [6].

[1] En el artículo citado al mencionar este resultado se utilizan las siglas AIM las cuales al revisar la fuente se corroboró que significa lo mismo que IAM pero el último es más utilizado en la literatura por lo que se hizo el cambio.

Al momento de adoptar un sistema de reconocimiento facial es necesario ser cautelosos. En un contexto donde la incorporación de estas herramientas muestra una tendencia creciente, se debe prestar atención a los usos que se le otorgan. Ningún sistema es invulnerable pero, al conocer mejor sus debilidades se puede trabajar en mitigar y visibilizar los riesgos, logrando transparencia y honestidad frente a la sociedad que, en última instancia, hará uso de la herramienta.

A pesar del crecimiento de inversión en el sector de reconocimiento facial, especialmente por parte del área APAC (Asia-Pacific), existen inquietudes con respecto a la seguridad y correcto uso de la tecnología, especialmente al tomar en consideración las áreas de mayor uso al día de hoy: seguridad (incluyendo vigilancia), salud, finanzas y ventas. Al mismo tiempo que ciertos países aumentan su nivel de adopción (como es el caso de China), en Estados Unidos, ciudades como San Francisco, Oakland y San Diego muestran recelo por la sensibilidad de los datos biométricos y su uso para acompañar a la ley, resultando en la proscripción de la tecnología de reconocimiento facial en 2019 [7].

Esta diferencia de tendencias provee evidencia sobre la falta de un marco regulatorio actual tanto para verificaciones de seguridad (sobre la tecnología y los datos a los que se puede acceder con autenticación biométrica) como para los correctos usos que se le debería dar a esta herramienta. En el presente informe, se profundiza sobre una problemática de seguridad que presenta el sistema de reconocimiento facial desde el punto de vista técnico.

Como se explica en el artículo de *MarketsandMarkets*, *deep learning* es la tecnología principal detrás de los sistemas de reconocimiento facial [6]. Si bien las redes neuronales *deep* son ampliamente utilizadas, en 2014 se publicó un estudio que analiza una vulnerabilidad de las mismas: los ejemplos adversarios [28]. En el contexto del sistema de reconocimiento facial es posible describir a los ejemplos adversarios como imágenes modificadas de manera tal que causen fallos en la clasificación o correcta identificación de las personas.

La potencialidad de confundir algoritmos considerados de alta precisión sobre la identidad de los individuos puede traer graves consecuencias. Dados los sectores donde se utilizan los sistemas de reconocimiento facial, podrían estar en riesgo datos médicos, cuentas bancarias y la autoría de acciones que realizamos día a día tanto en la calle como *online*.

A lo largo del trabajo se estudiará una nueva metodología de generación de ejemplos adversarios para sistemas de reconocimiento facial, basada en el uso de *autoencoders* y análisis de componentes principales.

2. Marco Teórico

2.1. Sistema de Reconocimiento Facial

Sea $S : I \to R$ un sistema de reconocimiento facial. S recibe una imagen i perteneciente al conjunto I de imágenes de rostros digitales, cuyo aspecto es realista para seres humanos y retorna una respuesta $r \in R$, la cual resulta de la comparación de i con un conjunto de rostros que S tiene almacenado. En r no hay etiquetas correspondientes a

clasificaciones específicas, sino que se incluye el porcentaje de similaridad de la imagen i con los rostros almacenados en S.

Por otra parte, se cuenta con un oráculo $O : I \to T$ que recibe una imagen $i \in I$ y devuelve una etiqueta $t \in T$, donde T es el conjunto de etiquetas posibles. La etiqueta se corresponde con la identidad de la persona en la imagen i. Los resultados del oráculo son consistentes con la identificación realizada por personas físicas. Se tomó la definición de oráculo de [1].

2.2. Ataque

Se define un ataque como el envío de una imagen i^* al sistema S, de manera tal que se altere el comportamiento del mismo de acuerdo a una (o ambas) de las definiciones establecidas en las secciones **2.3.** y **2.4.**

2.3. Evasión

Dada una imagen i_e perteneciente al conjunto de imágenes I con su respectiva etiqueta $t_e = O(i_e)$, se define evasión si se verifica que, para las imágenes almacenadas en S, el promedio de las similaridades de i_e con todas aquellas imágenes i_j que cumplan $t_e = O(i_j)$, se encuentra por debajo del 80%. La justificación de este umbral será explicada en la sección **5.2.2.**

2.4. Impersonificación

Dada una imagen i_i perteneciente al conjunto de imágenes I con su respectiva etiqueta $t_i = O(i_i)$ y sea t_x una etiqueta seleccionada por el atacante de manera tal que $t_i \neq t_x$.
Se produce un ataque de impersonificación si se verifica que, para las imágenes almacenadas en S, el promedio de las similaridades de i_i con todas aquellas imágenes i_j que cumplan $t_x = O(i_j)$, se encuentra por encima del 80%. La justificación de este umbral será explicada en la sección **5.2.2.**

2.5. Ejemplo Adversario

Un ejemplo adversario es un dato de entrada a un modelo de aprendizaje automático, diseñado por un atacante para provocar que el modelo devuelva un “error”. Por error se comprende un resultado diferente al esperable según los seres humanos.

En ***Fig. 1*** podemos ver un ejemplo de error de clasificación. A la izquierda se encuentra la imagen de un oso panda cuya clasificación según GoogleLeNet provee una confianza del 57,7% sobre la etiqueta de oso panda. Al manipular la imagen (con una de las numerosas técnicas de generación de ejemplos adversarios que existen) el clasificador determina que la imagen se trata de una especie de simio llamada gibón, con confianza del 99,3% [15].

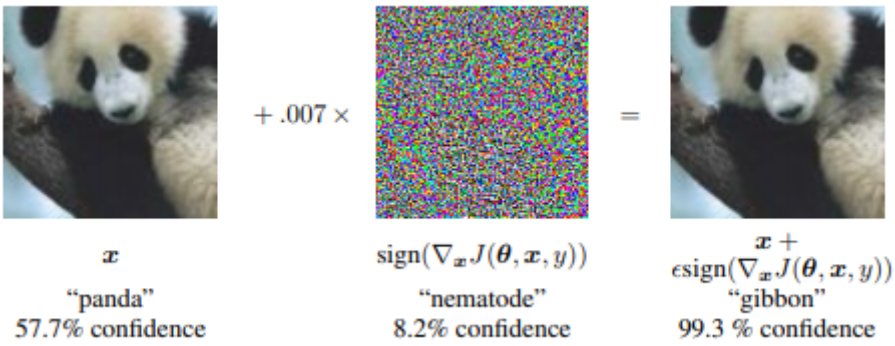


***Figura 1**: ejemplo adversario que engaña a la red GoogleLeNet.*

En el contexto del trabajo, se definirá ejemplo adversario como una imagen $i^* \in I$ que permita lograr un ataque A, según la definición de ataque de la sección **2.2.**, sobre un sistema de reconocimiento facial S.

Nótese que i^* puede ser producto de la manipulación de una imagen base o ser generada desde cero.

2.6. Análisis de Componentes Principales

Trabajar con conjuntos de datos cuyos elementos pertenecen a un conjunto de alta dimensionalidad, como por ejemplo imágenes, puede traer dificultades: no es posible visualizar los datos, lo cual puede traer problemas al momento de analizar propiedades interesantes. Sin embargo, existe una hipótesis empírica, llamada *Manifold Hypothesis*, la cual declara que datos en altas dimensiones suelen tener representaciones en espacios de menor dimensionalidad [22]. Esta hipótesis respalda los métodos utilizados para encontrar correlaciones en la información, para reorganizarla y obtener datos relevantes de la misma.

El Análisis de Componentes Principales (PCA por sus siglas en inglés) es una técnica cuyo uso más conocido es la reducción lineal de dimensionalidad en el área de aprendizaje automático. Provee la posibilidad, dependiendo a qué dimensión se reduzcan los datos, de graficar un conjunto de elementos cuya visualización resultaba imposible previamente.

PCA analiza la frecuente correlación entre los datos de alta dimensionalidad, buscando una rotación de los mismos para describirlos en términos de características no correlacionadas a través de combinaciones lineales [11]. El proceso de búsqueda de las componentes principales está fuertemente relacionado a la descomposición en valores singulares.

Sea N la cantidad de observaciones realizadas, p la cantidad de características presentes en una observación y q la cantidad de características objetivo $(q \leq p)$. La solución a la búsqueda de componentes principales está dada por la ecuación

$$X = UDV^T \quad (1)$$

Donde X refiere al conjunto de datos, U es una matriz ortogonal de dimensión $N \times p$, cuyas columnas son llamadas autovectores por izquierda. V es una matriz ortogonal de dimensión $p \times p$ cuyas columnas son llamadas autovectores por derecha. Por último, D es

una matriz diagonal de dimensión $p \times p$ donde los elementos de la diagonal cumplen $d_1 \geq d_2 \geq ... \geq d_p \geq 0$ y son conocidos como los valores singulares. Para la deducción completa hasta llegar a dicha expresión consultar [14].

Las columnas de UD son las proyecciones en componentes principales de X. Si se busca obtener las primeras q componentes principales basta con tomar la matriz $U_q D_q$ de dimensiones $N \times q$, que se corresponde con tomar las primeras q columnas del producto.

Como se puede observar, en ningún momento se habló de autovalores y autovectores dado que los mismos sólo aparecen al trabajar con matrices cuadradas, es decir, $N = p$. Cuando se da esta situación, las columnas de V son los autovectores, mientras que los autovalores están dados por D^2.

A lo largo de este trabajo **se estará trabajando con matrices que no son cuadradas** (las cuales son más frecuentes y menos restrictivas), sin embargo, se referirá como **autovectores** a los autovectores por derecha y como **autovalores** a los valores singulares (recordar que los autovalores son los valores singulares al cuadrado). Se tomó esta decisión ya que las herramientas utilizadas en el proyecto emplean esta nomenclatura y la documentación suele hablar en estos términos en vez de utilizar los matemáticamente correctos.

La descomposición en valores singulares ordena los autovectores (con sus autovalores) según la dirección de máxima variabilidad de los datos. Esto quiere decir que habrá una mayor dispersión de los mismos para las primeras componentes con respecto a las últimas. El porcentaje de variabilidad que representa cada autovector se calcula como el cociente entre el autovalor correspondiente y la suma de todos los autovalores.

Antes de aplicar análisis de componentes principales, se deben estandarizar las características del conjunto de datos: media cero y desvío estándar 1. Esto es sumamente importante ya que el proceso de PCA utiliza la matriz de covarianza y, en caso de tener características cuyas unidades de medición sean muy dispares, aquellas con mayor magnitud serán favorecidas al momento de analizar la ponderación que explica la variabilidad de los datos [16].

2.7. Autoencoder

El *autoencoder* es una red neuronal cuyo aprendizaje es no supervisado, es decir, no existe un agente que etiquete los datos. El objetivo es encontrar representaciones compactas de un conjunto de datos, desde las cuales puedan recuperarse los elementos originales con la menor pérdida de información posible.

Este objetivo puede modelarse como un problema de optimización a partir de la función (2), donde X es la entrada original y X' la salida del autoencoder. Durante el entrenamiento, el objetivo es minimizar esta función de pérdida.

$$L(X, X') = \|X - X'\|^2 \quad (2)$$

Se puede pensar la arquitectura de un *autoencoder* como dos redes perceptrones multicapa, donde la salida de la primera red se conecta con la entrada de la segunda. A la primera red se la suele llamar codificador y a la segunda decodificador. En ***Fig. 2*** se puede observar una representación visual de un *autoencoder*.

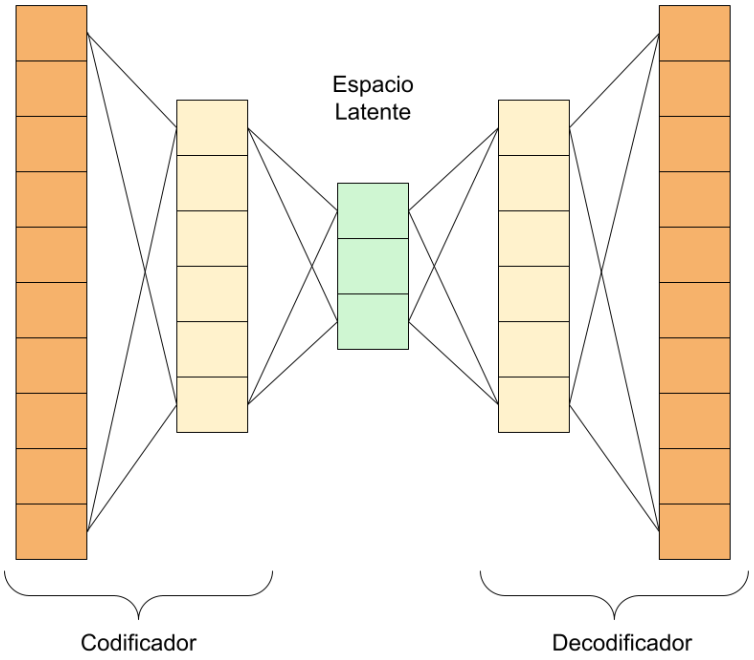


***Figura 2**: Representación gráfica de un autoencoder*

Al enviar una imagen por el *autoencoder*, ésta pasa por el codificador que reduce su dimensionalidad al tamaño correspondiente de la capa intermedia. A este espacio en R^n se lo denomina **espacio latente**.

El punto de mayor interés en el trabajo actual se encuentra en este espacio latente, pues es en esta capa en la cual se realiza la experimentación para la generación de ejemplos adversarios. Se explicará con mayor detalle este proceso en las secciones **4** y **8**.

3. Estado del Arte

Los algoritmos de reconocimiento facial han alcanzado una precisión muy alta en los últimos años. Desde una tasa de error de 4,1% en 2014, estos algoritmos lograron disminuir su tasa de error al 0,08% al día de hoy. No sólo eso, sino que la cantidad de algoritmos que alcanzaron dicha precisión también aumentó respecto de aquél año [21]. Esto genera la necesidad de poner especial atención tanto al correcto funcionamiento de la tecnología como también a las regulaciones que deben formularse para el uso apropiado de la misma.

En 2014 se publicó un estudio que analizaba propiedades inusuales en redes neuronales, una de las cuales se conoce como ejemplos adversarios: entradas modificadas de manera imperceptible que pueden causar fallos en la clasificación [28]. Uno de los investigadores referentes sobre esta temática, Nicholas Carlini, ha recopilado en un artículo de su sitio web

todos los papers que encontró sobre ejemplos adversarios en la plataforma *arXiv* [8]. Esta recopilación muestra una tendencia exponencial en la cantidad de papers publicados, lo cual pareciera ser indicio del interés que rodea a esta temática.

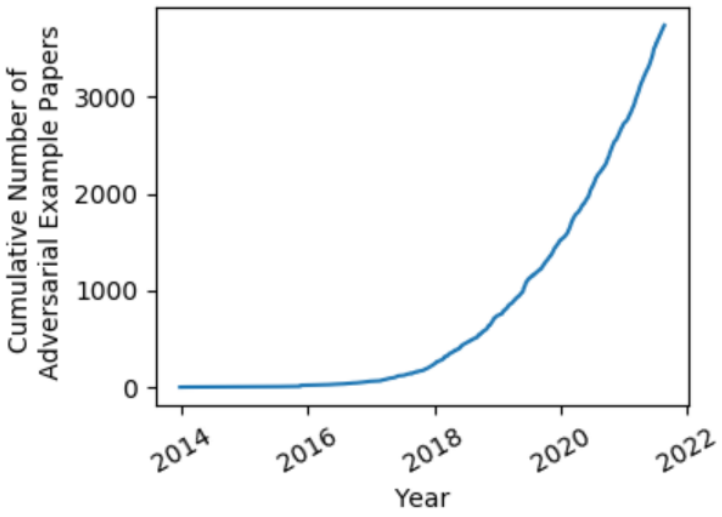


***Figura 3**: artículos académicos sobre ejemplos adversarios, acumulados desde 2014.*

Dos de los métodos más conocidos para generar estos ejemplos son el método de Carlini y Wagner (CW) [38] y el método de *Fast Gradient Descent* (FGS) [15]. Los ejemplos adversarios de estos papers son resultado de modificar un elemento original a través de perturbaciones. Además de la metodología mencionada, también existen los *Unrestricted Adversarial Examples*, los cuales son generados desde cero con algoritmos generativos, como pueden ser las Redes Generativas Adversarias, también conocidas como GAN por sus siglas en inglés [1]. En esta segunda categoría no existe la idea de modificaciones sobre una imagen pues no se tiene una imagen original de la cual partir.

Es de interés aclarar que, a lo largo del trabajo, al hablar de ejemplo adversario se referirá a la definición dada en la sección **2.5.**, la cual pone el foco en el efecto que el mismo tiene sobre un sistema de reconocimiento facial y no en su generación, ya que no es el objetivo de este trabajo estudiar la taxonomía de los ejemplos adversarios en profundidad.

En la literatura puede observarse que los investigadores estudian los ejemplos adversarios tanto desde el punto de vista del atacante como del defensor. Se puede dividir la metodología del ataque en dos modalidades: *white box* (se tiene acceso a la red que se busca vulnerar y a sus pesos) [29] y *black box* (se desconoce cómo funciona la red que se intenta atacar) [9]. Una característica interesante de los ejemplos adversarios que puede observarse en [28] es la verificación del llamado **principio de transferibilidad**: aquellos ejemplos que logran causar fallos en la clasificación de una red, replican este comportamiento en otras redes con arquitecturas diferentes o, inclusive, con un conjunto distinto de entrenamiento, provocando que los ejemplos adversarios sean robustos a distintos clasificadores.

Como se mencionó en la sección **1.**, es de interés analizar el rol que ocupan los *autoencoders* y el análisis de componentes principales en el estudio de ejemplos

adversarios. En relación a PCA se pudo observar que hay estudios tanto para ataque [24] como para defensa [25]. Por otra parte, el uso de *autoencoders* aparece mayoritariamente en la defensa de ejemplos adversarios, con lo cual se analizó esta tendencia en mayor profundidad.

Por un lado, se puede observar el caso de PuVAE [31] en el cual se utilizó un *autoencoder* variacional (VAE por sus siglas en inglés) para "purificar" un ejemplo adversario. La propuesta implica recibir una imagen y, antes de enviarla a una red de clasificación, procesarla con el VAE de manera tal que se vuelve inocuo el ataque por la minimización de las perturbaciones. Los autores destacan que resulta ser más rápido que metodologías que hacen lo mismo con GAN.

Además de purificación de ejemplos adversarios, los *autoencoders* también son utilizados en la detección de los mismos. En [32] estudian la diferencia en la distribución de imágenes sin modificar comparada con la de imágenes adversarias. Para esto, entrenan un *autoencoder* por cada capa oculta en el clasificador a defender: en el espacio latente de cada *autoencoder* se podrá encontrar el *manifold* (distribución) de las imágenes originales correspondiente a una capa. Se puede analizar la distancia de representaciones latentes de ejemplos adversarios a la distribución de imágenes originales. Esta métrica se utiliza como método de detección. El mecanismo propuesto fue probado con éxito experimental sobre 5 ataques distintos, uno de los cuales es el previamente mencionado CW.

Si bien no se encontraron papers que logren penetrar estas defensas, esto no quiere decir que las mismas sean válidas para todo tipo de ejemplos adversarios. Un aspecto discutido en [27] sobre el estudio de defensas es la falta de un marco riguroso para analizar si las mismas son efectivas. Existen trabajos que adaptan nuevos ataques para explotar defensas propuestas (en [33] se vulneran trece defensas y en [26] otras diez). Si bien [26] fue publicado en noviembre de 2017, inclusive hoy, continúa la preocupación por la falta de verificaciones apropiadas al momento de la publicación.

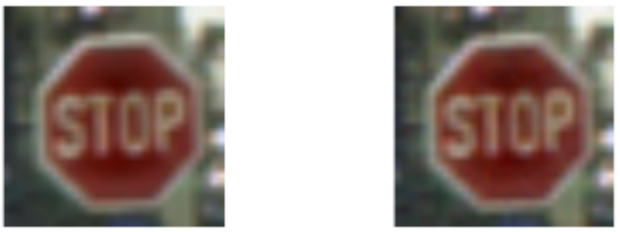


***Figura 4**: a la izquierda se encuentra la imagen original. A la derecha, la imagen modificada de manera imperceptible, que el clasificador reconoce como señal de "ceda el paso". Ejemplos obtenidos de [9].*

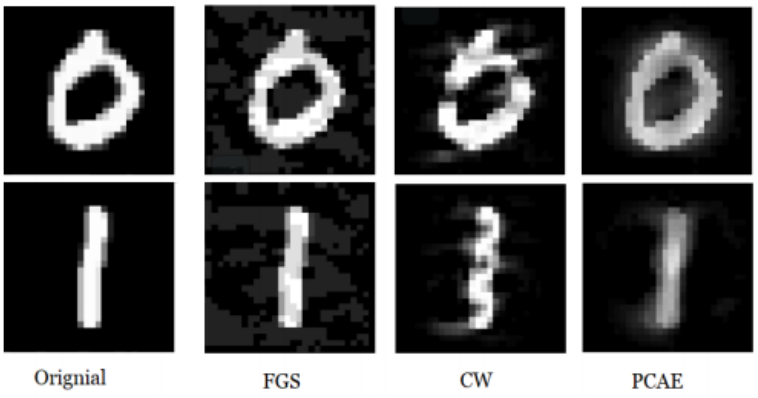


***Figura 5**: generación de ejemplos adversarios a partir de diferentes métodos publicados. Ejemplos obtenidos de [24].*

Con respecto al estudio de construcción de ejemplos adversarios, fue llamativa la falta de análisis de calidad en las imágenes generadas. Si bien existen ejemplos adversarios con modificaciones imperceptibles, como puede verse en ***Fig. 4***, también hay ataques que aparentan bajar la calidad de la imagen, como son FGS, CW y PCAE. En ***Fig. 5*** se encuentran ejemplos de estos ataques para los dígitos 0 y 1 del conjunto de datos MNIST. En dicha figura es posible notar que las perturbaciones añadidas no son equivalentes pues algunas imágenes tienen peor calidad que otras.

4. Propuesta

4.1. Contexto del Proyecto

Al momento de decidir la temática del proyecto final, se tuvieron en cuenta las dos áreas de principal interés de los autores: inteligencia artificial y seguridad informática. Se decidió continuar explorando el camino de inteligencia artificial y, dentro del mismo, se encontró la problemática de ejemplos adversarios en los algoritmos de *deep learning*. Dicha área capturó la atención de los autores, pues demostraba la existencia de vulnerabilidades que podrían traer graves consecuencias al momento de utilizar estos algoritmos en operaciones cotidianas.

Una elección importante fue determinar si la investigación se posicionaría desde el punto de vista de ataque o de defensa. La premisa inicial fue que la posición de ataque era más sencilla al momento de abordar la temática ya que permitiría entender cómo generar un ejemplo adversario y, siendo éste el primer proyecto de investigación de los autores, se decidió continuar por este rumbo.

4.2. Metodología de Ataque

Se propone una nueva metodología de generación de ejemplos adversarios combinando el uso de *autoencoders* con análisis de componentes principales. Como puede verse en ***Fig. 6***, a partir de la representación de una imagen en el espacio latente se obtiene su proyección en componentes principales. Una vez realizada la manipulación sobre la proyección obtenida anteriormente se aplica la operación inversa, obteniendo un vector

perteneciente al espacio latente. Por último, dicho vector se envía al decodificador del *autoencoder* para obtener un ejemplo adversario.

Nótese que esta metodología busca obtener una imagen cuya clase pueda ser reconocida por un oráculo O (ver sección **2.1.**). Por otro lado, en ***Fig. 6*** la respuesta del sistema de reconocimiento facial S no es real sino que busca ilustrar un ejemplo de ataque de evasión: S disminuye el porcentaje de reconocimiento del individuo al recibir el ejemplo adversario. La etiqueta verdadera del individuo, "Meryl Streep", es aquella que asigna el oráculo O, consistente con la clasificación humana.

Esta metodología se utilizará para vulnerar sistemas de reconocimiento facial, en un esquema de ataque *black box*. En la sección **2.6.** se mencionó que un uso común de PCA era la reducción de dimensionalidad pero, en este caso, es de interés la posibilidad de ordenar las características de acuerdo a la variabilidad que representan de la información. En este caso no se reduce la dimensionalidad pues ya se obtuvo una versión compacta del dato original al enviarlo al codificador.

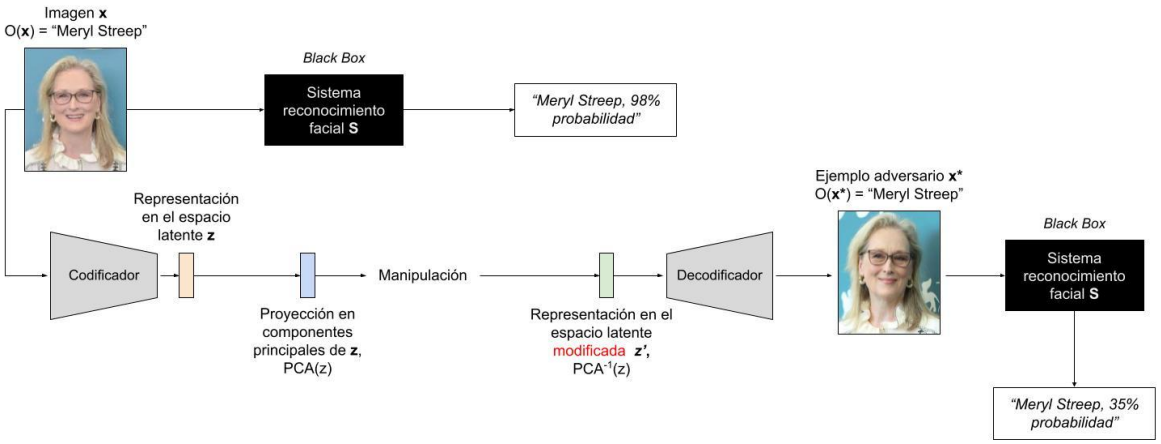


***Figura 6**: representación gráfica de la propuesta para generación de ejemplos adversarios.*

Por último, la falta sistemática de estudio de la calidad de los ejemplos adversarios es un diferencial que se propone en este trabajo. Servicios de reconocimiento facial, como Amazon Rekognition, proveen funcionalidades para analizar características de brillo, contraste, etc, proveyendo información sobre la calidad de una imagen. Se considera importante estudiar este aspecto ya que efectos negativos en la calidad disminuyen el reconocimiento.

4.3. Objetivos

El trabajo consta de una sección de experimentación, donde se intentará:

- Realizar ataques de evasión, según la definición provista en **2.3.**
- Realizar ataques de impersonificación, según la definición provista en **2.4.**

Se espera encontrar una relación entre la identidad de las personas y las proyecciones en componentes principales de la representación latente de las imágenes de dichas personas. Se supone que será posible describir la identidad de los individuos del conjunto de datos con las primeras componentes y que las mismas representarán una cantidad significativa de información respecto de las restantes.

