Escuela de Ingeniería
Ingeniería Civil en Computación

Detección de face liveness falsos mediante cámaras de eventos

Hernán Moreno Bustamante
Profesor guía: Rodrigo Verschae Tannenbaum
Profesor co-guía: Ignacio Bugueño Córdova
Comisión evaluadora: Stefan Escaida, M. de los Ángeles Rodríguez

Memoria para optar al título de Ingeniero Civil en Computación

Rancagua, Chile
Enero de 2024

Dedicatoria

Esta memoria está dedicada a mi familia, quienes me acompañaron a lo largo de este proceso, siendo el principal apoyo y motivo de seguir adelante.

Agradecimientos

En primer lugar, quiero agradecer a mis padres por su apoyo incondicional, cuyo sacrificio y aliento han sido la chispa que iluminó y guío este camino.

Agradezco a mi familia que me apoyó, en especial a Jaime Aguirre y Pamela Soto, quienes me acogieron en su hogar mientras realizaba mi práctica profesional.

Agradezco a mi profesor guía, Rodrigo Verschae, y a mi profesor coguía, Ignacio Bugueño, por haberme acompañado durante esta última etapa universitaria. Además, agradezco a M. de los Ángeles Rodríguez por sus valiosas contribuciones como miembros de mi comité evaluador. Sus comentarios y sugerencias enriquecieron significativamente mi trabajo.

Agradezco a cada uno de los profesores que me han guiado a lo largo de la carrera, y entre ellos quiero agradecer especialmente al profesor Rodrigo Delgado, que fue uno de los primeros profesores que tuve de la rama de computación y una de las motivaciones que tuve a lo largo de esta. Nuevamente agradezco a Ignacio Bugueño, quien confió en mí y me apoyo académicamente, brindándome oportunidades que me permitieron desarrollarme profesionalmente.

Agradezco especialmente a mis amigos más cercanos Felipe Gómez, Vicente González y Camila Rapu, con quienes he compartido enormemente durante los últimos años de la carrera y han sido un apoyo tremendo a lo largo de la misma. Además, muchas gracias a cada uno de mis compañeros de carrera y universidad con quienes he aprendido y crecido a lo largo de estos años.

Este trabajo no habría sido posible sin la contribución de todos ustedes. Aprecio enormemente su generosidad y apoyo. A cada uno de ustedes, gracias por estar a mi lado. Este logro no habría sido posible sin su constante aliento y amor.

Índice

Resumen

En el presente trabajo, se busca diseñar una base de datos de *face liveness detection* con datos recopilados de cámaras de eventos. El *face liveness detection* (o detección de vida facial, en español) es una técnica de autenticación biométrica y sistemas de seguridad que se utiliza para determinar la presencia física de un usuario frente al sensor mediante la identificación y análisis de un rostro y los movimientos del mismo. Las cámaras de eventos son sensores de visión neuromórficos, lo que quiere decir, que su funcionamiento se inspira en cómo lo realiza el cerebro humano difiriendo así de los sensores de visión tradicionales, ya que capturan y procesan la información espacio-temporal de forma diferente.

El diseño de la base de datos se basa en la base de dato NEFER, la que se presenta en el artículo *Neuromorphic Event-based Facial Expression Recognition* (NEFER) (Berlincioni et al., 2023), la que cuenta con datos de rostros y posee datos de eventos y frames en rgb. Además con la base de datos se planea abordar *replay attack* (ataque de reproducción, en español) este tipo de ataque consiste en la utilización de vídeos pregrabados, los cuales se reproducen frente a una cámara o sensor, para engañarlo y que este detecte que lo que tiene enfrente no es un video sino una persona.

Para lograr esta detección, se exploran modelos de aprendizaje profundo para utilizar los datos de la base de datos propuesta, decantándose por la implementación de redes convolucionales (CNN) y con una arquitectura ResNet34, con la que se entrenan y validan los datos, con diferentes configuraciones; se entrenan diferentes modelos basados en la configuración de los datos de entrada. Los datos de entrada corresponden visualizaciones de *time surface* de los eventos de los rostros que fueron recortados de las capturas de datos, estas visualizaciones se realizan con diferentes ventanas de tiempo (Δt), específicamente para ventanas de 5, 10, 33 y 50 ms (milisegundos), por último se aplica un filtro por peso a las imágenes, esto para disminuir la cantidad de imágenes que no aportan información a la red a la hora de ser entrenada ni validada.

Los modelos entrenados muestran un alto desempeño, en muchos casos con una precisión del 100%.

Palabras clave: Cámara de eventos, Deep Learning, Red Convolucional, Red Generativa Adversativa

Introducción

En esta investigación, se persigue el diseño y creación de una base de datos para *face liveness detection*, enfocándose en los ataques por reproducción (*replay attack*), para esto se aprovecha la innovadora tecnología de cámaras de eventos. Estas cámaras, poseen la capacidad única de capturar información espacio-temporal, lo que las presenta como candidatas a trabajar en entornos en los que se requiere medir los cambios en escena a lo largo del tiempo.

La inclusión de los sensores neuromórficos, presenta una ventaja al trabajar con estímulos pequeños, como pueden ser los movimientos naturales que ocurren en los rostros de las personas, el sensor detectar esto gracias a su alta resolución temporal, la que puede trabajar en el rango de los microsegundos (µs), la captura de la polaridad (p) asociada al cambio de luminosidad del píxel y el registro preciso de la ubicación (x, y) del evento. Además, estas cámaras poseen la capacidad de operar en condiciones de luminosidad variables, lo que las convierte en valiosas aliadas en situaciones donde las cámaras convencionales pueden presentar limitaciones por la luminosidad cambiante del ambiente.

La metodología asociada a la implementación de *face liveness detection* se centrará en el uso de modelos de aprendizaje profundo, con énfasis en redes neuronales convolucionales (CNN). Estos modelos se entrenarán con la base de datos recopilada, permitiendo una evaluación efectiva de su capacidad para discernir entre rostros reales y representaciones falsas.

En consecuencia, el objetivo primordial de este trabajo consiste en abordar el desafío de discriminar entre rostros auténticos e intentos de ataque de reproducción mediante el análisis de patrones visuales detectados por cámaras de eventos. En paralelo, se llevará a cabo un minucioso estudio del estado del arte en *face liveness detection*, identificando y evaluando los métodos aplicados a las cámaras convencionales, con la intención de contextualizar y validar la contribución significativa de las cámaras de eventos en este campo.

Es así que, uno de los objetivos principales de este trabajo, además de la creación de la base de datos, es lograr una discriminar efectiva entre rostros reales y representaciones falsas, mediante el análisis de patrones visuales detectados por una cámara de eventos. Para esto, durante el trabajo de título se realizará un estudio del estado del arte actual de *face liveness detection*, identificando los métodos utilizados actualmente para las cámaras convencionales.

Motivación

En la era digital actual, la autenticación biométrica basada en el reconocimiento facial se ha vuelto ampliamente utilizada en una gran variedad de aplicaciones, desde su presencia en dispositivos móviles, hasta sistemas de seguridad crítica. Lamentablemente, junto a su adopción masiva esto ha dado lugar a un aumento en los ataques de suplantación de identidad, lo que pone en riesgo la integridad y seguridad de los sistemas informáticos que dependen de la autenticación facial, por lo que la necesidad de desarrollar métodos efectivos de *face liveness detection* se ha vuelto crucial para garantizar la autenticidad de las identidades verificadas y proteger la privacidad y seguridad de los usuarios.

Problemática

La problemática fundamental que se planea abordar es robustecer los sistemas de *face liveness detection*, los cuales son propensos a ataques los cuales cada vez son más sofisticados, lo que subraya la necesidad de desarrollar soluciones más avanzadas. Si bien existen enfoques tradicionales que abordan este problema, estos se vuelven vulnerables en determinadas condiciones ambientales.

Hipótesis

Las cámaras de eventos presentan un notable beneficio ante los ataques de suplantación, reconociendo con alta precisión los ataques de *liveness*, independiente de las situaciones de iluminación, situación en la que las cámaras tradicionales (RGB) se suelen saturar debido a la sensibilidad de esta. Es así que planteamos en este trabajo que la cámara de eventos, sensor neuromórfico, presenta una mayor eficiencia en la detección de intentos de suplantación facial en comparación con enfoques tradicionales, lo que respaldara su aplicabilidad en una amplia gama de aplicaciones de seguridad y autenticación biométrica.

Propuesta de solución

Para abordar este problema se propone la utilización de un sensor de visión neuromórfico de eventos, cámara basada en eventos, que detecte con detalle los cambios en las escenas. Por otra parte, para procesar los datos se propone que sea mediante un modelo de aprendizaje profundo como, por ejemplo, el uso de las redes convolucionales para capturar patrones visuales complejos en los cambios de la imagen facial.

Objetivo general

Diseñar y crear una base de datos para *face liveness detection*, enfocándose en los ataques por reproducción (*replay attack*). Además, de explorar, diseñar, implementar, entrenar y validar métodos de *face liveness detection* basado en cámaras de eventos, validando su desempeño en diferentes escenarios. El trabajo buscará aprovechar las capacidades de las redes convolucionales para capturar patrones visuales complejos en los cambios de la imagen facial que se producen en respuesta a eventos específicos.

Objetivos específicos

- Diseñar y crear una base de datos para *face liveness detection*, con datos de rostros de personas y casos de ataques por reproducción (*replay attack*) basado en cámaras de eventos.
- Diseñar una metodología de investigación que incluya la selección de fuentes relevantes para el trabajo, junto al enfoque metodológico para recopilar, pre-procesar y analizar datos.
- Diseñar, desarrollar, implementar, entrenar y validar un método para abordar el problema de investigación, considerando diversas técnicas de inteligencia artificial. Validando su correctitud y optimalidad mediante métricas de desempeño.

Descripción del documento

En el presente documento se diseña y crea una base de datos para *face liveness detection*, enfocándose en los ataques por reproducción (*replay attack*). Además de la implementación de métodos *face liveness detection* probando las capacidades de la base de datos presentada. Para este se emplean métodos de detección de objetos del estado del arte de Aprendizaje Profundo (*Deep Learning*) para la búsqueda y determinación de patrones en los datos que ayuden a la detección, aprovechando el potencial de las cámaras de eventos para la captura de patrones visuales complejos y cambios en la imagen facial en tiempo real, permitiendo una detección de vida más precisa y eficaz.

Marco teórico y revisión de literatura

Machine learning (ML)

El *machine learning*, o aprendizaje automático en español, es una rama de la inteligencia artificial que se enfoca en el desarrollo de algoritmos y modelos que permiten procesar grandes cantidades de datos históricos e identificar patrones de datos, generando resultados con mayor precisión a partir de un conjunto de datos de entrada. Estos modelos se pueden aplicar en varios sectores, como por ejemplo: en procesos de fabricación, sanidad y ciencias biológicas, servicios financieros para una administración financiera personalizada o incluso en sistemas de recomendación de contenido multimedia (*What Is Machine Learning (ML)? - Enterprise ML Explained*, n.d.).

El *machine learning* incluye una amplia gama de modelos y algoritmos, los cuales se pueden clasificar en cuatro estilos de aprendizaje distintos en función de la salida esperada y del tipo de entrada, las que se describen a continuación (*What Are Neural Networks?*, n.d.):

- **Aprendizaje supervisado:** Los algoritmos se entrenan con datos etiquetados y definidos para evaluar las correlaciones, es decir, al entrenar los datos poseen la entrada y su salida correspondiente.
- **Aprendizaje sin supervisar:** Los algoritmos se entrenan con datos no etiquetados previamente. Estos algoritmos analizan los datos de entrada con la intención de establecer conexiones significativas entre las entradas y salidas predeterminadas.
- **Aprendizaje semisupervisado:** Este método combina el aprendizaje supervisado y el no supervisado. Esta técnica se basa en el uso de una pequeña cantidad de datos etiquetados y de una gran cantidad de datos sin etiquetar. Primeramente, el conjunto de datos etiquetados se utiliza para entrenar parcialmente el algoritmo, una vez realizado, con el algoritmo entrenado parcialmente se procede a etiquetar los datos no etiquetados, y finalmente, el modelo se vuelve a entrenar con la mezcla de datos resultante sin programarlo explícitamente.
- **Aprendizaje reforzado:** Este método utiliza valores de recompensa adjuntos a los diferentes pasos que debe dar el algoritmo. Así, el objetivo del modelo es acumular tantos puntos de recompensa como sea posible y alcanzar una meta final. Si bien este método funciona mejor en entornos de datos inciertos y complejos, no es eficiente para tareas bien definidas y el sesgo del desarrollador que diseñó el algoritmo puede afectar los resultados.

Deep Learning (DL)

El *deep learning*, o aprendizaje profundo en español, es un subcampo del machine learning, la cual trata de redes neuronales artificiales de múltiples capas ocultas sucesivas e

interconectadas que procesan y aprenden automáticamente a partir de grandes cantidades de datos de manera más efectiva, inspirándose en la forma en que lo hace el cerebro humano. Es así que, los datos fluyen desde la capa de entrada a través de varias capas de redes neuronales ocultas antes de llegar a la capa de salida. Las capas adicionales ocultas permiten un aprendizaje mucho más eficaz que el de los modelos estándar de *machine learning*.

Siendo por la gran cantidad de capas que las componen se suelen denominar aprendizaje profundo. Y la cantidad de capas que contribuyen a un modelo de datos se denomina profundidad del modelo.

Por otro lado, a diferencia del *machine learning*, el *deep learning* elimina parte del procesamiento previo de datos. Estos algoritmos pueden procesar datos no estructurados, como texto e imágenes, y automatizar la extracción de características.

Redes Neuronales Artificiales

Las Redes Neuronales Artificiales (ANN, por sus siglas en inglés) son modelos que se inspiran en la forma en que cerebro transmite la información, consistiendo así en un conjunto de nodos, o también denominados neuronas artificiales o perceptrón, interconectados entre sí para transmitirse combinaciones de entradas de datos, ponderaciones y sesgos. Cada uno de los nodos recibe una o varias entradas, de las antes mencionadas, con las que realiza una serie de cálculos y produce así una salida dirigida a otra neurona; la que a su vez se encuentra conectada a otra (What Are Neural Networks?, n.d.).

