Universidad de Buenos Aires
Facultad de Ciencias Exactas y Naturales
Departamento de Computación

Reconocimiento de Expresiones Faciales en secuencias a través de Citation-kNN y GANs con preservación de identidad

Tesis de Licenciatura en Ciencias de la Computación

Andreas Sturmer

Directores: Pablo Negri y Daniel Acevedo

Buenos Aires, 2025

RECONOCIMIENTO DE EXPRESIONES FACIALES EN SECUENCIAS A TRAVÉS DE CITATION-KNN Y GANS CON PRESERVACIÓN DE IDENTIDAD

En este trabajo proponemos un método de Reconocimiento de Expresiones Faciales para la tarea de *One-Shot* en secuencias de video.

El reconocimiento de las expresiones se aborda mediante la correlación entre las dinámicas de los videos de entrada, que corresponden a una expresión realizada por un sujeto, y una serie de secuencias artificiales generadas por modernas arquitecturas de Redes Neuronales Profundas.

La generación de estos cuadros artificiales se logra a través de una Red Generativa Antagónica (GAN[1]). Luego, la expresión de entrada es categorizada a partir de aquella secuencia artificial más similar en el espacio de clasificación.

Se plantea como hipótesis que la correlación dinámica se vería reforzada mediante el uso de arquitecturas de redes neuronales y estrategias que conserven la información de identidad del sujeto de test.

Realizamos experimentos para distintas variantes de representación de las imágenes, enfocándonos en la familia de métodos de 2DLDA, y reportamos métricas para cada expresión.

Palabras claves: Reconocimiento de Expresiones Faciales, GAN, Aprendizaje Automático, Visión por Computadora, Video, Citation-kNN, 2DLDA

[1] Del inglés Generative Adversarial Networks.

FACIAL EXPRESSION RECOGNITION IN SEQUENCES USING CITATION-KNN AND IDENTITY-PRESERVING GANS

In this work we propose a method for One-Shot Facial Expression Recognition in video sequences.

Our approach to expression recognition is based on the correlation between the dynamics of an input video, corresponding to an expression made by the subject, and a series of artificial sequences generated by modern Deep Neural Network architectures.

This artificial frame generation is achieved through Generative Adversarial Networks (GAN). The input expression is then classified according to the most similar artificial sequence in the comparison space.

The main hypothesis is that the dynamic correlation can be reinforced through the use of neural networks and strategies that preserve the identity information of the test subject.

We conducted experiments with different image representation variations, focusing on the 2DLDA family of methods, and report metrics for each expression.

Keywords: Facial Expression Recognition, GAN, Machine Learning, Computer Vision, Citation-kNN, Video, 2DLDA

AGRADECIMIENTOS

Este trabajo no habría sido posible sin el apoyo de un montón de gente. Quiero agradecer a todas mis amistades y familia, las personas con las que cursé esta carrera, y también a mis directores que me acompañaron en todo este trayecto. Además quiero darle las gracias al grupo de investigación que trabajó en partes de este problema en los momentos que no estuve presente. En particular, muchas gracias a Noe, Marto, Fabro, Philip, Dani, Uri, Lu, Jony, Mati, Ema, y mi psicóloga.

Índice general

1. INTRODUCCIÓN

Las expresiones faciales son uno de los medios de comunicación no-verbal más importantes para expresar información social entre personas, como su intención y sus emociones. Así, el reconocimiento de expresiones faciales utilizando sensores ópticos es de particular interés para diversas áreas de estudio, como interacción persona-computadora (HCI), diagnóstico y tratamiento de enfermedades psiquiátricas, educación, seguridad, entretenimiento y robótica, entre otras.

Si bien existen métodos multimodales [1] que utilizan información externa a la imagen, como podría ser audio o datos fisiológicos de los sujetos, la mayoría de los métodos populares se enfocan en el reconocimiento de expresiones faciales en imágenes o secuencias de las mismas.

Se han desarrollado diversos modelos para representar las expresiones y/o emociones. Ekman y Friesen definen seis emociones básicas [2] que afirman ser universales a través de distintas culturas. Estas emociones son Angry, Disgusted, Fearful, Happy, Sad y Surprised, que corresponden a enojo, asco, miedo, felicidad, tristeza y sorpresa, respectivamente. Luego este modelo fue expandido con una expresión de desprecio [3]. Avances en neurociencia y psicología pusieron en duda la universalidad del modelo de seis emociones básicas, considerando el impacto cultural en la interpretación de las mismas para las distintas expresiones faciales [4]. Sin embargo, el modelo continúa siendo de utilidad para la categorización de gestos faciales.

Ekman y Friesen definieron más tarde un sistema de codificación de acciones faciales (FACS[1]) [5]. A través de estos dos modelos, Du y Martinez [6] categorizan diversos gestos faciales compuestos a partir combinaciones de *Action Units* (AU). Una *Action Unit* corresponde a un grupo de activaciones musculares faciales que permiten codificar distintos gestos a través del tipo y magnitud de la AU. Así, por ejemplo, una AU podría indicar cuando un sujeto frunce el ceño, mientras que otra determina si abre la boca. En ambos casos, la magnitud de la AU indica la intensidad con la que se realiza el gesto.

Estos modelos de representación se condicen con la apariencia de imágenes faciales, haciendo posible la clasificación de distintos gestos a través de sistemas de aprendizaje automático [7].

1.1. Definición del Problema

Nuestro trabajo aborda la problemática del reconocimiento de expresiones faciales en secuencias de imágenes correspondiente a un único sujeto. Es decir, dada una secuencia $s \in \mathbb{R}^{N \times W \times H \times C}$ de cuadros de imágenes faciales de un sujeto particular, vamos a obtener una serie de secuencias sintéticas $g_{exp} \in \mathbb{R}^{K \times W \times H \times C}$ que consisten en diferentes expresiones faciales de la misma persona, donde N es la cantidad de cuadros para la secuencia original, K la cantidad de cuadros para una secuencia sintética, W y H son el ancho y alto de la imagen respectivamente, C la cantidad de canales, y exp la expresión facial de una secuencia sintética. La secuencia s será clasificada a partir de la secuencia sintética que mejor la represente en este espacio. En nuestro caso, trabajamos con el conjunto

[1] Facial Action Coding System.

de 6 expresiones que definen Ekman y Friesen: Angry, Disgusted, Fearful, Happy, Sad y Surprised.

En la figura 1.1 se presenta un ejemplo de clasificación. En el mismo, una secuencia con la expresión Happy es clasificada correctamente. A partir de la secuencia de imágenes faciales s que parten de un gesto neutro a uno de máxima intensidad, se desea clasificar la expresión en una de las 6 prototípicas. Para esto, se sintetizan 6 secuencias a partir de la primera imagen s_0 a través del generador G. Luego, a través de un método T, se transforman esas secuencias a vectores en un espacio que maximiza la separación entre clases y minimiza la separación intraclase. Una vista de un ejemplo de las secuencias generadas transformadas a ese espacio se puede apreciar en la figura 1.2. Finalmente, estos vectores pasan a un modelo clasificador C que determina la expresión de la secuencia original.

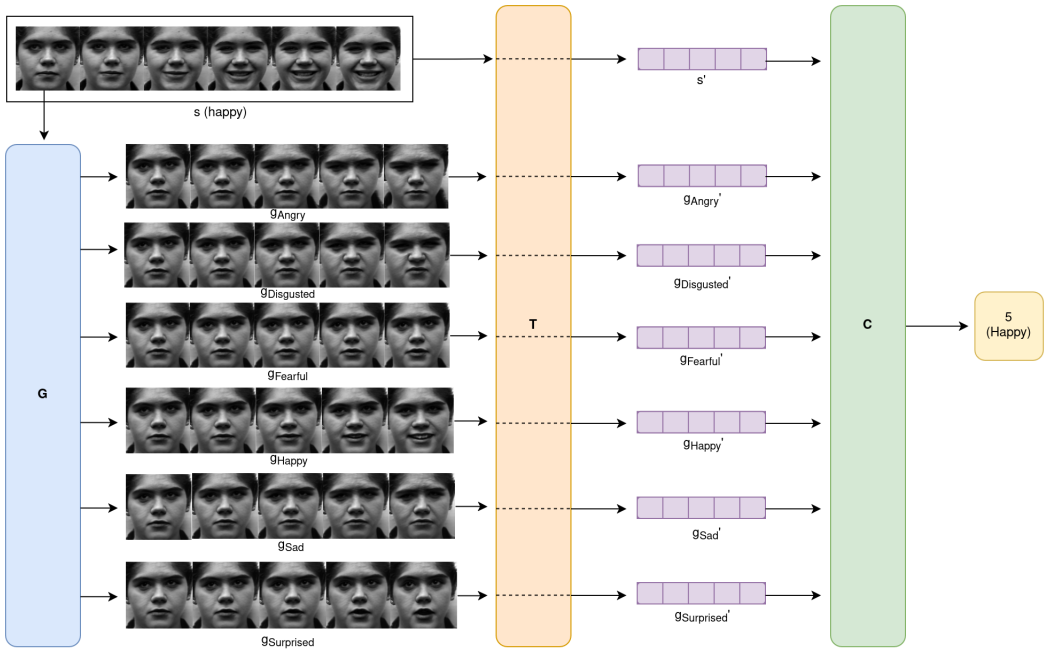


Fig. 1.1: Ejemplo de clasificación con nuestro método.