Por otro lado, se buscará asegurar que los niveles de calidad de las imágenes generadas se encuentren en un rango estipulado a partir de imágenes sin manipular. Como se mencionó en **3.**, si bien se habla de “modificaciones imperceptibles”, pueden notarse diferencias visibles con respecto a la imagen original, como se vió en ***Fig. 5***.

4.4. Marco del Problema

4.4.1. Hardware de propósito general

Se decidió utilizar *hardware* de propósito general en lugar de *hardware* especializado porque el proyectó inició de manera remota y los autores sólo tenían acceso a este tipo de *hardware*. Una vez avanzado el proyecto, surgió la posibilidad de utilizar una GPU NVIDIA que se encontraba en el ITBA pero, debido a que el proyecto se encontraba en un estado avanzado con dicha limitación en mente, se decidió no tomar ese camino.

4.4.2. Conjunto de Datos

Para la experimentación, se dispuso utilizar sólo dos sujetos para el conjunto de datos. Esto permite no sólo contar con un conjunto más pequeño (permitiendo entrenar al *autoencoder* en menos tiempo), sino que también simplifica el análisis posterior, ya que el objetivo será pasar de una persona a otra. En caso de agregarse más individuos, la cantidad de relaciones aumentaría.

A lo largo del trabajo, el conjunto de datos[2] evolucionó en tres aspectos:

- **Cantidad de imágenes**: se pasó de 40 a 100 imágenes por individuo.
- **Preprocesamiento**: se decidió transformar las imágenes de color a blanco y negro. En este caso, la decisión fue por optimización de entrenamiento a costa de perder información del individuo, como por ejemplo el color de ojos.
- **Fotos originales**: en la primera iteración se tenían dos individuos: Ignacio y Marina. Como puede observarse en ***Fig. 7***, las imágenes iniciales de Ignacio contaban con barba y lentes. Se decidió eliminar estas características para que el proceso de generación de ejemplos adversarios fuera más sencillo, ya que tanto la barba como los lentes eran características muy distintivas.

[2] El conjunto de datos utilizado para el proyecto puede encontrarse en la siguiente carpeta.

***Figura 7**: a izquierda, imágenes iniciales.*
A la derecha, las imágenes que se utilizan en el trabajo.

4.4.3. Amazon Rekognition como sistema objetivo

La última limitación impuesta se refiere al sistema de reconocimiento facial a utilizar. Se empleará únicamente el sistema Amazon Rekognition. En [9] y [10] se analizan ejemplos adversarios con distintos clasificadores para estudiar el principio de transferibilidad.

Dado que era el primer acercamiento a la investigación sobre la temática, se poseía un costo significativo en la implementación de infraestructura que permitiera la experimentación. Concentrarse en implementar un sistema únicamente para Amazon Web Services (AWS) tuvo como ventaja el ahorro de tiempo dedicado a planificar y diseñar una abstracción que permitiera utilizar cualquier sistema de reconocimiento facial. La desventaja de esta decisión es la pérdida de escalabilidad y de la posibilidad de verificar el principio de transferibilidad, pero, en este caso, se considera beneficioso pues permite dedicarle más tiempo a la elección del *autoencoder* y posterior experimentación.

5. Diagrama del sistema

A lo largo del desarrollo del trabajo se fue iterando la arquitectura del sistema para la mejora y adaptabilidad del mismo frente a requerimientos cambiantes. Con el objetivo de mostrar el sistema, se decidió separar la explicación en tres diagramas: entrenamiento del *autoencoder*, proceso de experimentación y, por último, visualización de resultados. Para cada uno de ellos se realizará una explicación de sus componentes y los módulos necesarios para la implementación.

5.1. Entrenamiento del autoencoder

Esta sección se compone principalmente de dos procesos: en primer lugar, el preprocesamiento del conjunto de datos, cuyas características se analizaron en la sección **4.** En segundo lugar se encuentra representado en ***Fig. 8*** el entrenamiento del *autoencoder* con el resultado del primer proceso.

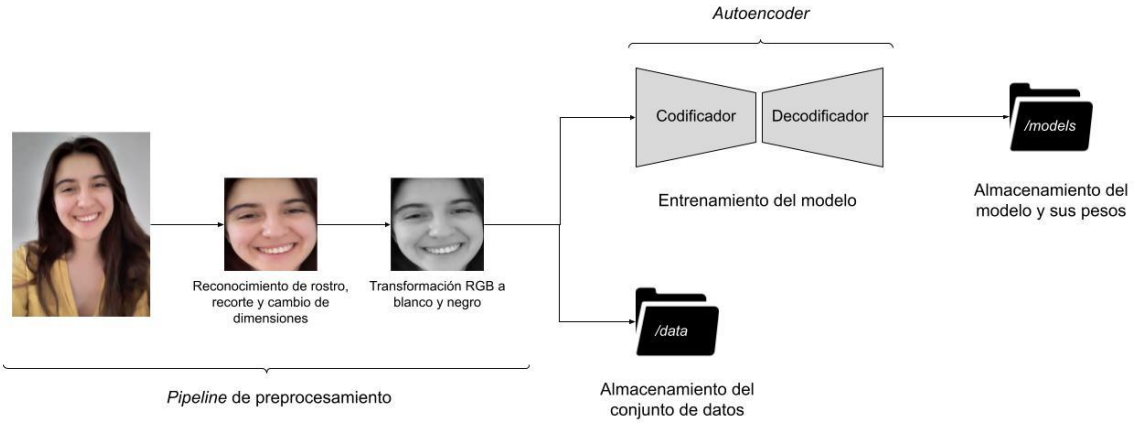


***Figura 8**: Diagrama de flujo sobre el proceso de entrenamiento del autoencoder.*

Fue necesario desarrollar varios módulos para esta sección. En primer lugar se cuenta con un módulo para el preprocesamiento llamado `face_detection`[3], cuya funcionalidad se describe en detalle durante la sección **5.1.1.** que se encuentra a continuación.

Por otro lado, se cuenta con la implementación correspondiente al *autoencoder* que comprende tres módulos: `autoencoder`[4], `encoder`[5] y `decoder`[6]. Éstos se encargan de la funcionalidad pertinente al entrenamiento y posterior predicción de datos. La separación en módulos permitió obtener mayor flexibilidad al momento de trabajar con las representaciones de las imágenes en la capa latente, lo cual resultó particularmente útil para la próxima sección, **5.2.**

Por último, se implementaron módulos auxiliares, como `data_loader`[7] y `model_loader`[8], para almacenamiento y recuperación del conjunto de datos y modelos. Luego de entrenar el *autoencoder*, se guarda el modelo del mismo en un archivo `.json`, junto con sus pesos en un archivo `.h5`, para posterior reutilización. Esto es necesario para evitar entrenar el *autoencoder* cada vez que se lo quiere utilizar, dado que este proceso puede consumir bastante tiempo.

5.1.1. Preprocesamiento de datos

Los datos originales son imágenes a color, tomadas con el celular de cada individuo. Antes de utilizar las imágenes en la aplicación (tanto para entrenamiento del *autoencoder* como para experimentación) debieron ser aplicadas las siguientes transformaciones:

- **Detección de rostro y recorte**: en las imágenes hay información que resulta de poco interés para el marco de la investigación, como por ejemplo la remera que utiliza la persona o algún objeto o textura del fondo. Se utilizó el algoritmo MTCNN

[3] Implementación: `modules.face_detection`
[4] Implementación: `modules.autoencoder`
[5] Implementación: `modules.encoder`
[6] Implementación: `modules.decoder`
[7] Implementación: `modules.data_loader`
[8] Implementación: `modules.model_loader`

[17] de detección de rostros para recortar la imágen y, de ese modo, dejar sólo el rostro.

- **Escalar la imagen a 256x256**: de esta manera se tiene un tamaño único de imagen y, a su vez, un tamaño manejable.
- **Blanco y Negro**: transformar las imágenes de color a blanco y negro trae como ventaja una gran simplificación de la imagen. Sin embargo, esta simplificación viene a costa de la pérdida de algunos detalles característicos del individuo como puede ser el color de ojos. Se decidió que la ganancia en *performance* era mayor que la pérdida en detalle por lo que se mantuvo esta modificación.

Estas transformaciones se realizaron para simplificar el problema y disminuir la carga que se genera sobre el *autoencoder*. Es importante resaltar esto debido a la limitación de *hardware* que se planteó en la sección **4.4.1.** Se puede observar que las imágenes en ***Fig. 9*** han pasado por el *pipeline* de preprocesamiento, dando como resultado las imágenes correspondientes en ***Fig. 10***.

Figura 9*: imágenes sin preprocesar del conjunto de datos.*

Figura 10*: imágenes del conjunto de datos luego del preprocesamiento.*

El *autoencoder* recibe las imágenes preprocesadas para ser entrenado con la configuración especificada (ver detalle del proceso de selección del autoencoder en **6.**).

5.2. Proceso de experimentación

El proceso de experimentación que se muestra en el flujo presente en ***Fig. 11*** cuenta con dos secciones importantes: la primera, encerrada en línea punteada, será ejecutada una única vez y luego se utilizarán esos resultados para la sección de experimentación, la segunda, fuera de la línea punteada.

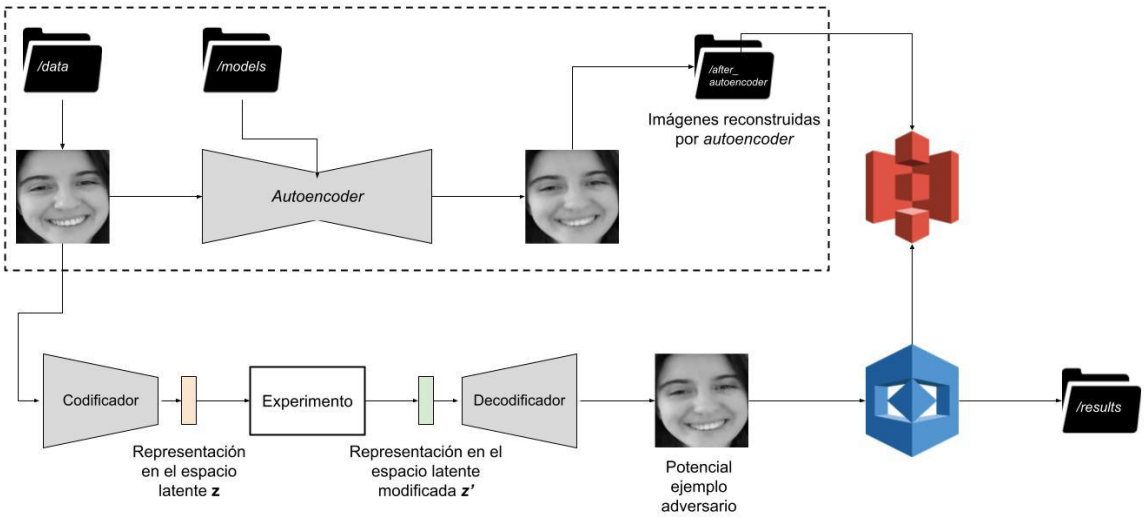


Figura 11*: Diagrama de flujo sobre el proceso de experimentación.*

Una problemática que surgió fue determinar contra qué conjunto de datos serían comparadas las imágenes alteradas (potenciales ejemplos adversarios). Dado que el *autoencoder* realiza una reconstrucción de una imagen a partir de una compresión de la misma y, además, que el mismo no es un modelo que se corresponda con el estado del arte ni con las capacidades de *hardware* especializado, las imágenes que se podrían obtener a partir de esta red neuronal no tienen la misma calidad que los rostros en el conjunto original de datos. Esto podría generar una pérdida de similaridad que se corresponda con la diferencia en calidad y no con la manipulación realizada. En consecuencia, las imágenes que se utilizan para comparar a los potenciales ejemplos adversarios son las que resultan de la salida del *autoencoder*.

Dentro de la línea punteada, se tiene a las imágenes preprocesadas almacenadas en `/data` y el *autoencoder* seleccionado para experimentación, almacenado en la carpeta `/models`. El detalle de selección del mismo puede hallarse en la sección **6.** El objetivo del proceso dentro del diagrama de flujo es obtener todas las imágenes reconstruidas por el *autoencoder*, como se mencionó previamente. El resultado de este procedimiento se almacenó en otra carpeta, llamada `/after_autoencoder`.

Estas imágenes se guardan en el servicio Simple Storage Service (S3) de AWS pues Amazon Rekognition hará uso de las imágenes allí almacenadas para obtener las características de estos rostros, a partir de las cuales realizará las comparaciones.

Por otro lado, para realizar un experimento se utilizarán los modelos del codificador y decodificador del *autoencoder* por separado. Para manipular una imagen, se utiliza el módulo `data_loader` para cargarla al programa y se la envía al codificador para obtener su

representación en el espacio latente. Luego, se realizará la manipulación correspondiente al experimento en cuestión (el detalle de los experimentos realizados se encuentra en la sección **8.**) y, una vez obtenido el vector correspondiente al espacio latente modificado, se enviará al decodificador para adquirir la reconstrucción correspondiente.

Por último, es de interés analizar cómo la imagen manipulada altera los resultados de similaridad contra imágenes inalteradas con lo cual se envía esta imagen modificada al servicio Amazon Rekognition para obtener los resultados pertinentes y almacenarlos en la carpeta `/results`.

5.2.1. Conexión a Amazon Web Services

Para uso de servicios necesarios para este trabajo, se decidió aprovechar el *Free Tier* de Amazon Web Services, el cual permite hacer uso de ciertas tecnologías (con algunos limitantes) sin costo, por un año.

Top Free Tier Services by Usage — View all

Service	Free Tier usage limit	Month-to-date usage
Amazon Rekognition	Analyze 5,000 images per month for Amazon Rekognition	66.78% (3,339.00/5,000 Images Processed)
Amazon Simple Storage Service	2,000 Put, Copy, Post or List Requests of Amazon S3	35.05% (701.00/2,000 Requests)
Amazon Simple Storage Service	20,000 Get Requests of Amazon S3	13.04% (2,607.00/20,000 Requests)
Amazon Simple Storage Service	5 GB of Amazon S3 standard storage	0.04% (0.00/5 GB-Mo)
AWS Lambda	1,000,000 free requests per month for AWS Lambda	0.01% (104.00/1,000,000 Requests)

***Figura 12**: servicios que son parte del Free Tier de Amazon Web Services. Amazon Rekognition es su sistema de reconocimiento facial.*

La integración de los servicios necesarios con la implementación de los autores se realizó utilizando `boto3` [18], un *Software Development Kit* (SDK) que permite conectarse con AWS utilizando Python.

5.2.2. Amazon Rekognition

El servicio más importante que se utiliza es Amazon Rekognition (el cual será abreviado como Rekognition a lo largo del trabajo), el sistema de reconocimiento facial de Amazon. Con `boto3` se implementaron las funcionalidades necesarias para la experimentación que se quería llevar a cabo:

1. Analizar la calidad de una imagen.
2. Comparar la similaridad entre dos imágenes.
3. Comparar la similaridad de una imagen contra un conjunto de imágenes.

Toda la funcionalidad relevante a este servicio fue implementada dentro del módulo `rekognition`[9]. Es importante mencionar que las funcionalidades 2 y 3 no se ejecutan de la misma forma. Para comparar una imagen contra un conjunto (funcionalidad 3) se crea una colección dentro del sistema. Las colecciones persisten las características de los rostros presentes en las imágenes del conjunto de datos. Es por ello que, con `boto3`, se agregó la funcionalidad de crear/borrar una colección y agregar/eliminar caras de una colección. La persistencia es llevada a cabo en Amazon Simple Storage Service (S3), el cual será presentado en la siguiente sección.

Para la funcionalidad de análisis de calidad de una imagen (1), se utiliza la función `DetectFaces` [19]: recibe una imagen y devuelve una gran cantidad de información sobre la misma, como puede ser el rango etario, emociones, género, etc. Dentro de este diccionario de datos, es de interés la sección *Quality* (calidad).

Por otra parte, para comparar la similaridad entre dos imágenes (2) se utiliza la función `CompareFaces` [19] que recibe una imagen fuente y una imagen objetivo. Además de retornar la similaridad entre ambas imágenes, otro dato de interés es la confianza con la cual reconoce que hay una cara en la imagen fuente. Ambos son retornados como porcentajes.

La confianza es un parámetro que puede servir como filtro de calidad, dado que busca indicar con qué confianza se encontró un rostro en la imagen que se quiere comparar. Si esta confianza es baja, quiere decir que Rekognition ni siquiera logró encontrar una persona en el dato en cuestión.

En referencia a la comparación de una imagen con una colección de imágenes (3), se utiliza la función `SearchFacesByImage` [19]: el objetivo es analizar la similaridad de la imagen alterada con respecto al conjunto de datos completo. En ***Fig. 13*** puede observarse un fragmento de la respuesta que envía Rekognition. Esto permitiría obtener un promedio de la similaridad que posee la imagen en cuestión respecto a cada uno de los individuos presentes en el conjunto de imágenes.

Por último, como se mencionó en las secciones **2.3.** e **2.4.**, fue necesario definir un umbral para determinar cuándo se producía un ataque de evasión y/o impersonificación. Dado que el valor por defecto de Rekognition para retornar resultados de similaridad es 80% se decidió utilizar el mismo como umbral. Si bien AWS recomienda modificar este parámetro a 95%, o más para usos sensibles, se cree que es importante analizar si es factible vulnerar la herramienta al momento de utilizarla con parámetros por defecto. En caso de existir esa posibilidad, sería punto de discusión si la empresa debiera utilizar valores por defecto más seguros ya que, si bien la responsabilidad de decidir un umbral recae sobre los clientes, es una barrera extra de protección en caso de que un usuario lo pase por alto.

[9] Implementación: `infra_aws.rekognition.Rekognition`

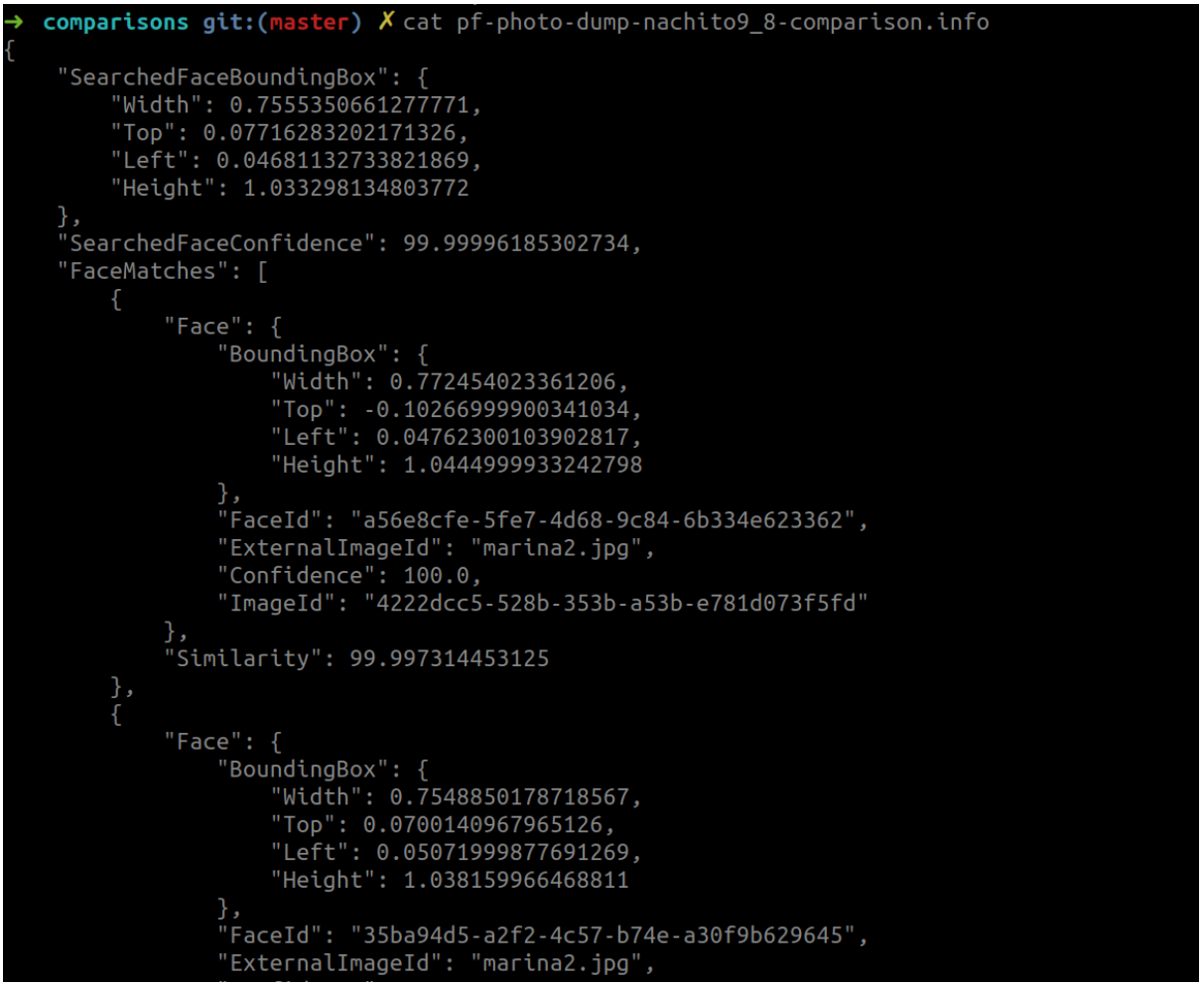


Figura 13: *Fragmento de respuesta resultante de comparar una imagen contra la colección de rostros.*

5.2.3. Amazon S3

Amazon Simple Storage Service es un servicio que permite almacenar objetos en la nube dentro de repositorios llamados *buckets*. Como se mencionó brevemente en la sección **5.2.2.**, fue utilizado para almacenar las imágenes contra las cuales se quiere realizar la comparación de una imagen provista por el usuario.

Dentro de las funcionalidades que fueron implementadas en el sistema de este trabajo, es posible:

- Crear nuevos *buckets*
- Borrar un *bucket* existente
- Subir y borrar archivos a elección

5.3. Visualización de resultados

Para el análisis de los resultados de la sección anterior se utilizó la librería `matplotlib` [20] (versión 3.3.2) de Python, que permitió obtener conclusiones e idear nuevos experimentos a partir de las mismas.

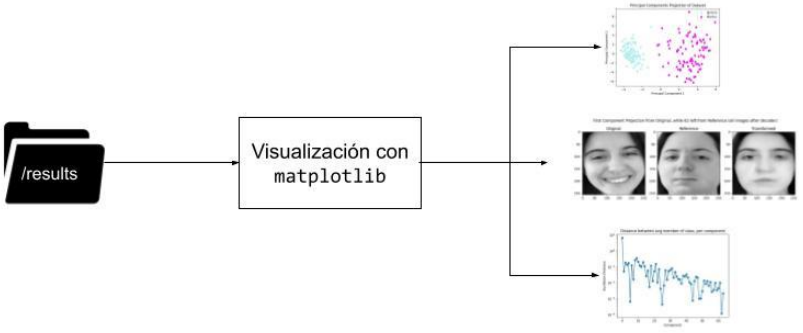


***Figura 14**: Diagrama de flujo sobre visualización de resultados. Las imágenes a derecha son resultados reales obtenidos a partir de los experimentos.*

6. Arquitectura del Autoencoder

El conjunto de datos que se estará utilizando es el resultante de la sección **5.1.1.**: 100 imágenes por sujeto (Marina e Ignacio), en blanco y negro, de dimensiones 256x256. La separación en conjuntos de entrenamiento y testeo será 80-20 respectivamente.

Para generar el *autoencoder* necesario, se tomó como base la arquitectura diseñada por Adrián Rosenbrock, en un artículo de su sitio web, PyImageSearch [2]. Dicha arquitectura usaba el conjunto de datos MNIST, un conjunto de dígitos del 0 al 9 escritos a mano, en blanco y negro, de dimensiones 28x28. Fue a partir de esta arquitectura que se intentó encontrar la mejor combinación de parámetros para el conjunto de imágenes de Marina e Ignacio. Al *autoencoder* de Rosenbrock lo denominaremos “*autoencoder* base”.

El proceso de decisión para la configuración óptima será definido a continuación, en el cual se distinguen cuatros secciones. En primer lugar, se describirá el *autoencoder* base. Luego, la elección de parámetros de la arquitectura a modificar y los valores que los mismos toman en la experimentación. En la tercera sección, se realizó un filtro de arquitecturas para analizar con más detalle. Por último, se diseñó una metodología de selección basada en métricas provistas por Rekognition, el sistema de reconocimiento facial.

Se decidió utilizar un *autoencoder* previo, en lugar de generar uno desde cero, porque al probar el *autoencoder* de Rosenbrock con el conjunto de datos de este trabajo, los resultados de reconstrucción (observados de manera subjetiva, por los autores) fueron muy buenos. Se tomó como supuesto que esta arquitectura tenía el potencial para generar resultados satisfactorios, en tanto se realice un análisis más estructurado de sus capacidades.

Por otra parte, se debió tener en cuenta el factor tiempo: el *hardware* del cual se dispone es aquél que provee una computadora de uso general. Esta limitación se trasladaba a las expectativas en cuanto a la calidad de las imágenes reconstruidas. Para entrenar las distintas arquitecturas, tanto el conjunto de datos como el mismo *autencoder* eran factores que influían fuertemente en el tiempo que llevaría el entrenamiento. Dado que el espacio de posibles *autoencoders* que pueden estudiarse es desmesurado, resulta coherente tomar un

punto de “anclaje” (el *autoencoder* base), a partir del cual se puede buscar una mejoría en la reconstrucción.

6.1. Autoencoder base

El primer punto a destacar es que se trata de un *autoencoder* convolucional, también llamado CAE, construido con la librería de Python, Keras (versión 2.4.3). Al momento de decidir qué tipo de red neuronal se utilizaría, se decidió enfocar los recursos de búsqueda en una con capas convolucionales, dado el éxito demostrado al momento de trabajar con imágenes.

Como se mencionó en la sección **2.7.**, el *autoencoder* puede pensarse como la composición de dos componentes: el codificador y el decodificador. A partir de esta red neuronal, se aplicaron algunos cambios para construir lo que se denominó “*autoencoder* base corregido”, que se abreviará como ABC.

Lo primero que se debió adaptar fueron las dimensiones de las imágenes, ya que no son las mismas. Por otro lado, se aumentó la dimensión de la capa latente, de 16 a 64 (en consecuencia del cambio en dimensión de las imágenes del conjunto de datos) y se añadió una tercera capa convolucional.