Al referirse a que poseen una ponderación asociada, o también llamados pesos, a la entrada que reciben estos indican la importancia relativa de las señales que fluyen a través de ellas. Al comienzo del entrenamiento, los pesos suelen ser inicializados de manera aleatoria y se van ajustando a medida que el entrenamiento se lleva a cabo. Entre las tareas en las que estas redes se pueden utilizar se puede encontrar la clasificación de imágenes, procesamiento de lenguaje natural, la detección de patrones, entre otros.

La organización de las neuronas dentro de la red se realiza según niveles, o también llamados capas, según su función y su posición en la arquitectura de la red. Cada tipo de capa tiene un propósito específico en el procesamiento de datos y en la modelación de relaciones en los datos. según su, en donde los tipos de capas más comunes en una ANN son (*The Neural Networks Model*, 2021):

- **Capa de entrada:** La función principal de esta capa es recibir los datos de entrada y simplemente transmitirlos a través de la red, y no realiza ninguna otra operación.
- **Capa oculta:** Estas capas son responsables de aprender representaciones intermedias y transformaciones de los datos. Cada capa oculta realiza cálculos basados en las entradas que recibe y las transmite a la siguiente capa oculta o a la capa de salida.

Tanto el número de capas ocultas, como la cantidad de neuronas en cada capa, es ajustable y depende de la arquitectura de la red y la complejidad de la tarea.

- **Capa de salida:** Esta es la capa final de la red y se encarga de producir la salida final de la red. La cantidad de neuronas en esta capa depende de la naturaleza de la tarea, como una neurona para clasificación binaria o varias neuronas para clasificación multiclase.

A continuación, en la figura 1 se muestra un esquema de una ANN, en donde se observa una capa de entrada de 3 neuronas, junto a dos capas ocultas de 6 neuronas cada una y finalmente en la capa de salida 2 neuronas. Además, en la misma figura se logra observar que cada neurona de cada capa se conecta son todas las neuronas de la capa siguiente, lo que se conoce como una red *fully-connected*.

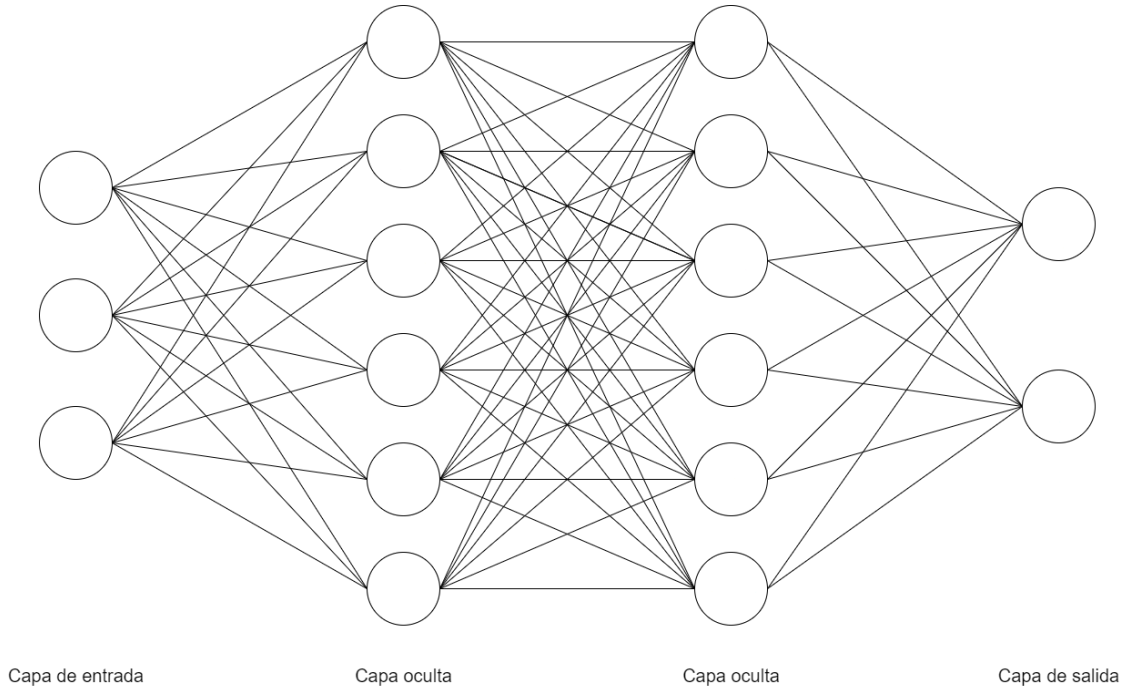

Figura 1. Diagrama de Red Neuronal Artificial (ANN) *fully-connected*. Elaboración propia.

Dentro de las operaciones que realizan las neuronas de la red, los valores pasan a través de una función de activación, la que actúa como un filtro que modifica el valor obtenido de la neurona, controlando el comportamiento de las neuronas de la red. A continuación, se mencionan algunas de las funciones de activación más comunes y algunas características de estas:

- **Función Sigmoide (Sigmoid):** Es una función matemática utilizada en redes neuronales y otros modelos de aprendizaje automático, esta toma cualquier número real como entrada y lo mapea a un valor en el rango de 0 a 1, siendo útil en problemas de clasificación binaria. Su forma se asemeja a una curva en forma de "S" y tiene la siguiente fórmula:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

En la función, si la entrada es un número grande y positivo, la salida obtenida se acerca a 1, y si la entrada es grande y negativa, la salida se acerca a 0. Por otro lado, cuando la entrada posee el valor 0, la salida es exactamente 0.5.

- **Función Tangente Hiperbólica (Tanh):** Esta función es similar en forma a la función sigmoide, y al igual que esta, la función $tanh$ es útil para problemas de clasificación binaria y regresión donde se necesita una función que aplique una transformación no lineal a los datos. Toma una entrada real y la mapea a un valor en el rango de -1 a 1, es decir, si la entrada es grande y positiva, la salida se acercará a 1, y si la entrada es grande y negativa, la salida se acercará a -1. Por otro lado, cuando la entrada es cercana a cero, la salida es cercana a cero también. La función $tanh$ tiene la siguiente fórmula:

$$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

- **Función Rectificadora Lineal (ReLU - Rectified Linear Unit):** Es una de las funciones de activación más utilizadas en capas ocultas debido a su eficacia y simplicidad, esta consiste en mantener los valores positivos de la entrada, mientras que los valores negativos los cambia por cero. La función tiene la siguiente fórmula:

$$f(x) = \max(0, x)$$

- **Función Softmax:** Es una función de activación usada principalmente en la capa de salida, cuando se suele abordar problemas de clasificación multiclase. Esta función calcula la distribución de probabilidades de cada clase objetivo sobre todas las clases objetivo-posibles.

$$\text{softmax}(x)_i = \frac{e^{x_i}}{\sum_j e^{x_j}}$$

A continuación, en la figura 2 se muestra la representación gráfica de las funciones de activación antes mencionadas.

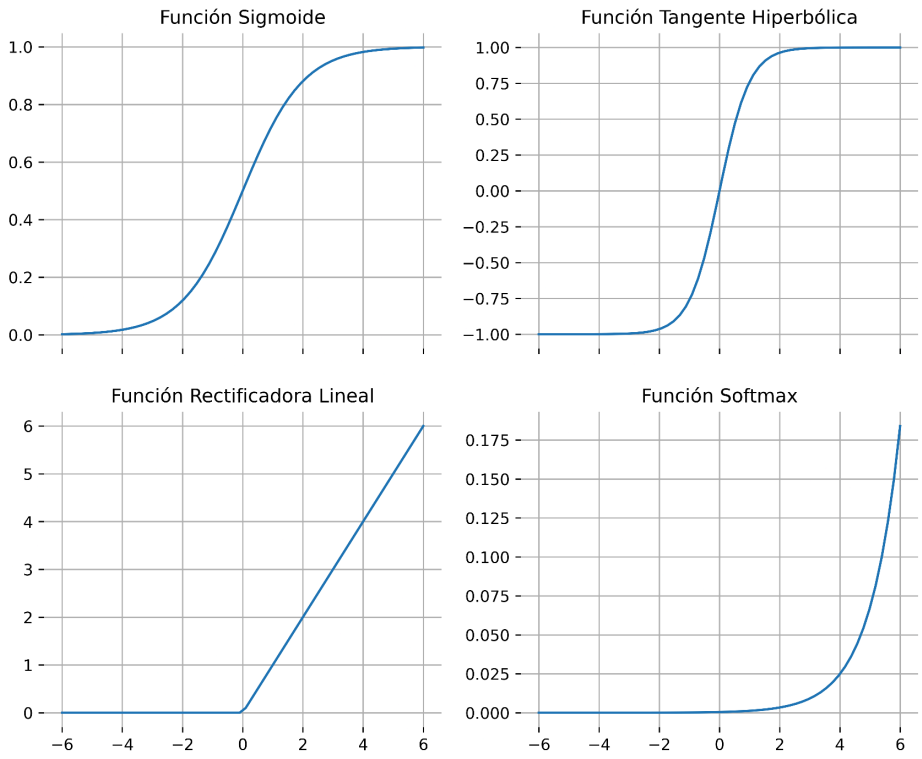

Figura 2. Representación gráfica de estas funciones de activación. Elaboración propia.

Es así que si representamos gráficamente un perceptrón con un con las entradas x_i, pesos w_i, pasando por un nodo en donde se suman las ponderaciones de las entradas por sus pesos respectivos, con el valor resultante se pasa por la funcion de activacion $f()$, obteniendo así la salida $\hat{y}$.

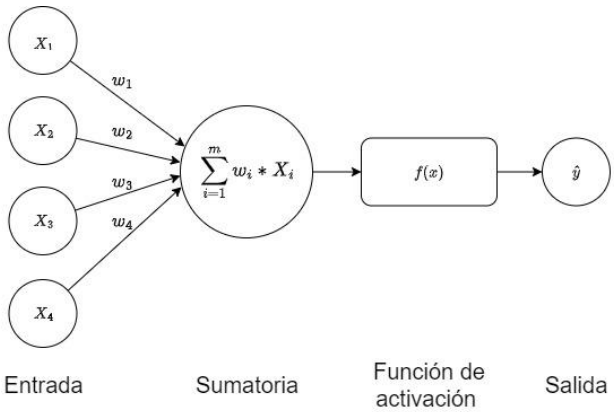

Figura 3. Representación gráfica de un perceptrón, basada en la visualización de la publicación *Perceptron Model: The Foundation of Neural Networks* del sitio web *Medium* (Kılıç, 2023).

Convolutional Neural Networks

Las redes neuronales convulsiones, CNN por sus siglas en inglés, o también llamadas ***convnets***, es una arquitectura de red para *Deep Learning* que se utilizan con mayor frecuencia

para tareas de clasificación y visión computacional. Estas redes proporcionan un enfoque escalable para las tareas de clasificación de imágenes y reconocimiento de objetos al aprovechar principios de álgebra lineal para identificar patrones. Se utilizan estos principios con el propósito de comprender las características presentes dentro de los datos, además que estas operaciones se llevan a cabo dentro de las capas dentro de la red.

Estas redes se componen de diferentes capas interconectadas, aunque su estructura general sigue las mismas bases:

- **Capa convolución:** el papel de esta capa es analizar las imágenes proporcionadas en la entrada y detectar la presencia de un conjunto de *features* mediante la utilización de *kernels* o filtros para escanear la imagen y detectar características como bordes, texturas y patrones. Estos filtros se desplazan a través de toda la imagen para extraer las características en diferentes ubicaciones. Al aplicar los *kernels*, cada uno produce un *feature map*, la cual es una representación espacial bidimensional de las características detectadas por el filtro en la imagen de entrada. Cada capa de convolución en una CNN puede tener múltiples *kernels*, cada uno de estos producir su propio *feature map*. Los mapas de características de las capas iniciales de una CNN tienden a capturar características simples y locales, como bordes y colores, mientras que los mapas de características en capas más profundas pueden capturar características más complejas y abstractas, como formas y patrones de alto nivel.

  Matemáticamente hablando, la convolución se utiliza en diversos campos, como el procesamiento de señales, visión por computadora, procesamiento de imágenes, entre otros. Esta es una operación matemática que consiste en combinar dos funciones para obtener una tercera función que representa la forma en que se superponen o se relacionan las dos funciones originales; y se puede expresar mediante una integral o una suma, dependiendo de si las funciones son continuas o discretas, respectivamente. (*Introducción a La Convolución*, 2021), (*La Convolución Discreta*, 2019)

  - Para funciones continuas $f(t)$ y $g(t)$, la convolución $f * g$ es:

$$(f * g)(t) = \int_{-\infty}^{\infty} f(\eta) g(t - \eta)\, d\eta$$

  - Para funciones discretas $f[n]$ y $g[n]$, la convolución $f * g$ es:

$$f * g[n] = \sum_{k=-\infty}^{\infty} f[k] g[n - k]$$

- **Capa de Pooling (POOL):** esta capa generalmente se aplica entre dos capas de convolución la operación de *pooling*, la cual es una operación utilizada para reducir la resolución y, a la vez, preservar las características más esenciales de los *feature map*

formados en la salida de la capa de convolución. Entre las operaciones más utilizadas, encontramos el *Max-pooling*, que selecciona el valor máximo en una región, del mismo modo existe el *Average-pooling*, para seleccionar el valor medio de la ventana, *Min-pooling* para el mínimo, por mencionar algunos.