Este problema puede enmarcarse dentro del área de *Few-Shot Learning* (FSL)[8] . FSL es una subárea del aprendizaje automático que trata con problemas en los que la cantidad de datos utilizado en el dataset de entrenamiento del modelo predictivo es reducida. La única entrada de nuestro pipeline de clasificación es la secuencia de imágenes y la clasificación se realiza sobre un conjunto de secuencias generadas a partir de la de entrada. En esta tesis abordamos un caso particular de FSL denominado *One-Shot*, debido a que el clasificador es provisto de una sola secuencia de entrada sin datos adicionales.

Un ejemplo concreto sería en el campo de la biometría, las denominadas pruebas de vida (*liveness tests*). Este tipo de pruebas son desarrolladas con el objetivo de determinar si un sistema está interactuando con un ser humano vivo y no, por el contrario, con objetos inanimados o artificales diseñados para engañarlo. Para ello, se le pide al usuario realizar un gesto específico que el sistema debe validar.

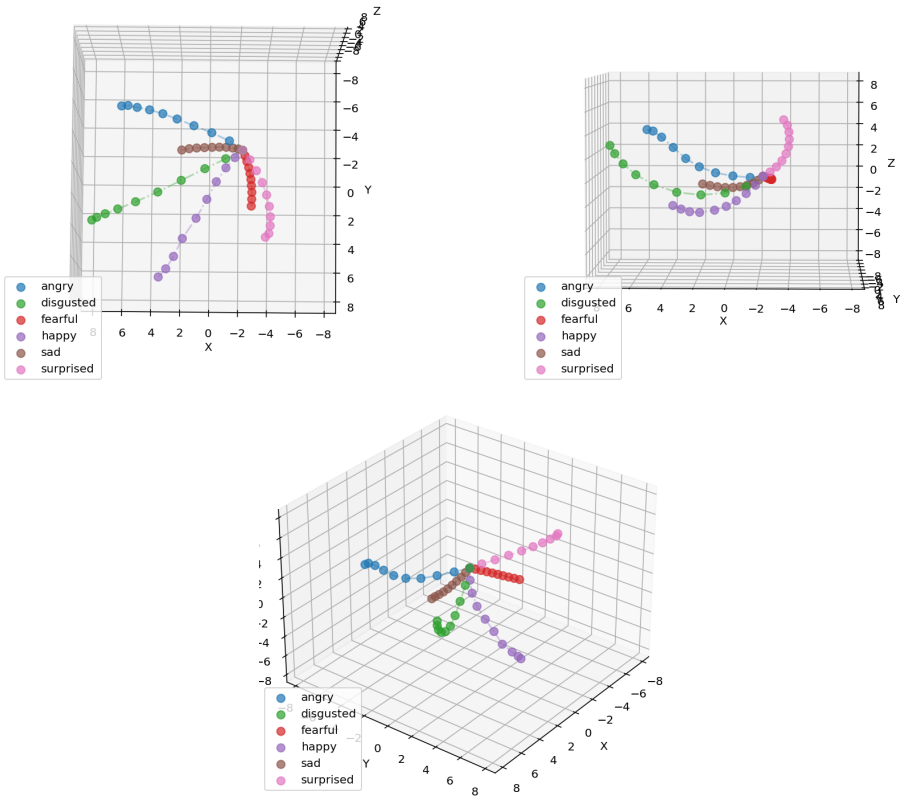


Fig. 1.2: Proyección de los cuadros de las secuencias generadas de un sujeto particular a un espacio que maximiza la separación interclase y minimiza la separación intraclase.

1.2. Hipótesis de Trabajo y Metodología

El objetivo principal de este proyecto es desarrollar un método para la clasificación *One-Shot* de secuencias de expresiones faciales. Es decir, deseamos clasificar la expresión de cada secuencia utilizando sólo datos adicionales generados con la propia secuencia. A diferencia de un método de aprendizaje automático tradicional, que entrenaríamos un clasificador con secuencias de un dataset, en el esquema *One-Shot* cada nueva secuencia genera un nuevo set de datos, donde se busca reconocer la expresión facial.

Un problema íntimamente relacionado al reconocimiento de expresiones es la síntesis de las mismas. Esto es, desarrollar métodos que tomen imágenes o videos de uno o varios sujetos y ajusten sus expresiones faciales a algún objetivo. Entonces, surge la idea de generar secuencias sintéticas para cada expresión con el objetivo de aumentar los datos. Así, nuestro clasificador podría usar estas secuencias generadas y compararlas contra la original (ver figura 1.1).

Un método general de síntesis de imágenes que ha obtenido cierto nivel de popularidad debido a sus buenos resultados en diversos problemas es el uso de Redes Adversarias Antagónicas (GAN). Introducidas por Goodfellow et al. en 2014 [9], son modelos que consisten en dos redes entrenadas alternadamente: un generador (G) que captura la distribución de los datos y un modelo discriminador (D) que estima la probabilidad de que una muestra provenga de los datos de entrenamiento.

Tradicionalmente, el generador toma de entrada un vector de ruido y devuelve una

imagen sintética. StyleGAN [10] propone una modificación a G en la que se mapea la entrada a un espacio latente intermedio, el cual controla el generador a través de *adaptive instance normalization* (AdaIN [11]) en cada capa convolucional. Esto hace posible el aprendizaje de una separación no-supervisada de los atributos de alto nivel (por ejemplo: pose e identidad) y la variación estocástica (por ejemplo: pecas, pelo) en las imágenes generadas, además de permitir un control intuitivo sobre la síntesis a través de la edición del vector latente de entrada. Esta característica de la red facilitaría la síntesis de rostros con distintas expresiones faciales manteniendo la identidad del sujeto.

La edición de imágenes con GANs es un desafío en sí mismo. El proceso de edición consiste en encontrar en principio la representación latente de la imagen original y un vector que represente la *"dirección"* del atributo a modificar. Si bien, la primera parte (encontrar una representación latente) es un problema estudiado, hay un *tradeoff* entre distorsión y capacidad de edición. HyperStyle [12] hace uso de una *hypernetwork*[2] que aprende a refinar los pesos del generador a partir de una imagen de entrada, y así logra obtener imágenes sintéticas de mayor calidad que a la vez sean editables sin provocar mucha distorsión. Decidimos utilizar StyleGAN2 [13] (la segunda versión de StyleGAN) en conjunto con el método de HyperStyle para generar las secuencias de expresiones.

Para la clasificación consideramos un método que tome las secuencias como conjuntos de datos con una única etiqueta, en vez de instancias etiquetadas por separado. Este tipo de aprendizaje supervisado se denomina *Multiple-Instance Learning* [14]. Citation-kNN [15] ofrece un enfoque que obtiene resultados comparativos al estado del arte. Describe un método que adapta kNN [16] (k vecinos más cercanos) a la tarea de *Multiple-Instance Learning* para el reconocimiento de expresiones faciales. El método utiliza un descriptor geométrico basado en landmarks ubicados en imágenes de caras, y una función particular para medir distancias entre bolsas de descriptores etiquetados con timestamps. Debido a su potencial y facilidad de implementación, decidimos explorar este método como clasificador.

El descriptor geométrico mencionado en el trabajo de Citation-kNN nos da pie a considerar otros tipos de representaciones de imágenes que faciliten la tarea del clasificador. El Análisis Discriminante Lineal (LDA) es un método que permite encontrar una combinación lineal de *features* que separan dos o más clases de objetos. Se usa tradicionalmente como un clasificador o para reducción dimensional. En nuestro caso, lo utilizamos para encontrar una proyección de las imágenes a un espacio de gestos faciales bien separados. En este trabajo probamos diferentes variantes de LDA para este proceso de transformación, y en particular 2DLDA [17], una versión bidimensional de esta metodología.

En resumen, nuestro sistema clasifica una secuencia utilizando Citation-kNN, comparandola contra secuencias sintéticas de distintas expresiones generadas con StyleGAN2. Para medir la distancia entre la secuencia original y las generadas, se proyectan a un espacio obtenido usando 2DLDA o sus variantes. Un diagrama de ejemplo del funcionamiento del método se encuentra en la figura 1.1. En el diagrama, el generador G corresponde a StyleGAN2, el proceso T a 2DLDA, y C a Citation-kNN.

Nuestra principal hipótesis es que es viable realizar el reconocimiento de gestos faciales a partir de generar un espacio de clasificación a partir de una sola imagen del sujeto a estudiar. Además, dentro de este espacio existe una separación entre las distintas expresiones, con distintas direcciones que corresponden a cada expresión particular. Además probaremos si el refuerzo en conservar la identidad del sujeto en la generación de las secuencias sintéticas impacta en la calidad de las predicciones.

[2] O hiperred, una red neuronal que aprende a generar pesos de otra red.