En el ABC se mantuvo la función de costo *Minimum Squared Error* (MSE), los parámetros del optimizador, los parámetros de las funciones de activación, el tipo de *padding* y el corrimiento de ventana al momento de aplicar los filtros de la capa convolucional, llamado *strides*. A continuación se muestra un resumen de la arquitectura del codificador y decodificador del ABC (en el Apéndice, en Arquitectura del autoencoder base, se encuentra el resumen correspondiente al *autoencoder* base), obtenido a partir de la función `summary` provista por `Keras`.

```
Model: "encoder"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 256, 256, 1)]     0
_________________________________________________________________
conv2d (Conv2D)              (None, 128, 128, 16)      160
_________________________________________________________________
leaky_re_lu (LeakyReLU)      (None, 128, 128, 16)      0
_________________________________________________________________
batch_normalization (BatchNo (None, 128, 128, 16)      64
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 64, 64, 32)        12832
_________________________________________________________________
leaky_re_lu_1 (LeakyReLU)    (None, 64, 64, 32)        0
_________________________________________________________________
batch_normalization_1 (Batch (None, 64, 64, 32)        128
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 32, 32, 64)        100416
_________________________________________________________________
leaky_re_lu_2 (LeakyReLU)    (None, 32, 32, 64)        0
_________________________________________________________________
```

```
batch_normalization_2 (Batch (None, 32, 32, 64)        256
_________________________________________________________________
flatten (Flatten)            (None, 65536)             0
_________________________________________________________________
dense (Dense)                (None, 64)                4194368
_________________________________________________________________
leaky_re_lu_3 (LeakyReLU)    (None, 64)                0
=================================================================
Total params: 4,308,224
Trainable params: 4,308,000
Non-trainable params: 224
```

```
Model: "decoder"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_2 (InputLayer)         [(None, 64)]              0
_________________________________________________________________
dense_1 (Dense)              (None, 65536)             4259840
_________________________________________________________________
reshape (Reshape)            (None, 32, 32, 64)        0
_________________________________________________________________
conv2d_transpose (Conv2DTran (None, 64, 64, 64)        200768
_________________________________________________________________
leaky_re_lu_4 (LeakyReLU)    (None, 64, 64, 64)        0
_________________________________________________________________
batch_normalization_3 (Batch (None, 64, 64, 64)        256
_________________________________________________________________
conv2d_transpose_1 (Conv2DTr (None, 128, 128, 32)      51232
_________________________________________________________________
leaky_re_lu_5 (LeakyReLU)    (None, 128, 128, 32)      0
_________________________________________________________________
batch_normalization_4 (Batch (None, 128, 128, 32)      128
_________________________________________________________________
conv2d_transpose_2 (Conv2DTr (None, 256, 256, 16)      4624
_________________________________________________________________
leaky_re_lu_6 (LeakyReLU)    (None, 256, 256, 16)      0
_________________________________________________________________
batch_normalization_5 (Batch (None, 256, 256, 16)      64
_________________________________________________________________
conv2d_transpose_3 (Conv2DTr (None, 256, 256, 1)       145
_________________________________________________________________
activation (Activation)      (None, 256, 256, 1)       0
=================================================================
Total params: 4,517,057
Trainable params: 4,516,833
Non-trainable params: 224
```

Tanto para el codificador como decodificador, puede observarse al final de su descripción que se definen parámetros entrenables y no entrenables. En el resumen que `Keras` otorga sobre la arquitectura de una red neuronal, los parámetros no entrenables son aquellos que no serán optimizados durante las iteraciones del *autoencoder*, sino que deben definirse a priori. Ejemplos de parámetros no entrenables podrían ser la cantidad de capas ocultas, cantidad de nodos por capa, etc. Por otra parte, los parámetros entrenables (como su

nombre lo indica) son aquellos que se modifican a medida que avanza el entrenamiento, como pueden ser los pesos de las neuronas.

6.2. Análisis de parámetros

El primer desafío al momento de estudiar el *autoencoder* base fue decidir qué parámetros serían sujetos a modificación y estudio, además de elegir cuáles serían los valores que los mismos tomarían.

Para el análisis de desempeño de cada uno de los parámetros se decidió estudiar, por época de entrenamiento, la reconstrucción de las imágenes en tanto se minimizaba el sobreentrenamiento. Esto quiere decir que no sólo se observaba la reconstrucción de las imágenes del conjunto de testeo, sino que también se buscaba que dicho valor no se encontrara a un distancia significativa del error de reconstrucción en el conjunto de entrenamiento (en referencia a las otras diferencias correspondientes a los valores que toma el parámetro en cuestión).

Para la reconstrucción, se tomó el promedio de la diferencia entre la salida del autoencoder, $\hat{x}$, y la imagen original del conjunto de entrenamiento, x, según la ecuación (3) que representa el valor de pérdida del *autoencoder*, al cual denominaremos *Loss*. El parámetro n corresponde a la cantidad de elementos del conjunto de testeo.

$$Loss = \frac{1}{n}\sum_{i=1}^{n} \|\hat{x} - x\| \quad (3)$$

Por otra parte, para examinar el sobreentrenamiento, se tomó la diferencia entre el valor de pérdida promedio para el conjunto de testeo $Loss^{test}$, reutilizando la ecuación (3), y el valor de pérdida promedio para el conjunto de entrenamiento $Loss^{train}$. Esta diferencia se la denomina *Loss Gap*, definida según la ecuación (4).

$$Loss\ Gap = Loss^{test} - Loss^{train} \quad (4)$$

A continuación, se encuentran las secciones correspondientes a los parámetros que han sido analizados. En la sección Apéndice, Parámetros iniciales del ABC, se puede encontrar una tabla que lista los parámetros iniciales del *autoencoder*.

Una vez que se han obtenido los datos correspondientes para estudiar los parámetros, se seleccionarán valores de acuerdo al siguiente criterio:

1. Se analiza en conjunto el gráfico de *Loss* y de *Gap Loss* (correspondiente a la diferencia entre conjunto de entrenamiento y testeo) para analizar reconstrucción y sobreentrenamiento.
2. Se analizan las reconstrucciones de las imágenes (en ***Fig. 15*** podemos ver un ejemplo de las mismas) para determinar si dicho valor se mantiene en la selección. En el Apéndice se pueden encontrar las reconstrucciones completas, por parámetro modificado.

***Figura 15**: reconstrucciones visuales del autoencoder según valores de la tasa de aprendizaje del optimizador Adam. Para cada parámetro, se contaba con dichas reconstrucciones.*

6.2.1. Tasa de aprendizaje del optimizador

Al momento de entrenar una red neuronal es necesario tener presente la importancia de la elección del optimizador, así como sus parámetros. En el ABC se utiliza como optimizador el algoritmo Adam. Se decidió mantener este algoritmo por ser computacionalmente eficiente, de gran relevancia al tener en cuenta que se utiliza *hardware* de propósito general, y por demostrar un desempeño mejor a otros optimizadores, como AdaGrad, RMSProp, Stochastic Gradient Descent (SGD) y AdaDelta, en la evaluación realizada con una red neuronal convolucional, tanto con el conjunto de datos MNIST como con CIFRAR10 [3].

Si bien Adam es un optimizador muy popular, existe el debate sobre su superioridad al momento de generalizar (evitar sobreentrenamiento) sobre un conjunto de datos en contraposición a SGD [4]. A pesar de ello, se determinó acotar la evaluación al uso de Adam con variaciones en su tasa de aprendizaje por la eficiencia computacional mencionada anteriormente. En caso de no obtener resultados satisfactorios, se tomaría como alternativa el cambio de optimizador.

Los valores que se tomaron en cuenta varían entre 10^{-1} y 10^{-5} con un paso de 10^{-1}.

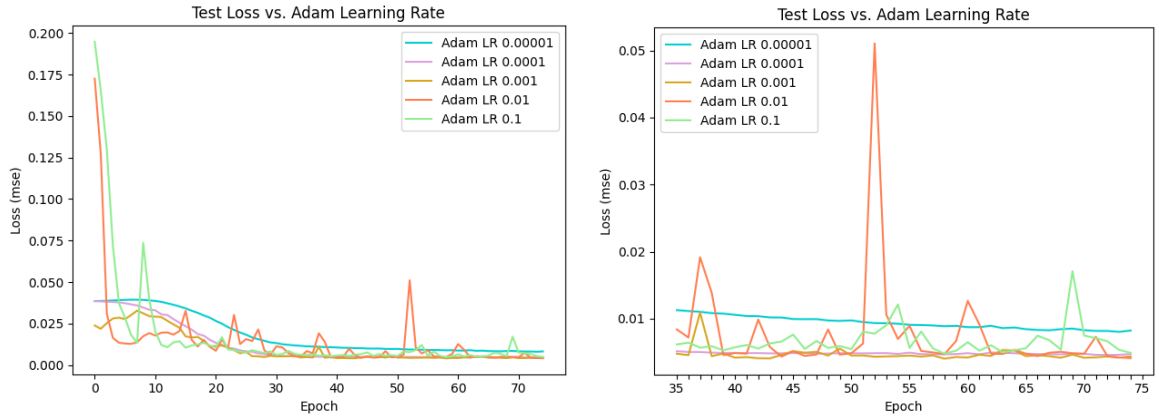


***Figura 16**: Loss para el conjunto de testeo durante 75 épocas, según tasa de aprendizaje del optimizador. A la derecha, los valores se encuentran a partir de la época 35 para mejor interpretación de los resultados.*

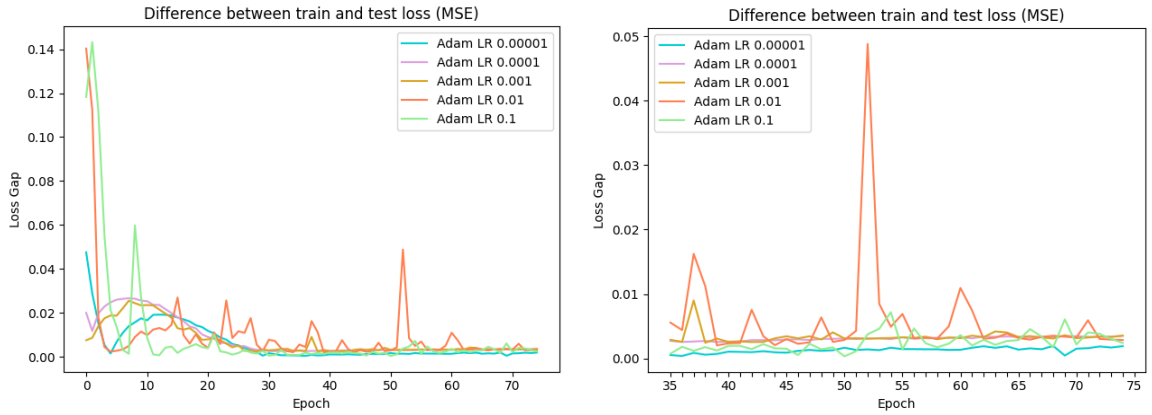


***Figura 17**: Gap Loss a lo largo del testeo de un autoencoder durante 75 épocas, según tasa de aprendizaje del optimizador. A la derecha, los valores se encuentran a partir de la época 35 para mejor interpretación de los resultados.*

Según el gráfico en ***Fig. 16***, es posible notar que la tasa de aprendizaje de 10^{-3} y 10^{-4} son de los valores más bajos en cuanto a la función de error. Por otra parte, en ***Fig. 17***, el valor de *Gap Loss* también se mantiene bajo con respecto a otras tasas de aprendizaje. Como complemento, la reconstrucción de las imágenes era satisfactoria según los autores.

Por último, se seleccionó la tasa de aprendizaje 10^{-5} ya que su *Gap Loss* es el más bajo de todos. Al momento de analizar la reconstrucción de imágenes, se pudo notar que éstas eran borrosas, lo cual podría ser indicio de que a esta arquitectura le faltaban épocas de entrenamiento. El gráfico de la derecha en ***Fig. 16*** acompaña esta reflexión, ya que se puede ver que la tendencia del valor de pérdida para 10^{-5} es decreciente.

6.2.2. Cantidad de capas convolucionales

Con respecto a la arquitectura de la red neuronal, una de las cosas que se buscó analizar fue la cantidad de capas convolucionales adecuadas para la red. Recordar que se comenzó con 3 capas convolucionales, con su respectiva función de activación y posterior *BatchNormalization*, pero se encontraba presente la duda de si sería suficiente para el conjunto de datos que se utiliza en la investigación. Otro elemento que se fue variando a

medida que se aumentaba la cantidad de capas (aumentando la cantidad de filtros en cada capa), fue el tamaño de núcleo para la capa en cuestión.

El objetivo de la variación de este parámetro es analizar si la arquitectura de la red es “suficiente” para aprender el conjunto de datos. Por otra parte, es necesario tener cuidado de no caer en un exceso de capas, lo cual puede ser contraproducente para la capacidad de generalización de la misma.

Los valores analizados en esta sección fueron:

- 3 capas convolucionales de 16 (núcleo 3x3), 32 (núcleo 5x5) y 64 (núcleo 7x7) filtros respectivamente.
- 4 capas convolucionales de 16 (núcleo 3x3), 32 (núcleo 5x5), 64 (núcleo 7x7) y 128 (núcleo 7x7) filtros respectivamente.
- 5 capas convolucionales de 16 (núcleo 3x3), 32 (núcleo 5x5), 64 (núcleo 7x7), 128 (núcleo 7x7) y 256 (núcleo 7x7) filtros respectivamente.

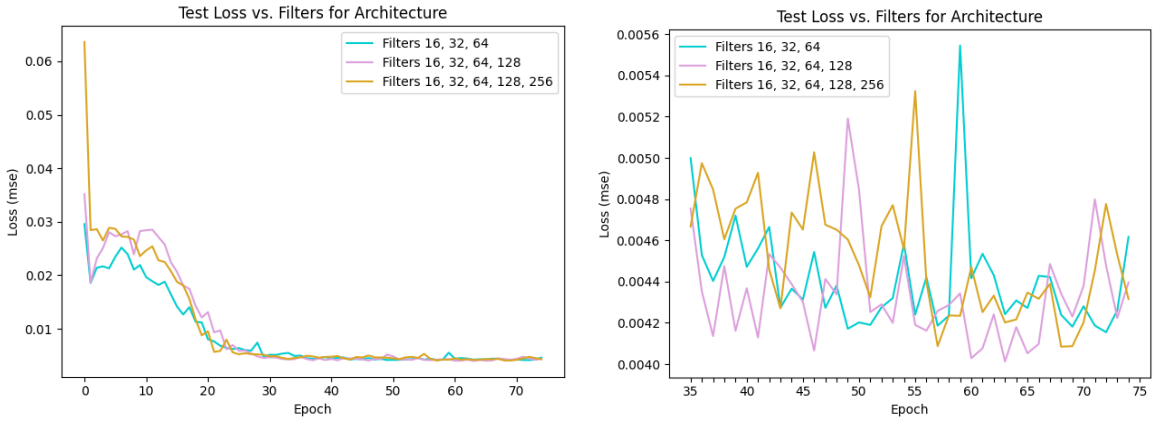


***Figura 18**: Loss para el conjunto de testeo durante 75 épocas, según cantidad de capas convolucionales. A la derecha, los valores se encuentran a partir de la época 35 para mejor interpretación de los resultados.*

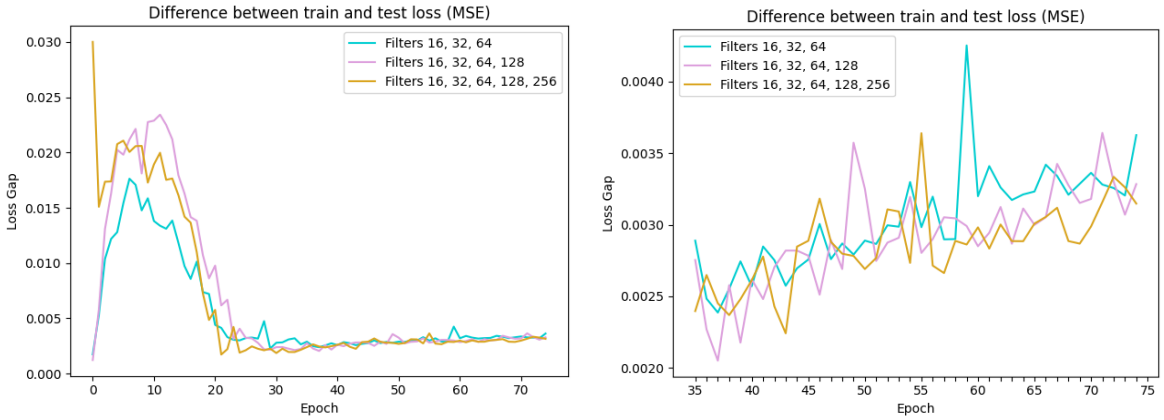


***Figura 19**: Gap Loss a lo largo del testeo de un autoencoder durante 75 épocas, según cantidad de capas convolucionales. A la derecha, los valores se encuentran a partir de la época 35 para mejor interpretación de los resultados.*

En este caso, es posible observar que los valores que se estudiaron arrojan resultados similares. En los datos que muestran los gráficos, no se encuentra razón para creer que una cantidad de capas (con sus respectivas especificaciones) es mejor que otra. Analizando las reconstrucciones, se notó que aquellas arquitecturas con 3 y 5 capas muestran los mejores resultados (e inclusive pareciera que el de 5 capas es el más satisfactorio).

Si bien se podrían haber seleccionado 5 capas para la sección posterior (o ambos valores para no dejar afuera posibilidades), se decidió mantener únicamente 3 capas convolucionales. Esto se debió a que no hay una superioridad significativa en cuanto a *Loss*, *Gap Loss* y las imágenes reconstruidas (si bien existe una pequeña mejoría, las imágenes no son las mismas para ambas arquitecturas, con lo cual esto no es concluyente). La ventaja de seleccionar el valor de 3 capas es que la red tarda menos en entrenar.

6.2.3. Cantidad de datos para actualización de gradiente

Otro parámetro que se decidió analizar fue el tamaño del *batch* durante el entrenamiento. Este valor define la cantidad de elementos del conjunto de entrenamiento que se utilizan para realizar una actualización del gradiente. Esta actualización busca ser una estimación del "verdadero" gradiente, ya que el mismo debería calcularse con todos los datos del conjunto.

Los valores que fueron tomados para este parámetro son 4, 8, 12 y 32.

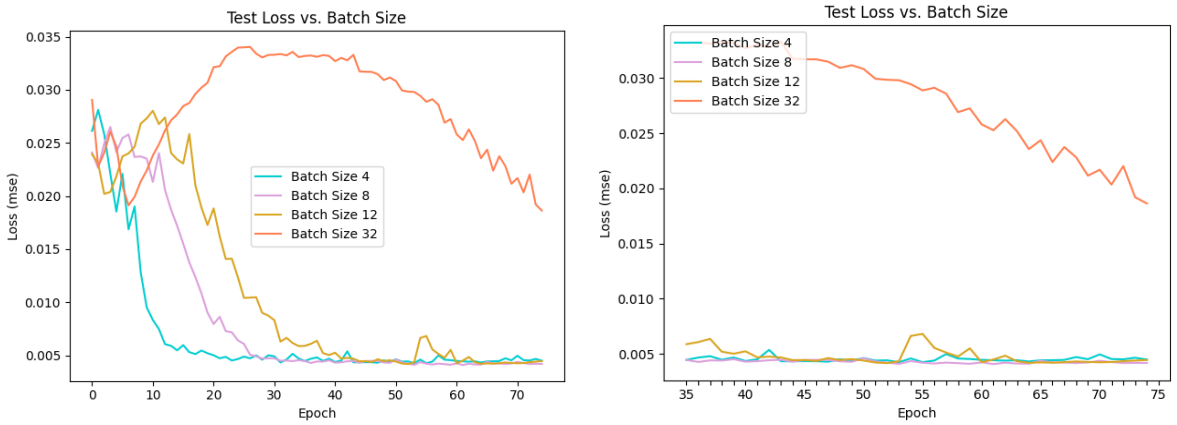


***Figura 20**: Loss para el conjunto de testeo durante 75 épocas, según cantidad de datos para actualización. A la derecha, los valores se encuentran a partir de la época 35 para mejor interpretación de los resultados.*

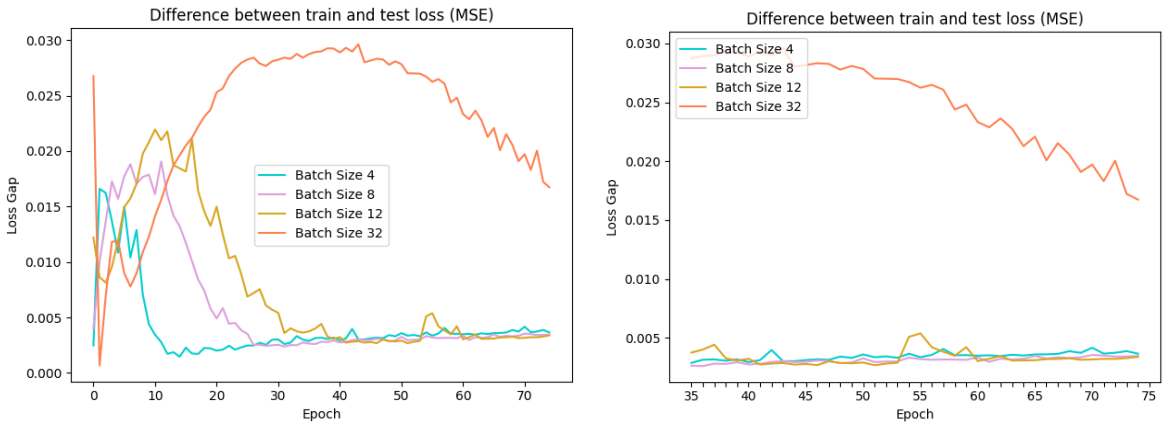


***Figura 21**: Gap Loss a lo largo del testeo de un autoencoder durante 75 épocas, cantidad de datos para actualización. A la derecha, los valores se encuentran a partir de la época 35 para mejor interpretación de los resultados.*

Los valores seleccionados fueron 8 y 32. Como es posible ver en ***Fig. 20*** y ***Fig. 21***, los mejores resultados son para los tamaños 4 y 8, lo cual se corresponde al analizar la reconstrucción de imágenes. Si bien se podría haber elegido el valor 4 para la sección posterior, el valor 8 pareciera estar levemente por debajo del 4 en *Loss* y, además, ya era el valor por defecto en los parámetros iniciales. Añadir el valor 4 hubiera resultado en más *autoencoders* para estudiar en las siguientes secciones.

Por otra parte, el valor 32 fue seleccionado por razones similares a la tasa de aprendizaje 10^{-5}, en la sección **6.2.1.** Si bien el desempeño en ***Fig. 20*** y ***Fig. 21*** es bajo con respecto a los otros valores, se nota claramente una tendencia decreciente. Adicionalmente, las imágenes reconstruidas son algo borrosas, lo cual puede ser indicio de falta de entrenamiento. Se decidió mantener el valor para analizar si las expresiones más exageradas (que suelen ser las de peor reconstrucción) podían ser aprendidas con estos modelos, al combinarlos con más épocas de entrenamiento.

6.2.4. Dimensión de la capa latente

La elección de dimensión de la capa latente era un parámetro de especial interés para este trabajo, dado que serían las representaciones de las imágenes en este espacio los elementos que luego se usarían para el análisis de componentes principales.

Para la obtención de datos, se tomaron 3, 9, 16, 32, 64 y 128 neuronas.

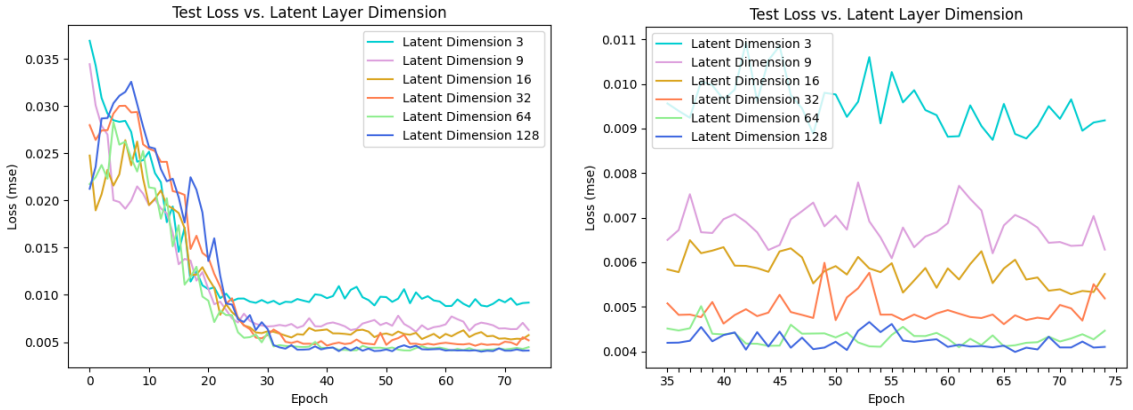


***Figura 22**: Loss para el conjunto de testeo durante 75 épocas, según cantidad de neuronas en la capa latente. A la derecha, los valores se encuentran a partir de la época 35 para mejor interpretación de los resultados.*

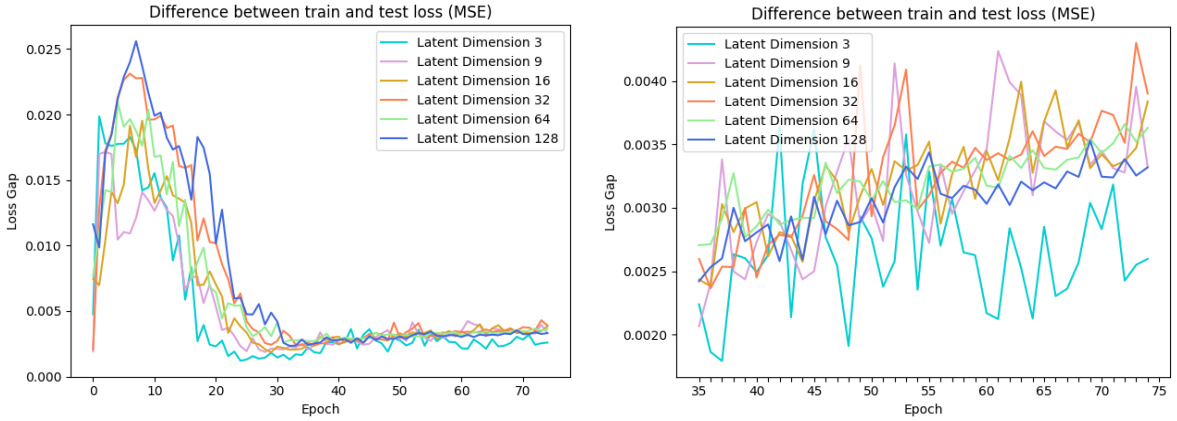


***Figura 23**: Gap Loss a lo largo del testeo de un autoencoder durante 75 épocas, según cantidad de neuronas en la capa latente. A la derecha, los valores se encuentran a partir de la época 35 para mejor interpretación de los resultados.*

Los valores seleccionados fueron 64 y 128. Como se puede observar en ***Fig. 22***, alcanzan los valores más bajos comparado a utilizar 3, 9, 16 o 32 dimensiones en la capa latente. Un detalle no menor es que todos los valores parecieran haber alcanzado una especie de convergencia en este gráfico. Esto trajo como consecuencia que se descarte la capa latente de dimensión 3 que, en ***Fig. 23***, pareciera tener el mejor desempeño en cuanto a *Gap Loss* (bajo sobreentrenamiento). Como el gráfico de ***Fig. 22*** no proveía indicio de que esta arquitectura de capa latente de dimensión 3 fuera a mejorar, se decidió descartarla.

Una de las ventajas de tener una capa latente con bastantes dimensiones es la posibilidad de alcanzar una mejor reconstrucción. Por otra parte, al momento de manipular el vector correspondiente a la representación de una imagen en el espacio latente, tenemos más "lugares" donde hacerlo. A su vez, trae como posible riesgo que la red memorice los datos en lugar de generalizar dado que el conjunto de imágenes no es grande.