- **Capa de activación:** Después de la capa de convolución, es común aplicar una función de activación no lineal, esto se conoce como activación, ya que solo las características activadas prosiguen a la siguiente capa. Esto introduce no linealidad en la red y permite un entrenamiento más rápido y eficaz. En el caso de las CNN, se suele utilizar ReLU (Rectified Linear Unit), la cual, como se mencionó anteriormente.
- **Capa Fully-Connected:** Estas capas son similares a las capas ocultas de las redes neuronales tradicionales, estas conectan por completo todas las neuronas (de ahí el término *fully-connected*). Además, estas capas se colocan al final de la arquitectura de la CNN y se utilizan para realizar la clasificación final o la regresión de las características extraídas en las capas anteriores.

Del mismo modo que CNN es una arquitectura de red para *Deep Learning*, para CNN se han desarrollado una serie de arquitecturas bien conocidas que han tenido un impacto significativo en campos como la visión por computadora. A continuación, se describen algunas de las arquitecturas de CNN más importantes:

- **LeNet:** Arquitectura de red neuronal convolucional (CNN) que fue propuesta por Yann LeCun, Léon Bottou, Yoshua Bengio y Patrick Haffner en 1998, siendo una de las primeras redes neuronales convolucionales diseñadas específicamente para el reconocimiento de patrones en imágenes, y se aplicó inicialmente al reconocimiento de números manuscritos en cheques bancarios, logrando establecer las bases para las CNN modernas.

  LeNet utiliza capas convolucionales para extraer características importantes de las imágenes de entrada, combinando convoluciones con funciones de activación no lineales (como la función sigmoide o la tangente hiperbólica) para introducir no linealidades en la red. En las capas de pooling, después de algunas capas convolucionales, se aplican capas de submuestreo para reducir la resolución espacial de las representaciones, ayudando así a reducir la cantidad de parámetros y a mejorar la eficiencia computacional. Y después de varias capas convolucionales y de pooling, la red utiliza capas *fully-connected* para realizar la clasificación final y la última capa a menudo utiliza una función de activación softmax para asignar probabilidades a las diferentes clases (LeCun et al., 1998).
- **AlexNet:** Esta arquitectura ganó la competencia ImageNet en el año 2012. Fue una de las primeras arquitecturas en utilizar profundidad y capas convolucionales en cascada,

popularizando la utilización de CNN profundas en tareas de visión por computadora. La arquitectura se compone por 8 capas, de las cuales 5 capas son convolucionales y 3 capas son *fully-connected*, además de ser una de las primeras en utilizar GPU para acelerar el entrenamiento (Wei, 2019).

Los *kernels* tienen dimensiones de 11×11 en la primera capa, 5×5 en la segunda, y 3×3 en el resto. La primera, cuarta y quinta capas convolucionales vienen seguidas cada una de ellas por una capa de agrupación (*max-pooling*) de 3×3, en la que existe un desplazamiento (*stride*) de dos, lo que provoca que haya un solapamiento de las celdas en la misma. Las capas convolucionales y de *pooling* vienen seguidas de tres capas densas. Las dos primeras tienen 4.096 neuronas cada una. La última es la de salida. Está compuesta con 1.000 neuronas dotadas de una función de activación *softmax*. Son ellas las encargadas de realizar la clasificación de la imagen (Krizhevsky et al., 2012).

- **Residual Networks (ResNet):** Arquitectura de red neuronal profunda que introdujo un enfoque innovador para abordar el problema del entrenamiento de redes profundas, el cual es el problema del desvanecimiento de gradiente en redes extremadamente profundas. Fue propuesta en el artículo *Deep Residual Learning for Image Recognition* publicado en el 2015 (He et al., 2015). Esta arquitectura introdujo bloques residuales que contienen conexiones de salto, también conocidas como conexiones de salto o mapeos de identidad, lo que permite que el gradiente fluya más fácilmente a través de la red; aliviando el problema del desvanecimiento de gradiente en redes neuronales muy profundas y permite que el modelo aprenda de manera más efectiva.

  Las arquitecturas ResNet vienen en varias profundidades, como ResNet-18, ResNet-34, ResNet-50, ResNet-101 y ResNet-152. Aquí, el número en el nombre de la arquitectura indica el número total de capas, incluyendo capas convolucionales y totalmente conectadas.
- **Densely Connected Convolutional Networks (DenseNet):** Arquitectura de red neuronal profunda que se introdujo para abordar la problemática de la propagación eficiente de gradientes en redes profundas. La caracteriza su densa conectividad entre capas, lo que significa que cada capa está conectada a todas las capas subsiguientes de la red.

  En esta arquitectura cada capa está conectada directamente a todas las capas posteriores de la red, en otras palabras, cada capa recibe como entrada todas las características generadas por las capas anteriores y, a su vez, envía sus propias características a todas las capas subsiguientes; permitiendo que la información fluya directamente a través de la red y facilita la propagación de gradientes durante el entrenamiento. Además de que está compuesta por bloques densos, que son conjuntos

de capas apiladas, y que cada bloque consiste en una secuencia de capas convolucionales, seguidas de funciones de activación ReLU, mencionada anteriormente.

Esta arquitectura ha demostrado un rendimiento sobresaliente en una variedad de tareas de visión computacional; como para la clasificación de imágenes, segmentación semántica y detección de objetos (Huang et al., 2018).

- **Squeeze Excite Net (SENets):** Arquitectura de red neuronal diseñada para mejorar el rendimiento de las CNN. En una capa convolucional típica de una CNN, los pesos de cada canal son uniformes al calcular la salida. Esto significa que la misma matriz 2D de pesos convoluciona con cada canal, ponderando cada uno de sus canales por igual al crear los mapas de características de salida. Las SENets pretenden cambiar esto introduciendo un enfoque adaptativo en el que la importancia de cada canal se evalúa individualmente en función de su contexto. En términos más simples, al incorporar este enfoque se tiene en cuenta la relevancia de cada canal al calcular la salida (Hu et al., 2019).

  Para lograr esto se incorpora un módulo de excitación que se centra en aprender la relación de ponderaciones entre los diferentes canales de una capa convolucional. Permitiendo asi a la red aprender la importancia relativa de cada canal para una tarea específica, lo que puede mejorar la capacidad de discriminación y la representación de características importantes (Pröve, 2017).

Visión computacional

¿Qué es la visión computacional?

La visión computacional es un campo interdisciplinario del mundo de la informática y la inteligencia artificial, esta se encarga de diseñar algoritmos computacionales capaces de interpretar y entender la información detectada por medio de sensores especializados.

Existen muchos aspectos claves dentro de este campo, entre los que se pueden incluir el procesamiento de imágenes o secuencias de vídeos, a los que se les puede extraer características de las imágenes, lo que implica identificar patrones, bordes, texturas y otros elementos visuales que son relevantes para la tarea en cuestión. Por otro lado, se puede realizar un reconocimiento y/o detección de objetos presentes, ya sea para rostros humanos o señales de tráfico, por mencionar algunos; a los que se les puede realizar un rastreo (*tracking*), lo que es esencial en aplicaciones como la vigilancia y la robótica.

Sensor de visión tradicional: cámara RGB

Al referirnos a sensores de visión, tradicionalmente pensamos en las cámaras RGB (Red, Green, Blue), o cámaras a color, basadas en frames; esto los solemos asociar debido a su

amplia disponibilidad y alcance. Estas cámaras capturan imágenes en un intervalo de tiempo, lo cual indica que el número de frames, o cuadros, que las cámaras pueden detectar en un segundo, y lo realizan en el espectro visible de la luz y son ampliamente utilizadas en aplicaciones de visión computacional, fotografía, videovigilancia, realidad aumentada, robótica, entre otras áreas.

Para la detección y representación de los colores estas cámaras tienen sensores para cada uno de los canales de color primario (rojo, verde y azul, RGB por sus siglas en ingles) y pueden combinar esta información para crear imágenes en color de alta calidad.

Evolución de los métodos de procesamiento de imágenes en visión computacional

En las primeras etapas de la visión computacional, se utilizaron técnicas de procesamiento de imágenes tradicionales, como la filtración, la convolución y la extracción de características simples para tareas como detección de bordes y reconocimiento de patrones. Estas técnicas eran costosas, computacionalmente hablando, y dependían en gran medida de la ingeniería de características manuales. También existían sistemas basados en reglas heurísticas y conocimiento experto para tareas específicas, los cuales solo servían para tareas específicas y requerían la codificación manual de reglas.

Con la aparición del aprendizaje automático, empezaron a aparecer técnicas para mejorar el reconocimiento de patrones y la clasificación de objetos. Y luego con el *Deep Learning*, y especialmente las redes neuronales convolucionales (CNN), demostraron ser altamente eficaces para la extracción automática de características a partir de datos de imágenes, eliminando la necesidad de características manuales. Esto llevó a mejoras significativas en la precisión en tareas como reconocimiento de objetos, detección de rostros y segmentación de imágenes.

Revisión análisis facial en cámaras basadas en frames (RGB)

Un sistema de análisis facial es una aplicación que emplea un algoritmo o modelo para identificar automáticamente a una persona en una imagen o vídeo. Son de gran importancia ya que agiliza los trámites y garantiza mayor seguridad. Las principales investigaciones se desarrollan en aplicaciones de video vigilancia o control de acceso, y posteriormente estas aplicaciones se han extendido en procesos de autenticación en teléfonos móviles y otros dispositivos electrónicos.

El primer paso en el análisis facial es la detección de rostros dentro de las imágenes de la entrada, lo cual se logra mediante algoritmos de visión computacional que realizan una búsqueda de patrones y características de interés.

Revisión y estado del arte de face liveness en cámaras basadas en frames (RGB)

El *face liveness detection* (detección de vida facial, en español) es una técnica de autenticación biométrica y sistemas de seguridad que se utiliza para determinar la presencia física de un usuario frente al sensor mediante la identificación de su rostro y análisis de los movimiento del mismo; o, en caso contrario determinar si pertenece a un rostro adulterado, el cual puede estar presente dentro de una imagen, una grabación de video pregrabada o, incluso, ser una persona con una máscara. Todos estos son intentos de suplantación o *spoofing* en términos de seguridad.

Las técnicas de *face liveness detection* pueden variar, pero generalmente involucran el análisis de diferentes características y señales en la imagen o el vídeo capturado para determinar si la persona es real o si se trata de una representación falsa. Algunas de las técnicas utilizadas son:

- **Detección de movimiento:** Verifica si hay movimientos faciales naturales, como parpadeos o movimientos de cabeza, que son difíciles de replicar en una imagen o vídeo pregrabado.
- **Análisis de textura:** Examina la textura de la piel y otras características faciales para buscar señales de autenticidad.
- **Detección de patrones:** Buscar patrones específicos en la imagen, como reflejos en la superficie de los ojos, para determinar si la persona está viva y tiene un parpadeo natural.
- **Análisis de profundidad:** Utilización de tecnologías 3D o sensores de profundidad para evaluar la distancia y la estructura tridimensional de la cara y verificar si es real o una representación plana.
- **Desafíos específicos:** Presentar desafíos al usuario, como requerir que realice acciones específicas, como seguir instrucciones en pantalla o mover la cabeza en una dirección particular, para demostrar que está vivo.

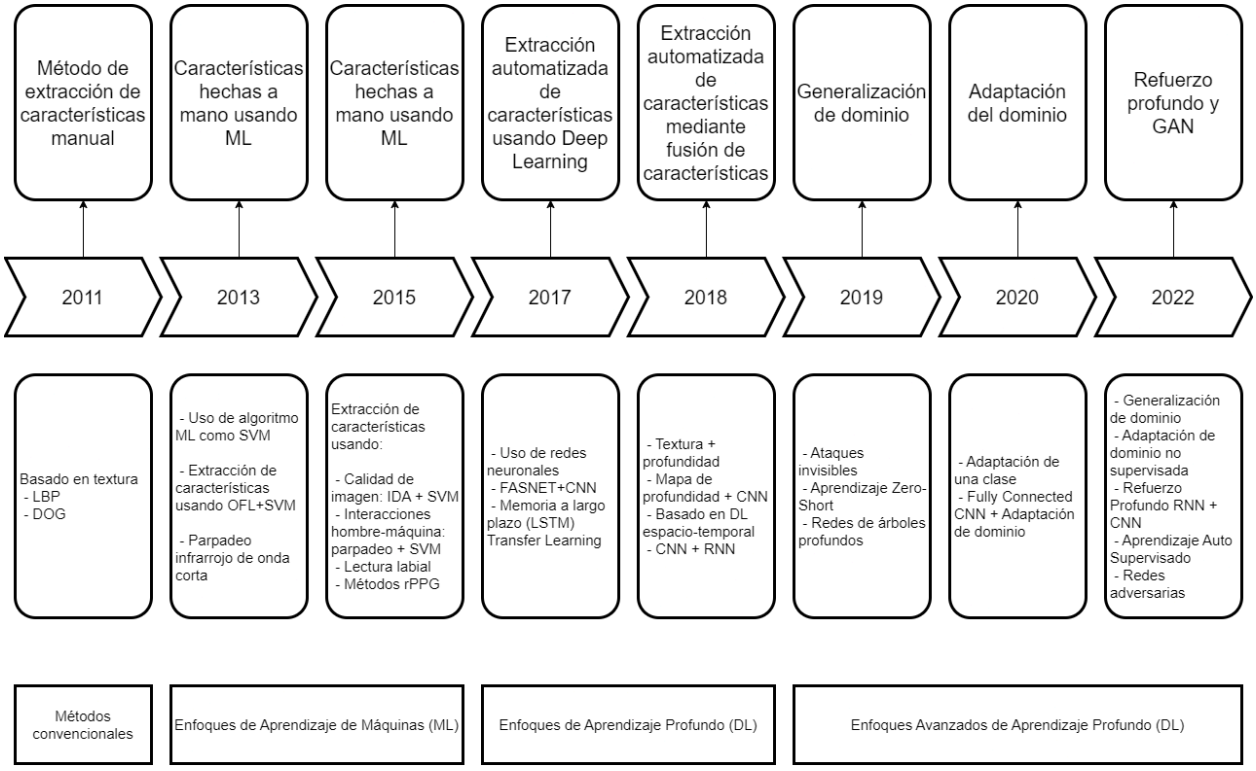

Figura 4. Diagrama de la progresión de cómo se ha abordado el tema de *liveness detection* a lo largo del tiempo. Figura basada en línea del tiempo presente en el artículo *Face Liveness Detection Using Artificial Intelligence Techniques: A Systematic Literature Review and Future Directions* (Khairnar et al., 2023)

Tipos de ataques de face liveness

Existen varios tipos de ataques que intentan engañar o burlar los sistemas de reconocimiento facial. Estos ataques están diseñados para imitar con éxito a una persona real frente al sensor y, por lo tanto, son una preocupación importante en la seguridad y la autenticación biométrica. La forma en la que se llevan a cabo estos ataques pueden variar mucho; desde solamente utilizar papel normal con rostros impresos, mostrarlos desde la pantalla de un teléfono inteligente mostrando imágenes de una persona real, hasta la utilización de máscaras de plástico. Algunos tipos de ataques de *face liveness* son:

- **Photo attack:** Consiste en ataques que involucran el uso de imágenes estáticas como fotografía frente a la cámara o sensor, en un intento de engañar al sistema.
- **Replay attack:** Consiste en la utilización de vídeos pregrabados, a diferencia del *photo attack*, este incorpora características como movimientos oculares, movimientos de labios y cambios de expresiones faciales, imitando la textura y el contorno del rostro y su dinámica. Los ataques de repetición suelen ser más difíciles de detectar que los ataques con fotografías (Khairnar et al., 2023).