2. ESTADO DEL ARTE

La tarea de reconocimiento de expresiones faciales (FER) apunta a identificar gestos producidos por la cara o las emociones representadas por estos. Si bien tiene sus limitaciones, el modelo más popular para FER es el modelo categórico de 6 emociones básicas de Ekman y Friesen [2], en donde se definen 6 expresiones: Angry, Disgusted, Fearful, Happy, Sad y Surprised. Otros modelos consideran expandir estas expresiones con nuevas emociones (por ejemplo desprecio) [3], o también con expresiones compuestas [6]. Por otro lado, existen modelos que consideran dimensiones afectivas (excitación, valencia, intensidad y expectativa) [18], y sistemas de codificación de acciones faciales (FACS) [5] que proveen mayor granularidad para expresar gestos faciales.

En este trabajo nos enfocamos en el modelo de las 6 expresiones básicas. El problema consiste en, dada una secuencia de imágenes de una cara realizando un gesto, clasificar la expresión en una de estas seis. Nuestra solución contempla tres subproblemas: la edición automática de expresiones faciales en imágenes, la representación de imágenes faciales, y la clasificación de expresiones faciales en secuencias. Detallamos a continuación distintos métodos posibles para resolver cada uno de estos problemas.

2.1. Edición automática de expresiones faciales en imágenes

El primer subproblema consiste en, dada una imagen de una cara, devolver una imagen modificada de la misma con una expresión facial específica. Esto se relaciona con la síntesis de imágenes faciales. Varios métodos generativos se enfocan en este campo. Entre estos, detallamos GANs, VAEs y modelos de difusión.

Como se mencionó en la sección 1.2, las Redes Generativas Antagónicas (GAN) resuelven este problema a través de un entrenamiento alternado de dos redes que se encargan de sintetizar y discriminar los resultados. Editar características de un sujeto puede lograrse a través del condicionamiento de la red, como es en el caso de cGANs [19], en donde se agrega a la entrada el aspecto a modificar. Así, se podría elegir directamente la expresión deseada como parte de la entrada de la red generadora. Una versión de cGAN con mayor expresividad en la edición es GANimation [20], en donde se editan directamente *Action Units* (activaciones musculares) específicas de la cara.

Otra opción es utilizar la inversión de la GAN y la modificación de "estilos" a través del vector latente, como podría ser el caso de StyleGAN [10]. En nuestro trabajo, utilizamos StyleGAN2 [13] por la calidad de las imágenes generadas y su facilidad de edición. Explicamos GANs y StyleGAN con mayor detalle en la sección 3.2. En la figura 2.1 se muestra un esquema de la edición de una expresión con StyleGAN2 operando en el vector latente de una imagen facial neutra. Así, por ejemplo, si encontramos un vector $\mathbf{v}_{Happy}$ que represente la expresión Happy, podemos sumarlo al vector latente correspondiente a la cara neutra del sujeto $\mathbf{w}$. Este nuevo vector $\mathbf{W}$ es pasado como entrada a StyleGAN2 generando una nueva imagen facial con la expresión Happy.

Los *Variational Autoencoders* [21] (VAE) son autoencoders[1] en las que el codificador

[1] Un tipo de red neuronal que aprende una representación de los datos de entrada de una manera no-supervisada. Consiste de un codificador que comprime los datos de entrada y un decodificador que intenta reconstruir los mismos a partir del espacio latente.

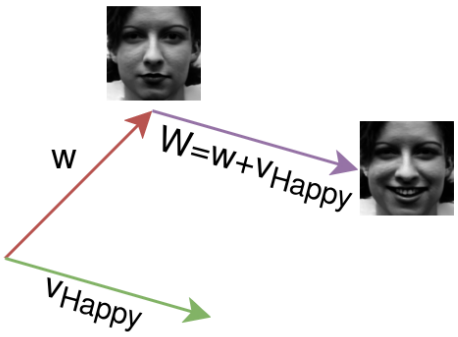


Fig. 2.1: Edición de expresión con StyleGAN2 a través de operación en el vector latente de una imagen facial neutra.

devuelve una distribución sobre el espacio latente en vez de un sólo punto, y a los que se le agrega un término de regularización a la función de pérdida para que este espacio esté mejor definido. Al igual que las GANs son métodos generativos: es posible utilizar VAEs para la edición de expresiones faciales. Yeh et al. describen un método en donde una VAE aprende a codificar el flujo óptico entre dos expresiones y luego al momento de test la edición se logra a través de aritmética del vector latente [22]. En la figura 2.2 presentamos un gráfico de su arquitectura. La red es un ejemplo de VAE que codifica el flujo óptico entre dos expresiones en vez de utilizar los píxeles de las imágenes directamente. Los bloques azules corresponden a capas de convolución y los bloques rojos a capas convolucionales de *upsampling*. Las flechas azules indican concatenación. El diseño de la red sigue una estructura de codificador-decodificador, donde el espacio latente, $\mathcal{Z}$, es el bloque central de $4 \times 4 \times 1024$ [22].

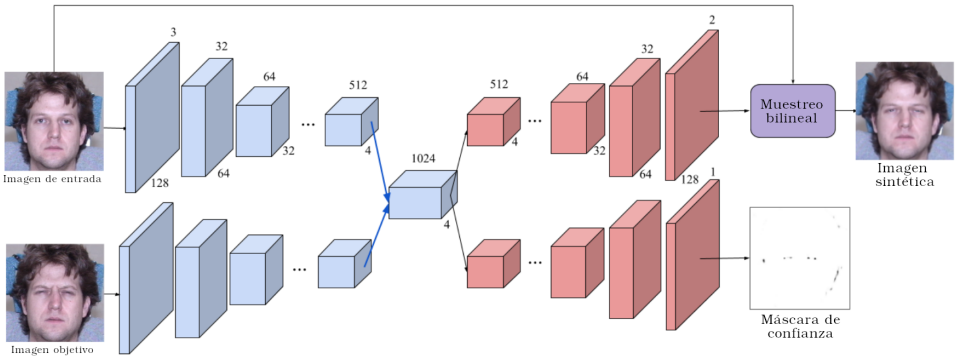


Fig. 2.2: Gráfico traducido de la arquitectura de FVAE tomado de [22].

Los modelos de difusión son modelos generativos que tienen de objetivo aprender un proceso de difusión que genera la distribución de los datos. Para esto, utilizan un proceso que agrega ruido y otro inverso que quita ruido. En conjunto, estos procesos permiten entrenar la red. Otro proceso de muestreo obtiene ruido aleatorio y utiliza la red entrenada

para la generación de datos [23]. Un ejemplo de edición de imágenes con este método es *MagicPose* [24], que utiliza un modelo de difusión para generar imágenes de un sujeto con una pose y expresión facial particular. Estas últimas se proveen de entrada en forma de un esqueleto y landmarks. En la figura 2.3 se muestra un gráfico de la arquitectura de *Stable Diffusion*, un modelo de difusión popular. Consiste en un codificador ε que codifica una imagen x a un espacio latente a través de un proceso de difusión, y un decodificador $\mathcal{D}$ que deconstruye la imagen a partir del latente resultando en $\tilde{x}$. La red es condicionada a través de un mecanismo de atención cruzada. *MagicPose* hace uso de *Stable Diffusion* en su método para generar imágenes de un sujeto con una pose y expresión facial particular [24].

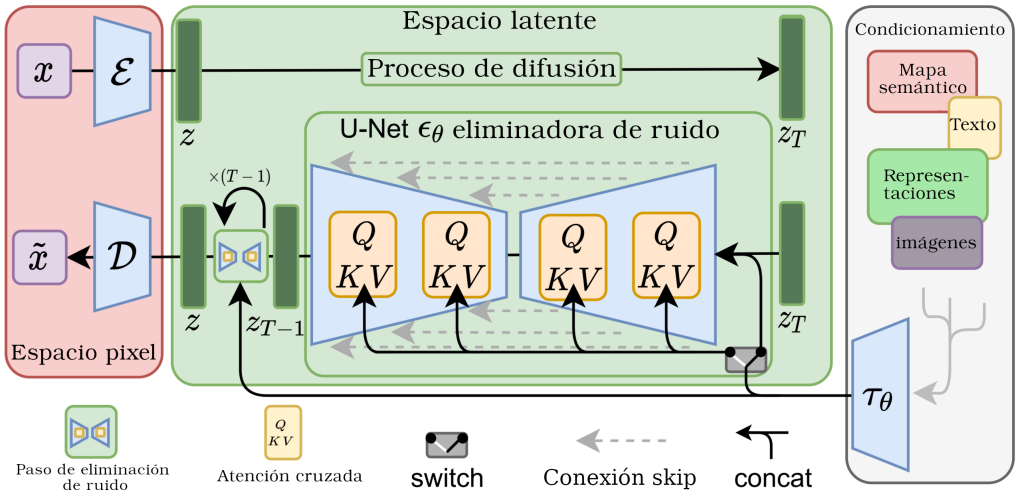


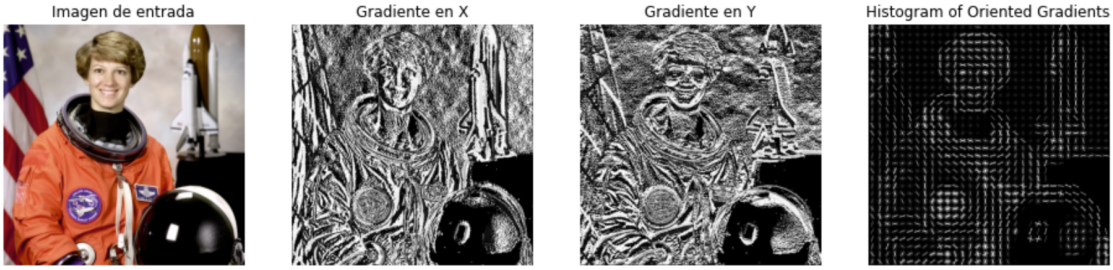
Fig. 2.3: Gráfico traducido de [25] de la arquitectura de *Stable Diffusion*, un modelo de difusión.