6.2.5. Función de activación

Como puede observarse tanto en el resumen de arquitectura del codificador como decodificador, a la salida de cada capa convolucional se aplica la función de activación

Leaky ReLU. Una de las principales ventajas de esta función de activación, que comparte con su predecesora ReLU, es el costo computacional en relación a otras funciones de activación no lineales, como puede ser la tangente hiperbólica o la función sigmoidea. Teniendo presente que los costos computacionales son importantes por las restricciones ya mencionadas en la sección **4.4.**, se decidió mantener esta función de activación y cambiar únicamente el parámetro α que representa la pendiente de la misma para el caso de valores menores a cero.

Podría haberse utilizado ReLU en lugar de Leaky ReLU, pero se decidió no tomar este camino por simplicidad ya que en una primera iteración, probando con el *autoencoder* base, habíamos obtenido resultados satisfactorios. Según [5], la función de activación ReLU puede provocar que varias neuronas “mueran” (su salida termina siendo siempre cero) con lo cual Leaky ReLU sería la alternativa para minimizar este problema. Esto es un punto a favor de la función de activación aquí elegida pero según este mismo artículo, esta problemática no afecta de manera significativa al entrenamiento, al menos utilizando el conjunto de datos MNIST, con lo cual ReLU sigue siendo una opción viable. Inclusive, en algunos foros de debate, los miembros de la comunidad no han notado diferencias significativas en el uso de una u otra.

Los valores con los cuales se varió la pendiente de la función de activación son 0.01, 0.05, 0.1, 0.3, 0.6, 0.9 y 1.3. Para evitar aumentar la dificultad de análisis, se usó el mismo valor de α para todas las instancias en que se usa la función de activación, tanto para el codificador como decodificador.

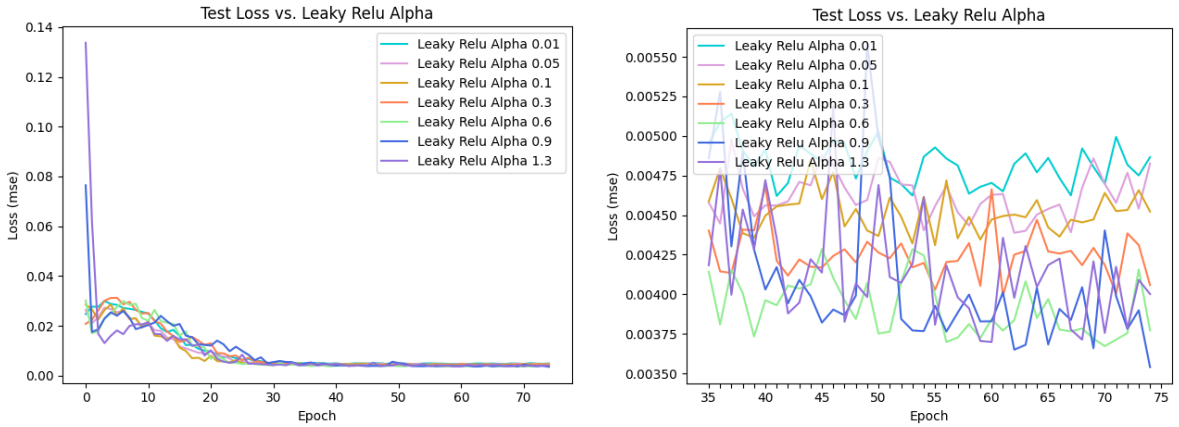


***Figura 24**: Loss para el conjunto de testeo durante 75 épocas, según el valor α de la función Leaky ReLU. A la derecha, los valores se encuentran a partir de la época 35 para mejor interpretación de los resultados.*

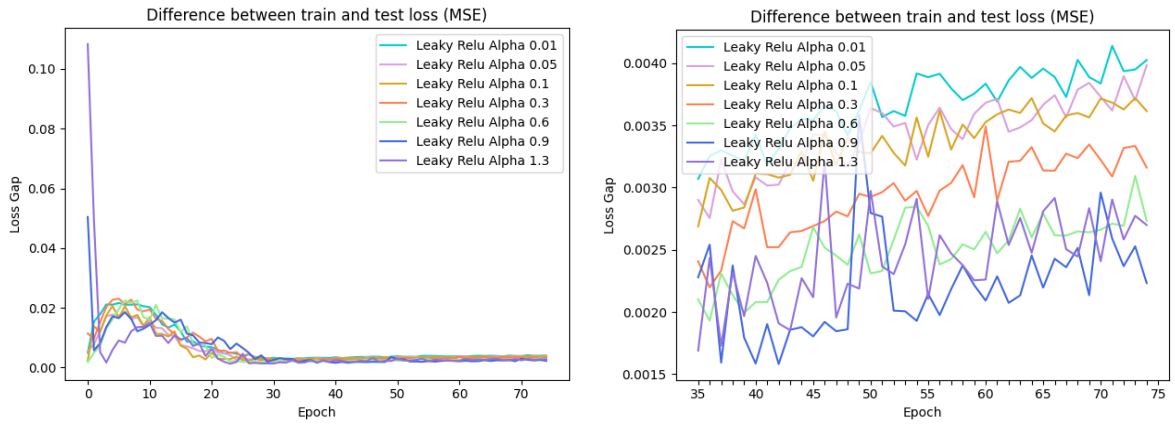


***Figura 25**: Gap Loss a lo largo del testeo de un autoencoder durante 75 épocas, según el valor α de la función Leaky ReLU. A la derecha, los valores se encuentran a partir de la época 35 para mejor interpretación de los resultados.*

Los valores seleccionados fueron 1.3 y 0.9. Al momento de observar ***Fig. 24*** y ***Fig. 25***, 0.6, 0.9 y 1.3 parecieran tener el mejor desempeño, a pesar de su comportamiento errático. Sin embargo, al observar las reconstrucciones que generan los mismos, es posible notar que el valor α de 0.6 reconstruye ciertas imágenes de forma no satisfactoria. Si bien fue mencionado el problema de la reconstrucción al momento de sacar conclusiones, la necesidad de limitar las combinaciones posibles para la sección siguiente fue lo que empujó a los autores a eliminar 0.6 de la selección.

6.3. Filtro de arquitecturas

Una de las principales falencias del análisis de parámetros, fue la aleatoriedad en la selección de las imágenes para analizar la reconstrucción. Dado que estudiar 200 imágenes por *autoencoder* es una tarea tediosa, que probablemente provocará que los últimos revisados no tuvieran un criterio similar a los primeros, se decidió elegir 6 imágenes aleatorias por *autoencoder* para el análisis. En su momento no pareció un error seleccionar estas imágenes al azar, pero luego fue evidente que aquellos *autoencoders* que tuvieran imágenes con expresiones exageradas en esta selección eran los que tenían las peores reconstrucciones.

La razón por la cual no se utilizó una selección uniforme fue porque no se habían almacenado los modelos entrenados, lo cual fue un aprendizaje para las siguientes etapas del proyecto. Se decidió continuar con los *autoencoders* correspondientes a los valores seleccionados, teniendo en consideración que no debía tomarse como una prueba concluyente, como fue mencionado varias veces en la sección **6.2.** Al realizar una rápida revisión de las 200 imágenes del grupo reducido que se describe a continuación, se pudieron ver buenas reconstrucciones para la mayoría de las imágenes. De todas formas, es importante tener presente que pudo haber sucedido que un *autoencoder* más adecuado para este conjunto de datos hubiera sido descartado.

Como se mencionó previamente, los valores de parámetros seleccionados fueron producto del análisis de *Loss*, *Gap Loss* y las reconstrucciones de imágenes. A modo de resumen, serán listados aquí las elecciones finales:

- Tasa de aprendizaje del optimizador: 10^{-3}, 10^{-4} y 10^{-5}.

- Cantidad de capas convolucionales: 3
- Cantidad de datos para actualización del gradiente: 8 y 32
- Dimensión de la capa latente: 64 y 128.
- Función de activación: 0.9 y 1.3.

Una de las opciones consideradas fue realizar la combinatoria de estos parámetros y analizar todos los modelos correspondientes. Esto daba como resultado 24 modelos pero se decidió no tomar este camino por las siguientes razones: en primer lugar, la cantidad de épocas de entrenamiento de los *autoencoders* anteriores fue 75 y, como se mencionó, ciertos valores de parámetros necesitaban más iteraciones para lograr una reconstrucción comparable a las arquitecturas que pudieron converger. En segundo lugar, la separación entrenamiento-testeo del conjunto de datos se mantuvo fija en todo momento. Era de interés realizar validación cruzada para analizar cómo esta separación estaba afectando al desempeño de los *autoencoders*.

Se tomó como alternativa realizar una especie de "filtro" de los valores anteriores, reduciendo la cantidad de arquitecturas a estudiar en la tercera etapa a 8. Por un lado se encontrarían los *autoencoders* cuyas épocas de entrenamiento se mantendrían similares al paso anterior. Esto da como resultado la selección que se presenta en ***Tabla 1***.

Nombre	Épocas	Tamaño batch	Filtros	Leaky ReLU Alpha	Adam LR	Capa Latente	Núcleos
ID1	80	8	[16, 32, 64]	0,2	1,00E-03	64	[(3, 3), (5, 5), (7, 7)]
ID2	80	8	[16, 32, 64]	0,2	1,00E-04	64	[(3, 3), (5, 5), (7, 7)]
ID3	80	8	[16, 32, 64]	0,9	1,00E-03	64	[(3, 3), (5, 5), (7, 7)]
ID4	80	8	[16, 32, 64]	1,3	1,00E-03	64	[(3, 3), (5, 5), (7, 7)]

***Tabla 1**: selección de los autoencoders, manteniendo cantidad de épocas.*

Se mantuvieron los valores iniciales que se observan en la sección Apéndice, Parámetros iniciales del ABC y, para cada uno, se ha modificado la variación correspondiente para dos parámetros: tasa de aprendizaje (Adam LR en ***Tabla 1***) y parámetro α de la función de activación Leaky ReLU (Leaky ReLU Alpha en ***Tabla 1***).

Nótese que se ha descartado el uso del *batch* 32 y de tasa de aprendizaje de 10^{-5}, ya que éstos requieren más iteraciones de entrenamiento. Por otra parte, se decidió no utilizar la capa latente de dimensión 128 debido a que los resultados obtenidos cuando se utilizó el valor por defecto 64 eran satisfactorios.

Nombre	Épocas	Tamaño batch	Filtros	Leaky ReLU Alpha	Adam LR	Capa Latente	Núcleos
ID5	1300	32	[16, 32, 64]	1,3	1,00E-05	64	[(3, 3), (5, 5), (7, 7)]
ID6	1300	32	[16, 32, 64]	1,3	1,00E-05	128	[(3, 3), (5, 5), (7, 7)]
ID7	1300	32	[16, 32, 64]	0.9	1,00E-05	64	[(3, 3), (5, 5), (7, 7)]
ID8	1300	32	[16, 32, 64]	0.9	1,00E-05	128	[(3, 3), (5, 5), (7, 7)]

***Tabla 2**: selección de los autoencoders, aumentando cantidad de épocas.*

En segundo lugar, se tienen aquellos autoencoders que requerían más épocas de entrenamiento, los cuales se listan en ***Tabla 2***. Aquí puede verse que se utilizan los valores

para tasa de aprendizaje del optimizador y tamaño del *batch* que necesitan más iteraciones. Por otra parte, se agregó la combinatoria de los valores seleccionados para la dimensión de la capa latente y el parámetro α de la función de activación Leaky ReLU.

Para los *autoencoders* en ***Tabla 1*** se realizó validación cruzada con parámetro k=5, siendo k el valor que determina la cantidad de bloques de fragmentación. Esto no fue llevado a cabo para los *autoencoders* en ***Tabla 2*** por la cantidad de tiempo que llevaba entrenarlos: 14.5 horas para cada *autoencoder*, lo cual daría un total (incluyendo validación cruzada) de 232 horas, aproximadamente 10 días.

Esta decisión dió como resultado un total de 24 modelos: 20 como consecuencia de los experimentos de validación cruzada de ID_1 a ID_4 y los últimos cuatro son aquellos de ***Tabla 2***, con 1300 épocas de entrenamiento cada uno.

6.4. Metodología de comparación con Rekognition

El objetivo de la última etapa, mencionado previamente, es elegir el *autoencoder* que se va a utilizar para los experimentos. Para cada uno de los modelos resultantes del paso anterior, se generaron dos carpetas, que se denominarán `/recon` y `/data`. En la carpeta `/data` se almacenan las imágenes del conjunto de datos sin modificación. Nótese que la carpeta `/data` tendrá la misma información para cada uno de los modelos. Por otra parte, en la carpeta `/recon` se encontrarán las imágenes reconstruidas por el modelo en cuestión. Cada imagen en la carpeta `recon` tendrá su correspondiente en la carpeta `/data`.

Para evaluar el desempeño de los *autoencoders* se resolvió utilizar la herramienta que, en última instancia, se tiene como objetivo de ataque: Amazon Rekognition. De las funcionalidades mencionadas en la sección **5.2.2.**, para obtención de las siguientes métricas, con respecto a las imágenes de la carpeta `/recon`:

- ***Similarity***: para obtener dicho valor se usa la funcionalidad de comparación de dos imágenes: la imagen reconstruida almacenada en `/recon` contra su respectiva imagen original en `/data`. Esta similitud está definida por el algoritmo de Amazon Rekognition y su cálculo funciona como una caja negra para los autores.
- ***Confidence***: este parámetro se encuentra en la respuesta obtenida a partir de la detección de un individuo e indica la confianza con la cual Amazon Rekognition determina que encontró el rostro en la imagen.
- ***Quality Sharpness***: este valor surge de la funcionalidad de detección de rostros. El objetivo es analizar esta métrica para la imagen reconstruida, de manera tal que se pueda evaluar qué modelo toma valores más altos en cuanto a calidad.
- ***Quality Brightness***: este valor tiene un objetivo similar al mencionado anteriormente en lo que respecta al análisis de calidad de la imagen. Se obtiene con la misma funcionalidad y se realiza sobre la imagen reconstruida por el *autoencoder*.

Para cada modelo se tomará el promedio y desvío estándar de las métricas obtenidas. A continuación se muestran los resultados (en el Apéndice, Datos obtenidos con Rekognition para selección del Autoencoder se puede encontrar la tabla con los datos correspondientes). Es importante destacar que la métrica prioritaria para la selección del modelo es la *Similarity*. Se buscaron los valores más altos de la misma que pudieran

alcanzar los modelos, en conjunto con la confianza con que Amazon Rekognition reconoce que hay un rostro en la imagen reconstruida por el *autoencoder*. Otro aspecto complementario a la decisión fue el análisis de calidad. Si bien la calidad no determinó la elección, sí fue necesario corroborar que las imágenes del modelo elegido por la métrica *Similarity* no perdieran calidad de manera significativa en referencia a los otros modelos.

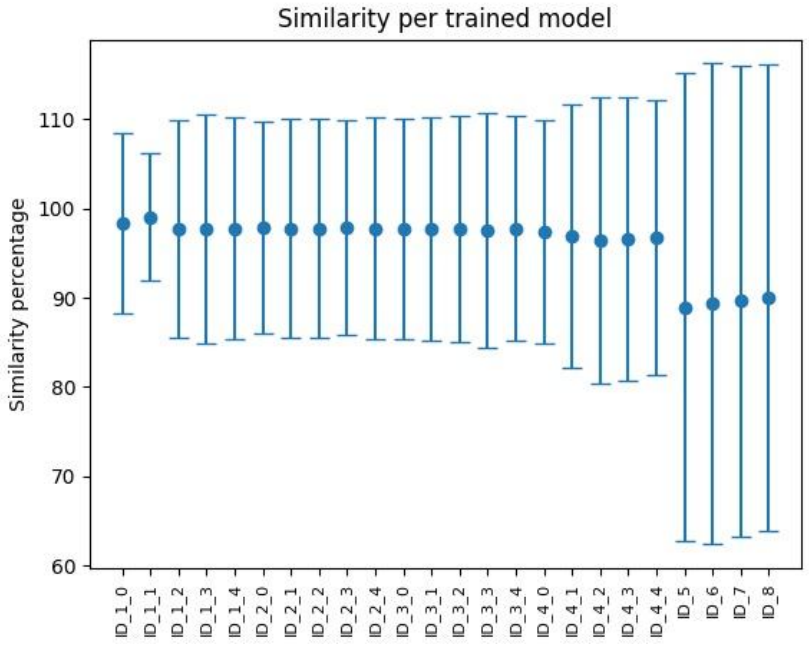


***Figura 26**: resultados obtenidos con respecto a la métrica Similarity para los modelos entrenados. Aquellos cuya etiqueta sea de la forma ID_n_m se corresponden a los modelos con validación cruzada.*

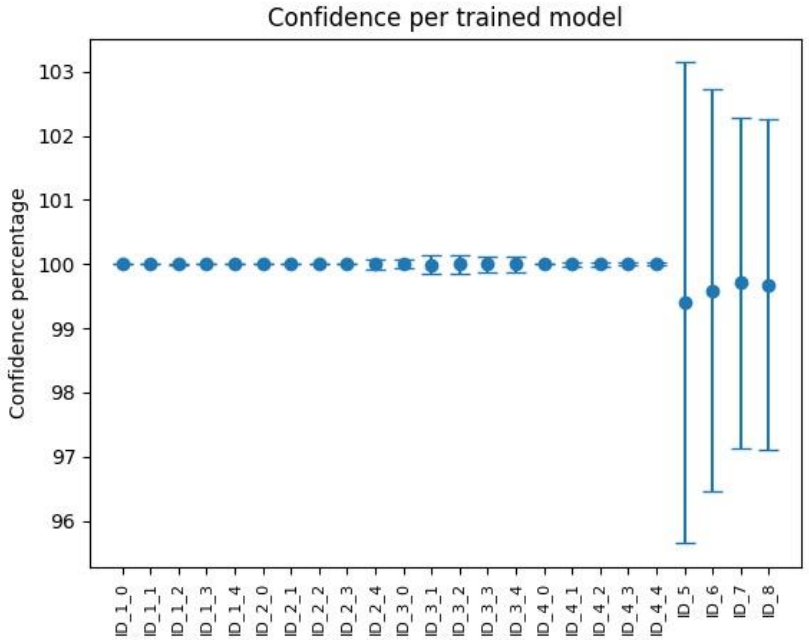


***Figura 27**: resultados obtenidos con respecto a la métrica Confidence para los modelos entrenados. Aquellos cuya etiqueta sea de la forma ID_n_m se corresponden a los modelos con validación cruzada.*

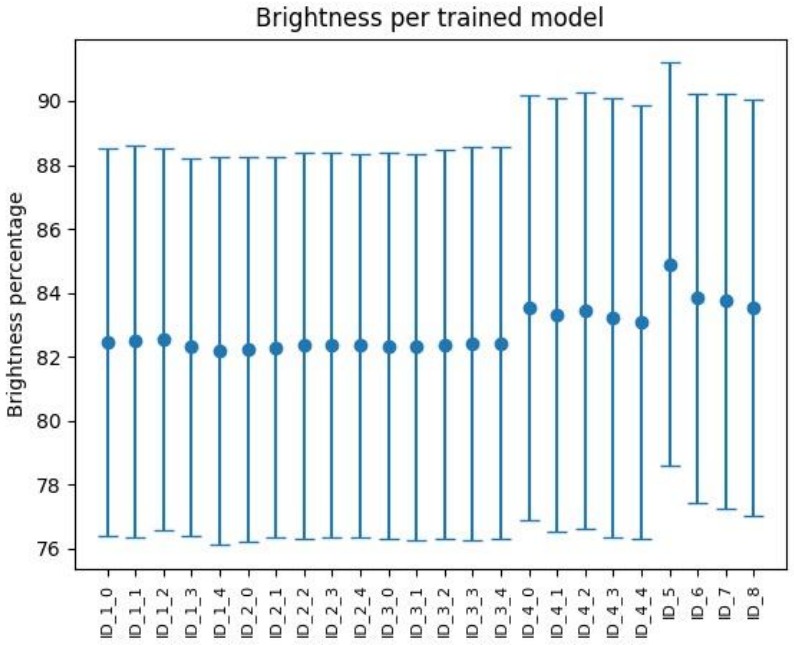


***Figura 28**: resultados obtenidos con respecto a la métrica Quality Brightness para los modelos entrenados. Aquellos cuya etiqueta sea de la forma ID_n_m se corresponden a los modelos con validación cruzada.*

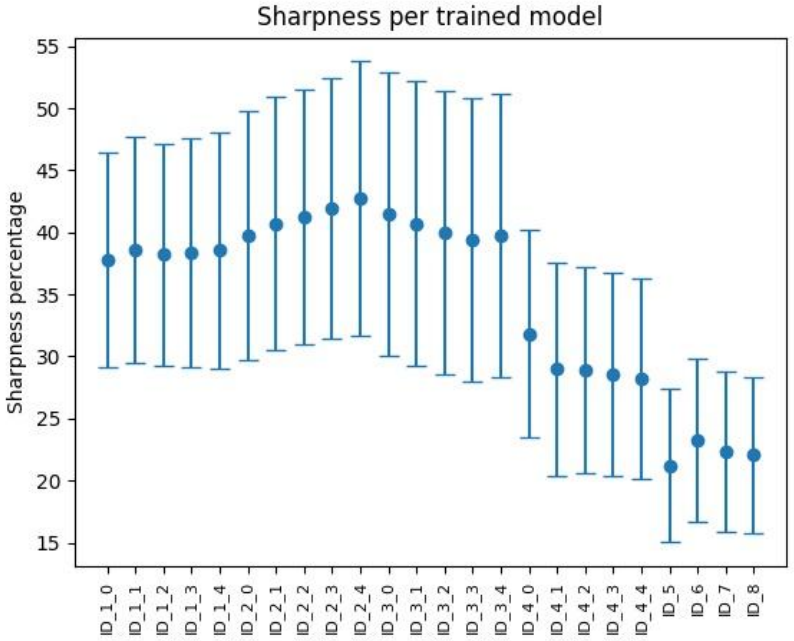


***Figura 29**: resultados obtenidos con respecto a la métrica Quality Sharpness para los modelos entrenados. Aquellos cuya etiqueta sea de la forma ID_n_m se corresponden a los modelos con validación cruzada.*

En ***Fig. 26*** se puede observar que el modelo ID_1_1 posee la métrica de *Similarity* más alta al mismo tiempo que su desvío estándar es el más pequeño. Por ello, se decidió tomar este modelo para la experimentación. Véase que en ***Fig. 27*** la confianza con la cual se reconocen los rostros está centrada en el 100% con un desvío estándar muy pequeño. Por otro lado, si bien el porcentaje de calidad, tanto para la métrica *Brightness* como para *Sharpness* es bajo, los resultados no son muy diferentes de los otros modelos entrenados.

A partir de las próximas secciones del trabajo, cuando se hable de *autoencoder* se hará referencia al ID_1_1.

Es importante aclarar que los modelos ID_5 a ID_8 se comportan de manera diferente a los demás. Los valores de las métricas, a excepción del brillo de las imágenes, son notoriamente inferiores a los modelos ID_1 a ID_4. Una inspección de las imágenes reconstruidas por estos modelos (de los cuales pueden verse algunos ejemplos en ***Fig. 30***) mostró que las mismas aún se encontraban borrosas, siendo esto un posible indicio de la necesidad de más épocas. Se decidió no continuar con el análisis para selección de *autoencoder* en favor de disponer mayor tiempo para la experimentación.

Figura 30*: reconstrucción de tres imágenes utilizando el modelo ID_8.*

Una pregunta que puede surgir al momento de estudiar la metodología completa (desde análisis de parámetros hasta esta misma sección) es el hecho de que en el análisis de parámetros y posterior filtro de arquitecturas se utilizan criterios diferentes a los recién descritos. Esto puede verse como un error o una desventaja, pues usar las mismas métricas permitiría mantener un enfoque unificado de análisis para saber si los resultados mejoraron o empeoraron al combinar características de los *autoencoders* iniciales.

Al momento de definir un *autoencoder* de entre los 24 modelos disponibles, comenzaron a surgir preguntas sobre la simplificación de análisis realizada al tomar 6 imágenes de 200 como subconjunto de estudio, además de la problemática de calidad y de confianza en el reconocimiento de un rostro. La metodología usada al principio posee un nivel de subjetividad considerado muy alto según los autores, ya que las métricas aquí mencionadas se analizaron de forma manual, todas al mismo tiempo.

Al buscar alternativas, se reconoció la potencialidad de Rekognition como herramienta para incrementar la objetividad de la elección del *autoencoder*. El resultado de Rekognition proveía la información que implícitamente se analizaba de manera manual. Una ventaja importante en el cambio de metodología es la obtención de valores analizados de manera

consistente, sin importar la cantidad de imágenes a estudiar (lo cual no es posible para un sujeto humano de percepción subjetiva).

7. Análisis de Componentes Principales

En esta sección se explica en detalle el uso de análisis de componentes principales en el contexto del trabajo. Como se mencionó en la sección **4.2.**, se intentará generar ejemplos adversarios que permitan realizar ataques de evasión y/o impersonificación. PCA se utiliza sobre las representaciones en el espacio latente del conjunto de imágenes y, a través de la manipulación de las proyecciones resultantes, se busca obtener potenciales ejemplos adversarios.

7.1. Generación de Modelos

En esta primera parte se describe el procedimiento que permite adquirir las herramientas necesarias para obtener proyecciones en componentes principales a partir de imágenes y visceversa. Se realizarán los siguientes pasos una única vez, ya que las libreŕias utilizadas permiten almacenar los modelos (con sus respectivos parámetros) para su posterior reutilización.

1. Se obtienen las representaciones en el espacio latente del conjunto de datos utilizando el codificador del *autoencoder*. Esto da como resultado 200 vectores.
2. Se estandariza el conjunto de representaciones obtenidas en 1, utilizando una instancia de la clase `StandardScaler`, de la librería `sklearn.preprocessing`.
3. Se aplica análisis de componentes principales sobre el conjunto obtenido en 2, utilizando una instancia de la clase `PCA`, de la libraría `sklearn.decomposition`.
4. Se almacenan, en dos archivos `.joblib`, los modelos correspondientes al `StandardScaler` y a `PCA`.

Dado que cada representación en el espacio latente es de dimensión 64x1 y que se cuenta con 200 representaciones, la cantidad de autovalores (con sus autovectores correspondientes) resultantes de aplicar PCA es 64. Se puede observar en ***Tabla 3*** el ratio de varianza explicada de los autovalores. En el Apéndice, en la sección Análisis de Componentes Principales puede encontrarse la lista de autovalores.