- **Cut-photo:** Similar al *photo attack*, con la diferencia que en este se cortan secciones de la fotografía como los ojos y se sitúa una persona detrás y así se incluyen movimientos oculares en la captura de la cámara.
- **Silicone Mask attack:** Consiste en ataques que involucran el uso de máscaras se silicona, las cuales una persona utiliza mientras está frente a la cámara o sensor, en un intento de engañar al sistema.

Limitaciones y desafíos a la fecha en liveness en cámaras basadas en frames

La utilización de cámaras basadas en frames en *liveness* son varias, siendo la principal de estas la alta dependencia de las condiciones de iluminación; esto provoca que al momento de realizar la toma de muestras, ocasionando saturación de píxeles en situaciones de alta luminosidad, en estas situaciones, la iluminación es tan alta que alcanzan su valor máximo y no pueden representar detalles, dando como resultado la pérdida de información en áreas sobre expuestas a esta. Sumado a esto, se le suma su alta sensibilidad ante cambios en las condiciones de iluminación, como por ejemplo deslumbramientos, sombras o reflejos pueden afectar la calidad de las imágenes capturadas.

Por otro lado, las cámaras basadas en frames pueden tener dificultades para distinguir entre un rostro real y una representación artificial bidimensional, como una fotografía. Por lo que, los algoritmos de detección de *liveness* deben ser robustos para identificar señales de autenticidad, como movimientos faciales sutiles o parpadeos, que no pueden ser replicados por imágenes estáticas, pero el problema vendría al momento en que se esté representando un rostro mediante un video.

Otra limitación que se puede detectar al utilizar estas cámaras, es la baja resolución temporal de los frames, la cual se es limitada a los frames por segundo que puede capturar la cámara; por ejemplo, en una cámara que captura 30 frames por segundo, el tiempo que pasa entre un frame y otro es de alrededor de 33 ms, tiempo que cuando se trabaja con liveness puede ser demasiado y pueden haber cambios en escena que la cámara no logra detectar. Por otro lado los eventos, poseen una resolución temporal en el orden de los microsegundos (µs).

¿Qué es una cámara de eventos?

Las cámaras basadas en eventos son sensores especializados de captura de imágenes que registran la información visual del entorno de manera diferente a como lo hacen las cámaras tradicionales. Ya que estos sensores detectan y registran cambios significativos en la escena en tiempo real, determinando esto gracias a los cambios de iluminación.

Estos sensores se inspiran en el funcionamiento del sistema visual humano y en la necesidad de capturar y procesar información visual de manera eficiente en situaciones donde los cambios son rápidos y significativos; imitando la forma en que el cerebro humano y los

sistemas biológicos procesan y responden a los estímulos visuales. Y por ello, también se les denomina como sensores de visión neuromórficos.

Estas cámaras en lugar de registrar cada cuadro (frame) completo, sólo capturan los momentos en que hay un cambio en la intensidad de la luz en una región específica de la imagen que, sumado a su alto rango dinámico, se logra obtener un gran detalle de los cambios en la escena. Con esto solamente se activan los píxeles en los que realmente ocurre algo relevante; con esto se logra tener un menor consumo energético, con una latencia extremadamente baja y la capacidad de capturar objetos en movimiento rápido, teniendo un flujo de eventos con una resolución de microsegundos.

Estos sensores representan una tecnología prometedora debido a su capacidad única para capturar eventos visuales relevantes en tiempo real y con una baja latencia.

Principio de funcionamiento

El principio de funcionamiento de las cámaras de eventos se basa en la detección y registro de cambios significativos en la intensidad de la iluminación del entorno que está observando y lo realiza de forma asincrónica, lo que quiere decir que, registran la información en función de los cambios detectados y no en intervalos de tiempo regulares predefinidos.

Representaciones de los eventos

Los eventos detectados por las cámaras de eventos se representan generalmente en forma de una secuencia continua de datos que describe cada evento individual. Cada evento detectado contiene información sobre la ubicación en la imagen en la que ocurrió el cambio de intensidad de iluminación, así como el momento exacto en que este cambio se produjo.

A cada uno de los eventos que son detectados por el sensor, se le asocia un par de coordenadas espaciales (x, y) que indican la posición precisa en la imagen donde ocurrió el cambio de luz, representando la ubicación del evento en el plano bidimensional de la imagen. Además de esto, poseen una marca de tiempo correspondiente a cuándo ocurrió el cambio, la cual se puede expresar en microsegundos o incluso nanosegundos, permitiendo una representación extremadamente precisa del momento en que se produjo el evento.

Sumado a esto, en algunas representaciones de eventos se pueden incluir información sobre la polaridad del evento, la que indica si el cambio en la intensidad de la luz fue un aumento (polaridad positiva) o una disminución (polaridad negativa). Y en algunos casos, también se puede asociar una medida de validez o confianza con cada evento para indicar cuán seguro está el sistema de que el cambio detectado es un evento real y no un ruido o una falsa alarma.

Métodos/arquitecturas y aplicaciones

Las cámaras de eventos, debido a su capacidad única para capturar cambios visuales en tiempo real con baja latencia, han generado interés en una variedad de implementaciones en aplicaciones; y han dado lugar al desarrollo de varios métodos y arquitecturas.

- **Algoritmos de Detección de Eventos:** Consiste en algoritmos que procesan la secuencia de eventos generados por la cámara, ya sea para detectar objetos en movimiento, cambios en la escena y otros eventos relevantes.
- **Fusión de Datos Multisensorial:** Estas se utilizan en sistemas de percepción multisensorial para fusionar datos de diferentes sensores, como datos recopilados de cámaras RGB, LiDAR y/o sensores inerciales, permitiendo así una percepción completa del entorno.
- **Redes Neuronales Basadas en Eventos:** Se basa en redes neuronales específicas para el procesamiento de datos de eventos, como las redes neuronales de impulsos (SNN) y las redes convolucionales basadas en eventos; en donde, estas redes son eficientes en términos energéticos y adecuadas para aplicaciones en tiempo real.

Las aplicaciones más comunes para estos es algoritmos se pueden encontrar aplicaciones de visión por computadora en tiempo real, como el seguimiento de objetos, la detección de gestos y la interacción humano-computador; la cual además de su aplicación en sistemas de vigilancia y seguridad para la detección de movimientos sospechosos, puede ser implementada en robots o vehículos autónomos para la percepción del entorno, detección de obstáculos en tiempo real y seguimiento de objetos en movimiento, el cual puede ser en altas velocidades. También estas cámaras se utilizan para mejorar la realidad aumentada al permitir una interacción más precisa y en tiempo real con el entorno físico.

Revisión de métodos en análisis facial

El análisis facial en eventos se refiere al uso de técnicas de procesamiento de imágenes y visión por computadora para detectar y analizar tanto las expresiones faciales, como las reacciones emocionales de las personas. A continuación, se mencionan algunos métodos de análisis facial.

- **Reconocimiento de Expresiones Faciales:** Se utilizan algoritmos de aprendizaje automático, como redes neuronales convolucionales (CNN), para detectar y reconocer expresiones faciales en imágenes o vídeos. Y pueden identificar emociones en los rostros detectados, ya sea identificando felicidad, tristeza, ira, sorpresa, etc.
- **Análisis de Movimientos Oculares:** Se realiza el seguimiento de los movimientos de los ojos y los parpadeos, lo que puede proporcionar información sobre la atención y el interés de la persona a la que se le está observando por medio de la cámara.

- **Detección de Microexpresiones:** Se detectan expresiones faciales fugaces y sutiles, las que pueden revelar emociones ocultas; estas se utilizan para detectar estas expresiones rápidas y a menudo involuntarias.
- **Análisis de Movimientos Labiales:** Se enfoca en el seguimiento y análisis de los movimientos de los labios para comprender el habla y la comunicación no verbal.
- **Modelos Generativos de Rostros:** Se implementan modelos generativos de rostros, como *Generative Adversarial Networks* (GANs), las que se utilizan para generar expresiones faciales sintéticas que pueden utilizarse para entrenar y evaluar algoritmos de análisis facial.

Metodos destacados de liveness

En la siguiente tabla se muestran algunos modelos creados para *liveness detection*, con sus respectivas precisiones.

Artículo	Año de publicación	Modelo	Base de datos	Precisión (accuracy)
Face-Fake-Net: The Deep Learning Method for Image Face Anti-Spoofing Detection	2021	CNN (ResNet50)	CASIA-SURF	99.7%
			CelebA-spoof	99.4%
Adversarial examples for replay attacks against CNN-based face recognition with anti-spoofing capability	2020	CNN	Replay-Mobile	94.56%
Deep Learning for Face Anti-Spoofing: An End-to-End Approach	2017	CNN	CASIA-FASD	92.52%

Tabla 1. Articulos destacados de *liveness* y *anti-spoofing*

En la tabla anterior se muestran tres artículos referentes a liveness que presentan un buen desempeño en cuanto a la métrica accuracy, además se puede destacar que en todos los casos mencionados se optó por un modelo del estilo de una red convolucional. Cabe mencionar que, si bien no todos los artículos mencionados en la tabla anterior son de *liveness*, ya que hay algunos que son de *anti-spoofing*, estos dos conceptos están relacionados, debido a que se utilizan en el contexto de la biometría y la seguridad, especialmente en sistemas de reconocimiento facial. Sin embargo, se centran en aspectos ligeramente diferentes, *liveness* se centra en verificar si la muestra biométrica proviene de una fuente en tiempo real y no es una representación artificial, mientras que el *anti-spoofing* se enfoca en prevenir o detectar intentos

de falsificación, ya sea mediante fotografías, grabaciones u otros métodos. Por lo que, de igual forma se están considerando como trabajos previos.

Bases de datos para análisis facial con eventos

Existen varias bases de datos públicas y privadas que se utilizan para el análisis facial y el entrenamiento de algoritmos, de entre las cuales se investigaron las siguientes para su consideración.

Faces

Esta es una base de datos utilizada en una investigación publicada en el año 2020, titulada *Event-Based Face Detection and Tracking Using the Dynamics of Eye Blinks* (Lenz et al., 2020), en la que se presenta el primer método puramente basado en eventos, con el cual, se busca detectar rostros utilizando las propiedades de alta resolución temporal de una cámara basada en eventos para detectar la presencia de un rostro en una escena mediante parpadeos; utilizándolos como una firma temporal, dinámica, natural, única y estable de rostros humanos en toda la población que puede ser capturada completamente por sensores basados en eventos. Una vez localizado un rostro, se aplica un marco probabilístico para rastrear su ubicación espacial para cada evento entrante mientras se utilizan parpadeos para corregir errores de deriva y seguimiento.

Neuromorphic Event-based Facial Expression Recognition (NEFER)

La base de datos *Neuromorphic Event-based Facial Expression Recognition* (NEFER por sus siglas en inglés) es un conjunto de muestras para reconocimiento de expresiones faciales basado en eventos neuromórficos, fue presentada en el año 2023, en el artículo homónimo *Neuromorphic Event-based Facial Expression Recognition* (Berlincioni et al., 2023). Esta se compone de frames de videos en RGB y de eventos emparejados que representan rostros humanos etiquetados con las emociones respectivas y también anotados con cuadros delimitadores de rostros y puntos de referencia faciales. Esta base de datos se compone de un total de 609 capturas de datos, que corresponden a 29 usuarios y 7 emociones expresadas. A continuación, se muestra una imagen de referencia de la base de datos, la que es un fragmento de la figura 5 presente en el artículo original; en donde se muestran cuatro muestras del conjunto de datos: la primera fila muestra felicidad, segunda fila miedo, tercera fila disgusto y cuarta fila sorpresa.

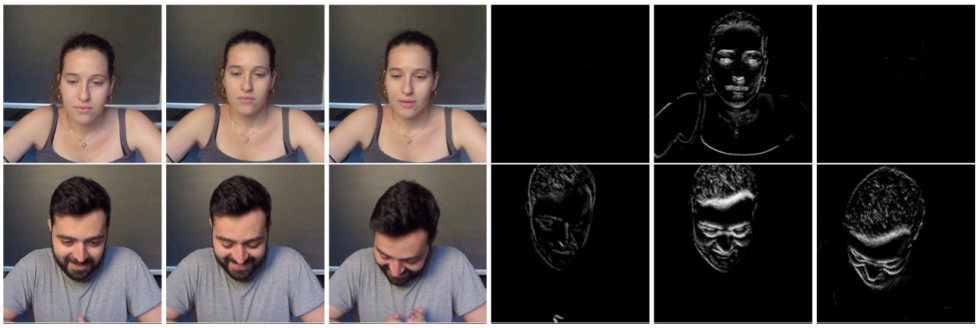

Figura 5. Fragmento de la imagen de referencia de la base de datos, obtenida del artículo original de la base de datos NEFER (Berlincioni et al., 2023); en donde se muestran dos muestras del conjunto de datos, de dos personas (o usuarios) diferentes, mostrando *frames* en RGB de los datos y los eventos acumulados de los mismos.