2.2. Representación de imágenes faciales

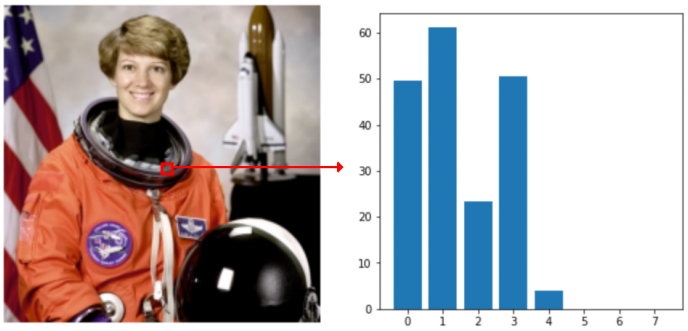
Con el objetivo de facilitar la tarea de clasificación, las imágenes faciales de cada cuadro en la secuencia pueden ser transformadas a otra representación vectorial o numérica. Es decir, a partir de una imagen definida como un arreglo bidimensional de píxeles, lo que se busca es obtener un vector o número que tenga alguna relación con la imagen original a través de algún proceso que la transforme. Una forma es a través de métodos que toman en consideración la forma de la cara, como podría ser usar landmarks, un descriptor geométrico, grupos de puntos, etc. Otras representaciones se basan en el uso de histogramas, como podrían ser LBP [26], LPQ [27], HOG [28] o QLZM [29]. Por otro lado, existen métodos de extracción de features nuevas que mapean una imagen de entrada a un espacio de baja dimensionalidad para descubrir una estructura de representación latente. Ejemplos de estos últimos podrían ser DCT [30], PCA [31] y LDA [7]. A continuación, explicamos brevemente las representaciónes mencionadas.

En los métodos basados en histogramas se calculan descriptores en distintas regiones de la imagen, para luego calcular sus histogramas y estos ser concatenados, dando de resultado un descriptor visual. *Local Binary Patterns* (LBP) [26] describe la variación local de una imagen a través de números enteros en una región circular de la misma. *Local Phase Quantisation* (LPQ) [27] usa la transformada de Fourier en distintas regiones.

Histogram of Oriented Gradients (HOG) [28] representa las imágenes por las direcciones de sus bordes. Extrae *features* locales aplicando operaciones de gradientes a través de la imagen y codificando su salida en términos de la magnitud y el ángulo del gradiente. En la figura 2.4 podemos apreciar un ejemplo de su aplicación. En (a) se presenta la imagen de entrada, sus gradientes en X e Y, y una visualización del HOG. En (b) una celda de 16x16 píxeles es extraída de la imagen, y un histograma es calculado a partir de sus gradientes. El descriptor es la concatenación de estos histogramas. Por último, *Quantised Local Zernike Moments* (QLZM) [29] se refiere a un método en donde una imágen se descompone en una serie de bases ZM (de momentos Zernike) ortogonales y coeficientes. Los coeficientes se cuantizan y luego combinan para formar una nueva imagen. Esta última se particiona en regiones, para luego calcular los histogramas como en los otros métodos. En la figura 2.5 se ilustra este método.



(a)



(b)

Fig. 2.4: Histogram of Oriented Gradients (HOG) computa los gradientes de la imagen, y luego calcula histogramas de celdas de la imagen a partir de la magnitud del gradiente en distintas orientaciones.

La Transformada de Coseno Discreta (DCT) [30] expresa una secuencia de datos en términos de sumas de funciones coseno de distintas magnitudes y frecuencias. DCT por lo general se utiliza para compresión de imágenes, ya que por lo general la mayoría de la información de una imágen está concentrada en algunos coeficientes de la DCT. Por esta razón también es utilizada en selección de *features*. Para esto se seleccionan los coeficientes de baja frecuencia y se disponen en un vector, mientras que los de alta frecuencia son descartados. Existen criterios de selección variados, entre ellos la técnica de selección en zigzag [32].

El Análisis de Componentes Principales (PCA) [31] es una técnica de reducción dimensional en donde se computa una transformación lineal con el objetivo de extraer *features*

decorrelacionadas. En el contexto de FER, en condiciones controladas de pose e imagen, capturan la estructura estadística de las expresiones de una manera eficiente [7]. PCA puede aplicarse directamente sobre las imágenes faciales, dando como resultado *Eigenfaces* [33]: los autovectores obtenidos del método (ver figura 2.6). Un descriptor posible es la combinación de un subconjunto de las *Eigenfaces* de una imagen. Por otro lado, es común la utilización de PCA sobre otra representación previa, y su posterior selección de *features* decorrelacionadas. Un ejemplo de esto es en el trabajo de Nicolle et al. [34], donde se extraen *features* utilizando landmarks y texturas de la cara, para luego aplicarles PCA.

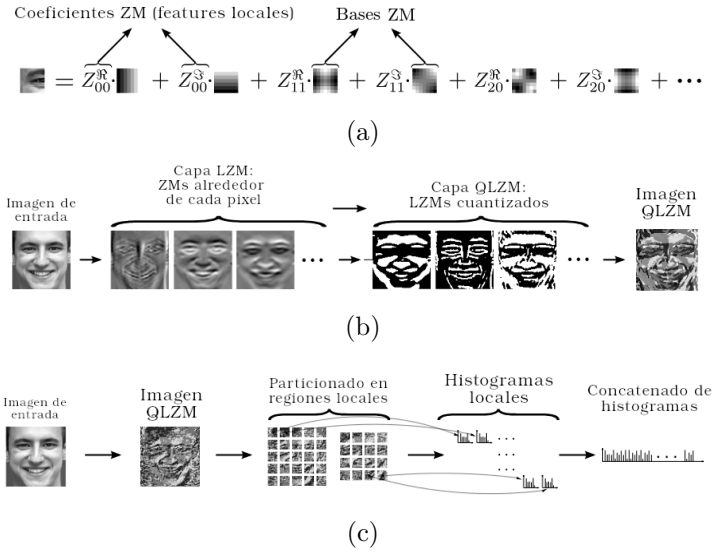


Fig. 2.5: Ilustraciones de QLZM tomadas de [29] y traducidas al español. (a) Coeficiente ZM y bases ZM. (b) Proceso de cómputo de una imagen QLZM. (c) Ilustración de todo el proceso de representación: Luego de computarse la imágen QLZM, se particiona en distintas regiones, se crean histogramas locales y se concatenan. LBP, LPQ y HOG realizan un proceso similar.

Mientras que PCA es un método no-supervisado, el Análisis Discriminante Lineal (LDA) utiliza información de etiquetas para aprender a cómo discriminar entre grupos de datos relacionados [7]. Esto se logra encontrando una transformación que proyecte los datos a un subespacio que maximice la separación entre clases y minimice la separación intraclase. Deng et al. [35] utilizan una combinación de filtros de Gabor locales, PCA y LDA para extraer *features* de imágenes faciales y luego clasificar las distintas expresiones. Damos una explicación más detallada de LDA y sus variantes en la sección 4.1.

Una alternativa a estas representaciones es utilizar *embeddings* obtenidos de entrenar una red neuronal para la clasificación de expresiones faciales. En *A Compact Embedding for Facial Expression Similarity* [36] se utilizan las últimas capas de una red entrenada con imágenes etiquetadas por similaridad de expresión para obtener esta representación vectorial.

Fig. 2.6: Algunas *Eigenfaces* de las imagenes faciales correspondientes a la expresión *Happy* del dataset Cohn-Kanade extendido.

2.3. Clasificación de expresiones faciales en secuencias

En cuanto a la clasificación, los métodos pueden dividirse entre los tradicionales y los que utilizan aprendizaje profundo. En el primer grupo podemos encontrar SVM, Naive Bayes y kNN, entre otros.

SVM es un modelo de clasificación que mapea la entrada a un espacio de alta dimensionalidad en el que luego se construye un hiperplano que separa los datos acorde a sus clases [37]. En la figura 2.7 podemos observar un ejemplo de clasificación con el método. Los vectores de soporte, marcados con cuadrados grises, definen el margen de la mayor separación entre las dos clases.