Eigenvalues, Explained Variance Ratio							
A1	0,16794744	A2	0,1064457	A3	0,064212084	A4	0,0529059
A5	0,04945347	A6	0,047460143	A7	0,04176735	A8	0,032769542
A9	0,03206953	A10	0,027543474	A11	0,02566085	A12	0,024135994
A13	0,023233654	A14	0,021846538	A15	0,01997548	A16	0,019750385
A17	0,017400328	A18	0,016722362	A19	0,015929572	A20	0,014889501
A21	0,012427767	A22	0,0115133375	A23	0,011165145	A24	0,010974049
A25	0,010307819	A26	0,009446151	A27	0,008347811	A28	0,0075190715
A29	0,0072452654	A30	0,0067230794	A31	0,0065918486	A32	0,006465569
A33	0,005721348	A34	0,0054692533	A35	0,004821858	A36	0,004697566
A37	0,0045182505	A38	0,003935131	A39	0,003792166	A40	0,0035469485

A41	0,0029739758	A42	0,0028897158	A43	0,0027488023	A44	0,0026673602
A45	0,0024066838	A46	0,0021420482	A47	0,0019885378	A48	0,0017795902
A49	0,0016003761	A50	0,0013683785	A51	0,0013173097	A52	0,0011832417
A53	0,0011094314	A54	0,0010149184	A55	0,0008864566	A56	0,00082319847
A57	0,00077462324	A58	0,00061722647	A59	0,0005423088	A60	0,0005210481
A61	0,0004198526	A62	0,00038504144	A63	0,00036332934	A64	0,000127693

***Tabla 3**: ratio de varianza explicada por autovalor.*

7.2. Funcionalidades

Una vez obtenidos los modelos, se utilizarán los mismos para:

- Obtener la proyección en componentes principales de una imagen i:
    a. Se envía la imagen i al codificador del *autoencoder* y se obtiene la representación z en el espacio latente.
    b. Se estandariza z utilizando el modelo de `StandardScaler`.
    c. Se obtiene la proyección en componentes principales utilizando la función `transform` del modelo de `PCA`.
- Obtener una imagen a partir de la proyección en componentes principales:
    a. Se utiliza la función `inverse_transform` del modelo de `PCA` sobre el vector en cuestión.
    b. Se aplica la operación inversa a la estandarización obteniendo un vector z, utilizando la función `inverse_transform` del modelo `StandardScaler`.
    c. Se envía el vector z al decodificador del *autoencoder* para obtener la imagen.

8. Experimentación

Como se puede recordar de la sección **4.3.**, los objetivos para la experimentación incluyen lograr, en primer lugar, ejemplos adversarios que permitan realizar un ataque de evasión y, en segundo lugar, un ataque de impersonificación.

Una problemática de los *autoencoders* regulares es que no poseen una estructura para el espacio latente. Esto trae como consecuencia dificultades al momento de generar imágenes a partir del recorrido de dicho espacio, ya que no hay una relación entre proximidad (por ejemplo, según la distancia euclidiana) y resultado de la decodificación, como sí sucede con los *autoencoders* variacionales.

Otro desafío refiere a la forma en que se recorre el espacio latente. Una solución ingenua y de gran costo computacional sería recorrer el espacio por fuerza bruta (con un paso definido), analizando la existencia de ejemplos adversarios. Esta metodología no resulta muy atractiva, teniendo en cuenta las limitaciones de *hardware* expuestas en la sección **4.4.1.**

El uso de análisis de componentes principales busca proporcionar un ordenamiento jerárquico al espacio latente. La jerarquía viene dada por el orden decreciente de variabilidad que representa cada autovector del análisis de componentes principales. Es de

interés la posibilidad de relacionar determinadas componentes con la identidad permitiendo el armado de una estrategia de generación de ejemplos adversarios orientada a las mismas.

A continuación se presentarán los experimentos en orden cronológico. Estos están divididos en las siguientes secciones:

- **Información Precedente**: sección opcional que contiene información necesaria para la realización del experimento.
- **Hipótesis**: incluye no sólo las proposiciones a probar sino también la explicación por la cual se eligió dicha hipótesis.
- **Procedimiento**: se detallan los pasos para realizar el experimento.
- **Análisis**: se discute si la evidencia apoya o no la hipótesis previamente formulada y se incluyen comentarios adicionales sobre nuevos caminos de exploración.
- **Información Adicional**: sección opcional que contiene información necesaria pero no perteneciente al procedimiento y/o resultados.

8.1. Experimento 1

8.1.1. Hipótesis

Es posible encontrar una relación entre las dos primeras componentes principales y la identidad de cada individuo. Si bien la varianza explicada que suman estas componentes es del 26%, lo cual no resulta particularmente alto, es importante tener en cuenta que los autovalores correspondientes a estas componentes son un orden de magnitud mayor que el resto.

Modificar estas componentes no modificará la etiqueta asignada por el oráculo O (ver sección **2.1.**), ya que las otras 62 componentes (con un 74% de la varianza explicada) quedarán fijas, pero generará cambios sustanciales en Rekognition pues son las dos más importantes.

8.1.2. Procedimiento

1. Se obtienen las proyecciones en componentes principales de todas las imágenes del conjunto de datos.
2. Se obtiene la proyección promedio de todas las componentes principales, de Marina y de Ignacio por separado. Se obtienen las imágenes correspondientes a estas componentes y se almacenan las imágenes “promedio” de cada individuo.
3. Se grafican las proyecciones de las dos primeras componentes principales y se analiza si se puede realizar una separación entre los puntos correspondientes a Marina e Ignacio. Si dicha separación no es posible, finaliza el experimento.
4. Se toma, del gráfico realizado en 3, los límites del grupo de Ignacio para cada componente. A partir de la imagen promedio de Marina, se modifican las componentes 0 y 1 de acuerdo a los límites obtenidos anteriormente. Para recorrer estas componentes se utilizará un paso de 0.2. Se obtiene la imagen correspondiente al resultado y se almacena.
5. Se repite 4, tomando los límites del grupo de Marina y utilizando la imagen promedio de Ignacio.

8.1.3. Análisis

Al momento de analizar el gráfico realizado en el paso 3 se puede observar una clara separación de los datos entre los dos individuos. Esto aporta evidencia a favor de la primera parte de la hipótesis, ya que las dos primeras componentes principales parecen tener relación con la identidad de la persona en cuestión.

***Figura 31**: a izquierda, imagen promedio de Ignacio. A derecha, imagen promedio de Marina.*

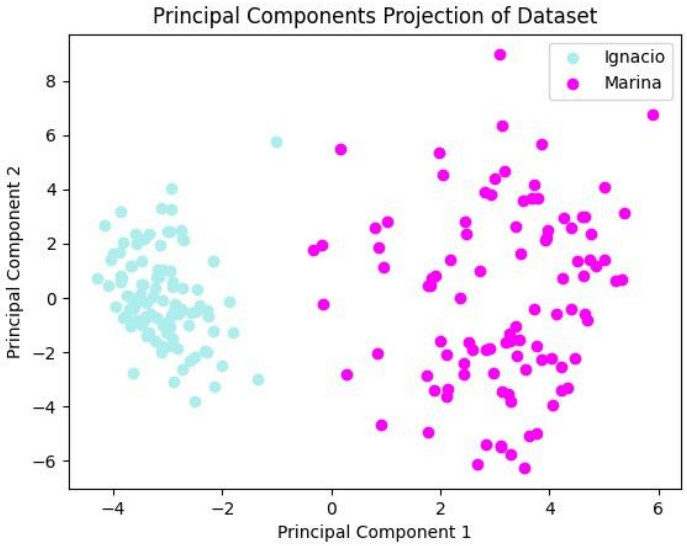


***Figura 32**: proyección de las primeras dos componentes del conjunto de datos.*

Como se puede observar en ***Fig. 32*** el grupo de datos correspondiente a Marina se encuentra más esparzo que el de Ignacio. Se puede explicar esta diferencia al observar que las imágenes de Marina tienen expresiones y ángulos de enfoque más variados que las de Ignacio.

***Figura 33**: resultados de la manipulación a partir de la imagen promedio de Ignacio.*

***Figura 34**: resultados de la manipulación a partir de la imagen promedio de Marina.*

Tanto en ***Fig. 33*** como en ***Fig. 34*** se encuentra la imagen original en la fila superior, con su correspondiente manipulación en la fila inferior. En ***Fig. 33*** es posible observar que, a partir de una imagen cuya etiqueta es Ignacio (fila superior), las imágenes resultantes de la manipulación verifican que la etiqueta asignada por O es Marina (fila inferior). Luego, en ***Fig. 34*** se producen los mismos resultados para la situación inversa: la modificación de las dos primeras componentes principales provoca un cambio de etiqueta de Marina a Ignacio. Los cambios realizados en los valores de las dos primeras componentes se encuentran detallados en el Apéndice, en la sección Detalle de Modificaciones para Experimento 1.

El experimento mostró que no se verifica la segunda parte de la hipótesis ya que al modificar las proyecciones de las primeras dos componentes principales de un individuo con las del otro, la clasificación dada por el oráculo O cambia.

8.2. Experimento 2

8.2.1. Hipótesis

La primera componente principal es la única dirección en la cual se observa una separación clara entre los datos de Marina y de Ignacio.

Como puede observarse en ***Fig. 32*** la separación en grupos sólo se da a lo largo del eje X, correspondiente a la primera componente principal. Se espera que el comportamiento de la segunda componente se repita en las 62 restantes.

8.2.2. Procedimiento

1. Se obtienen las proyecciones en componentes principales de todas las imágenes del conjunto de datos.
2. Se grafican las proyecciones obtenidas en 1 para cada componente principal. En cada proyección se distinguen los puntos de Marina e Ignacio con distintos colores para analizar si existe una separación.

8.2.3. Análisis

La experimentación apoya la hipótesis. Se puede observar que la separación en grupos se da únicamente en la primera componente principal. En el resto, los valores de Marina e Ignacio se encuentran superpuestos y, en general, los de Marina tienen mayor rango de variabilidad que los de Ignacio.

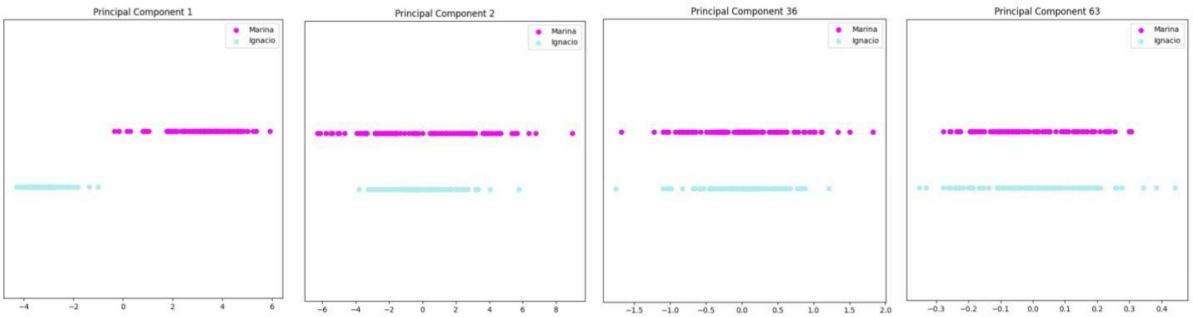


Figura 35*: representación de los elementos del conjunto de datos según i-ésima componente principal*[10]*.*

[10] Los gráficos correspondientes a todas las componentes principales se encuentran en esta carpeta.

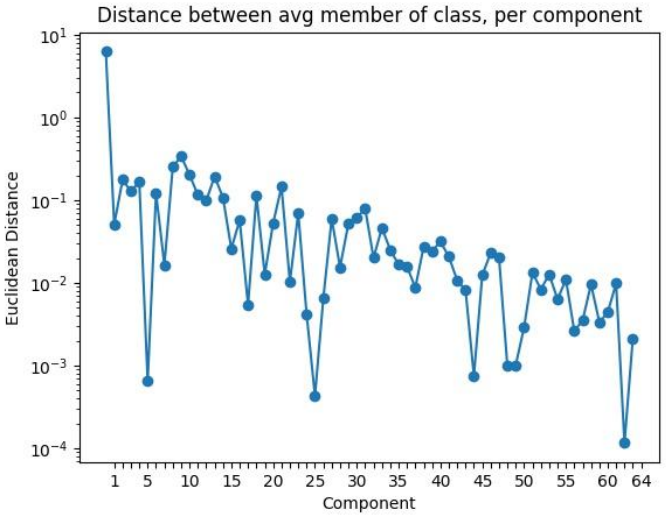


***Figura 36**: distancia euclídea entre el promedio de Ignacio y Marina, por componente principal.*

Una vez obtenidas las proyecciones en componentes principales de los datos, se puede analizar en ***Fig. 36*** la separación entre el valor promedio de los datos de Marina y de Ignacio, por componente principal. Este gráfico aporta mayor claridad al momento de analizar la hipótesis, pues se ve claramente que la primera componente principal muestra una distancia euclidiana entre promedios significativamente mayor al resto de las componentes. Inclusive, pareciera existir una tendencia descendente respecto a la distancia entre los promedios.

8.2.4. Información Adicional

Durante el transcurso del experimento se decidió replicar el procedimiento para las representaciones de los datos en el espacio latente. Se buscaba analizar si se observa dicha separación en alguna dimensión de este espacio.

Se pudo observar que, si bien no hay dimensiones con una separación disjunta entre Marina e Ignacio, hay direcciones que parecieran tener una “predominancia” de un individuo u otro. En ***Fig. 37***, la dimensión 53 posee predominancia de Ignacio, con los datos de Marina agrupados, con poca variabilidad. La situación inversa se da en la dimensión 3, con predominancia de Marina.

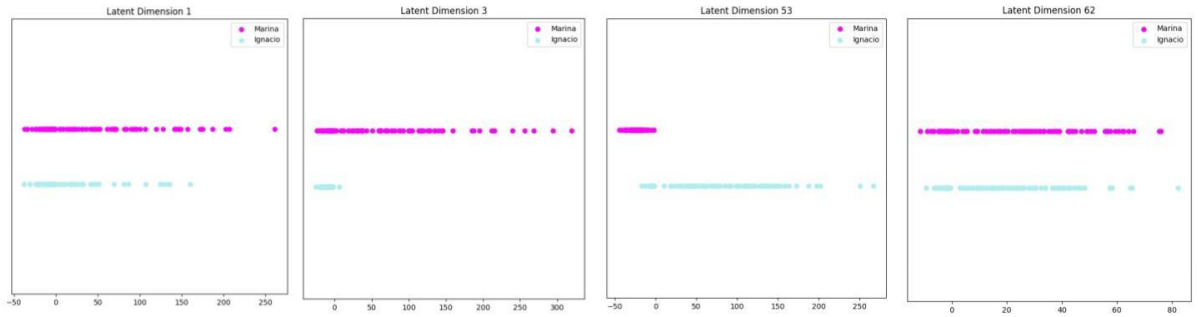


***Figura 37**: representación de los elementos del conjunto de datos según i-ésima dimensión del espacio latente*[11].

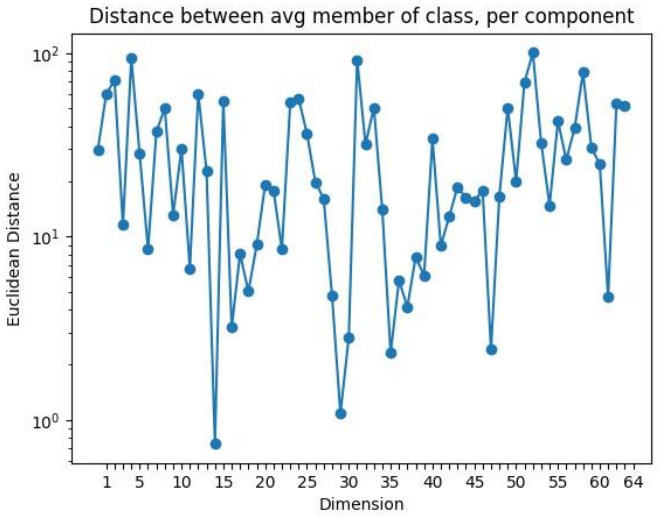


***Figura 38**: distancia euclídea entre el promedio de Ignacio y Marina, por dimensión del espacio latente.*

Por cada dimensión del espacio latente, al igual que en ***Fig. 36***, se calculó el promedio para los datos de Ignacio y, por separado, para los datos de Marina. Luego, se midió la distancia euclidiana entre los promedios de cada individuo. Como se observa en ***Fig. 38***, no pareciera existir una tendencia clara sobre las distancias que presentan dichos promedios.

En conclusión, se cree que el comportamiento de las direcciones del espacio latente provee la posibilidad de armar una nueva estrategia de generación de ejemplos de adversarios, además de aquella que se encuentra apalancada en el uso de componentes principales. Esto se discutirá en mayor profundidad en la sección **9.1.**

8.3. Experimento 3

8.3.1. Hipótesis

A partir del promedio de las proyecciones en componentes principales del conjunto de datos, la modificación de la primera componente principal permite controlar la etiqueta asignada por el oráculo O.

[11] Los gráficos correspondientes a todas las dimensiones del espacio latente se encuentran en esta carpeta.

8.3.2. Procedimiento

1. Se obtienen las proyecciones en componentes principales de todas las imágenes del conjunto de datos.
2. Se obtiene el promedio de las proyecciones obtenidas en 1. Se obtiene la imagen correspondiente al resultado y se almacena.
3. Se toma el valor mínimo y máximo que puede tomar la primera componente principal según ***Fig. 32***. A partir del promedio obtenido en 2 se varía la primera componente en el rango adquirido con un paso de 0.2.

8.3.3. Análisis

Figura 39*: imágenes decodificadas luego de modificar la primera componente principal a partir de la imagen promedio.*

Los resultados obtenidos respaldan la hipótesis. Como se observa en ***Fig. 39***, según el valor de la primera componente principal se modifica la etiqueta asignada por O. Es importante tener en cuenta que esta variación se realiza sobre la imagen promedio, la cual tiene una expresión vacía (ver ***Fig. 40***).

Figura 40*: imagen promedio*

8.4. Experimento 4

8.4.1. Hipótesis

Las componentes principales 2 a 64 controlan la expresión y/o ángulo de la imagen. Modificar estas componentes permite que un rostro cambie su expresión sin modificar la etiqueta asignada por el oráculo O.

8.4.2. Procedimiento

1. Se seleccionan nueve imágenes manualmente de manera tal que sean diversas en sus expresiones. Las imágenes elegidas pueden observarse en ***Fig. 41***.
2. Se realizan combinaciones a partir de las imágenes en ***Fig. 41***. Cada combinación contará con una imagen original y una de referencia. De la original se busca mantener la etiqueta asignada por O y de la de referencia, obtener la expresión del rostro.

Oiriginal	Referencia
marina0	ignacio40
marina69	ignacio40
marina41	ignacio40
ignacio37	marina69
test-ignacio9	marina69
ignacio50	marina69
ignacio40	marina64
marina69	ignacio22

Tabla 4*: combinaciones de imágenes utilizadas en el experimento.*

3. Se realizan los siguientes pasos sobre cada combinación en ***Tabla 4***:
    a. Se obtienen las componentes principales de la imagen original, a las cuales llamaremos y_o y de la imagen de referencia, y_r.
    b. Se crea un nuevo vector, $y*$ utilizando el valor de la primer componente de y_o y, para el resto de las componentes, los valores de y_r.
    c. Se obtiene la imagen correspondiente a $y*$ y se almacena el resultado.
    d. Se grafica la imágenes correspondientes a y_o, y_r e $y*$ para observar los cambios.

***Figura 41**: imágenes seleccionadas para experimento 4.*

8.4.3. Análisis

Los resultados obtenidos aportan evidencia a favor de la hipótesis propuesta. Para distintas y_o, se decidió utilizar la misma imagen de referencia y_r pues se esperaba que la expresión del resultado $y*$ sea similar a la de y_r e independiente de la expresión de y_o. La característica que se buscaba mantener de y_o era la etiqueta asignada por O. Se puede observar en ***Fig. 42*** que todas las imágenes $y*$ resultan en la misma expresión manteniendo $O(y_o)$.

***Figura 42**: imágenes originales de marina con referencia de ignacio*

Los mismos resultados se alcanzan al invertir la situación: y_o corresponde a distintas imágenes de Ignacio e y_r corresponde a una imagen de Marina. En ***Fig. 43*** podemos observar que las imágenes $y*$ se comportan de manera similar a lo analizado anteriormente. La diferencia es que la decodificación resultante genera imágenes con un ruido significativo que distorsiona las características del rostro.

***Figura 43**: Imágenes originales de Ignacio con referencia de Marina*

La aparición de ruido en $y*$ no necesariamente implica que el cambio de expresión haya fracasado. Para ello se decidió utilizar una imagen y_r con una expresión más distintiva que hiciera aparente su presencia en $y*$ a pesar del ruido. En ***Fig. 44*** puede observarse que la sonrisa de y_r aparece en y_* a pesar de la distorsión introducida.

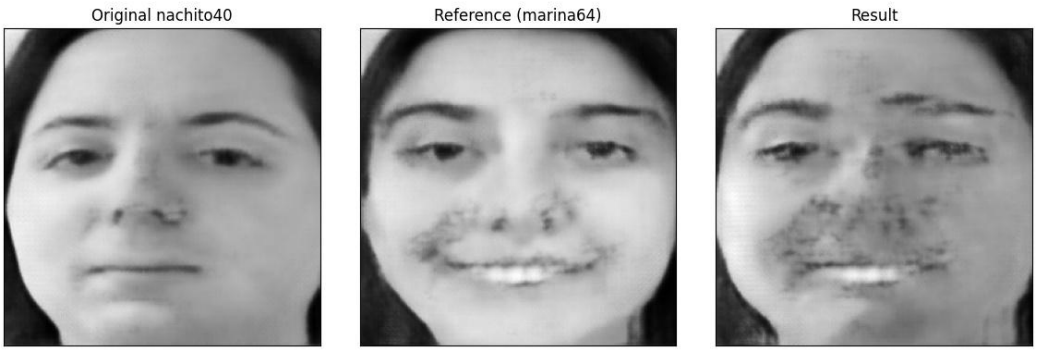


***Figura 44**: Imagen original de Ignacio con referencia de Marina con expresión distintiva*

En los ejemplos presentados en ***Fig. 42*** y ***Fig. 43***, se observa que la imagen de referencia posee una expresión seria, similar a la imagen promedio. Era de interés analizar si la situación inversa era posible: pasar de una expresión seria a otra más exagerada. Los resultados de las combinaciones correspondientes a ***Fig. 44*** y ***Fig. 45*** muestran que y_* cambia radicalmente su expresión, manteniendo $O(y_o)$.

Figura 45*: Imagen original de Marina con referencia de Ignacio con expresión distintiva*

En conclusión, los resultados incrementan la evidencia a favor de la hipótesis planteada en el experimento anterior: la primera componente principal controla la etiqueta del oráculo que se le asigna a la imagen. Por otra parte, la combinación de las 63 restantes manipula la expresión de la misma. No se tiene conocimiento sobre el significado semántico de cada componente por separado e inclusive podría ocurrir que dicho significado no exista y sólo surja de la combinación de las mismas.

Finalmente, la presencia de distorsiones al pasar de Ignacio a Marina o al pasar de expresiones simples a complejas, como en ***Fig. 45***, muestra la necesidad de estudiar la calidad de las imágenes resultantes del proceso de experimentación. A partir de esta reflexión es que surge el planteo de un análisis objetivo de calidad propuesto en la sección **4.** y descrito en la sección **8.6.1.**

8.5. Experimento 5

8.5.1. Hipótesis

Si bien las últimas 63 componentes principales controlan la expresión del rostro, cada una por separado tiene un significado semántico. Por ejemplo, una componente principal controla la posición de los ojos, otra la forma de la boca, etc.

8.5.2. Procedimiento

1. Se obtienen las proyecciones en componentes principales de todas las imágenes del conjunto de datos.
2. Se calcula el promedio de las proyecciones de 1.
3. Se parte del vector promedio y, para cada componente i se realiza el siguiente procedimiento:
    a. Se toma el mínimo y máximo que toma la componente i para las proyecciones obtenidas en 1. Luego se divide el rango en ocho valores.

b. Por cada uno, se mantienen fijas todas las componentes principales, a excepción de la i-ésima componente, la cual toma el valor en cuestión.
c. Se decodifica el vector de componentes principales en una imagen y se almacena[12].

8.5.3. Análisis

Los resultados obtenidos en este experimento no permiten definir si la evidencia apoya la hipótesis o si se encuentra en contra. Al modificar ciertas componentes se generan cambios visuales a los cuales se les puede asignar significado (como puede ser el ángulo de rotación de la cara). El problema es que dicho significado no suele ser exclusivo de una única característica en el rostro y, además, suele aparecer ruido a medida que la componente va siendo modificada.

***Figura 46**: decodificación de imagen promedio al variar la componente principal 1.*

***Figura 47**: decodificación de imagen promedio al variar la componente principal 5.*

[12] Se debió fraccionar el cómputo de las 512 (64 dimensiones x 8 valores) imágenes resultantes en pequeños lotes de 40 por restricciones de *hardware*.

***Figura 48**: decodificación de imagen promedio al variar la componente principal 3.*

***Figura 49**: decodificación de imagen promedio al variar la componente principal 9.*

***Figura 50**: decodificación de imagen promedio al variar la componente principal 41.*

Por ejemplo, en ***Fig. 47***, se puede observar una rotación del rostro. Sin embargo, también parece que la expresión cambia a una de enojo a medida que se llega a los valores máximos de la componente cinco. La rotación del rostro no sólo aparece en ***Fig. 47*** sino que también se puede encontrar en la componente tres como se ve en ***Fig. 48***. A diferencia de la componente anterior, en este caso se produce una cantidad considerable de ruido, al igual que en ***Fig. 49***. Finalmente, sucede que para las últimas componentes principales los cambios se vuelven más sutiles al ojo humano, como se evidencia en ***Fig. 50***, la cual muestra los resultados para la componente 41.

8.5.4. Información Adicional

Se decide replicar el experimento 5 sobre las imágenes del conjunto de datos, codificadas en el espacio latente. Dado que PCA no logró aportar evidencia que apoye la hipótesis, se busca analizar si el espacio latente desorganizado permite modificar características puntuales a través de movimientos en direcciones específicas en dicho espacio.

El procedimiento se mantiene igual, a excepción del punto 1 en el cual se obtienen las representaciones en el espacio latente en lugar de las proyecciones en componentes principales.

***Figura 51**: decodificación de imagen promedio al variar la dimensión 4 del espacio latente*

***Figura 52**: decodificación de imagen promedio al variar la dimensión 10 del espacio latente*

Figura 53: *decodificación de imagen promedio al variar la dimensión 20 del espacio latente*

Figura 54: *decodificación de imagen promedio al variar la dimensión 34 del espacio latente*

A partir de los resultados obtenidos, se pueden realizar observaciones similares al experimento original: existen dimensiones que no producen cambios significativos y dimensiones que sí lo hacen pero que no generan un único cambio y que, muchas veces, agregan ruido.

En conclusión, si el objetivo es modificar características específicas del rostro, sería difícil hacerlo a través de direcciones particulares tanto en el espacio latente como en las direcciones de las componentes principales.

Se podría intentar añadir más individuos al conjunto de datos para estudiar cómo impacta en la separación de características a lo largo de diferentes direcciones. Se cree que al tener poca variabilidad de rostros, el *autoencoder* no generaliza lo suficiente sobre elementos de un rostro.