Faces in Event Streams (FES)

La base de datos FES (*Faces in Event Streams*, por sus siglas en inglés) fue presentada en el año 2023, en el artículo *Faces in Event Streams (FES): An Annotated Face Dataset for Event Cameras* (Bissarinova et al., 2023). Esta publicación aborda el problema publicando en el primer conjunto de datos FES grande y variado con una duración de 693 minutos y con la participación de 73 personas para su realización. Esta base de datos está destinada para la detección de rostros y puntos de referencia faciales para una salida de cámara directa basada en eventos, presentando 12 modelos entrenados con este conjunto de datos para predecir el cuadro delimitador y las coordenadas de los puntos de referencia faciales. En la siguiente figura, se muestran tres registros del mismo rostro.

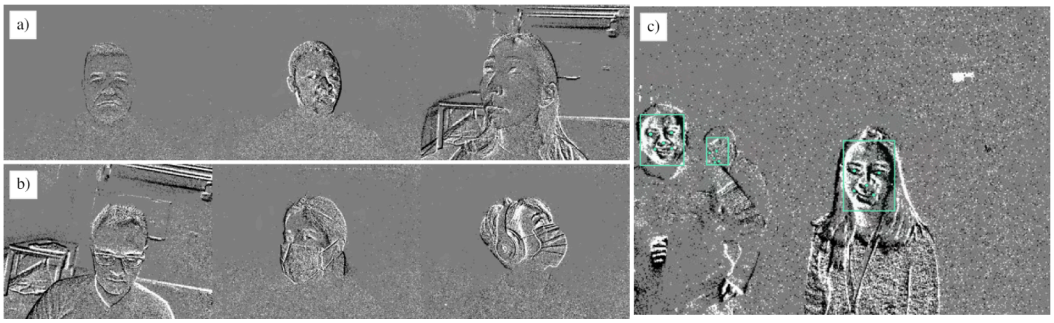

Figura 6. Capturas de datos del conjunto que muestran la diversidad de los sujetos, la etiqueta en la parte superior derecha de cada una corresponde a: a) Participantes de diferentes etnias y géneros, b) participantes con diversos accesorios (lentes, mascarilla, audífonos), y c) participantes en el entorno salvaje con los cuadros delimitadores de caras anotados y cinco puntos de referencia. La figura se obtuvo del artículo original de la base de datos (Bissarinova et al., 2023).

Marco metodológico

El diseño de esta investigación se centra en la elaboración de una base de datos de *replay attack* para *face liveness detection* de datos de eventos, a la cual se le aplicaran modelos de inteligencia artificial para explorar de forma preliminar las posibilidades de la base de datos, evaluando su desempeño. Es así que, la metodología de trabajo se puede dividir en las siguientes etapas claves:

- **Adquisición de Datos:** Se elaborará una base de datos, utilizando los datos de las base de datos NEFER (Berlincioni et al., 2023) como base, y a partir de estos datos se obtendrán casos de ataques, más específicamente: *replay attack* y *photo attack*.
- **Preprocesamiento de Datos:** Los eventos detectados por la cámara de evento se procesan para eliminar ruido y se registran con precisión en una secuencia temporal.
- **Entrenamiento del Modelo:** Con la base de datos elaborada se entrenarán modelos existentes de *liveness detection* basado en aprendizaje profundo utilizando los eventos visuales preprocesados como entrada.
- **Validación y Evaluación:** Se evaluará el modelo en términos de su capacidad para *face liveness detection* en diferentes situaciones. Esto se realizará utilizando métricas de desempeño, como precisión, sensibilidad, especificidad, entre otras; para evaluar la eficacia del sistema.

Elaboración base de datos

La fuente de los datos que se utilizará en este estudio será la base de datos NEFER (Berlincioni et al., 2023); utilizando los datos de eventos directamente como casos de no ataque. Por otra parte, también se utilizaran los datos de frames en RGB de esta misma base de datos, los que se juntarán para formar así secuencias de vídeos de 15 imágenes por segundo (fps) y de un aproximado de 10 segundos de duración. Estos vídeos serán los que más tarde se utilizarán para generar los casos de ataque.

Para generar los casos de ataque dentro de la base de datos, se abordará el problema de *replay attack*, para los cuales se plantean propuestas experimentales para cada uno. Estas propuestas mencionan los objetivos asociados a cada escenario experimental, consideraciones asociadas y procedimiento de ejecución. A continuación se se pasa a describir la propuesta experimental:

Replay attack

El proceso de recopilación de datos de *replay attack*, se basa en la configuración utilizada en el artículo *Adversarial examples for replay attacks against CNN-based face recognition with anti-spoofing capability* (Zhang et al., 2020), esta configuración consiste en

ubicar el sensor frente a un monitor en el que se reproducen los videos, las capturas obtenidas corresponden a casos de reproducción (*replay*). En el artículo antes mencionado, como sensor se utiliza una cámara basada en frames, en cambio en este se reemplaza por una cámara basada en eventos, más específicamente la cámara DAVIS 346. A continuación se muestra una fotografía de la cámara a utilizar:

Figura 7. Fotografía de cámara DAVIS 346, obtenida del artículo *CED: Color Event Camera Dataset* (Scheerlinck et al., 2019).

Esta cámara posee la capacidad de capturar tanto secuencias de eventos y videos (frames) en escala de grises, ambos con una resolución de 346x260 píxeles. A continuación, se muestra un esquema de referencia de la configuración del entorno de captura de los datos de *replay attack*.

Figura 8. Configuración para *replay attack*. Basado en configuracion del artículo: *Adversarial examples for replay attacks against CNN-based face recognition with anti-spoofing capability* (Zhang et al., 2020).

En este escenario se ubicará una computadora, la cual se encarga de correr un código el cual automatiza la captura de datos, este código consiste en (1) iniciar la grabación del sensor, (2) reproducir en pantalla completa el video a ser replicado y (3) esperar a que se termine el video para parar la grabación y guardarla; esto para todos los videos de la base de datos NEFER. Cabe resaltar que, la configuración mostrada en la figura 8, se cubre por una caja sellada, para bloquear la luz y reflejos externos, provenientes tanto superior, como lateral; y el avance correcto de la captura de datos se realiza con un monitor externo.

El objetivo asociados a este escenario experimental se centran en la elaboración de una toma de muestras de datos de *replay attack* para tener una base de datos variada en un ambiente controlado.

Por el lado de las consideraciones asociadas al escenario experimental son principalmente que, no se podrá modificar el brillo del monitor durante la captura de datos, no se consideran variaciones de iluminación tanto natural como artificial, ya que estas son bloqueadas por la caja mencionada anteriormente. La captura de muestras se realizará con el sensor DAVIS 346, el cual realiza captura de eventos y frames en escala de grises. Por último, durante la captura de datos, la cámara será sostenida por un trípode y que esta será de 10 segundos aproximadamente.

De acuerdo al procedimiento de captura de muestras, este se dividió en tres partes, la pre-ejecución, ejecución y post-ejecución. En la pre-ejecución se lleva a cabo el montaje de los diferentes elementos que se utilizan en la ejecución de la captura, como el monitor y el sensor (DAVIS 346) en el trípode; no se considera iluminación externa ya que esta será proporcionada por el mismo monitor en el que se reproduzcan las muestras. Además, se procuró que el monitor se encuentre lo más recto posible respecto a la cámara, en la figura 8 se muestra una fotografía de la configuración de captura de datos.

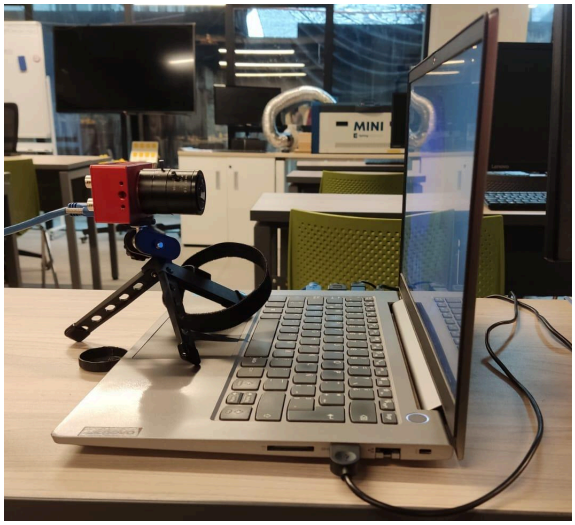

Figura 9. Fotografia de configuración de captura de datos

Durante la ejecución de captura de datos, se ejecutarán los vídeos de la base de datos en el monitor en frente de la cámara y serán registrados por la cámara.

Finalmente en la post-ejecución, se revisarán las muestras tomadas, procurando que se hayan tomado correctamente, se enumeraron las muestras tomadas y se dividirá el dataset en subconjuntos de entrenamiento, validación y prueba.

Pre-procesamiento de datos

Recopilados los datos, estos se pre procesan, removiendo los extremos de las muestras, en los que se encuentran los momentos en los que el video comenzó y terminó de reproducir, esto se logra gracias a que se sabe la duración de cada uno de los videos y los conjuntos de eventos resultantes se guardan en formatos más intuitivo a trabajar, como por ejemplo: un archivo numpy.

Para preparar las imágenes a utilizar durante los entrenamientos se leen los archivos mencionados anteriormente, de los cuales se extraen los eventos a visualizar. El tipo de visualización que se optó a utilizar es *Time Surface*, la cual también es llamada superficie de eventos activos (*surface of active events (sae)*, en inglés), que incluye información tanto espacial como temporal. El cálculo de las matrices de *Time Surface* utilizó como base el código del archivo *time_surface.py* presente en el repositorio *event_representation* (*LarryDong/event_representation: Event-Based Camera Data Representation and Processing. Some Common Representations and Reference Codes.*, n.d.), el cual calcula la decadencia exponencial en el tiempo de los eventos que se presentan en la ventana de tiempo correspondiente, el cálculo se realiza según la polaridad del evento en la celda de la matriz correspondiente a la imagen, además los resultados de esta se encuentran en el conjunto de . La fórmula de la misma se muestra a continuación.

$$\text{Time Surface}(y_i, x_i) = \begin{cases} \frac{e^{-(t_{ref}-t_i)}}{\tau}) & si \quad p_i > 0 \\ -\frac{e^{-(t_{ref}-t_i)}}{\tau}) & si \quad p_i \leq 0 \end{cases}$$

Esta fórmula se utilizó para el cálculo en diferentes ventanas de tiempo (Δt), específicamente para ventanas de 5, 10, 33 y 50 ms (milisegundos). En donde en cada una de estas se obtuvieron 20 imágenes por captura de eventos, del mismo modo esto se realiza con los datos originales de NEFER, los cuales no corresponden a casos de ataques. Por otro lado, para la transformación de matrices a imágenes, estas se pasaron al mapa de color *viridis*.

Posteriormente, se realiza un recorte del rostro de la persona en la imagen, obteniendo así solo los eventos de la zona de interés para el modelo, descartando los eventos que no aportaran información no relevante. Las zonas de recorte se realizaron utilizando como referencia los frames RGB proporcionados en la base de datos original, para ello a cada una de

las muestras originales se tomó el frame 10 al que se le en donde se identifica el rostro en la imagen y de este se obtienen las coordenadas que forman el *bounding box* del rostro del usuario en la imagen, y así efectuar el recorte. La delimitación de la *bounding box* se logra con un etiquetado manual de las imágenes el cual se llevó a cabo en la plataforma *Roboflow* (*Roboflow: Give Your Software the Power to See Objects in Images and Video*, n.d.) subiendo previamente los frames correspondientes y posteriormente delimitando la zona del rostro en cada una de las imágenes.

Con los *bounding box* de los frames realizados, se logra obtener un archivo CSV con anotaciones, en donde se encuentran las coordenadas de los rostros según las *bounding box*. Cabe destacar que, al no estar en la misma resolución los frames de la base de datos original, con los de las capturas, a los valores de las anotaciones se les aplican transformaciones geométricas para escalar las dimensiones de las *bounding box*, ya que en la base de datos original los frames poseían una resolución de 600x600 píxeles, mientras que los eventos poseen una resolución de 1280x720 píxeles, sin mencionar que para los de las capturas con la cámara DAVIS se obtiene una resolución de 346x260 pixeles.

Por otro lado, debido a que al realizar los recortes no todos los rostros se encuentran completamente centrados en los recortes, se optó por aplicar un efecto espejo a las muestras y así, además de obtener un mayor número de muestras, se obtiene una mayor variabilidad dentro de los datos. A continuación se muestra un diagrama del proceso de obtención de la imagen de *Time Surface*, recorte de rostro y aplicación de efecto espejo.

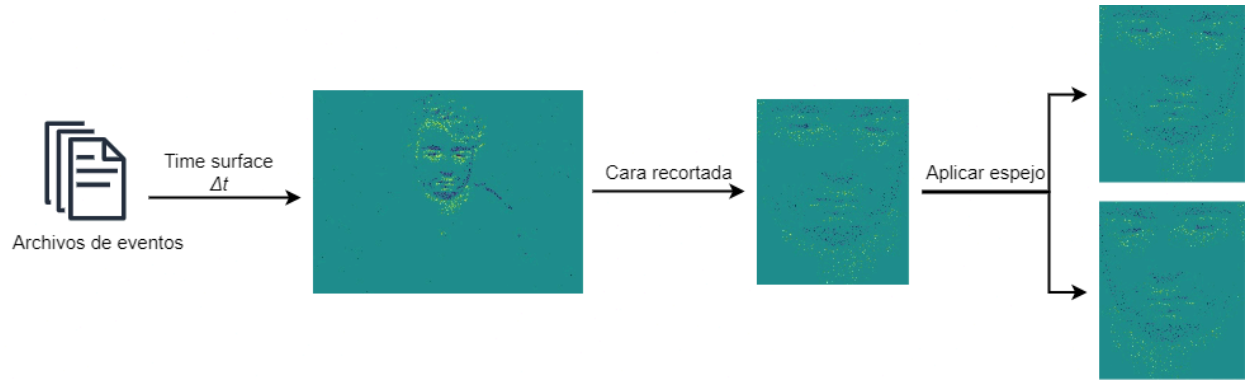

Figura 10. Diagrama del proceso de obtención de la imagen de *Time Surface*, recorte de rostro y aplicación de efecto espejo. Elaboración propia.