Naive Bayes toma de suposición principal que las *features* de los datos son independientes entre sí, aplicando el teorema de Bayes combinado con una regla de decisión para determinar a qué clase pertenecen los datos. En *Facial expression recognition from video sequences: temporal and static modeling* [38] se propone usar Naive Bayes para la clasificación estática de expresiones faciales.

Por otro lado, kNN es un método supervisado en el que los objetos son clasificados por un proceso de votación que considera la clase más representativa de sus k vecinos más cercanos. Se define esta cercanía con alguna métrica de distancia (por ejemplo, euclidiana) [16]. En la figura 2.8 se muestra un ejemplo de clasificación con kNN. Para K=3, el dato representado con el círculo naranja es clasificado en la clase de datos representados con triángulos azules, ya que de los 3 vecinos más cercanos, 2 son triángulos. De la misma forma, para K=5 se clasifica como la clase de los cuadrados verdes.

En el trabajo de Dino y Abdulrazzaq [39] se comparan tres métodos de clasificación: SVM, kNN y un perceptrón multicapa (MLP), en un proceso automático de clasificación de expresiones faciales. En el mismo se utiliza el algoritmo de Viola-Jones [40] para la detección facial. El descriptor utilizado es HOG para la extracción de features, al que luego se le aplica PCA para reducción dimensional, obteniendo las features más relevantes [39].

Entre los métodos que utilizan aprendizaje profundo, las Redes Neuronales Convolu-

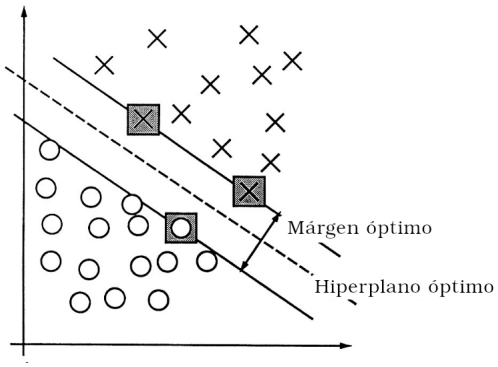


Fig. 2.7: Ejemplo de clasificación utilizando SVMs de un problema separable en 2D. Gráfico tomado de [37] y traducido al español.

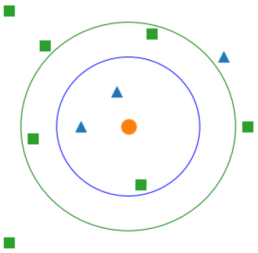


Fig. 2.8: Ejemplo de clasificación con kNN.

cionales (CNN) han obtenido popularidad por su efectividad en la tarea de clasificación de imágenes. Las arquitecturas de las CNN constan por lo general de tres tipos de capas distintas: convolucionales, de pooling y totalmente conectadas. La capa de convolución tiene tres beneficios principales: conectividad local, distribución de pesos en el mismo mapa de *features* e invarianza a cambios en la ubicación del objeto. La capa de pooling se usa para reducir costos computacionales y espaciales. Por último, la capa totalmente conectada por lo general se incluye al final de la red para convertir las imágenes en vectores de *features* unidimensionales para mejor representación y/o clasisficación [41]. Algunos modelos conocidos de redes neuronales convolucionales que fueron aplicados para FER son AlexNet [42](en [43]), VGGNet [44] (en [45] y [46]), GoogleNet [47] (en [48] y [45]) y ResNet [49] (en [50] y [51]) [41].

En la figura 2.9 podemos observar un gráfico de la vista interior de una red neuronal convolucional. Se aprecia la salida de cada capa de una arquitectura típica de CNN aplicada a la imagen de un perro Samoyedo (abajo a la izquierda, y los distintos canales de entrada (RGB), abajo a la derecha). Cada imagen es un mapa de *features* correspondiente a la salida de una de las *features* aprendidas detectadas en cada una de las posiciones de la imagen. La información fluye *bottom-up*: las *features* de bajo nivel actúan como detectores de bordes, y un puntaje es calculado para cada clase de salida (arriba) [52].

Si bien las redes neuronales profundas suelen obtener mejores resultados que los métodos tradicionales, también sufren de problemas de interpretabilidad. Los modelos son muy

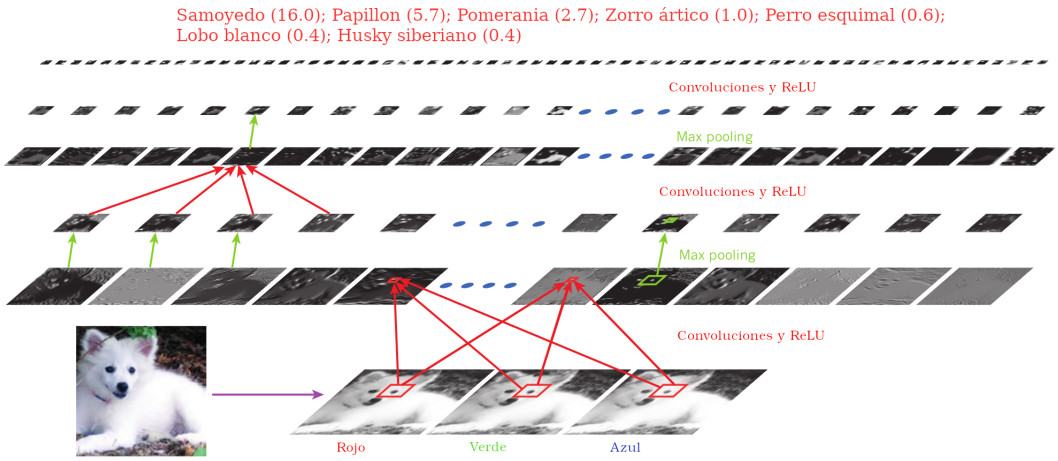


Fig. 2.9: Gráfico de la vista interior de una red neuronal convolucional tomado de [52] y traducido al español.

complejos, con millones de parámetros (por ejemplo, AlexNet [42] tiene 62 millones), por lo que suelen exhibir comportamientos inesperados. Por ejemplo, se puede modificar la entrada de la red de manera imperceptible para el ojo humano y esto puede llevar a cambiar arbitrariamente sus predicciones. También existen formas de producir imágenes que se ven como ruido blanco y sin embargo las redes reconocen como ciertos objetos con un 99.99 % de confianza [53].

En nuestro trabajo optamos por un método tradicional de clasificación, para hacer más sencilla la interpretación de los resultados. En particular, expandimos el trabajo expuesto en *A Citation k-NN Approach for Facial Expression Recognition* [15], donde se propone usar una variante de kNN para el problema de Multiple Instance Learning. Para más información sobre Citation-kNN, consultar la sección 4.

2.4. Pipeline

A partir de las soluciones seleccionadas construimos un método de reconocimiento de expresiones faciales en secuencias. En esta sección describimos el *pipeline* del mismo y cada una de sus componentes, desde su entrada hasta llegar a la clasificación de una secuencia de imágenes faciales correspondientes a una expresión particular.

La entrada del *pipeline* es una secuencia $s \in \mathbb{R}^{N \times W \times H \times C}$ de cuadros de imágenes faciales de un sujeto particular, a clasificar en una expresión particular. Siendo N el tamaño de la secuencia, y W, H y C el ancho, alto y cantidad de canales de cada cuadro, respectivamente. En nuestro caso particular, fijamos W=H=128 y C=1. Es decir, imágenes en escala de grises cuadradas sin transparencia.

Los cuadros $s_0, ..., s_{N-1} \in s$ están distribuidos en orden creciente de intensidad de la expresión, con s_0 correspondiendo a la expresión neutra del sujeto, y s_{N-1} la de máxima intensidad.

Las expresiones de las secuencias pueden ser una de 6 expresiones: Angry, Disgusted, Fearful, Happy, Sad o Surprised. La salida del modelo es una expresión $exp_{out} \in [1, 6]$. Cada uno de los valores en este rango corresponde a una de las expresiones anteriormente mencionadas.

(a) Fearful

(b) Happy

Fig. 2.10: Secuencias de entrada para un sujeto particular con distintas expresiones y cantidad de cuadros. A medida que progresa la secuencia (de izquierda a derecha), la intensidad de la expresión se acrecenta. (a) muestra una secuencia correspondiente a la expresión Fearful de N=9 cuadros, mientras que (b) muestra una secuencia correspondiente a la expresión Happy de N=12 cuadros.

El *pipeline* está compuesto por tres partes:

1. Un **Generador** (**G**) de secuencias de imágenes faciales con expresiones particulares.

   Es representado como una función $\mathbf{G} : \mathbb{R}^{W \times H \times C} \times \mathbb{N}^{[1,6]} \rightarrow \mathbb{R}^{K \times W \times H \times C}$, que toma de entrada una imagen facial neutra y una expresión particular $exp \in \mathbb{N}^{[1,6]}$ y devuelve una nueva secuencia de cuadros de imágenes faciales correspondientes a *exp*. En nuestro caso, **G** es el modelo generador de StyleGAN2 que se entrena *offline*. Es decir, usamos el modelo preentrenado, con los pesos iniciales fijos. A través de HyperStyle obtenemos tanto los *embeddings* de la imagen de entrada como modificaciones a los pesos del generador para poder generar la imagen facial lo más similar posible a la cara neutra. Luego, la secuencia para una expresión particular se genera modificando los *embeddings* obtenidos. Este proceso se realiza para cada expresión de nuestro conjunto de comparación y se explica en más detalle en la sección 3.1.