Otra alternativa, ya que la anterior podría ser insuficiente, consiste en observar qué ocurre al combinar distintas componentes principales o direcciones del espacio latente. Quizás es necesario estudiar las direcciones en conjunto para encontrar formas de realizar cambios puntuales en los rostros.

8.6. Experimento 6

8.6.1. Información Precedente

El objetivo de esta sección es obtener métricas sobre cómo son reconocidas las imágenes en Rekognition, tanto de Marina como de Ignacio. Es importante recordar que Rekognition cuenta con una colección de imágenes contra las cuales realizará las comparaciones en caso de usar la funcionalidad "Comparar la similaridad contra una colección" descrita en la sección **5.2.2.** Esta colección se compone de las características de las imágenes del conjunto de datos luego de pasar por el *autoencoder*, por las razones explicadas en **5.2.**

Se busca obtener medidas "base" sobre la confianza en reconocimiento del rostro, similaridad y calidad de la colección que almacena Rekognition para luego analizar potenciales ejemplos adversarios en función de estos valores iniciales. Se explicó anteriormente que los ejemplos adversarios serían generados por el *autoencoder*, con lo cual se considera que los mejores valores de calidad y similaridad que pueden alcanzar son aquellos que se corresponden a las imágenes utilizadas para entrenar y evaluar al *autoencoder*. Un ejemplo adversario sólo será considerado como tal si sus valores de *Quality Brightness* y *Quality Sharpness* se encuentran alrededor del promedio obtenido para las imágenes sin manipular.

Rekognition permite determinar un parámetro denominado `face_threshold`, entre 0% y 100%, permitiendo definir qué resultados son considerados: si la similaridad con una imagen es mayor o igual al `face_threshold`, el "*match*" será incluido en el resultado. Caso contrario, será descartado. Se decidió utilizar un `face_threshold` de 0% para analizar todas las comparaciones contra la colección. El procedimiento puede definirse de la siguiente manera:

1. Se pasan las imágenes del conjunto de datos por el *autoencoder* y se almacenan. Cada imagen
    a. Se envía a Rekognition y se utiliza la funcionalidad "Comparar la similaridad contra una colección". Esto permite obtener los resultados de confianza en reconocimiento del rostro y similaridad contra la colección que tiene almacenada Rekognition.
    b. Se envía a Rekognition y se utiliza la funcionalidad "Analizar la calidad de una imagen" descrita en la sección **5.2.2.** El objetivo es obtener las métricas de calidad, *Quality Sharpness* y *Quality Brightness*, para cada imagen.
2. Cada resultado obtenido en 1a, correspondiente a un individuo (Marina o Ignacio), cuenta con una lista de "*matches*" entre la imagen en cuestión y aquellas almacenadas en Rekognition. Se realiza un promedio, para esa imagen, de la similaridad correspondiente a las imágenes de Marina y, luego, a las imágenes de Ignacio. Se almacena este promedio con la etiqueta del individuo de la imagen.

3. Se separan los valores obtenidos en aquellos correspondientes a la etiqueta Marina (serán 93 promedios pues se tienen 93 imágenes[13] de Marina) y a la etiqueta Ignacio (misma cantidad de promedios). A partir de esta separación se computa, para cada etiqueta, promedio y desvío estándar de los 93 valores.
4. Se computan las métricas promedio y desvío estándar para confianza en el reconocimiento del rostro y calidad a partir de los resultados obtenidos en 1b.

Target	Marina Mean	Marina Standard Deviation	Ignacio Mean	Ignacio Standard Deviation
Marina	99,72605991	0,83894468	1,845080529	2,18497707
Ignacio	1,842878439	2,00393929	98,94514483	3,01514133

Tabla 5: análisis de similaridad de Rekognition

Metric	Mean	Standard Deviation
Confidence	99,99999224	1,48E-05
Quality Brightness	83,02473108	5,978395951
Quality Sharpness	40,98942438	8,595760332

Tabla 6: *análisis de calidad de Rekognition*

En ***Tabla 5*** se encuentran los resultados del ítem 2 del procedimiento y en ***Tabla 6*** los del ítem 3.

8.6.2. Hipótesis

Los sistemas de reconocimiento facial no se basan únicamente en el aspecto general de una imagen para el reconocimiento del individuo. Como se explicó en el experimento 3, se considera que la etiqueta asignada a una imagen por el oráculo O puede ser controlada al modificar el valor de la primera componente principal. Al manipular una imagen para modificar su etiqueta original, los porcentajes de similaridad, resultantes de analizar la imagen con Rekognition, no necesariamente serán cercanos a los de ***Tabla 5***.

Por ejemplo, si se envía a Rekognition una imagen manipulada etiquetada por O como “Marina” puede ocurrir que el promedio de “*matches*” con Ignacio tome un valor mayor a 1,84508052% y, por otra parte, que el de Marina tome un valor menor a 99,72605991%. Lo mismo sucedería con una imagen etiquetada por el oráculo como “Ignacio”.

8.6.3. Procedimiento

1. Se obtienen las proyecciones en componentes principales de las imágenes del conjunto de datos.
2. Se definen dos rangos para modificar el valor de la proyección en la primer componente principal

[13] Si bien el conjunto de datos tiene 100 imágenes por cada individuo, las reconstrucciones correspondientes al *autoencoder* seleccionado no son buenas para todas las imágenes. Algunas de ellas presentan una cantidad excesiva de ruido que disminuía significativamente los valores de reconocimiento. Para evitar que estos valores atípicos impacten en el promedio de similaridad y calidad se decidió eliminarlos. De cada individuo fueron removidas las 7 imágenes más dañadas por el *autoencoder*, para mantener el balance en términos de cantidad de rostros.

    a. Para lograr que la identidad se corresponda con Marina, se utiliza el intervalo [1, 6.5] con un paso de 0.2.
    b. Para lograr que la identidad se corresponda con Ignacio, se utiliza el intervalo [-5, 1.5] con un paso de 0.2.
3. Por cada vector obtenido en 1 se generan nuevas proyecciones modificando la primera componente de acuerdo al rango definido en 2 según la etiqueta asignada por O a la imagen correspondiente al vector.
    a. Si la etiqueta del oráculo es Ignacio, se utilizará el rango definido en 2a.
    b. Si la etiqueta del oráculo es Marina, se utilizará el rango definido en 2b.
4. Se decodifican los vectores resultantes en imágenes y se almacenan los resultados.
5. Se seleccionan 18 imágenes cuya etiqueta original cambió y cuya calidad parece mantenerse, de acuerdo al ojo humano. Dado que esta selección fue manual de entre **4600** imágenes, se debe tener en cuenta la subjetividad de este paso[14]. Cada una de las imágenes fue enviada a Rekognition para obtener
    a. Métricas de similaridad con respecto a las imágenes almacenadas en Rekognition.
    b. Confianza en el reconocimiento de rostro.
    c. Métricas de calidad: *Quality Sharpness* y *Quality Brightness*.

8.6.4. Análisis

Luego del experimento, se analizaron las 18 imágenes seleccionadas en el ítem 5 del procedimiento, las cuales tuvieron un cambio en la etiqueta asignada por O. Los resultados del análisis de Rekognition sobre calidad se encuentran en ***Tabla 7*** y los correspondientes al análisis de similaridad, en ***Tabla 8***.

Potential Adversarial Examples, Quality Analysis			
Source Image	Confidence	Quality Brightness	Quality Sharpness
ignacio0	100	87,70657349	32,20803452
ignacio11	99,99998474	89,30258179	26,17736816
ignacio14	100	85,84529114	26,17736816
ignacio21	99,99998474	90,79774475	32,20803452
ignacio29	99,99998474	88,12119293	26,17736816
ignacio49	99,99998474	89,27024841	32,20803452
ignacio51	99,99998474	84,99997711	38,89601135
ignacio54	99,99998474	90,41576385	32,20803452
ignacio76	100	87,24095917	32,20803452
ignacio87	100	83,67180634	46,02980042
ignacio88	99,99998474	89,55757904	32,20803452
test-ignacio0	99,99998474	90,46258545	32,20803452

[14] Del conjunto de imágenes obtenidas, podrían haberse ordenado las mismas de acuerdo a la calidad determinada por Rekognition y, una vez ordenadas, seleccionar las 18 mejores que verifiquen que la etiqueta original asignada por el oráculo cambió. El problema que se presentó en este experimento, y que no se había presentado hasta el momento, fue la limitación de la *Free Tier* de Rekognition, mencionada en la sección **5.2.1.** Tanto para el análisis de calidad como el de similaridad, fue necesario restringir las imágenes seleccionadas debido al límite de 5000 imágenes mensuales que impone el paquete gratuito de Amazon.

marina17	99,99990845	82,61359406	32,20803452
marina21	99,99987793	83,24923706	32,20803452
marina29	100	82,19504547	46,02980042
marina33	99,99993846	85,35240936	38,89601135
marina50	99,99998474	77,64185333	32,20803452
test-marina7	99,99997711	77,64492035	38,89601135

Tabla 7: *análisis de calidad de Rekognition sobre potenciales ejemplos adversarios*

Potential Adversarial Examples, Similarity Analysis				
Source Image	Mean Marina Similarity	Stdev Marina Similarity	Mean Ignacio Similarity	Stdev Ignacio Similarity
ignacio0	98,07464083	3,76530556	3,42836851	3,28099627
ignacio11	99,84451138	0,44299136	2,67762129	2,72107566
ignacio14	98,04071176	3,12131098	9,27779568	9,02577925
ignacio21	99,55909073	1,05749549	3,92870466	4,17965053
ignacio29	99,94787491	0,18264075	2,00511547	2,19638218
ignacio49	99,75134868	0,70359387	1,96552379	2,94280761
ignacio51	99,42177319	1,47528882	2,15327145	2,34993002
ignacio54	99,26387926	1,56206441	2,06201546	1,97096455
ignacio76	96,17442076	7,39962654	6,47935669	7,31280913
ignacio87	99,58807677	1,34167541	2,65176981	2,70953122
ignacio88	99,47252073	2,23569500	3,59192174	4,39843860
test-ignacio0	99,75705399	0,91449670	2,81552409	2,62947759
marina17	1,08416005	0,86099082	91,84194524	11,20301741
marina21	3,83634580	3,71051312	46,84344557	22,6654669
marina29	35,37624416	18,83294737	62,60839676	23,33784421
marina33	64,07814949	17,81382552	69,9908113	21,38174483
marina50	26,38905084	16,75245342	81,39943349	17,91575476
test-marina7	17,73802099	12,09566476	66,42649591	22,02305054

Tabla 8: *análisis de similaridad de Rekognition sobre potenciales ejemplos adversarios*

Los resultados arrojados no son concluyentes con respecto a la hipótesis planteada. En primer lugar, se analizó si las imágenes de Ignacio manipuladas para que parezcan Marina tomaban valores distintos a aquellos en ***Tabla 5*** con respecto a las imágenes de Marina originales.

En ***Fig. 55*** se visualiza el comportamiento de las imágenes de Ignacio modificadas. En el eje x tenemos la imagen fuente que, luego de la manipulación, O asigna la etiqueta Marina. Por otra parte, en el eje y se grafica para cada imagen el promedio (con su desvío estándar) de similaridad con respecto a Marina e Ignacio por separado.

Por otra parte, dos rectas en color más claro representan el comportamiento de una imagen de Marina original cuando se la envía a Rekognition. Se grafica, para cada individuo, los valores de similaridad promedio junto con su desvío estándar (en una franja más clara).

En ***Fig. 55*** se observa que las imágenes manipuladas tienen un comportamiento similar al de las imágenes originales, aportando evidencia en contra de la hipótesis. Este no es el caso para las imágenes de Marina que se manipulan para que O las etiquete como Ignacio.

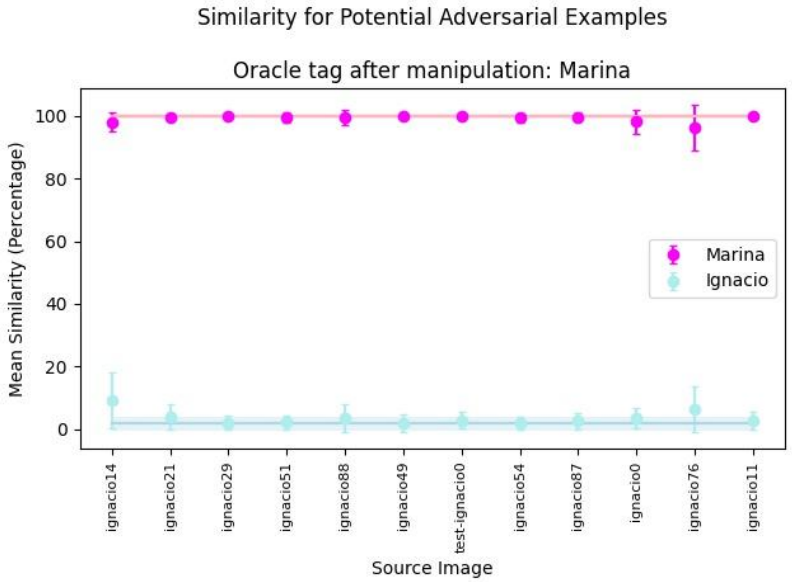


Figura 55*: promedio de similaridad para imágenes de Ignacio manipuladas.*

En ***Fig. 56*** se tiene una figura similar a la explicada anteriormente, con ciertas diferencias. Las imágenes fuente son imágenes de Marina y su primera componente principal ha sido manipulada de manera tal que la O asigna la etiqueta Ignacio. Por otra parte, las líneas rectas correspondiente al promedio y desvío estándar de similaridad refieren al comportamiento esperado de una imagen original de Ignacio.

En este caso, sí hay evidencia que respalda la hipótesis ya que el comportamiento de los potenciales ejemplos adversarios difiere notoriamente del de las imágenes originales. Es especialmente interesante el caso de la manipulación de marina33, la cual se encuentra en ***Fig. 57***, ya que los valores de similaridad para ambos individuos se encuentran prácticamente solapados.

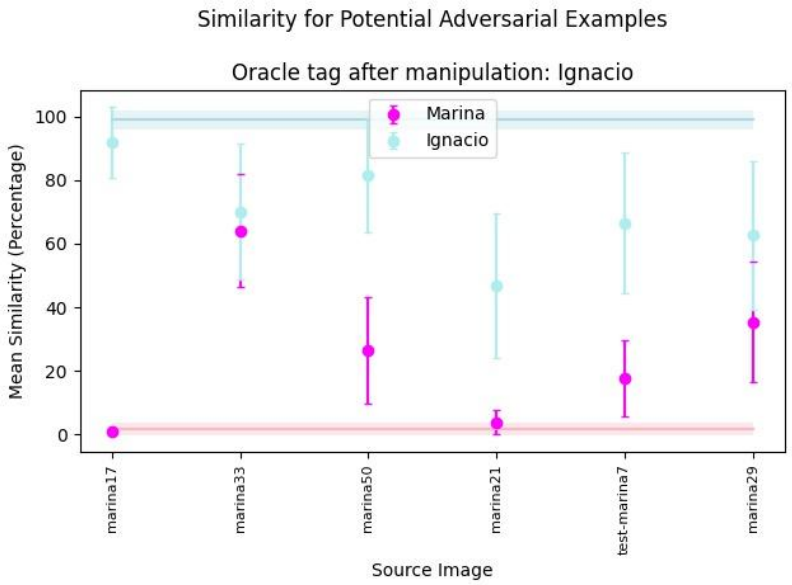


Figura 56: promedio de similaridad para imágenes de Marina manipuladas.

Figura 57*: Resultado de la manipulación de marina33.*

Al generar las 4600 imágenes del ítem 4 se observó un fenómeno inesperado: no todas las imágenes manipuladas resultaban en un cambio de etiqueta. Se decidió estudiar esta situación por separado y añadir un sexto paso al procedimiento, ya que la existencia de estas imágenes pone en jaque las conclusiones observadas en el experimento 3.

De la misma manera que en ítem 5, se seleccionaron 8 imágenes de las 4600 cuya etiqueta original no cambió (a pesar de haber cambiado el valor de la primera componente) y cuya calidad parece mantenerse. En este paso aplica la misma restricción sobre el análisis de calidad explicada anteriormente.

Cada una de las imágenes fue enviada a Rekognition para obtener las métricas obtenidas para las imágenes del ítem 5 del procedimiento. A continuación, en ***Tabla 9***, se encuentra el análisis de calidad de estas imágenes y, en ***Tabla 10***, el análisis de similaridad.

Images with same Oracle's Tag, Quality Analysis			
Source Image	Confidence	Quality Brightness	Quality Sharpness
marina2	100	88,01103210	46,02980042
marina3	100	83,47825623	38,89601135
marina12	100	91,77685547	53,33004761

marina13	100	85,69889069	53,33004761
marina18	100	82,27753448	53,33004761
marina20	99,99811554	85,59648132	53,33004761
test-marina6	100	75,19393158	60,49041748

Tabla 9: *análisis de calidad de Rekognition sobre imágenes modificadas que mantuvieron la etiqueta asignada por el oráculo.*

Images with same Oracle's Tag, Similarity Analysis				
Source Image	Mean Marina Similarity	Stdev Marina Similarity	Mean Ignacio Similarity	Stdev Ignacio Similarity
marina2	92,61780532	10,9613069	2,96117127	2,91782896
marina3	83,49153317	16,35864252	5,24387771	5,4755931
marina12	97,38220371	5,39847509	8,23557135	7,09698961
marina13	99,14090499	2,37094935	15,19867614	13,65096206
marina18	96,13959216	6,84118404	0,27803256	0,23591804
marina20	68,91432025	22,96477827	3,34106165	5,63893883
test-marina6	99,70805014	1,21200391	6,47128051	8,14745347

Tabla 10: *análisis de similaridad de Rekognition sobre imágenes modificadas que mantuvieron la etiqueta asignada por el oráculo.*

Nuevamente, se realizó un gráfico que permitiera estudiar el comportamiento de estas imágenes modificadas. En ***Fig. 58*** se puede observar que, si bien el comportamiento no cambió de manera tan drástica como en ***Fig. 56***, tampoco se comporta de la misma forma que las imágenes de Marina originales.

Es interesante observar los casos de marina20 y marina3, ya que la similaridad con Marina disminuye, mientras que con Ignacio se mantiene dentro de los valores originales. Si bien es posible notar cierto ruido en la imagen, los resultados de calidad de ***Tabla 9*** no muestran un descenso significativo en los valores de *Quality Brightness* y *Quality Sharpness* con respecto al promedio de las imágenes sin manipular, presentes en ***Tabla 6***.

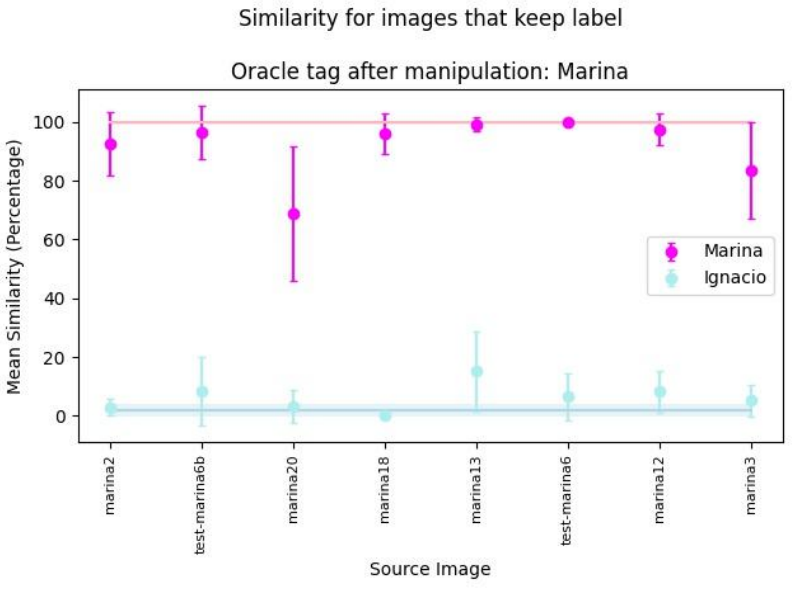


***Figura 58**: promedio de similaridad para imágenes de Marina manipuladas, que conservan la etiqueta de O.*

***Figura 59**: Resultado de la manipulación de marina12 y marina20, a izquierda y derecha respectivamente.*

Como conclusión, la existencia de estos ejemplos muestra que la primer componente principal no permite controlar la etiqueta asignada por O en todos los casos. Pareciera que hay más factores en juego dentro del lo que es identidad, aparte de la primera componente.

Por otra parte, para alterar los promedios de similaridad obtenidos por Rekognition con esta metodología, la manipulación no necesariamente debe incluir un cambio de etiqueta por parte de O.

8.6.5. Información Adicional

Al modificar el valor de la primera componente principal correspondiente a la imagen marina33 se logró una similaridad de 69% para Ignacio y 64% para Marina, aproximadamente. Si bien la imagen resultante es un ejemplo adversario, pues al enviarla a Rekognition se cumple la definición de ataque de evasión, se decidió investigar el entorno de esta imagen para buscar un ejemplo adversario que no ́solo produjera ataques de evasión sino también de impersonificación.

Con respecto a la primera componente principal, se decidió mantener un entorno cercano al valor con el cual fue manipulada la imagen marina33 durante el experimento. Por otra parte, se modificaron componentes dos y tres. El objetivo era analizar si, utilizando otras componentes aparte de la primera, se podía lograr que Rekognition inclinara aún más la balanza hacia el reconocimiento de Marina, sin afectar la etiqueta asignada por O a la imagen (Ignacio). A continuación se detalla el procedimiento:

1. Obtenemos las proyecciones en componentes principales de marina33.
2. A partir del vector de proyecciones original, se itera realizando las siguientes manipulaciones:
    a. La primera componente se modifica según el rango [-1.7, -1.4] con un paso de 0.1. Durante el experimento el valor de modificación fue -1.6.
    b. La segunda componente se modifica según el rango [-6, 8] con un paso de 0.5.
    c. La tercera componente se modifica según el rango [-6, 6] con un paso de 0.5.

3. De las 2688 imágenes resultantes, se seleccionan 36 imágenes[15] para realizar un análisis de calidad y similaridad, como en el ítem 5 del procedimiento.

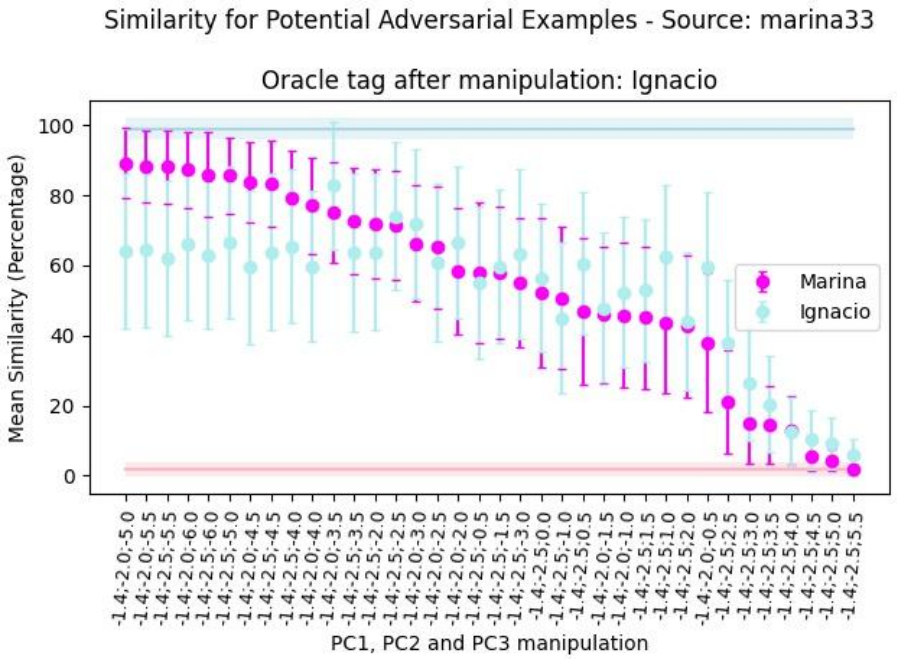


***Figura 60**: promedio de similaridad para imágenes de Marina manipuladas.*

En ***Fig. 60*** se encuentran los resultados de similaridad de las nuevas manipulaciones, ordenadas de manera descendente por el promedio de similaridad respecto a Marina. Se busca analizar el comportamiento de las mismas en referencia a los valores de una imagen original de Ignacio, al enviarla a Rekognition. Se puede observar que varias imágenes han superado el umbral del 80% para el reconocimiento de Marina, con lo cual, si la calidad se encuentra dentro de los parámetros esperados, dichas imágenes permitirían realizar ataques de impersonificación.

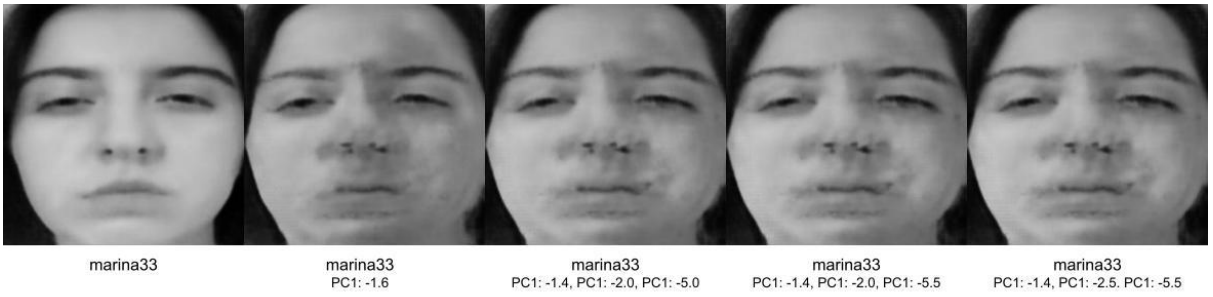


***Figura 61**: manipulaciones de marina33 en relación a la imagen original.*

Potential Adversarial Examples, Quality Analysis					
PC1	PC2	PC3	Confidence	Quality Brightness	Quality Sharpness
-1,4	-2,0	-5,0	99,99998474	85,28433228	46,02980042
-1,4	-2,0	-5,5	99,99998474	84,90164948	53,33004761

[15] Se procedió a realizar una selección manual, por las mismas restricciones que fueron mencionadas en el pie de página anterior.

-1,4	-2,5	-5,5	99,99998474	84,76454926	53,33004761

Tabla 11: *análisis de calidad de Rekognition sobre manipulaciones a partir de la imagen marina33.*

Potential Adversarial Examples, Similarity Analysis						
PC1	PC2	PC3	Mean Marina Similarity	Stdev Marina Similarity	Mean Ignacio Similarity	Stdev Ignacio Similarity
-1,4	-2,0	-5,0	89,14839300	9,908472096	64,15828444	22,19039942
-1,4	-2,0	-5,5	88,14728263	10,24077961	64,30149917	21,93384265
-1,4	-2,5	-5,5	88,06548047	10,51835873	62,03947989	22,24786561

Tabla 12: *análisis de similaridad de Rekognition sobre manipulaciones a partir de la imagen marina33.*

En ***Tabla 11*** se encuentran las características de calidad de las tres imágenes manipuladas que alcanzaron mayor reconocimiento para Marina. Allí podemos ver que tanto *Quality Brightness* como *Quality Sharpness* se encuentran dentro de los parámetros de calidad promedio, obtenidos en ***Tabla 6***.