Cabe destacar que las imágenes no se llevan a cabo si estas no poseen eventos en la zona del recorte, esto con la intención de no entrenar los modelos con imágenes que no posean información y así no afectar en el rendimiento de los mismos.

Post-procesamiento de Datos

Una vez obtenidas las imágenes a utilizar en entrenamientos se realiza un filtrado de las mismas según el peso de las imágenes de los rostros; este peso de filtrado se toma según el

peso de los datos y la cantidad de imágenes que queden luego de que se aplica. Tratando de que la cantidad de imágenes sea pareja entre las dos etiquetas.

Con las imágenes filtradas, se separan las imágenes entre los conjuntos de entrenamiento y validación, esta división no se realiza al azar, sino que la división se tomó según los usuario presentes en la base de datos original, tomando que los usuarios para entrenar fueron: 0, 1, 2, 4, 5, 6, 8, 9, 10, 11, 12, 13, 14, 15, 16, 21, 22, 23, 24, 25 y 26; y para el conjunto de validación se utilizan los usuarios: 3, 7, 17, 19, 27 y 28. La decisión de que usuarios se utilizan para entrenamiento y validación se basa en los mismos conjuntos de la base de datos de NEFER, esto para obtener una proporción en torno al 80% y 20% respectivamente, y al mismo tiempo, validar los modelos con datos de usuarios no presentes en el entrenamiento.

Aplicación de modelos

Los conjuntos de datos se utilizan en el entrenamiento de modelos, los cuales están basados en arquitecturas de aprendizaje profundo, por ello se utilizo la libreria de python Fastai, esta es una biblioteca de aprendizaje profundo desarrollada sobre PyTorch y diseñada para facilitar la implementación de modelos de aprendizaje profundo. Con esta librería se crea un objeto Learner que representa el componente central para el proceso de entrenamiento de los modelos a entrenar, como modelo se seleccionó una ResNet-34 la cual es parte de la familia de arquitecturas ResNet y el 34 corresponde al número total de capas en el modelo.

Este modelo, consiste en un clasificador binario de liveness detection, el cual identifica si las imágenes de entrada corresponden a imágenes de rostros del dataset original de NEFER o si corresponden a ataques de *replay attack*. Para llevar a cabo esto se utiliza la librería antes mencionada, con la cual se realiza el entrenamiento. Para realizar el entrenamiento, los datos previamente se preparan cargándose en el dataloader respectivo, se establece un tamaño de batch de 64 y re-escalando las imágenes a imágenes cuadradas de 224x224 píxeles. Además, la duración del entrenamiento es de unas 10 épocas (*epoch*).

Evaluación del Modelo

Para evaluar el comportamiento de los modelos entrenados, se utiliza el conjunto de datos de validación, que corresponda al modelo, según las características de los datos de entrenamiento con los que se entrenó. Aquí se pasará a evaluar su desempeño utilizando métricas como la precisión (*accuracy*) y la matriz de confusión de las predicciones con estos datos; además de analizar la pérdida obtenida durante las épocas de entrenamientos para ambos conjuntos de datos (entrenamiento y validación).

Resultados y análisis

Captura experimental

De forma experimental se realizó una captura experimental con la cámara DAVIS 346, en donde se realizó un total de 8 capturas, 4 de estas siendo de *replay attack* y 4 de *photo attack*, siendo 2 con papel normal y 2 con papel fotográfico.

Para explorar de mejor manera los datos capturados, se desarrolló un programa simple en *python*, con el cual se abre una captura y se realiza una visualización *time-surface 2D*, para lograr esto se introducen dos números uno siendo el tiempo de inicio en segundos, es decir, desde que segundo se analizaran en la visualización, y el segundo parámetro es el delta en milisegundos, el cual corresponde al tiempo que se considerarán para realizar la visualización. El código de esta visualización está basado en el GitHub *event_representation* de *LarryDong* (*LarryDong/event_representation: Event-Based Camera Data Representation and Processing. Some Common Representations and Reference Codes.*, n.d.).

A continuación en la figura 11, se muestra una captura de la aplicación con una visualización de una de las capturas preliminares.

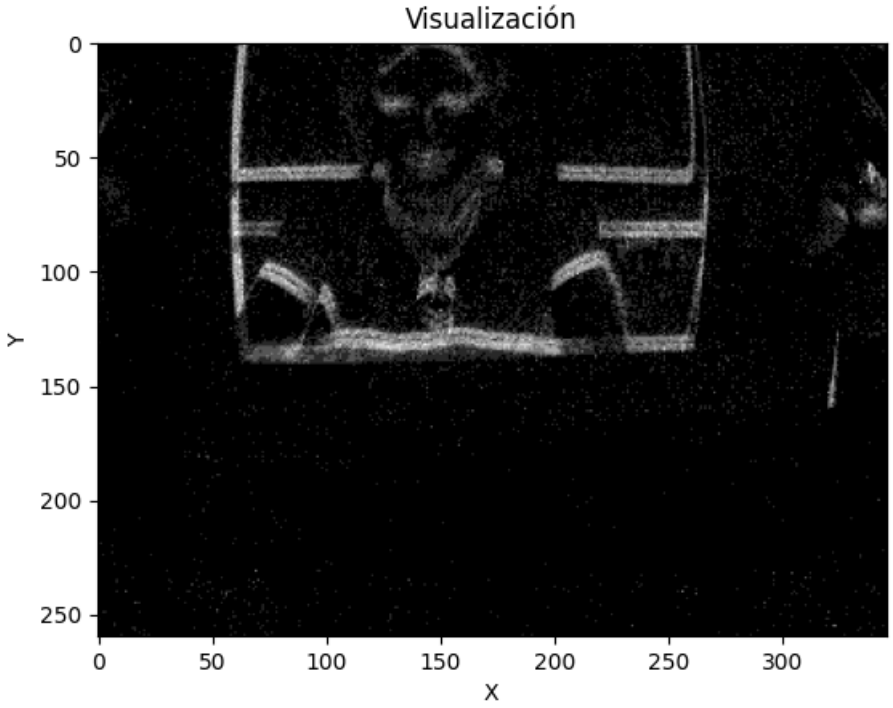

Figura 11. Captura de aplicación con una visualización de *adaptative time-surface* de una captura preliminar de *photo attack* graficando los eventos de 50 ms después del segundo 4.1.

Captura de datos

Se realizó la captura de datos de *replay attack* con la que se obtuvo un total de 589 muestras de datos capturados por la cámara DAVIS 346. Como referencia para próximas comparaciones, en la figura 12 se muestra el frame 15 de la captura GOPR0762, la cual corresponde al usuario 1; este se extrajo de la base de datos NEFER (Berlincioni et al., 2023), se muestra esta *frame*, ya que sirve como referencia para las figuras 13 y 14 las cuales son visualizaciones de *time surface* de los eventos de cuando este *frame* está siendo capturado.

Figura 12. Frame 15 de la captura GOPR0762 del usuario 1. Extraído de la base de datos NEFER (Berlincioni et al., 2023).

Es así que, luego de analizar el primer conjunto de capturas se observó la presencia de ruido ocasionada por reflejos de la iluminación del lugar. En la figura 13 se observa una visualización de *Time Surface* en escala de grises, en la que se observa un conjunto de eventos provocando una gran cantidad de ruido en la imagen.

Figura 13. Captura de eventos de *replay attack*, con mancha de eventos producida por reflejo de iluminación. Elaboración propia.

Como se observa en la figura 13, existe una acumulación de eventos en la captura de datos, revisando el entorno de captura de datos, se determina que este es ocasionado por unos reflejos de luz que llegan de forma perpendicular al monitor. Para solucionar esto se optó por aislar el sistema mediante una caja la cual se encarga de obstruir el ingreso de luz externa, tanto el ingreso frontal y lateral de la luz, como la iluminación superior. Es así que, se lograron obtener las 589 capturas de datos de *replay attack* correctamente y que no cuentan con la presencia de ruido ocasionado por la iluminación, como la que se muestra en la siguiente figura, la cual corresponde al mismo instante que la de la figura 12.

Figura 14. Captura de eventos de *replay attack*, dentro de un ambiente controlado. Elaboración propia.

Además de la captura de eventos por parte de la cámara DAVIS, simultáneamente este sensor también realiza captura de frames en escala de grises de lo que está frente al mismo, la velocidad de captura es de aproximadamente 30 frames por segundo

Figura 15. Muestra de frames capturados por la cámara DAVIS 346. Elaboración propia.

Si bien los frames entregados por el sensor no poseen una definición muy alta (346x260 píxeles) sirven como una primera aproximación de cómo se vería un ataque de reproducción capturado por el sensor.

Clasificador binario

Se entrenó un total de 8 clasificadores binarios diferentes, los cuales varían en su configuración en las ventanas de tiempo con las que se obtuvieron las imágenes de *Time Surface* de las áreas de los rostros de los usuarios y posteriormente se filtraron según el peso de las imágenes obtenidas. Las ventanas de tiempo trabajadas en los diferentes conjuntos de datos son 5, 10, 33 y 50 ms (milisegundos).

Para cada una de estas ventanas se entrenaron dos modelos, realizando cambios en sus conjuntos, siendo los primeros, modelos entrenados con la totalidad de los datos obtenidos y otro al que se le aplicó un filtrado por peso a las imágenes. Este filtrado se realizó para remover de forma automática, gran parte de las imágenes que no aportan demasiada información al modelo que se esté entrenando y este se realiza por etiqueta de las muestras (NEFER y replay), ya que no se puede aplicar el mismo filtro a ambas, debido a que esto puede cargar la cantidad

de los datos en gran medida, ocasionando un desequilibrio dentro del mismo conjunto de datos.

Pasando a ver los resultados, primeramente se observan los resultados de los modelos entrenados con todos los datos y posteriormente a los datos a los que se les aplicó el filtrado.

Modelos entrenados con todos los datos

En la siguiente tabla se muestran los conjuntos de datos según la ventana de tiempo con la que fueron formados, además de indicar si esto pertenece al total del conjunto, o si es solo del entrenamiento o validación, indicando también la etiqueta y con el número de muestras para cada una.

		Ventana de tiempo			
Conjunto	**Etiqueta**	**5 ms**	**10 ms**	**33 ms**	**50 ms**
Total	NEFER	22446	23340	23556	23560
	replay	17826	20118	22852	23258
Entrenamiento	NEFER	17962	18614	18878	18880
	replay	14204	16004	18016	18318
Validacion	NEFER	4484	4626	4678	4680
	replay	3622	4114	4836	4940

Tabla 2. Conjuntos de datos según la ventana de tiempo con la que fueron formados. Elaboración propia.

En si el comportamiento de los modelos ante sus conjuntos de datos de validación respectivos, fue altamente positivo, superando en la mayoría de los casos un *accuracy* del 99% ante estos datos, mostrando así un bajo sobreajuste de los modelos en los datos de entrenamiento. Otro valor obtenido ante la prueba del modelo ante este conjunto fue *loss* (pérdida), y esta se calculó tanto con este conjunto de datos como con el de entrenamiento; obteniendo así todos los casos un *loss* bajo en entrenamiento y aún más bajo para los datos de validación, lo que indica un bajo o nulo sobreajuste (overfitting), indicando que los modelos logran generalizar bien las decisiones ante datos nuevos.

En la siguiente tabla se muestra un resumen de las pérdidas y *accuracy* obtenidos con los modelos en el último epoch del entrenamiento.

Δt	**Train Loss**	**Valid Loss**	**Accuracy**
5 ms	0,017111	0,005941	0,997779
10 ms	0,017180	0,004309	0,998627
33 ms	0,001697	0,000320	0,999895

50 ms	0,003279	0,000033	1,000000

Tabla 3. Resumen de las pérdidas y accuracy obtenidos con los modelos en el último epoch, según el conjunto de datos. Elaboración propia.

Además, en el anexo 2 se encuentran tablas con el detalle de los *loss* y *accuracy* obtenidos a lo largo de los *epoch* del entrenamiento.

Por otro lado, también al evaluar los modelos con las imágenes del conjunto de validación, se obtiene la matriz de confusión de los resultados al clasificar, por lo que se obtienen las siguientes matrices.