2. Una **función o proceso T** que transforma los datos a una representación que facilite la clasificación con **C**. Se representa como una función $\mathbf{T} : \mathbb{R}^{W \times H \times C} \rightarrow \mathbb{R}^{M}$. Que toma de entrada una imagen y la transforma a un vector de tamaño M. Nuestro enfoque consiste en utilizar 2DLDA (y variantes) como T (ver sección 4.1). Este proceso se aplica a todas las imágenes en las secuencias, tanto de entrada como las generadas.

3. El **Clasificador** (**C**).

   Que a su vez es representado como una función $\mathbf{C} : \mathbb{R}^{N \times M} \rightarrow \mathbb{N}^{[1,6]}$, que tomará de entrada la secuencia s transformada de longitud N y devolverá la expresión correspondiente. Para este trabajo, utilizamos *Citation-kNN* (consultar la sección 4).

Los pasos del *pipeline* son los siguientes:

1. **Generación de las secuencias sintéticas** a partir de un cuadro neutro de la secuencia de entrada.

2. **Transformación de las secuencias con $\boldsymbol{T}$**. Tanto la secuencia de entrada como las generadas.

3. **Clasificación de la secuencia original** (transformada por **T**).

La figura 2.11 expone el flujo de estos pasos. En las secciones 3 y 4 se explican en más detalle los mismos.

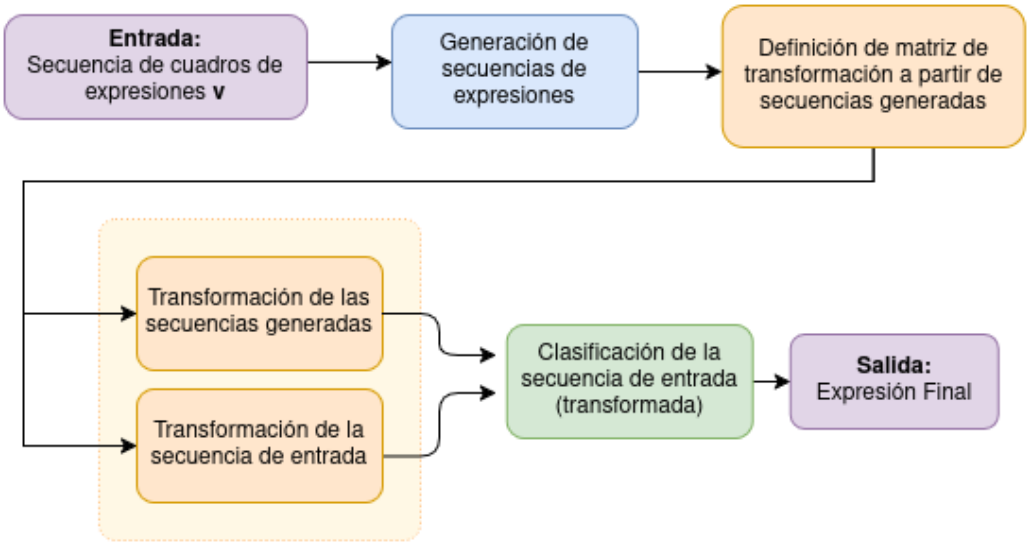


Fig. 2.11: *Pipeline* del método. Cada paso depende del que apunta a él con una flecha.

3. GENERACIÓN SINTÉTICA

La parte más importante de nuestro pipeline es la generación de secuencias de imágenes faciales con distintos gestos. En esta sección detallamos el proceso de generación de las secuencias sintéticas, y describimos las GANs, enfocándonos en StyleGAN2 y HyperStyle.

3.1. Generación de las secuencias sintéticas

Este paso consiste en generar secuencias de cuadros artificiales a partir del cuadro inicial $\mathbf{s}_0$ de la secuencia de entrada $\mathbf{s}$. Para esto, se utiliza el modelo generador $\mathbf{G}$ que toma de entrada tanto $\mathbf{s}_0$ como una expresión particular y devuelve una secuencia correspondiente a esa expresión. Esta última tiene el mismo formato que $\mathbf{s}$, pero con distinto tamaño fijo. Es decir, son secuencias $g_{exp} \in \mathbb{R}^{K \times W \times H \times C}$, con $g_{exp0}, ..., g_{expK-1} \in g_{exp}$ distribuidos en orden creciente de intensidad de la expresión exp, con g_{exp0} correspondiendo a la expresión neutra del sujeto, y g_{expK-1} la de máxima intensidad. En nuestro caso, fijamos K=10.

Se repite este proceso para cada expresión del conjunto, resultando en 6 secuencias (una para cada expresión: g_{Angry}, $g_{Disgusted}$, $g_{Fearful}$, g_{Happy}, g_{Sad} y $g_{Surprised}$). Estas son utilizadas como base de comparación del clasificador $\mathbf{C}$. Es posible generar más de una secuencia por expresión, lo cual hipotéticamente ayudaría a la clasificación. En la sección 5 se detallan experimentos que intentan probar esta hipotesis.

El entrenamiento del modelo generador no es parte de este pipeline. Específicamente utilizamos StyleGAN2 [13], con el método de HyperStyle [12] para conseguir los pesos correspondientes a la imagen neutra. En la sección 3.2 se detalla la arquitectura y detalles de esta red.

Fig. 3.1: Secuencias generadas para todas las expresiones de un sujeto.

Para generar una imagen, es necesario pasarle de entrada al generador un vector latente.

En principio, necesitamos encontrar un vector $\mathbf{w}$ que resulte en la imágen más similar a la de la cara neutra del sujeto. Una vez obtenido este vector, el próximo paso es editarlo de forma que genere la imagen de la cara con el gesto a expresar, conservando la identidad del sujeto. Esto se puede lograr sumando otro vector $\mathbf{v}$ correspondiente al gesto. Entonces, nuestro cuadro inicial en la secuencia se genera utilizando el vector latente $\mathbf{W}_0 = \mathbf{w}$, mientras que nuestro cuadro final con $\mathbf{W}_{K-1} = \mathbf{w} + \mathbf{v}$. Los cuadros intermedios se pueden obtener por interpolación lineal entre estos dos:

$$\mathbf{W}_i = \mathbf{w} + \mathbf{v}\frac{i}{K-1}, i = 0, ..., K-1 \tag{3.1}$$

Para obtener el vector $\mathbf{w}$ que genera la cara neutra, hacemos uso del encoder e4e [54]. Este, dada una imagen facial, devuelve un vector que pasado al generador de StyleGAN2 genera una imagen similar a la de entrada, garantizando la capacidad de edición de la misma. Luego, utilizamos el método propuesto en HyperStyle [12] para encontrar los pesos que modifican al generador con el objetivo de producir una imagen más fiel a la original.

Obtener $\mathbf{v}$ es un poco más complicado. Para cada expresión, queremos obtener un $\mathbf{v}$ representativo de esa expresión. Sin embargo, el problema está en que hay múltiples formas distintas de gesticular una expresión particular. Por ejemplo, un sujeto con la expresión Happy (feliz) podría realizar una sonrisa mostrando los dientes, mientras que otro sujeto podría no hacer esto y limitarse a una sonrisa más sutil con la boca cerrada.

Fig. 3.2: Ejemplos de diferencias en la expresión Happy. Algunos sujetos tienen la boca cerrada, otros abren la boca. Algunas expresiones son más sutiles que otras.

Para obtener este $\mathbf{v}$ representativo, hacemos uso de una regresión. Utilizamos el dataset de *Compound facial expressions of emotion* [6], que incluye distintas imágenes de sujetos realizando gestos faciales de máxima intensidad, etiquetados con el tipo de expresión correspondiente. Utilizamos únicamente las expresiones básicas, ya que el dataset también incluye imágenes para expresiones compuestas. Para cada una de las imágenes, extraemos el vector latente correspondiente con el encoder e4e.

Una opción sencilla para obtener estos vectores es realizar una regresión lineal con los residuos de los vectores latentes de las imágenes de las expresiones del dataset y las correspondientes a la caras neutras de los sujetos.

En este trabajo usamos Support Vector Machines (SVMs [37]) para encontrar los $\mathbf{v}$. La idea es tomar los vectores latentes de las expresiones y los correspondientes a las caras neutras, para cada expresión, y a partir de sus diferencias realizar una regresión para obtener vectores representativos para cada expresión. Se presenta pseudocódigo del método en el algoritmo 1.