Si bien las imágenes manipuladas son similares al ojo humano con respecto al potencial ejemplo adversario del experimento 6, como puede observarse en ***Fig. 61***, movimientos en el entorno del mismo permitieron aumentar significativamente el reconocimiento para Marina, superando el umbral definido para Rekognition y permitiendo obtener ejemplos adversarios que provoquen ataques de evasión e impersonificación.

8.7. Experimento 7

8.7.1. Hipótesis

En el experimento anterior se cuestiona la hipótesis planteada en el experimento 3 sobre la relación entre identidad y la primera componente principal. Esto provoca reflexiones sobre la metodología de generación de ejemplos adversarios propuesta. Antes, se buscaba definir una metodología utilizando la primera componente principal como base, con la posibilidad de añadir modificaciones secundarias. Una vez puesta en duda la hipótesis del experimento 3, se planteó la estrategia de encontrar ejemplos adversarios a través de una búsqueda exhaustiva del espacio de proyecciones en componentes principales. A raíz de ello, se plantea la siguiente hipótesis,

Es posible encontrar ejemplos adversarios a través de la manipulación de componentes principales por fuerza bruta. Modificar una componente principal, no necesariamente la primera, es suficiente para la generación de estos ejemplos.

8.7.2. Procedimiento

1. Se obtienen las proyecciones en componentes principales de todas las imágenes del conjunto de datos[16].
2. Se seleccionan 2 imágenes al azar, una con la etiqueta “Marina” y otra con la etiqueta “Ignacio”. Imágenes seleccionadas: **ignacio74** y **marina19**
3. Se calculan las proyecciones en componentes principales de las imágenes seleccionadas en 2.

[16] El conjunto de datos generado puede encontrarse en la siguiente carpeta.

4. Para cada uno de los vectores generados en el paso anterior, se realiza el siguiente procedimiento para cada componente i[17]:
    a. Se busca el mínimo y máximo que toma i para las proyecciones obtenidas en 1. Luego se divide el rango en ocho valores.
    b. Por cada uno, se mantienen fijas todas las componentes principales, a excepción de la i-ésima componente, la cual toma el valor en cuestión.
5. Se envían a Rekognition las imágenes generadas a partir de **ignacio74** y se guardan los resultados de similaridad en un `.csv`.
6. Se repite el ítem 5 con las imágenes generadas a partir de **marina19**.
7. Se busca si hay ataques de impersonificación o evasión en los archivos de 5 y 6, observando manualmente las similaridades. Una vez seleccionados los ejemplos adversarios, se verifica la calidad del mismo. Si los parámentros se encuentran dentro de los esperados según ***Tabla 6***, se toma en consideración.
8. De los posibles ejemplos adversarios hallados en 7 se seleccionan 2 de cada etiqueta.

8.7.3. Análisis

Se pueden observar los resultados del análisis de calidad en ***Tabla 13*** y su correspondiente análisis de similaridad en ***Tabla 14*** para los ejemplos adversarios obtenidos en el ítem 8. Finalmente, en ***Fig. 62*** se ilustran las respectivas imágenes adversarias.

Source Image	Confidence	Quality Brightness	Quality Sharpness
PC17: -2.0368	99,99999893	93,24709320	46,02980042
PC15: 4.0341	99,99999893	91,68709564	53,33004761
PC8: -2.968	99,99999893	97,41493225	46,02980042
PC3: 6.9605	99,99999893	96,46032715	53,33004761

Tabla 13: Análisis de calidad de los posibles ejemplos adversarios

Source Image	Mean Marina Similarity	Stdev Marina Similarity	Mean Ignacio Similarity	Stdev Ignacio Similarity	Oracle Result	Attack Type
PC17: -2.0368	0,75797176	0,78659789	72,94214247	24,83002875	Ignacio	Dodging
PC15: 4.0341	0,22895857	0,19176988	60,44409301	26,26751010		Dodging
PC8: -2.968	78,95486565	21,02921946	2,15884985	2,19278473	Marina	Dodging
PC3: 6.9605	42,57506853	20,30322823	6,54301130	8,16298712		Dodging

Tabla 14: Análisis de similaridad de los posibles ejemplos adversarios

[17] Se limita a las primeras 20 componentes principales por restricciones en la *Free Tier* de AWS.

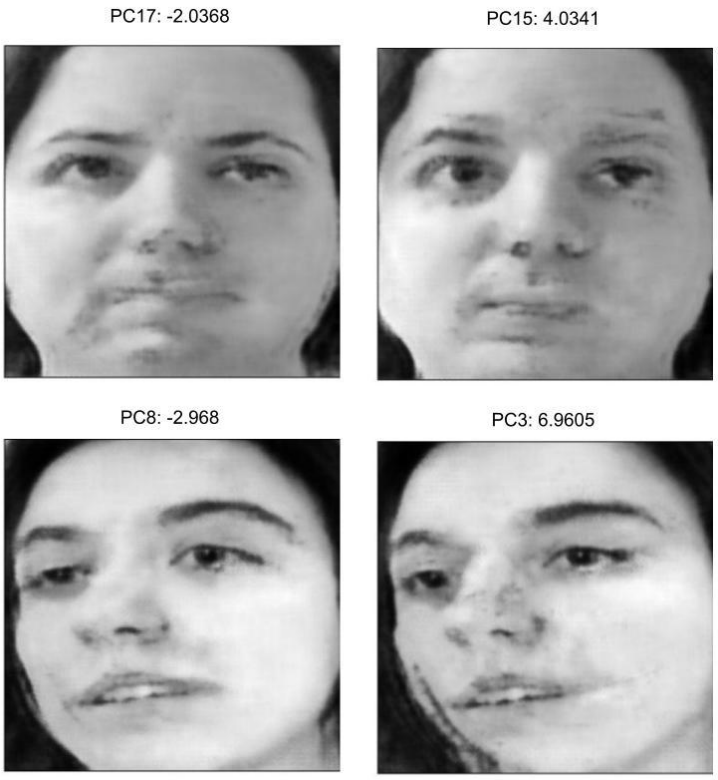


Figura 62: ejemplos adversarios seleccionados.

En los casos seleccionados la similaridad promedio es menor a 80, con lo cual se cumple con la definición de evasión provista en la sección **2.3.** Sin embargo, si bien se observa en ***Tabla 13*** que los valores de *Quality Brightness y Quality Sharpness* se encuentran dentro de los parámentros esperados (e inclusive un poco por encima de la media), resulta evidente que la manipulación generó una degradación en la imagen mediante a la observación directa en la ***Fig. 62*.**

Se mencionó también en el paso 7 del procedimiento que se buscarían ataques de impersonificación pero, al revisar la información generada, no se encontraron ejemplos permitieran realizar este ataque. Dado que en el experimento 6 pudo encontrarse un ejemplo adversario que permita un ataque de este tipo, se cree que la modificación de una única componente principal es insuficiente para la generación de dichos ejemplos adversarios.

9. Discusión

9.1. Sobre las decisiones tomadas durante la investigación

Una de las decisiones principales tomadas en el trabajo refiere al uso de análisis de componentes principales como forma de organización del espacio latente para la generación de ejemplos adversarios. Si bien el *autoencoder* variacional ha tenido éxito dentro de los modelos generativos, no se creía necesario un *autoencoder* con capacidad de generación que se correspondiera con el estado del arte porque el objetivo era lograr la construcción de ejemplos adversarios a partir de imágenes preexistentes. Por otro lado, el

análisis de componentes principales provee una jerarquización en la organización, la cual era de gran interés para los autores y, además, se tenía familiaridad con la herramienta. En resumen, se determinó que era una mejor alternativa.

Otro aspecto importante del proyecto fue la decisión del esquema de ataque. En la sección **3** se definió el principio de transferibilidad: existen ejemplos adversarios robustos a distintos clasificadores. Esto podría ser indicativo de que, para la generación de los mismos, quizás sea conveniente realizar un esquema de ataque *white box*, por la disponibilidad de información desde la perspectiva del atacante y, luego, analizar si se cumple este principio. Para esta alternativa era necesario contar con un sistema de reconocimiento facial al cual se tuviera acceso. Se decidió no seguir este camino, optando por un esquema *black box*, lo cual permitió dedicar más tiempo al *autoencoder* y a la generación de imágenes adversarias. Por otra parte, una mejora sobre la metodología propuesta implica añadir más de un sistema de reconocimiento facial para validación de los resultados. La elección de Amazon Rekognition como sistema objetivo es consecuencia de los conocimientos preexistentes de los autores sobre la infraestructura de AWS.

Por otra parte, sería recomendable el uso de *hardware* especializado ya que permite ampliar posibilidades respecto a qué tipo de arquitecturas y con qué conjunto de datos es posible experimentar. En este proyecto, se buscó mantener un conjunto de datos pequeño para acelerar el entrenamiento a pesar del costo que dicha decisión implica en términos de variabilidad dentro del conjunto.

Con respecto a la arquitectura del *autoencoder*, se decidió tomar un *autoencoder* base, el cual disminuye las posibilidades al momento de armar la arquitectura, ahorrando tiempo dedicado a la selección del mismo. Si bien la construcción de la red desde cero tiene numerosas ventajas a nivel personal por el aprendizaje que conlleva, no parece recomendable para un trabajo académico cuyo objetivo no consiste en estudiar arquitecturas de redes neuronales. Por otra parte, en caso de existir, se recomienda usar una red preexistente por el tiempo invertido en analizar diferentes arquitecturas, inclusive partiendo de un *autoencoder* base.

Si bien se planteó una metodología para elegir la arquitectura del *autoencoder*, se encontraron dificultades al momento de analizar los datos ya que existía una componente importante de decisiones manuales que aportaban subjetividad a la selección. Se intentó, paso a paso, mejorar este procedimiento hasta llegar a un proceso de toma de decisiones basado en los resultados de Amazon Rekognition, descrito en la sección **6.4.** Sin embargo se cree acertada la decisión de no comenzar desde cero al encontrar falta de objetividad en algunos pasos pues el tiempo invertido en seleccionar la arquitectura del *autoencoder* no debía ser excesivo. Como recomendación, debería plantearse una metodología de selección más ordenada.

Elegir qué parámetros serían modificados durante el análisis de arquitecturas fue un proceso complejo. Un gran aprendizaje durante esta etapa es la atención que se debe prestar al momento de estudiar la influencia de diferentes variables. Cuando se realizó la distinción de aquellos *autoencoders* que necesitaban más épocas para entrenar se decidió combinar dos parámetros que, al estudiarlos por separado, requerían un aumento de iteraciones para alcanzar la convergencia: una tasa de aprendizaje del optimizador pequeña

y un tamaño mayor del *batch* para actualizar el gradiente. El problema es que esta combinación es exactamente lo opuesto a lo que debería haberse hecho, según muestra un análisis del efecto del tamaño del *batch* en redes neuronales [39]. En su momento, no pareció relevante analizar en profundidad qué efecto podían tener ambas variables en conjunto y, más adelante, se descubrió que no era el deseado.

Pasando al proceso de experimentación, un camino alternativo inexplorado fue la manipulación de proyecciones de los datos en el espacio latente. Al analizar cómo se distribuyen los datos de cada clase (Marina e Ignacio) en estas dimensiones, se observan direcciones "predominantes" para cada individuo. Por ejemplo, en ***Fig. 37***, la dimensión 3 tiene como individuo predominante a Marina, ya que sus datos se encuentran dispersos a lo largo de dicha dirección y, los de Ignacio, comprimidos a izquierda. Investigando el autovector correspondiente al primer autovalor se encontró que los coeficientes más altos (en módulo) referían a estas dimensiones y que el signo del coeficiente se correspondía con el individuo predominante. Una estrategia alternativa podría basarse en el uso de estas direcciones características, en lugar de enfocar la atención en la manipulación de la primera componente principal.

En el párrafo anterior se menciona la importancia en el uso de la primera componente principal durante la experimentación. Sin embargo, no se debe perder de vista que el ratio de varianza explicada de la misma no es significativamente mayor al resto, en especial respecto a la segunda componente, como se observa en ***Tabla 3***. Quizás el peso que se dió a la primera componente por ser la única que mostraba una separación completa de clases debería haber sido más cauteloso.

Para finalizar, se considera que uno de los cambios más importantes y necesarios refiere al análisis de calidad. El objetivo, y uno de los rasgos característicos del proyecto, era obtener ejemplos adversarios cuya calidad fuera similar a la de imágenes sin manipular. Si bien los casos presentados poseen valores de calidad similares a las imágenes del conjunto de datos originales, utilizando el ojo humano es posible notar que dichos ejemplos son resultado de alguna forma de manipulación. Sería necesario estudiar alternativas a las métricas de *Quality Sharpness* y *Quality Brightness* provistas por Amazon Rekognition, ya que la falta de un estudio objetivo de calidad de los ejemplos adversarios es una de las críticas centrales que se le hace a otras propuestas de generación de los mismos.

9.2. Sobre la relevancia del trabajo en el contexto del problema

Al momento de reflexionar sobre las posibles aplicaciones del trabajo, surge la pregunta sobre los riesgos de los ejemplos adversarios en situaciones de la vida cotidiana. Dentro de la bibliografía mencionada, la única modalidad de ataque a sistemas de reconocimiento facial que parece razonable se presenta en un artículo cuyo objetivo es la feasibilidad de trasladar los ejemplos adversarios a la vida real [29]: se imprimen parches o lentes con diferentes patrones sobre el rostro provocando errores en el reconocimiento, quedando pendiente estudiar cómo se desempeña esta técnica en video. En el caso de la metodología propuesta por los autores, el ataque se encuentra restringido a condiciones muy específicas, como las características de la foto, y requeriría de esfuerzos no triviales para ser aplicado en el mundo real.

Se cuestiona si la problemática de los ejemplos adversarios requiere la atención que recibe o si, por el contrario, debería ser una preocupación en situaciones específicas. Por ejemplo, un caso de uso de ejemplos adversarios es la manipulación de contenido para evitar restricciones de software que bloquea publicidades. Por otra parte, las redes neuronales aún no generalizan con la misma precisión aquellos datos con baja probabilidad de ocurrir. En autos de conducción autónoma, un cartel doblado por el viento continúa siendo una señal válida pues no constituye un ejemplo adversario, sino un caso atípico de la vida real. Entonces, ¿no debería ponerse mayor énfasis en otras problemáticas de las redes neuronales?

Actualmente, no se tiene consenso sobre la razón de existencia de los ejemplos adversarios. En un artículo presentado en 2019 se analizan las características de los datos, clasificándolas en robustas y no robustas [30]. Las no robustas son una posible causa de los ejemplos adversarios pero resultan de gran importancia para los clasificadores ya que, al remover estas características, la precisión del mismo desciende. En general, las características no robustas representan propiedades poco (o nada) útiles para la clasificación por parte de un actor humano. A partir de lo discutido pareciera existir en las redes de clasificación un *trade-off* entre precisión y seguridad. Quizás sería positivo acudir a otras disciplinas en busca de herramientas complementarias para resolver la problemática de ejemplos adversarios.

El trabajo propuesto, si bien el objetivo principal era la expansión de conocimiento en materia de ejemplos adversarios, finalizó como un estudio de la temática. Desde el punto de vista académico, se considera que faltó perspectiva: se tomaron caminos alternativos y se profundizaron aspectos improductivos a fin de entender, de manera temprana, que la metodología propuesta no respaldaba la hipótesis planteada.

10. Conclusión

Si bien no se encontró un proceso metódico que permita llevar a cabo un ataque realista, entendiéndose éste como un escenario factible en la vida real, la metodología empleada de experimentación (hipótesis, procedimiento y posterior análisis) fue de gran utilidad para explorar la temática y se recomienda su reutilización para el estudio de otros caminos.

Respecto al uso de análisis de componentes principales como forma de organización del espacio latente, a pesar de encontrar ejemplos puntuales de ataques de evasión e impersonificación, no se encontró evidencia concluyente sobre su utilidad para los objetivos planteados. Podrían utilizarse otras técnicas, como redes de Kohonen o un *autoencoder* variacional, en pos de la búsqueda de mejores resultados.

Otra cuestión a examinar es la forma de recorrer las direcciones en componentes principales. Durante el trabajo se limitó a una manera constante de movimiento, esto es, con un paso fijo. Quizás otras estrategias para trazar trayectorias son más útiles para los objetivos planteados. En este caso, se debería realizar una investigación sobre desplazamiento en $\mathbb{R}^n$ ya que se desconocen alternativas por parte de los autores.

Por otra parte, es necesario tener en cuenta las restricciones definidas al momento de delimitar los límites del problema. La experimentación podría arrojar resultados muy diferentes si se contase con un conjunto de datos diverso, como puede ser CelebA (*CelebFaces Attributes*) [40]. Además, el entrenamiento de una red neuronal con *hardware* especializado posibilitaría la obtención de un generador de imágenes más realistas.

A fin de enfatizar uno de los ítems discutidos en la sección **9.1**, resulta de interés resaltar la importancia de realizar un análisis de calidad independientemente del camino seleccionado para organizar el espacio latente, recorrerlo, etc. El estudio llevado a cabo debe ser superior al presentado en este informe, ya que las métricas demostraron que imágenes de resolución pobre (de acuerdo al ojo humano) tenían valores de calidad similares a imágenes sin modificar.

Como valoración final, desde la perspectiva de investigadores novatos, se considera importante realizar una análisis profundo sobre las certezas que se tienen sobre el aŕea de estudio y se cree que esto aplica no sólo a la temática de ejemplos adversarios sino a aquellos campos donde abunda la información pero falta un esquema estructurado para incorporar conocimiento. A lo largo de la realización del trabajo fue desafiante decidir cuándo poner límites en los objetivos, así como discriminar la validez de los contenidos analizados.

Bibliografía

[1] Y. Song, R. Shu, N. Kushman and S. Ermon. Constructing Unrestricted Adversarial Examples with Generative Models. *arXiv:1805.07894 [cs.LG], 2018*.

[2] A. Rosenbrock. Autoencoders with Keras, TensorFlow and Deep Learning. Retrieved from https://www.pyimagesearch.com/2020/02/17/autoencoders-with-keras-tensorflow-and-deep-learning/

[3] D. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. *arXiv:1412.6980 [cs.LG], 2017.*

[4] S. Park. A 2021 Guide to improving CNNs-Optimizers: Adam vs SGD. Retrieved from https://medium.com/geekculture/a-2021-guide-to-improving-cnns-optimizers-adam-vs-sgd-495848ac6008

[5] C. Versloot. Leaky ReLU: improving traditional ReLU. Retrieved from https://www.machinecurve.com/index.php/2019/10/15/leaky-relu-improving-traditional-relu/

[6] A. Mehra. Facial Recognition Market worth $8.5 billion by 2025. Retrieved from https://www.marketsandmarkets.com/PressReleases/facial-recognition.asp

[7] Facial Recognition: top 7 trends (tech, vendors, markets, use cases & lateset news). Retrieved from https://www.thalesgroup.com/en/markets/digital-identity-and-security/government/biometrics/facial-recognition

[8] N. Carlini. A Complete List of All (arXiv) Adversarial Example Papers. Retrieved from https://nicholas.carlini.com/writing/2019/all-adversarial-example-papers.html

[9] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik and A. Swami. Practical Black-Box Attacks against Machine Learning. *arXiv:1602.02697 [cs.CR], 2017*.

[10] Adversarial Image Translation: Unrestricted Adversarial Examples in Face Recognition Systems. K. Kakizaki and K. Yoshida. Retrieved from http://ceur-ws.org/Vol-2560/paper4.pdf

[11] M.P. Deisenroth, A. A. Faisa and C.S. Ong, “Dimensionality Reduction with Principal Component Analysis” in *Mathematics For Machine Learning* 1st Ed, Cambridge, Reino Unido: Cambridge University Press, cap. 10

[12] I. Jolliffe and J. Cadima Principal component analysis: a review and recent developments. Retrieved from https://royalsocietypublishing.org/doi/10.1098/rsta.2015.0202

[13] M. Stewart. A Comprehensive Introduction to Autoencoders. Retrieved from https://towardsdatascience.com/generating-images-with-autoencoders-77fd3a8dd368

[14] T. Hastie, R. Tibshirani, J. Friedman “Principal Components, Curves and Surfaces” in *The Elements of Statistical Learning: Data mining, Inference, and Prediction* 2nd Ed, Springer.

[15] I. J. Goodfellow, J. Shlens and C. Szegedy. Explaining and Harnessing Adversarial Examples. *arXiv:1412.6572 [stat.ML], 2015*.

[16] I. Jolliffe and J. Cadima. Principal Component Analysis: a review and recent developments. Retrieved from https://royalsocietypublishing.org/doi/10.1098/rsta.2015.0202

[17] MTCNN Documentation. Retrieved from https://pypi.org/project/mtcnn/

[18] AWS SDK for Python (Boto3). Retrieved from https://aws.amazon.com/sdk-for-python/

[19] Amazon Rekognition Documentation for its API (used with Boto3). Retrieved from https://docs.aws.amazon.com/rekognition/latest/dg/API_Reference.html

[20] Matplotlib Documentation. Retrieved from https://matplotlib.org/stable/index.html

[21] W. Clumper. How Accurate are Facial Recognition Systems. Retrieved from https://www.csis.org/blogs/technology-policy-blog/how-accurate-are-facial-recognition-systems-%E2%80%93-and-why-does-it-matter
[22] DeepAI Machine Learning Glossary. What is the manifold hypothesis? Retrieved from https://deepai.org/machine-learning-glossary-and-terms/manifold-hypothesis
[23] Scikit Learn. Python Principal Components Analysis Documentation. Retrieved from https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html
[24] Y. Zhang, X. Tian, Y. Li, X. Wang and D. Tao. Principal Component Adversarial Example. Retrieved from http://staff.ustc.edu.cn/~xinmei/publications_pdf/2020/09018372.pdf
[25] D. Hendrycks and K. Gimpel. Early Method for Detecting Adversarial Examples. *arXiv:1608.00530 [cs.LG], 2017*.
[26] N. Carlini and D. Wagner. Adversarial Examples Are Not Easily Detected: Bypassing Ten Detection Methods. Retrieved from https://dl.acm.org/doi/abs/10.1145/3128572.3140444
[27] N. Carlini, D. Wagner, F. Tramèr. Adversarial Examples, Machine Learning Street Talk. Retrieved from https://www.youtube.com/watch?v=2PenK06tvE4
[28] C. Szegedy et al. Intriguing Properties of Neural Networks. *arXiv:1312.6199 [cs.CV], 2014*.
[29] M. Pautov, G. Melnikov, E. Kaziakhmedov, K. Kireev and A. Petiushko. On adversarial patches: real-world attack on ArcFace-100 face recognition system. *arXiv:1910.07067 [cs.CV], 2020*.
[30] A. Ilyas, S. Santurkar, D. Tsipras, L. Engstrom, B. Tran and A. Madry. Adversarial Examples are not Bugs, they are Features. *arXiv:1905.02175 [stat.ML], 2019*.
[31] U. Hwang, J. Park, H. Jang, S. Yoon, N. Ik Cho. PuVAE: A Variational Autoencoder to Purify Adversarial Examples. *arXiv:1903.00585 [cs.LG]*.
[32] B. Wójcik, P. Morawiecki, M. Śmieja, T. Krzyżek, P. Spurek, J. Tabor. Adversarial Examples Detection and Analysis with Layer-wise Autoencoders. *arXiv:2006.10013 [cs.LG]*.
[33] F. Tramèr, N. Carlini, W. Brendel, A. Madry. On Adaptative Attacks to Adversarial Example Defenses. *arXiv:2002.08347v2 [cs.LG]*.
[34] R. Bird. The Post-Pandemic Workforce Requires Greater Identity Security To Achieve The New Normal. Retrieved from https://www.forbes.com/sites/forbestechcouncil/2021/08/11/the-post-pandemic-workforce-requires-greater-identity-security-to-achieve-the-new-normal/?sh=42364cba370d
[35] A. Wang. The importance of secure remote authentication during lockdown (and beyond) Retrieved from https://dis-blog.thalesgroup.com/corporate/2020/06/19/the-importance-of-secure-remote-authentication-during-lockdown-and-beyond/
[36] McKinsey Global Institute. The future of work after COVID-19. Retrieved from https://www.mckinsey.com/featured-insights/future-of-work/the-future-of-work-after-covid-19
[37] M. Dubie. Another Layer of Security for Your Remote Workforce, On Us Retrieved from https://blog.lastpass.com/2020/05/another-layer-of-security-for-your-remote-workforce-on-us/
[38] N. Carlini and C. Wagner. Towards Evaluating the Robustness of Neural Network. *arXiv:1608.04644v2 [cs.CR]*.
[39] D. Chang and A. Pathak. Effect of Batch Size on Neural Net Training. Retrieved from https://medium.com/deep-learning-experiments/effect-of-batch-size-on-neural-net-training-c5ae8516e57
[40] CelebA Dataset. Retrieved from https://www.kaggle.com/jessicali9530/celeba-dataset

Apéndice

Arquitectura del autoencoder base

```
Model: "encoder"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 28, 28, 1)]       0
_________________________________________________________________
conv2d (Conv2D)              (None, 14, 14, 32)        320
_________________________________________________________________
leaky_re_lu (LeakyReLU)      (None, 14, 14, 32)        0
_________________________________________________________________
batch_normalization (BatchNo (None, 14, 14, 32)        128
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 7, 7, 64)          18496
_________________________________________________________________
leaky_re_lu_1 (LeakyReLU)    (None, 7, 7, 64)          0
_________________________________________________________________
batch_normalization_1 (Batch (None, 7, 7, 64)          256
_________________________________________________________________
flatten (Flatten)            (None, 3136)              0
_________________________________________________________________
dense (Dense)                (None, 16)                50192
=================================================================
Total params: 69,392
Trainable params: 69,200
Non-trainable params: 192
```

```
Model: "decoder"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_2 (InputLayer)         [(None, 16)]              0
_________________________________________________________________
dense_1 (Dense)              (None, 3136)              53312
_________________________________________________________________
reshape (Reshape)            (None, 7, 7, 64)          0
_________________________________________________________________
conv2d_transpose (Conv2DTran (None, 14, 14, 64)        36928
_________________________________________________________________
leaky_re_lu_2 (LeakyReLU)    (None, 14, 14, 64)        0
_________________________________________________________________
batch_normalization_2 (Batch (None, 14, 14, 64)        256
_________________________________________________________________
conv2d_transpose_1 (Conv2DTr (None, 28, 28, 32)        18464
_________________________________________________________________
leaky_re_lu_3 (LeakyReLU)    (None, 28, 28, 32)        0
_________________________________________________________________
batch_normalization_3 (Batch (None, 28, 28, 32)        128
_________________________________________________________________
conv2d_transpose_2 (Conv2DTr (None, 28, 28, 1)         289
_________________________________________________________________
```

```
activation (Activation)      (None, 28, 28, 1)         0
=================================================================
Total params: 109,377
Trainable params: 109,185
Non-trainable params: 192
```

Parámetros iniciales del ABC

Parámetro	Valor
Dimensiones	
Alto	256
Ancho	256
Canales	1
Dimensión capa latente	64
Capas Convolucionales	
Padding	*Same*
Strides	2
Filtros Capa Convolucional 1	16
Filtros Capa Convolucional 2	32
Filtros Capa Convolucional 3	64
Núcleo Capa Convolucional 1	3
Núcleo Capa Convolucional 2	5
Núcleo Capa Convolucional 3	7
Funciones de Activación	
Activación para codificador	Leaky ReLu
Valor alpha para codificador	0,2
Activación para decodificador	Leaky ReLu
Valor alpha para decodificador	0,2
Entrenamiento	
Optimizador	Adam
Tasa de aprendizaje	1,00E-03
Beta 1	0,9
Beta 2	0,9999
Epsilon	1,00E-07
amsgrad	Falso

Reconstrucción para selección de arquitecturas

Tasa de aprendizaje del optimizador

Tasa de aprendizaje 10^{-1} | Tasa de aprendizaje 10^{-2} | Tasa de aprendizaje 10^{-3} | Tasa de aprendizaje 10^{-4} | Tasa de aprendizaje 10^{-5}

Cantidad de capas convolucionales

3 capas convolucionales | 4 capas convolucionales | 5 capas convolucionales

Cantidad de datos para actualización del gradiente

Batch Size 4 | Batch Size 8 | Batch Size 12 | Batch Size 32

Dimensión de la capa latente

Dimensión capa latente 3 | Dimensión capa latente 9 | Dimensión capa latente 16

Dimensión capa latente 32 Dimensión capa latente 64 Dimensión capa latente 128

Función de activación

Leaky ReLU Alpha 0,01 | Leaky ReLU Alpha 0,05 | Leaky ReLU Alpha 0,1 | Leaky ReLU Alpha 0,3

Leaky ReLU Alpha 0,6 | Leaky ReLU Alpha 0,9 | Leaky ReLU Alpha 1,3

Datos obtenidos con Rekognition para selección del autoencoder

Los valores presentados en la tabla han sido redondeados manteniendo 4 decimales luego de la coma.