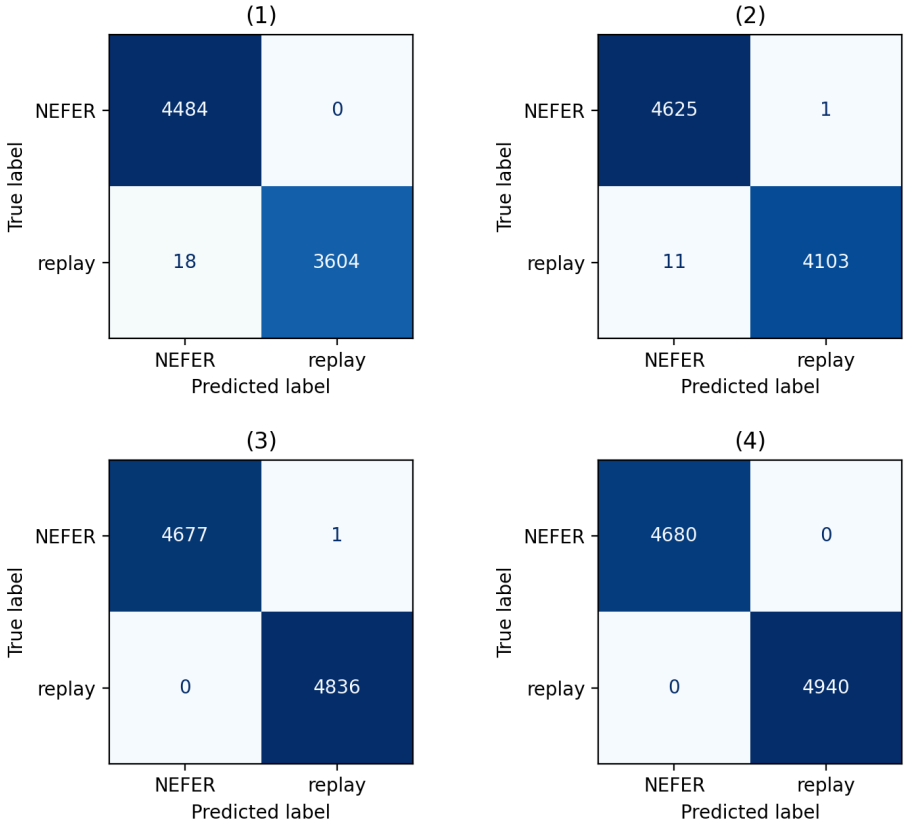

Figura 16. Matrices de confusión de la evaluación de los modelos con los datos del conjunto de validación respectivos, en donde la matriz (1) corresponde al modelo de 5 ms, (2) a 10 ms, (3) a 33 ms y (4) a 50 ms. Elaboración propia.

Como se puede observar en las matrices de la figura 16, la precisión de los modelos es alta, fallando en pocos de los casos en comparación al volumen de imágenes con los que se evaluó.

Modelos entrenados con los datos filtrados según peso (tamaño)

Para el entrenamiento de estos modelos, antes de subdividir las imágenes a los subconjuntos de entrenamiento y validación, se optó por el descarte de imágenes que no aporten mayor información al modelo cuando este sea entrenado. El filtrado que se realizó se

realizó considerando el peso de las imágenes, es decir, se establece un umbral arbitrario por conjunto de la base de datos, sea NEFER o replay según el etiquetado y origen de las muestras, y las imágenes que poseen un peso superior a este umbral eran consideradas para el dataset de entrenamiento, mientras que las demás se descartan. Además este umbral era determinado según el subconjunto de datos, debido a que las imágenes del subconjunto realizado con la bases de datos de NEFER no poseen el mismo peso de las imágenes utilizadas para replay attack, debido a la resolución de las mismas, siendo la resolución de eventos de NEFER de 1280x720 y la de replay, realizado con la DAVIS 346 de 346x260, influyendo al mismo tiempo en la resolución del recorte de los rostros. En la siguiente tabla se muestran los umbrales considerados para cada uno de los subconjuntos y el número de imágenes original y resultante de cada uno.

Ventana de tiempo	Subconjunto	Umbral de filtrado	Cantidad de imágenes antes de filtrado	Cantidad de imágenes después de filtrado
5 ms	NEFER	2 KB	22446	4957
	replay	1 KB	17826	6660
10 ms	NEFER	2 KB	23340	7033
	replay	1 KB	20118	10149
33 ms	NEFER	3 KB	23556	5095
	replay	2 KB	22852	4340
50 ms	NEFER	3 KB	23560	6841
	replay	2 KB	23258	6742

Tabla 4. Cantidad de imágenes presentes en los subconjuntos antes y después del filtrado por peso de las imágenes. Elaboración propia.

En la tabla anterior se muestran las cantidades de imágenes por subconjunto luego del filtrado, de igual manera se trató de realizar un filtrado con la intención de que las cantidades entre una clase y otra fuesen parejas, sin embargo esto fue más complicado para el subjuntivo de replay de la ventana de tiempo de 10 ms, debido a que si se aplica un umbral mayor, como por ejemplo 2 KB, la disminución de imagenes disminuye drásticamente, teniendo así, una diferencia mayor que la actual.

Por otro lado, del mismo modo que en el punto anterior, a continuación se muestra una tabla con los subconjuntos de datos según la ventana de tiempo con la que fueron formados, con la diferencia que esta vez solo se indica la cantidad de imágenes de los conjuntos de entrenamiento o validación, debido a que el número total de estas se indica en la tabla 3.

		Ventana de tiempo			
Conjunto	**Etiqueta**	**5 ms**	**10 ms**	**33 ms**	**50 ms**
Entrenamiento	NEFER	3779	5434	3719	5042
	replay	5218	8046	3379	5291
Validacion	NEFER	1178	1599	1376	1799
	replay	1442	2103	961	1451

Tabla 5. Conjuntos de datos según la ventana de tiempo con la que fueron formados. Elaboración propia.

Nuevamente el comportamiento de los modelos ante sus conjuntos de datos de validación respectivos fue altamente positivo, superando los modelos entrenados anteriormente con un *accuracy* del 100% ante estos datos. Y al mismo tiempo mostrando *loss* significativamente menores que la vez anterior, mostrando un sobreajuste extremadamente bajo, y hasta en el caso particular del modelo de 50 ms el cual fue prácticamente nulo ante el conjunto de imágenes de validación. Cabe destacar que de igual forma para el resto de modelos también lograron *loss* extremadamente bajos, indicando un bajo o nulo sobreajuste (overfitting) ante, indicando que los modelos logran generalizar bien las decisiones ante datos nuevos.

En la siguiente tabla se muestra un resumen de las pérdidas y *accuracy* obtenidos con los modelos en el último *epoch* del entrenamiento.

Δt	**Train Loss**	**Valid Loss**	**Accuracy**
5 ms	0,000915	0,000001	1,000000
10 ms	0,000141	0,000001	1,000000
33 ms	0,000180	0,000013	1,000000
50 ms	0,000092	0,000000	1,000000

Tabla 6. Resumen de las pérdidas y accuracy obtenidos con los modelos en el último epoch, según el conjunto de datos. Elaboración propia.

Además, del mismo modo que en punto anterior, en el anexo 3 se encuentran tablas con el detalle de los *loss* y *accuracy* obtenidos a lo largo de los *epoch* del entrenamiento.

Por otro lado, también al evaluar los modelos con las imágenes del conjunto de validación, se obtiene la matriz de confusión de los resultados al clasificar, por lo que se obtienen las siguientes cuatro matrices.

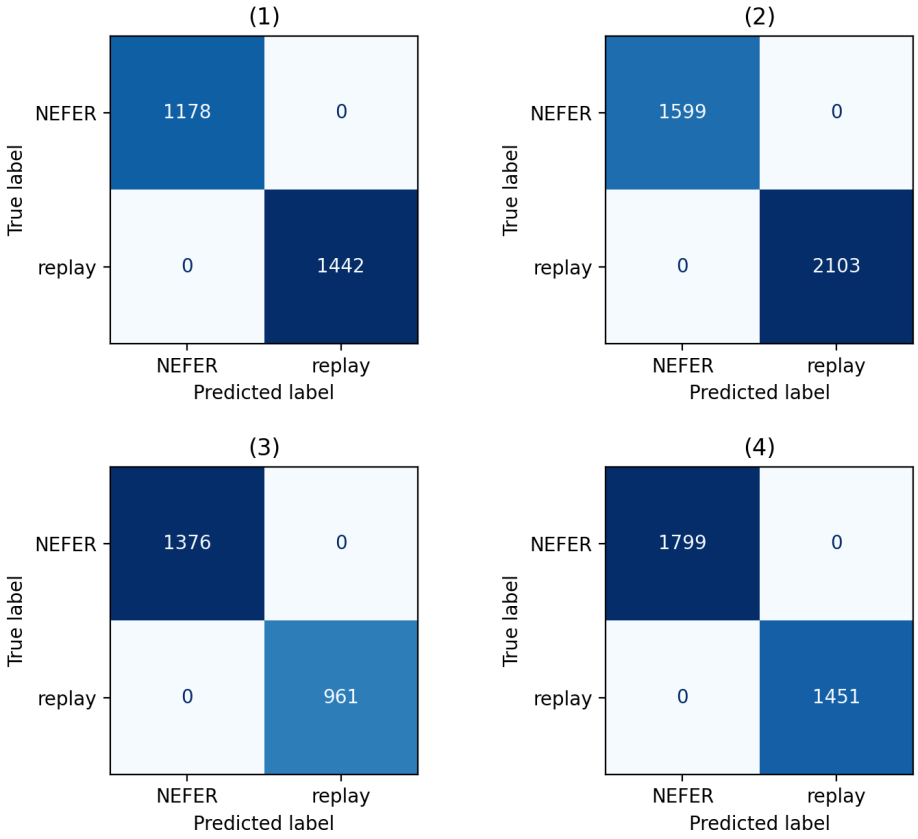

Figura 17. Matrices de confusión de la evaluación de los modelos con los datos del conjunto de validación respectivos, en donde la matriz (1) corresponde al modelo de 5 ms, (2) a 10 ms, (3) a 33 ms y (4) a 50 ms. Elaboración propia.

Como se puede observar en las matrices de la figura 17, la precisión de los modelos es alta, no fallando al clasificar ninguna de las imágenes de los conjuntos de validación respectivos para cada modelo.

Discusión

Como se pudo observar en los resultados de los modelos mostrados anteriormente, se logra observar que los mismos logran clasificar satisfactoriamente entre las imágenes que se realizaron con la base de datos de NEFER y los que corresponden a intentos de *replay attack*; mostrando una alta precisión al clasificar casos desconocidos a los modelos, identificando claramente los casos de ataque de reproducción, lo que ya se esperaba con la hipótesis inicial.

Si bien en los modelos entrenados con las imágenes sin filtrar, los modelos mostraban errores en algunas predicciones, aunque esta cantidad de errores era mínima en comparación al volumen de datos (comparado con el número total de predicciones realizadas) esto se pudo deberse al mismo filtrado realizado, afectando en la variabilidad de uno de los conjuntos de datos. Además algo que es importante destacar es que los fallos en la clasificación de los modelos fueron mayor en ventanas de tiempo más pequeñas, por lo que estos casos de error se pudieron deber a imágenes con una cantidad de eventos mínima, lo cual puede ser difícil para

el modelo discriminar, es más, podemos visualizar los casos con mayor pérdida del modelo, lo cual se muestra en la siguiente figura.

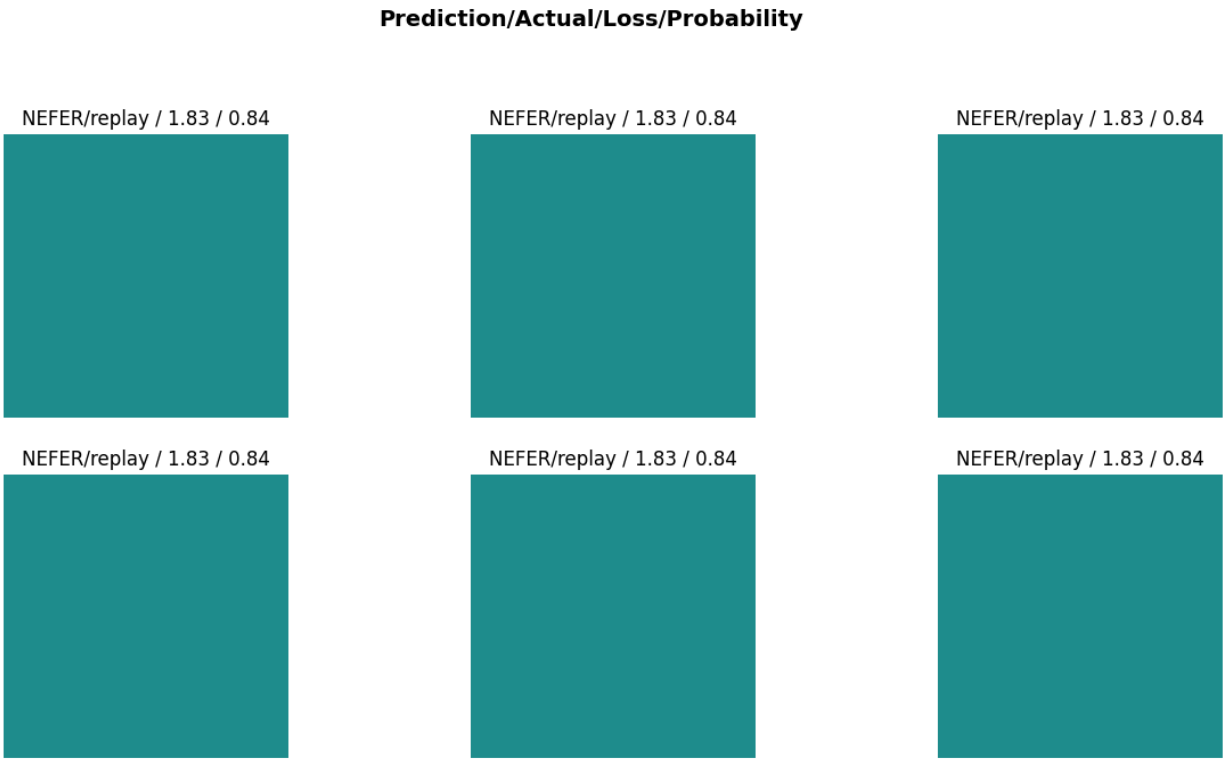

Figura 18. Casos con mayor pérdida en la predicción del conjunto de validación para visualizaciones de time surface de una ventana de 5 ms. Elaboración propia.

Como se puede observar en la figura anterior, se muestran seis casos de visualizaciones en las que el clasificador fallo, lo que al analizar el caso se puede deber a que no se contaba con la información suficiente para determinar una clasificación correcta. Una posible solución a esto puede ser determinar un umbral referente a la cantidad de eventos para asi recien ingresar la visualización al clasificador.

Por otro lado, no se exploró si la iluminación podría variar en el comportamiento de los modelos, sin embargo por el alto rango dinámico de la cámaras de eventos se puede suponer que esto no debería afectar en las predicciones.