Algorithm 1 Selección de vectores latentes para las expresiones

$direcciones \leftarrow \{\}$
for `exp` $\in$ `expresiones` **do**
  $pares \leftarrow []$
  $target \leftarrow []$
  for sujeto $\in$ sujetos **do**
    $w_0 \leftarrow latente[sujeto]["neutra"]$
    $w_1 \leftarrow latente[sujeto][exp]$
    $pares.append([w_0, w_1])$
    $target.append([0, 1])$
  end for
  $u \leftarrow SVM.fit(pares, target)$
  $u \leftarrow |u|$
  $u \leftarrow u - min(u)$
  $u \leftarrow \frac{u}{max(u)}$
  $deltas \leftarrow []$
  for sujeto $\in$ sujetos **do**
    $w_0 \leftarrow latente[sujeto]["neutra"]$
    $w_1 \leftarrow latente[sujeto][exp]$
    $delta \leftarrow (w_1 - w_0) * u$
    $deltas.append(delta)$
  end for
  $ud_0 \leftarrow promedio(deltas)$
  $direcciones[exp] \leftarrow ud_0$
end for

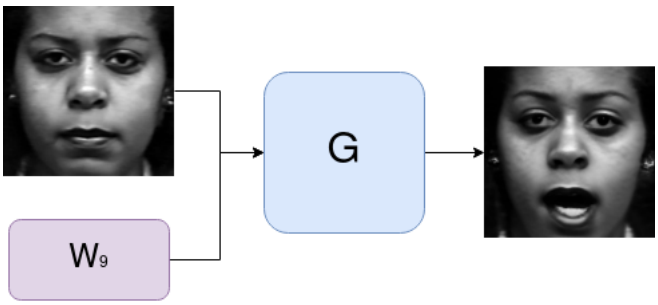


Fig. 3.3: Ejemplo de generación de un cuadro. El generador (**G**) toma de entrada una imagen neutra y un vector latente $\mathbf{W}_9$ (compuesto por $\mathbf{w} + \mathbf{v}$) para la expresión de sorpresa y devuelve un cuadro de la cara del sujeto con la expresión correspondiente.

Este método de generación de secuencias tiene sus limitaciones. En principio, la imagen neutra generada no es completamente igual a la original, y aunque es una muy buena aproximación, el sumarle la expresión introduce más modificaciones. El vector de expresión $\mathbf{v}$ no está completamente separado de otros atributos que no nos interesan, como podrían ser cambios en la pose, iluminación o hasta identidad del sujeto. Esto resulta en cambios accidentales en el último cuadro y los cuadros intermedios. En la figura 3.4 se muestra un ejemplo de las diferencias en la generación para un sujeto. Notar que hay

detalles que se pierden de la imagen correspondiente a la expresión neutra al pasarla por el generador. También, existen variaciones no intencionales como la pose en la expresión de enojo (Angry) generada. En este caso, la cara está orientada ligeramente hacia abajo comparada a la neutra.

(a) Expresión neutra original. (b) Expresión neutra generada. (c) Expresión de enojo (Angry) generada.

Fig. 3.4: Ejemplo de diferencias en la generación para un sujeto.

3.2. GANs

A continuación introducimos y detallamos el método de generación de secuencias sintéticas que utilizamos, y las redes relacionadas. En particular, en nuestro trabajo nos enfocamos en el uso de GANs. Ahondamos en la arquitectura de StyleGAN, una variante de GAN, y su segunda versión. Luego, describimos el uso que le damos al método descripto en HyperStyle [12] para inversión de GANs.

Como se mencionó en la sección 1, las GAN son un conjunto de redes neuronales, un generador G y un discriminador D, que se entrenan alternadamente en un juego que involucra la competencia entre ambas.

G se encarga de generar imágenes a partir de un vector de entrada ($\mathbf{z}$). Mientras que D recibe una muestra real o fabricada por el generador y debe distinguir entre las dos. G entonces es entrenado para generar imágenes que engañen al discriminador [55].

Formalmente, D y G optimizan el siguiente objetivo [9]:

$$\min_G \max_D \mathbb{E}_{x\sim p_{data}(x)}[logD(x)] + \mathbb{E}_{z\sim p_z(\mathbf{z})}[log(1 - D(G(\mathbf{z}))] \tag{3.2}$$

Donde p_{data} es la distribución de los datos y p_z es una distribución aleatoria (como podría ser una distribución uniforme).

3.3. StyleGAN y HyperStyle

Para la generación de secuencias usamos el generador de StyleGAN2 [13] preentrenado sobre el dataset Flickr-Faces-HQ (FFHQ) [10]. StyleGAN construye un *mapeo* a un espacio latente intermedio $\mathcal{W}$ con poco entrelazamiento, lo que facilita la edición de las imágenes generadas a través de la manipulación vector latente de entrada. Esto permite modificar atributos de alto nivel (como la pose o iluminación) de una manera coherente, sólo con variaciones estocásticas de detalle fino (por ejemplo, el pelo).

En StyleGAN, el discriminador no sufre ninguna modificación comparado a una GAN tradicional, y sólo varía la arquitectura del generador. La diferencia en el generador radica

en que el vector de entrada $\mathbf{z} \in \mathcal{Z}$ pasa por una red de mapeo $f : \mathcal{Z} \to \mathcal{W}$ que devuelve un vector latente intermedio $\mathbf{w} \in \mathcal{W}$. Este a su vez pasa por la red de síntesis, siendo modificado por transformaciones afines para convertirse en "estilos" y que controlan operaciones AdaIN [11] luego de cada capa convolucional [10], como puede observarse en la figura 3.5. AdaIN se refiere a *Adaptive Instance Normalization*, un método de normalización que alínea la media y varianza de las *features* del contenido (x) a aquellas del estilo (y). Está definida como:

$$AdaIN(x, y) = \sigma(y) \left(\frac{x - \mu(x)}{\sigma(x)} \right) + \mu(y) \tag{3.3}$$

Para generar detalle estocástico, se introducen explícitamente entradas de ruido. Estas son imágenes de ruido gaussiano que se pasan a cada capa de la red de síntesis.

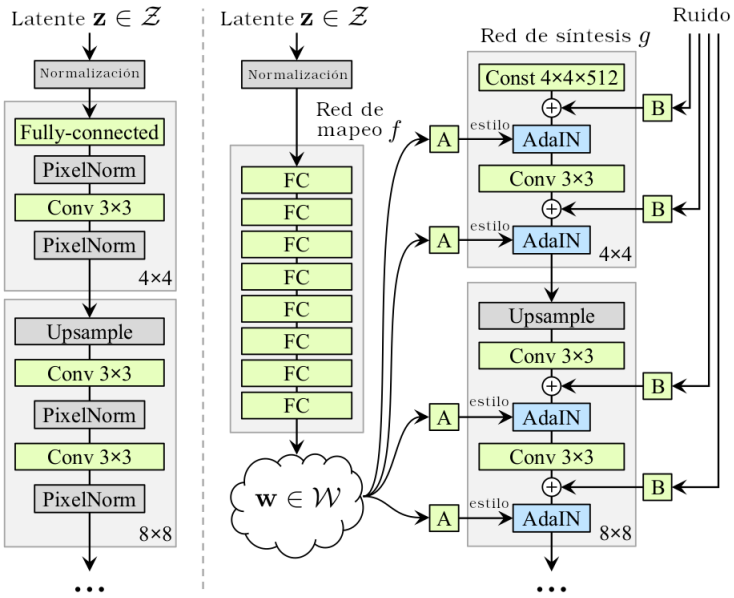


Fig. 3.5: Arquitectura de un generador tradicional (izquierda) vs el utilizado en StyleGAN (derecha). Este último mapea la entrada a un espacio latente intermedio $\mathcal{W}$, el cual luego controla el generador a través de *Adaptive Instance Normalization* (AdaIN) en cada capa convolucional [10]. Gráfico de *A Style-Based Generator Architecture for Generative Adversarial Networks* [10] traducido.

El entrenamiento de estas redes se realiza siguiendo el método descripto en *Progressive GANs* [56]. Esto es, se entrena haciendo crecer progresivamente la GAN, empezando con imágenes de resoluciones pequeñas y agregando capas a las redes para trabajar con resoluciones cada vez mayores.

La segunda versión de StyleGAN, StyleGAN2 [13], se enfoca en corregir artefactos comunes de su predecesor. Para esto, introduce cambios a la arquitectura del generador, al que también se le aplica regularización, y se modifica el entrenamiento del modelo para no utilizar el crecimiento progresivo.

Las imágenes generadas por StyleGAN contenían artefactos en forma de manchas. Estas provenían de la operación AdaIN, que normaliza el promedio y varianza de cada

mapa de *features* de forma separada, destruyendo potencialmente cualquier información de las distancias entre las *features*. Al eliminar la normalización, los artefactos desaparecen completamente. Por lo tanto, se reemplaza *instance normalization* con una operación de demodulación, la cual se aplica a los pesos asociados a cada capa convolucional. Además, se decide mover las operaciones de ruido y sesgo por fuera del bloque de estilos.

La métrica de *perceptual path length* (PPL), introducida como un método para estimar la calidad de las interpolaciones en el espacio latente $\mathcal{W}$, está correlacionada con la consistencia y estabilidad de las formas y la calidad percibida de la imagen. Se aplica regularización a la red de síntesis, considerando esta métrica. Como esta operación es costosa computacionalmente, se realiza una vez cada 16 *minibatches*.