Dataset	Mean Confidence	SD Confidence	Mean Similarity	SD Similarity	Mean Brightness	SD Brightness	Mean Sharpness	SD Sharpness
ID_1_0	100,0000	0,0003	98,3428	10,0982	82,4791	6,0569	37,7729	8,6112
ID_1_1	100,0000	0,0002	99,0623	7,1970	82,4895	6,1173	38,6177	9,1183
ID_1_2	99,9997	0,0044	97,6836	12,1530	82,5429	5,9556	38,1909	8,9645
ID_1_3	99,9998	0,0036	97,6715	12,8152	82,3019	5,8837	38,3637	9,2180
ID_1_4	99,9998	0,0033	97,7735	12,4828	82,1778	6,0637	38,5492	9,4988
ID_2_0	99,9998	0,0030	97,8291	11,8882	82,2344	6,0022	39,7759	10,0466
ID_2_1	99,9998	0,0038	97,7302	12,2530	82,2973	5,9347	40,7180	10,1970
ID_2_2	99,9998	0,0035	97,7605	12,2355	82,3485	6,0196	41,2278	10,2701
ID_2_3	99,9998	0,0033	97,8329	12,0351	82,3689	6,0049	41,9026	10,4844
ID_2_4	99,9976	0,0709	97,7380	12,4298	82,3611	5,9931	42,7289	11,0332
ID_3_0	99,9978	0,0672	97,7560	12,3442	82,3425	6,0406	41,4821	11,4005
ID_3_1	99,9954	0,1403	97,7328	12,4784	82,3171	6,0302	40,6911	11,4524
ID_3_2	99,9957	0,1344	97,6895	12,6126	82,3890	6,0788	39,9569	11,4458
ID_3_3	99,9961	0,1291	97,5304	13,1587	82,4201	6,1352	39,3538	11,4106
ID_3_4	99,9958	0,1271	97,7629	12,6137	82,4201	6,1188	39,7518	11,4493
ID_4_0	99,9999	0,0007	97,4381	12,5033	83,5325	6,6583	31,7963	8,3458
ID_4_1	99,9983	0,0323	96,9212	14,7676	83,3255	6,7770	28,9734	8,5647
ID_4_2	99,9987	0,0266	96,4541	16,0063	83,4483	6,8107	28,9305	8,3102
ID_4_3	99,9990	0,0231	96,5180	15,8799	83,2203	6,8739	28,5142	8,2005
ID_4_4	99,9991	0,0207	96,7389	15,3207	83,0847	6,7891	28,1898	8,1105
ID_5	99,3956	3,7466	88,9110	26,2089	84,8908	6,3000	21,2068	6,1944
ID_6	99,5937	3,1381	89,3062	26,9383	83,8281	6,4139	23,2503	6,6237
ID_7	99,7056	2,5816	89,6576	26,3961	83,7425	6,4889	22,3525	6,4552
ID_8	99,6801	2,5723	90,0398	26,1199	83,5294	6,4926	22,0458	6,2983

Análisis de Componentes Principales

Los valores de la tabla se corresponden a la magnitud de los autovalores resultantes de aplicar PCA.

Eigenvalues							
A1	10,802648	A2	6,846757	A3	4,1302238	A4	3,4029918
A5	3,180926	A6	3,052712	A7	2,6865425	A8	2,1077893
A9	2,0627632	A10	1,7716402	A11	1,6505468	A12	1,5524656
A13	1,4944257	A14	1,4052042	A15	1,2848547	A16	1,2703762
A17	1,1192168	A18	1,075609	A19	1,0246155	A20	0,95771646
A21	0,79937375	A22	0,74055624	A23	0,7181599	A24	0,7058683
A25	0,6630153	A26	0,60759145	A27	0,5369445	A28	0,48363867
A29	0,46602702	A30	0,43243918	A31	0,4239982	A32	0,4158757
A33	0,36800623	A34	0,35179108	A35	0,31014958	A36	0,30215493
A37	0,29062107	A38	0,2531139	A39	0,24391817	A40	0,22814539

A41	0,19129087	A42	0,18587112	A43	0,17680734	A44	0,17156887
A45	0,15480174	A46	0,13777995	A47	0,12790592	A48	0,11446608
A49	0,10293874	A50	0,08801629	A51	0,08473146	A52	0,07610799
A53	0,071360394	A54	0,065281175	A55	0,0570183	A56	0,052949436
A57	0,049825	A58	0,03970099	A59	0,034882165	A60	0,033514645
A61	0,027005587	A62	0,024766479	A63	0,023369921	A64	0,008213418

Detalle de Modificaciones para Experimento 1

Los valores de la tabla corresponden a las modificaciones de la primera y segunda componente principal realizadas sobre las imágenes de Marina e Ignacio durante el experimento 1. Las imágenes correspondientes a las manipulaciones se encuentran en ***Fig. 33*** y ***Fig. 34*** respectivamente.

Potential Adversarial Examples, Modifications				
Source Image	PC1 Old Value	PC1 New Value	PC2 Old Value	PC2 New Value
ignacio average	-3,4602	0,5000	-0,3558	-3,5000
ignacio average	-3,4602	2,0000	-0,3558	-0,5000
ignacio average	-3,4602	4,5000	-0,3558	1,0000
ignacio average	-3,4602	5,5000	-0,3558	-3,0000
marina average	3,5583	-1,5000	-1,1882	-3,0000
marina average	3,5583	-3,0000	-1,1882	-1,5000
marina average	3,5583	-3,5000	-1,1882	-3,5000
marina average	3,5583	-4,0000	-1,1882	-4,0000

Detalle de Modificaciones para Experimento 6

Los valores de la tabla corresponden a las modificaciones de la primera componente principal realizadas sobre los ejemplos adversarios potenciales del ítem 5 del procedimiento del experimento 6. Debajo de dicha tabla se encuentran las imágenes resultantes de modificar dicha componente.

Potential Adversarial Examples, Modifications		
Source Image	PC1 Old Value	PC1 New Value
ignacio0	-3,551283	5,00
ignacio11	-3,813199	6,40
ignacio14	-3,199674	4,40
ignacio21	-3,987424	6,40
ignacio29	-3,389351	6,20
ignacio49	-2,824564	5,40
ignacio51	-2,916997	6,40
ignacio54	-3,654767	6,40
ignacio76	-3,037767	6,20

ignacio87	-2,562436	6,00
ignacio88	-3,223754	4,20
test-ignacio0	-3,774843	6,00
marina17	3,251978	-4,20
marina21	3,875732	-1,60
marina29	3,099534	-2,80
marina33	3,993810	-1,60
marina50	2,776832	-4,20
test-marina7	3,143551	-4,00

A continuación se presenta una segunda tabla, correspondiente a las modificaciones de la primera componente principal de aquellas imágenes que mantuvieron la etiqueta asignada por el oráculo (a pesar de dicha manipulación). Debajo de dicha tabla se encuentran las imágenes resultantes de la decodificación de las componentes principales manipuladas.

Images with same Oracle's Tag, Modifications		
Source Image	PC1 Old Value	PC1 New Value
marina2	1,8581791	-1,60
marina3	1,8956704	-1,60
marina12	1,0469455	-1,60
marina13	-0,1853556	-1,80
marina18	0,2062798	-1,60
marina20	-0,2197045	-3,00
test-marina6	-0,2140693	-1,60


source: marina2
source: marina3
source: marina12
source: marina13
source: marina18
source: marina20
source: test-marina6

Anexo A

A medida que se acercaba el final del trabajo, el tutor propuso una serie de preguntas para reflexionar sobre el proyecto final. A continuación se encuentran las preguntas con las respectivas respuestas de los autores.

¿Qué problema resuelve? Este proyecto no busca resolver un problema particular sino expandir la frontera de conocimiento respecto al área de estudio de ejemplos adversarios. Como mencionamos durante el estado del arte, los ejemplos adversarios fueron presentados por primera vez en 2014 y todavía hoy no se tiene consenso sobre su causa. A través de la creación de una nueva metodología de generación de ejemplos adversarios intentamos encontrar datos relevantes para la problemática.

¿Cuál es el estado del arte? Fue presentada en la sección Estado del Arte del informe.

¿Qué conocimientos de la carrera les fueron útiles? De las numerosas materias cursadas durante la carrera, aquellas que se revelaron de mayor utilidad son:

- Sistemas de Inteligencia Artificial, ya que tanto el área de estudio como las principales herramientas para la metodología de ataque fueron presentadas en dicho curso.
- Ingeniería de Software II, principalmente durante el planteo de un camino alternativo analizado en Marzo de 2021: una aplicación web para implementar el entorno de experimentación. Nos permitió diagramar funcionalidades, supuestos, *trade-offs*, etc.
- Machine Learning, pues en esta materia se pone especial atención en la correcta evaluación de los resultados obtenidos al momento de implementar distintas soluciones.
- Cloud Computing, dado que se utilizaron las herramientas de computación en la nube, incluido el objetivo de ataque: Amazon Rekognition.
- Criptografia y seguridad, pues ambos teníamos un especial interés en elegir un tema relevante para la seguridad informática por la curiosidad presente durante la materia. También fue de utilidad para analizar la viabilidad al momento considerar ataques en un contexto real.
- Álgebra, ya que permitió comprender la teoría detrás del análisis de componentes principales.

¿Qué es único de esta tésis/PF que no hayan encontrado implementado o investigado en otros lugares? Principalmente, hay dos elementos de este proyecto final que no se han encontrado durante la investigación, los cuales son:

- El uso de un *autoencoder* combinado con análisis de componentes principales en su espacio latente para la generación de los ejemplos adversarios.
- El análisis de calidad de potenciales ejemplos adversarios con respecto a imágenes originales. Si bien en este caso era de gran relevancia, dado que el autoencoder no se correspondía con el estado del arte, no encontramos mención a la calidad en ninguno de los papers de la bibliografía ni tampoco a la posibilidad de que la misma influya como factor durante el reconocimiento o la clasificación de un objeto.

¿Por qué eligieron este problema, y cómo se relaciona con el equipo? Seleccionamos esta problemática pues ambos compartíamos interés por los temas relacionados con la seguridad y, durante la búsqueda dentro de este área, surgió la duda sobre posibles vulnerabilidades en sistemas de reconocimiento facial. A lo largo de años trabajando como equipo, siempre sucedió que uno de nosotros demostraba mayor interés por el aspecto teórico de un tema y, el otro, por el técnico. Resultó ser una excelente combinación para introducirnos en la investigación, a la cual ninguno de los dos se había dedicado anteriormente.

¿Existe un producto o servicio que pueda dar uso al trabajo que realizaron? No, el trabajo actual tiene una finalidad puramente académica y, además, concluimos que la metodología no era una alternativa idónea de investigación, mucho menos para aplicar en un escenario de la vida real.

¿Qué harían si tuvieran más tiempo? Coincidimos en que se realizaría un análisis más profundo del estado del arte, en especial ahondar en hipótesis e investigación sobre posibles explicaciones de los ejemplos adversarios. Por otra parte, desde el lado técnico, creemos que es importante validar que los resultados obtenidos para los individuos del conjunto de datos se replican cuando se cambian estos individuos.

¿Qué decisiones importantes tuvieron que tomar? Las decisiones que consideramos más relevantes fueron analizadas en la sección Discusión y en dos preguntas posteriores:

- ¿Qué errores creen haber cometido en el camino?
- ¿Qué aciertos creen haber cometido en el camino?

¿Qué aprendieron en el proceso? Desde el punto de vista técnico y metodológico, nuestros principales aprendizajes fueron:

- Metodología de investigación: como abordar un problema de investigación.
- El funcionamiento de *autoencoders* y análisis de componentes principales.
- El funcionamiento de Amazon Rekognition.

Por otro lado, desde la perspectiva de gestión de procesos, nuestros principales aprendizajes fueron:

- De ser posible, contar con gente activa en el área de investigación a la que uno se acerca.
- Al momento de realizar estimaciones, tener presente la posibilidad de una subestimación sobre la tarea a realizar. Nos sucedió con la aplicación web que intentamos desarrollar (ver **Anexo B**).
- Delimitar los límites de los objetivos y tareas pues el trabajo puede extenderse indefinidamente.
- Esquematizar claramente el corto, mediano y largo plazo. Luego, realizar verificaciones periódicas ya que es sencillo perder de vista el objetivo principal al tener tantas posibilidades. Sucedió que ciertos aspectos del trabajo, como el entrenamiento del *autoencoder*, llevaron mucho tiempo en relación a su aporte a la problemática que buscábamos estudiar.

- Comunicación honesta y temprana entre los miembros del equipo. Hablar de las preocupaciones que teníamos con respecto al rumbo elegido y a la metodología con la cual se trabajaba fue esencial para debatir sobre las ventajas y desventajas de las distintas opciones con las que contábamos.

¿Qué difiere entre lo que tenían pensado realizar el primer mes y lo que terminaron realizando? Esperábamos encontrar una relación más clara entre las proyecciones de las primeras componentes principales y los resultados obtenidos con Amazon Rekognition. También esperábamos obtener una metodología genérica de manipulación, mientras que al final no se encontró un vínculo definido y los experimentos resultaron bastante más manuales.

¿Qué errores creen haber cometido en el camino? Si bien varios errores fueron comentados en la sección Discusión, nos gustaría resaltar tres decisiones que, en última instancia, perjudicaron el desarrollo:

- Al principio de la selección del *autoencoder*, no fuimos diligentes en almacenar todos los resultados obtenidos. Esto trajo retrasos temporales en la selección del mismo.
- La falta de una metodología objetiva para la elección del autoencoder. Muchas decisiones fueron basadas en observaciones subjetivas.
- Por último, el que consideramos el mayor desacierto es el desvío a la aplicación web. Creíamos que era una buena forma de concretar el proyecto final con un entregable pero subestimamos el tiempo necesario para el desarrollo de la misma y sobreestimamos la relevancia que tenía en nuestro trabajo.

¿Qué aciertos creen haber cometido en el camino? Dos aciertos que nos parece importante mencionar son, en primer lugar, abordar la investigación sin miedo a que la hipótesis planteada sea falsa. En segundo lugar, haber vuelto a la investigación original luego de dedicar dos meses al desarrollo de la aplicación Dexter. Si bien fue una decisión difícil, creemos que fue la correcta para darle un cierre satisfactorio al trabajo.

¿Por qué creen que lo que realizaron es meritorio de aprobación? Se hizo un estudio en profundidad sobre la temática de investigación y, si bien no se llegó a probar las hipótesis planteadas, el proceso de investigación y de toma de decisiones se realizó buscando la mayor claridad y objetividad que pudimos lograr. Probar la hipótesis no era el foco del proyecto, sino realizar un trabajo de investigación exhaustivo y crítico de la propuesta.

¿Qué tipo de audiencia estaría interesada en lo que realizaron? El trabajo está orientado a personas interesadas en la temática de *deep learning*, no sólo desde una perspectiva práctica, sino desde el aspecto académico y quizás filosófico. Este proyecto es un ejemplo particular de un fenómeno curioso que ocurre con los algoritmos que, si bien se encuentra bajo investigación, trae aparejado consigo muchas preguntas sobre la idoneidad de las herramientas de *deep learning* en la vida cotidiana.

¿Qué perfil de estudiantes podrían continuar con su trabajo? En primer lugar, creemos que es un trabajo interesante para estudiantes que busquen adentrarse en el mundo de la investigación, ya que realizamos, a nuestro parecer, críticas constructivas sobre el abordaje a la investigación desde la perspectiva de novatos. Dado que la temática principal del

trabajo es de inteligencia artificial, se esperaría que dichos alumnos tengan curiosidad sobre la materia y conocimientos sobre los conceptos básicos.

Anexo B

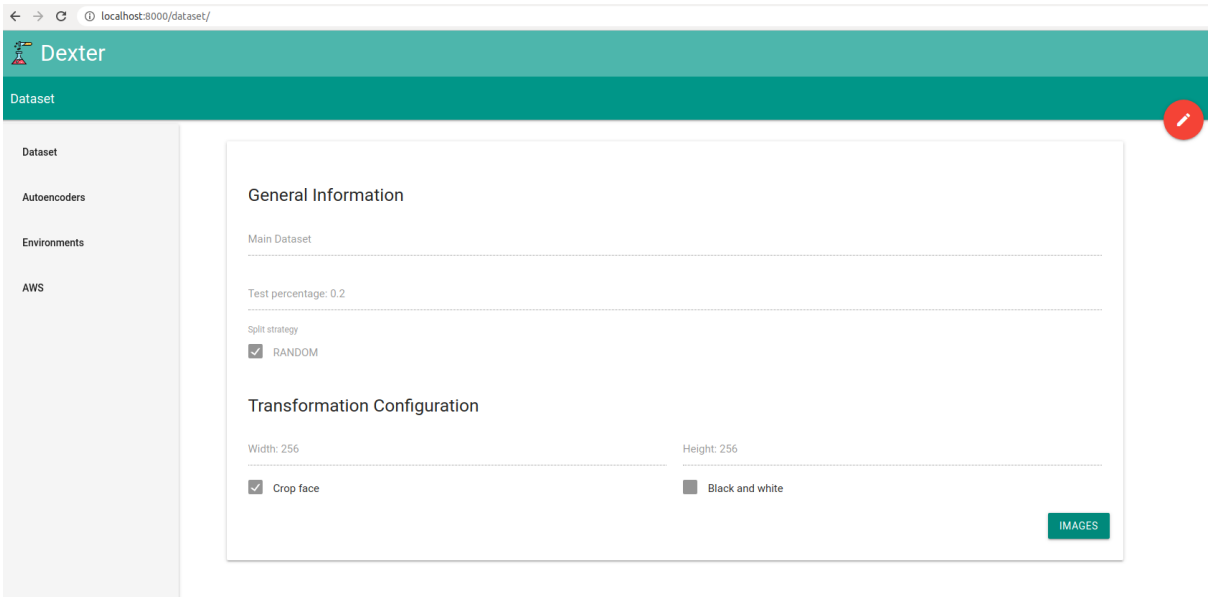
Dexter es el nombre que se dio a una aplicación web cuyo desarrollo empezó en Marzo, 2021. El objetivo era realizar una plataforma que permitiera llevar a cabo experimentos de forma sencilla a través del desarrollo de una GUI, incluyendo

- Creación y entrenamiento de *autoencoders* con distintos conjuntos de datos y arquitecturas.
- Creación de diferentes entornos de experimentación que incluyeran un *autoencoder* y sistema de reconocimiento facial específico.
- Armado de experimentos y obtención de resultados para posterior análisis.
- Integración con sistemas de reconocimiento facial para probar el principio de transferibilidad. En principio se buscaba añadir Amazon Rekognition.

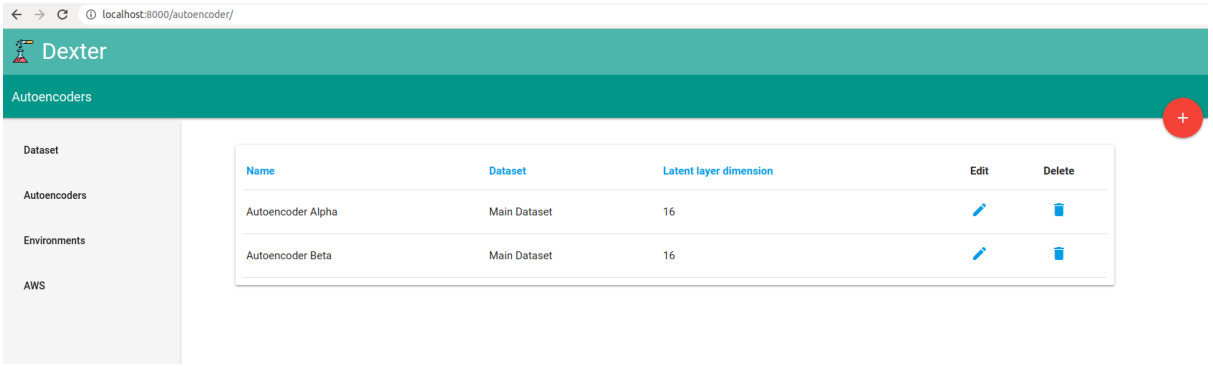
Este proyecto se desarrolló por dos meses pero fue durante Junio que se decidió retomar el camino original ya que la aplicación era un proyecto muy ambicioso como elemento complementario de la investigación. Teniendo en cuenta que ya se había dedicado mucho tiempo a elementos complementarios pero no centrales a la metodología de ataque (como la selección del *autoencoder*) se decidió volver al camino original y dejar este proyecto en pausa indefinida.

A continuación, algunas vistas de la aplicación:

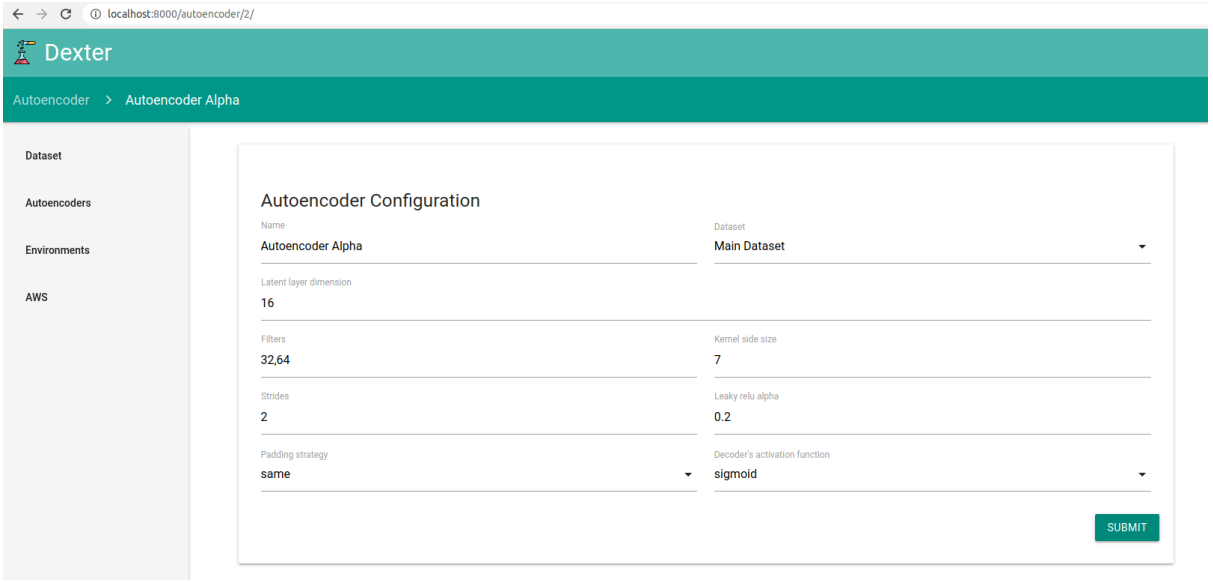
Vista en detalle de las opciones de creación o edición del conjunto de datos. En el caso de la captura de pantalla se observa la edición de un conjunto de datos ya creado.



Vista en detalle de la lista de *autoencoders* creados. En el caso de la captura de pantalla se tienen dos *autoencoders*.

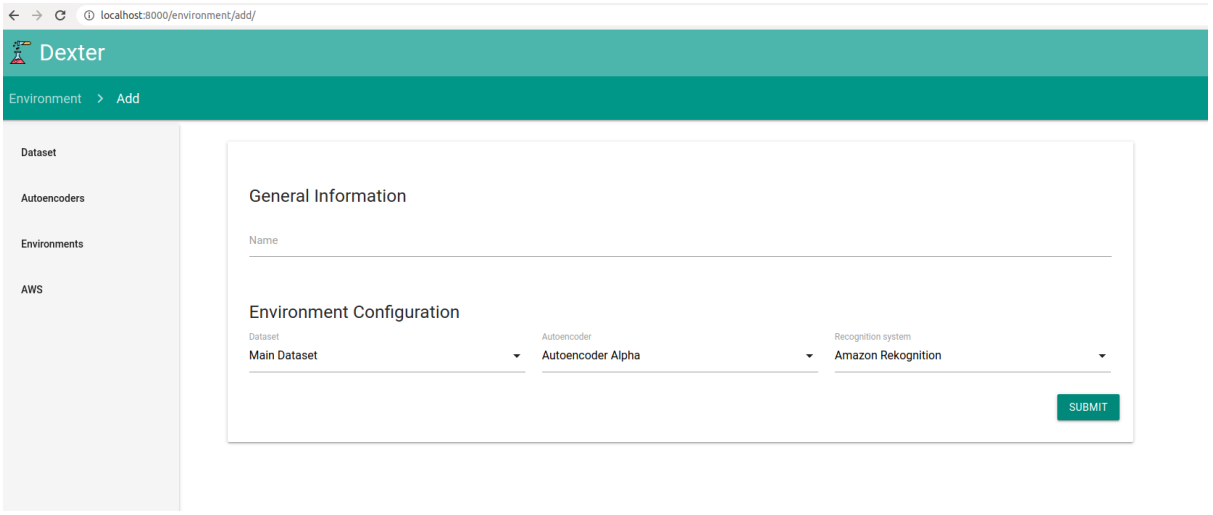


Vista en detalle de las opciones de creación o edición de un *autoencoder*.



Vista en detalle de la lista de ambientes de experimentación.

Vista en detalle de las opciones de creación o edición de un ambiente de experimentación.



Vista en detalle de la lista de experimentos para un ambiente en particular, en este caso, Env0.