Volviendo al tema el gran desempeño obtenido, lo que significa claramente que los modelos logran detectar entre sí la imagen de entrada corresponde a un caso de *replay attack* o si es originario a la base de datos NEFER. Es por ello que, deben existir elementos característicos dentro de las imágenes de entrada para que el modelo logre identificarlo correctamente, omitiendo agentes externos gracias a la caja que bloqueo la luz durante la captura de datos, este elemento puede ser provocado por el mismo monitor que reproduce los videos, por ejemplo, puede estar afectando la tasa de refresco del monitor, lo que puede generar ruido imperceptible para nosotros, pero que sí puede detectar el sensor.

Conclusión

En el transcurso de este trabajo, se ha presentado una base de datos *replay attack* en *face liveness detection* basada en eventos, utilizando la base de datos NEFER como base, lo que proporcionó un conjunto de datos rico y diverso para el entrenamiento de los modelos de aprendizaje profundo. La incorporación de eventos capturados por las cámaras de eventos en este conjunto de datos ha sido clave en la detección de características faciales, permitiendo a los modelos aprender patrones más complejos y específicos. Además de las capturas de datos de eventos, con información espacio-temporal, de intentos de ataques de reproducción dentro de una ambiente controlado.

La implementación de modelos de aprendizaje profundo, en particular, redes neuronales convolucionales (CNN) con la arquitectura *ResNet34*, ha demostrado ser eficaz en la tarea de *face liveness detection*, entregando una alta precisión en los resultados Por lo que, la capacidad de estos modelos para procesar la información espacio-temporal de los eventos ha mostrado un excelente rendimiento en la detección, lo que ha mostrado ser una herramienta prometedora en la discriminación entre rostros reales y livenessrepresentaciones falsas.

Además, este trabajo no solo aborda el desafío técnico de la *face liveness detection*, sino que también contribuye al estado del arte mediante la introducción y evaluación de cámaras de eventos en este contexto. La capacidad de estas cámaras para operar en tiempo real y capturar eventos con una resolución temporal asombrosa abre nuevas posibilidades en la autenticación facial y detección de ataques en ciberseguridad, especialmente su implementación en situaciones donde las cámaras convencionales presentan limitaciones. Por otra parte, esto podría inducir a un trabajo más extenso, en el que se utilice un conjunto de datos aún mayor, y abordando una mayor variedad de ataques y no solo ataques de reproducción.

Referencias

Alshaikhli, M., Elharrouss, O., Al-Maadeed, S., & Bouridane, A. (2021). Face-Fake-Net: The Deep Learning Method for Image Face Anti-Spoofing Detection. *2021 9th European Workshop on Visual Information Processing (EUVIP)*, 1-6. https://doi.org/10.1109/EUVIP50544.2021.9484023

Berlincioni, L., Cultrera, L., Albisani, C., Cresti, L., Leonardo, A., Picchioni, S., Becattini, F., & Del Bimbo, A. (2023, April 13). Neuromorphic Event-based Facial Expression Recognition. *arXiv*. https://doi.org/10.48550/arXiv.2304.06351

Bissarinova, U., Rakhimzhanova, T., Kenzhebalin, D., & Atakan Varol, H. (2023, August). Faces in Event Streams (FES): An Annotated Face Dataset for Event Cameras. https://doi.org/10.36227/techrxiv.22826654.v2

davisking/dlib-models: Trained model files for dlib example programs. (n.d.). GitHub. https://github.com/davisking/dlib-models

dlib C++ Library. (n.d.). dlib C++ Library. http://dlib.net/

He, K., Zhang, X., Ren, S., & Sun, J. (2015, december). Deep Residual Learning for Image Recognition. *arXiv*. https://doi.org/10.48550/arXiv.1512.03385

Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, *9*(8), 1735-1780. https://doi.org/10.1162/neco.1997.9.8.1735

Hu, J., Shen, L., Albanie, S., Sun, G., & Wu, E. (2019). Squeeze-and-Excitation Networks. *arXiv*. https://doi.org/10.48550/arXiv.1709.01507

Huang, G., Liu, Z., Maaten, L. v. d., & Weinberger, K. Q. (2018, August). Densely Connected Convolutional Networks. https://doi.org/10.48550/arXiv.1608.06993

Introducción a la convolución. (2021). MathWorks. https://la.mathworks.com/discovery/convolution.html

K., R. R., Vidya, K. R., & Wilscy, M. (2021, February 03). Detection of Deepfake Images Created Using Generative Adversarial Networks: A Review. *Springer, Cham*, 25-35. https://doi.org/10.1007/978-3-030-49500-8_3

Khairnar, S., Gite, S., Kotecha, K., & Thepade, S. (2023). Face Liveness Detection Using Artificial Intelligence Techniques: A Systematic Literature Review and Future Directions. *Big Data and Cognitive Computing*, *7*(37). https://www.mdpi.com/2504-2289/7/1/37

Kılıç, İ. (2023, September 15). *Perceptron Model: The Foundation of Neural Networks*. Medium. https://medium.com/@ilyurek/perceptron-model-the-foundation-of-neural-networks-4db25b0148d

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. *Advances in Neural Information Processing Systems*, *25*. https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf

La convolución discreta. (2019, March 9). https://ridertechblog.wordpress.com/acerca-de/la-convolucion-discreta

LarryDong/event_representation: Event-based camera data representation and processing. Some common representations and reference codes. (n.d.). GitHub. https://github.com/LarryDong/event_representation

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-Based Learning Applied to Document Recognition. *Proceedings of the IEEE*. http://yann.lecun.com/exdb/publis/pdf/lecun-98.pdf

Lenz, G., Ieng, S.-H., & Benosman, R. (2020, July 27). Event-Based Face Detection and Tracking Using the Dynamics of Eye Blinks. *Frontiers in Neuroscience*, *14*. https://doi.org/10.3389/fnins.2020.00587

The neural networks model. (2021, August 17). IBM Documentation. https://www.ibm.com/docs/en/spss-modeler/saas?topic=networks-neural-model

Pröve, P.-L. (2017, October 17). *Squeeze-and-Excitation Networks*. Towards Data Science. https://towardsdatascience.com/squeeze-and-excitation-networks-9ef5e71eacd7

Rehman, Y. A. U., Po, L. M., & Liu, M. (2017). Deep Learning for Face Anti-Spoofing: An End-to-End Approach. 195–200. https://doi.org/10.23919/SPA.2017.8166863

Roboflow: Give your software the power to see objects in images and video. (n.d.). Roboflow. https://roboflow.com/

Scheerlinck, Cedric and Rebecq, Henri and Stoffregen, Timo and Barnes, Nick and Mahony, Robert and Scaramuzza, DavideCED: Color Event Camera Dataset. (2019). CED: Color Event Camera Dataset.

Wei, J. (2019, July 2). *AlexNet: The Architecture that Challenged CNNs*. Towards Data Science. https://towardsdatascience.com/alexnet-the-architecture-that-challenged-cnns-e406d5297951

What are Neural Networks? (n.d.). IBM. https://www.ibm.com/topics/neural-networks

What is Machine Learning (ML)? - Enterprise ML Explained. (n.d.). AWS. https://aws.amazon.com/what-is/machine-learning/

Zhang, B., Tondi, B., & Barni, M. (2020). Adversarial examples for replay attacks against CNN-based face recognition with anti-spoofing capability. *Computer Vision and Image Understanding*, *197-198*, 102988. https://doi.org/10.1016/j.cviu.2020.102988

Anexos

Anexo 1: Códigos utilizados

Los códigos utilizados se en este trabajo se tituló se encuentran almacenados en el repositorio *hm_event-based_face_liveness_detection*, perteneciente el laboratorio de robótica de la Universidad de O'Higgins, *Robotics and Intelligent Systems Laboratory (RIS LAB)*, la direccion web para acceder a este repositorio es: https://github.com/uoh-rislab/hm_event-based_face_liveness_detection, sin embargo, se requiere autorizacion por el administrador del repositorio para obtener acceso y visualizar los codigos almacenados.

Anexo 2: Logs entrenamientos realizados con la totalidad de los datos

En las siguientes tablas se muestran los valores de pérdida (*loss*) en los conjuntos de entrenamiento y validación, además del *accuracy* y tiempo de ejecución por época de los modelos entrenados con la totalidad de los datos, según corresponda por la ventana de tiempo de los datos.

Anexo 2.1: Ventana de tiempo: 5 ms

	5 ms			
Epoch	**Train Loss**	**Valid Loss**	**Accuracy**	**Time**
0	0,091917	0,029983	0,981495	2:35
1	0,043364	0,013404	0,994695	2:35
2	0,040542	0,011579	0,997039	2:34
3	0,030029	0,007901	0,996792	2:32
4	0,026385	0,012228	0,996669	2:36
5	0,020547	0,007941	0,997409	2:33
6	0,018427	0,008058	0,996792	2:37
7	0,017085	0,006223	0,997903	2:32
8	0,016891	0,006296	0,997656	2:32
9	0,017111	0,005941	0,997779	2:37

Anexo 2.2: Ventana de tiempo: 10 ms

	10 ms			
Epoch	**Train Loss**	**Valid Loss**	**Accuracy**	**Time**
0	0,055998	0,024670	0,990847	2:48
1	0,035652	0,014814	0,993249	2:48
2	0,022366	0,009780	0,996110	2:50

3	0,021141	0,009440	0,995652	2:48
4	0,016114	0,006674	0,997826	2:48
5	0,016741	0,004570	0,998169	2:50
6	0,012873	0,004701	0,998513	2:48
7	0,010354	0,004828	0,998169	2:47
8	0,013132	0,004464	0,998627	2:52
9	0,017180	0,004309	0,998627	2:48

Anexo 2.3: Ventana de tiempo: 33 ms

	33 ms			
Epoch	**Train Loss**	**Valid Loss**	**Accuracy**	**Time**
0	0,016571	0,005818	0,997793	2:53
1	0,090070	0,004974	0,999054	2:52
2	0,006078	0,001889	0,999264	2:53
3	0,004905	0,000993	0,999685	2:51
4	0,004545	0,000253	1,000000	2:53
5	0,002452	0,012048	0,996531	2:54
6	0,003384	0,000789	0,999790	2:51
7	0,000393	0,000495	0,999790	2:53
8	0,001835	0,000282	0,999895	2:56
9	0,001697	0,000320	0,999895	2:54

Anexo 2.4: Ventana de tiempo: 50 ms

	50 ms			
Epoch	**Train Loss**	**Valid Loss**	**Accuracy**	**Time**
0	0,016487	0,001892	0,999376	2:56
1	0,007533	0,001244	0,999584	2:59
2	0,003608	0,000185	0,999896	2:56
3	0,006164	0,000014	1,000000	2:53
4	0,002739	0,000177	0,999896	2:54
5	0,004671	0,000029	1,000000	2:53
6	0,004140	0,000017	1,000000	2:53
7	0,001458	0,000011	1,000000	2:56
8	0,001523	0,000056	1,000000	2:54
9	0,003279	0,000033	1,000000	2:57

Anexo 3: Logs entrenamientos realizados con datos filtrados por peso

En las siguientes tablas se muestran los valores de pérdida (*loss*) en los conjuntos de entrenamiento y validación, además del *accuracy* y tiempo de ejecución por época de los modelos entrenados con los conjuntos de datos filtrados por el peso de las imágenes, según corresponda por la ventana de tiempo de los datos.

Anexo 3.1: Ventana de tiempo: 5 ms

	5 ms			
Epoch	**Train Loss**	**Valid Loss**	**Accuracy**	**Time**
0	0,118965	0,007194	0,997710	0:44
1	0,019012	0,001428	0,999237	0:42
2	0,007397	0,000191	1,000000	0:44
3	0,003792	0,000076	1,000000	0:44
4	0,007690	0,000022	1,000000	0:45
5	0,001627	0,000002	1,000000	0:44
6	0,001418	0,000001	1,000000	0:45
7	0,002300	0,000001	1,000000	0:45
8	0,000324	0,000001	1,000000	0:43
9	0,000915	0,000001	1,000000	0:44

Anexo 3.2: Ventana de tiempo: 10 ms

	10 ms			
Epoch	**Train Loss**	**Valid Loss**	**Accuracy**	**Time**
0	0,054696	0,002426	0,999730	1:11
1	0,007692	0,000488	0,999730	1:15
2	0,005622	0,000127	1,000000	1:09
3	0,005626	0,000169	1,000000	1:08
4	0,007291	0,002270	0,999460	1:08
5	0,001087	0,000010	1,000000	1:07
6	0,001122	0,000001	1,000000	1:07
7	0,000552	0,000002	1,000000	1:11
8	0,000143	0,000001	1,000000	1:08
9	0,000141	0,000001	1,000000	1:08

Anexo 3.3: Ventana de tiempo: 33 ms

	33 ms			
Epoch	**Train Loss**	**Valid Loss**	**Accuracy**	**Time**
0	0,126139	0,003140	0,999144	00:34
1	0,017685	0,000302	1,000000	00:33
2	0,003979	0,000095	1,000000	00:33
3	0,001927	0,000061	1,000000	00:34
4	0,000336	0,000011	1,000000	00:36
5	0,000334	0,000014	1,000000	00:35
6	0,000380	0,000009	1,000000	00:34
7	0,000162	0,000010	1,000000	00:34
8	0,000144	0,000005	1,000000	00:34
9	0,000180	0,000013	1,000000	00:38

Anexo 3.4: Ventana de tiempo: 50 ms

	50 ms			
Epoch	**Train Loss**	**Valid Loss**	**Accuracy**	**Time**
0	0,062460	0,001273	1,000000	00:53
1	0,005891	0,000129	1,000000	00:50
2	0,002675	0,000501	0,999692	00:51
3	0,001543	0,000002	1,000000	00:52
4	0,000173	0,000001	1,000000	00:51
5	0,000746	0,000001	1,000000	00:50
6	0,000734	0,000004	1,000000	00:51
7	0,000113	0,000000	1,000000	00:52
8	0,000084	0,000000	1,000000	00:53
9	0,000092	0,000000	1,000000	00:51