El crecimiento progresivo tiene sus propios artefactos. Por ejemplo, *features* como dientes u ojos deberían moverse de manera suave sobre la imagen al rotar la cabeza de un sujeto. Sin embargo, en la versión original de StyleGAN, quedan fijas para luego saltar instantáneamente a la ubicación deseada. Durante el crecimiento progresivo, cada resolución sirve momentáneamente como la resolución de salida, forzando al modelo a generar detalles de máxima frecuencia. Esto lleva a que la red entrenada tenga frecuencias excesivamente altas en las capas intermedias, comprometiendo la invarianza de desplazamiento.

Se exploran otras arquitecturas para no utilizar el crecimiento progresivo, como una modificación de MSG-GAN [57], un modelo que conecta las resoluciones del generador y discriminador usando *skip connections*, y una red residual [49] para el discriminador. Por último, se entrenan modelos más grandes para obtener mayor calidad.

En nuestro trabajo, necesitamos tener la capacidad de generar imágenes de distintas expresiones faciales para el sujeto que esté en la secuencia de imágenes de entrada. Particularmente, deseamos generar la misma imagen de una expresión neutra del sujeto y a partir de esta editarla. Para esto necesitamos obtener el vector latente que pasado como entrada al generador logre formular una imagen lo más similar posible a la original. Este problema se denomina *inversión de GAN*.

Utilizamos HyperStyle [12]: un método de inversión de GAN que permite encontrar un vector latente y una modificación a los pesos del generador de StyleGAN para lograr imágenes con alta similaridad a la original, manteniendo una buena capacidad de edición. Para lograr esto, se utiliza un *encoder* e4e [54] preentrenado que da una estimación inicial, y una hiperred [58] que es entrenada para predecir los pesos del generador minimizando la distorsión de las imágenes reconstruidas. Este proceso es mejorado a través de un refinamiento iterativo.

Una vista simplificada de la generación de un cuadro sintético de ejemplo puede observarse en la figura 3.6, en este caso para la expresión Happy. StyleGAN2 toma de entrada un vector latente $\mathbf{W}$ que se compone de la suma de otros dos, $\mathbf{v}_{Happy}$ y $\mathbf{w}$. Realizamos el proceso de inversión de GAN utilizando e4e [54] para obtener el vector latente $\mathbf{w}$ correspondiente a la cara neutra del sujeto (primer cuadro de la secuencia de entrada). El vector $\mathbf{v}_{Happy}$ corresponde al vector latente de la expresión Happy, y se obtiene utilizando una SVM [37] entrenada sobre los vectores latentes de imágenes faciales que realizan esa expresión del dataset de *Compound facial expressions of emotion* [6]. En este esquema no se muestra el ajuste de pesos de la red utilizando HyperStyle.

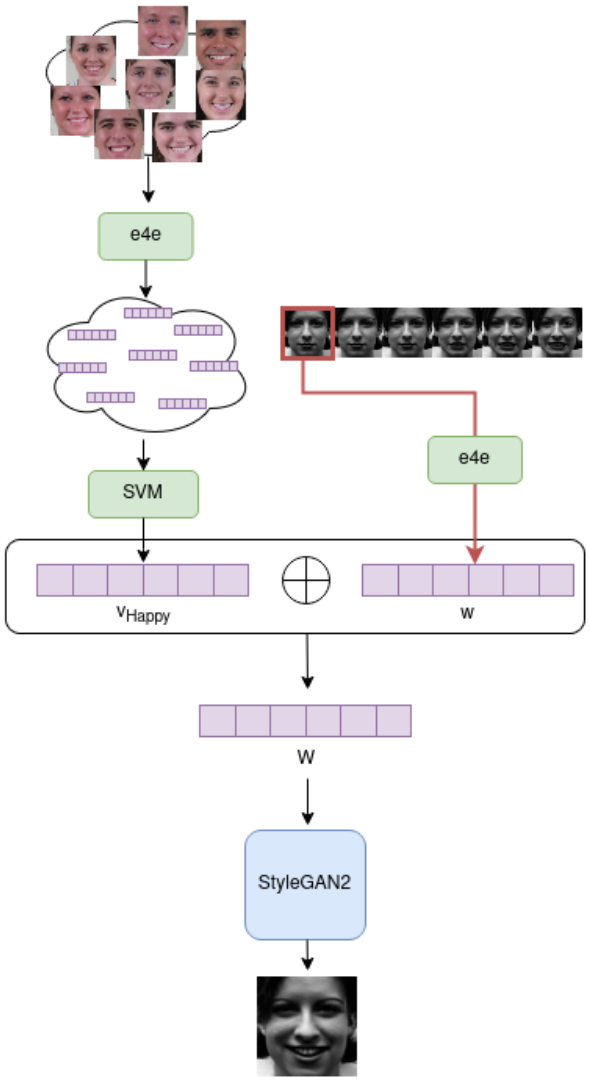


Fig. 3.6: Vista simplificada de la generación de un cuadro sintético para la expresión Happy.

4. CLASIFICACIÓN

El último paso de nuestro método es la clasificación. En esta sección se explican los métodos de transformación aplicados a la entrada del clasificador, el modelo clasificador utilizado (Citation-kNN) y la métrica de distancia seleccionada.

4.1. Transformación de las secuencias: 2DLDA

Utilizamos una transformación de los datos para facilitar la tarea de clasificación. Tanto la secuencia original como las generadas pasan a través de este proceso. Para esto, utilizamos 2DLDA y sus variantes. La transformación se obtiene a través de las secuencias generadas de un sujeto, y luego todas las secuencias (incluída la original) son proyectadas a este nuevo espacio. Las proyecciones se realizan individualmente, cuadro a cuadro.

En la figura 4.1 se observa la proyección de los cuadros de las secuencias generadas para un sujeto particular utilizando 2DLDA. Los vectores resultantes son reducidos a tres dimensiones con PCA para permitir la visualización. Cada color corresponde a una expresión.

A continuación, damos una breve explicación de 2DLDA y los métodos derivados explorados en este trabajo.

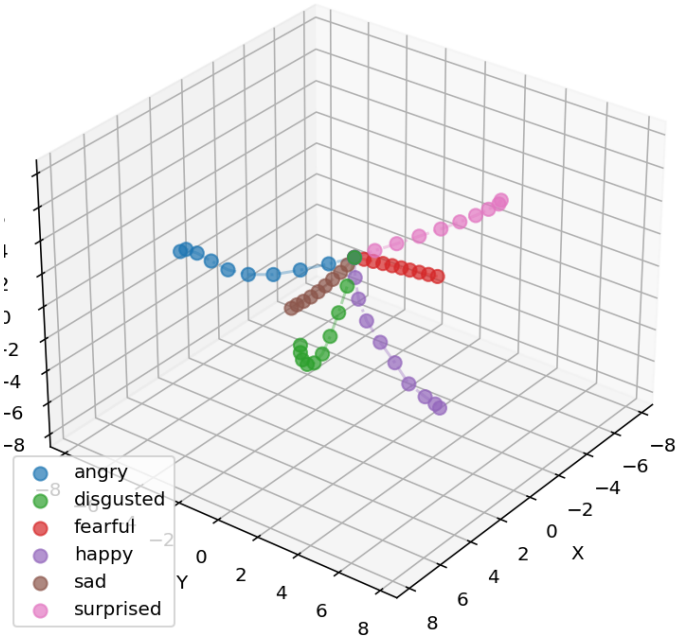


Fig. 4.1: Proyección con 2DLDA de los cuadros de las secuencias generadas para un sujeto particular, reducidos dimensionalmente con PCA para la visualización.

4.1.1. LDA

LDA es un método que permite encontrar una combinación lineal de *features* que separan dos o más clases de objetos. Es una generalización del análisis discriminante de Fisher [59, 60]. El método consiste en encontrar una matriz $W = [\mathbf{w}_1, \mathbf{w}_2, ..., \mathbf{w}_{r_1}] \in \mathbb{R}^{d_1 \times r_1}$, con $r_1 \leq d_1$ que proyecte los datos a un subespacio que maximice la separación entre clases y minimice la separación intraclase.

Trabajamos con un conjunto de datos $[X_1, ..., X_N]$ y sus etiquetas asociadas $[y_1, ..., y_N]$, donde $X_l \in \mathbb{R}^{d1}$ es un vector e $y_l \in \{1, ..., c\}$ la etiqueta asociada, y siendo $l = 1, ..., N$. Cada clase i contiene N_i muestras, con $i = 1, ..., c$.

Definimos $\overline{X} = \frac{1}{N} \sum_{N}^{l=1} X_l$ como el promedio del total de las muestras, y $\overline{X_i} = \frac{1}{N_i} \sum_{N_i}^{j=1} X_{ij}$ como el promedio de las muestras de la clase i.

Entonces, las matrices de separación entre clases (S_b) e intraclase (S_w) se definen como [61]:

$$S_b = \frac{1}{N} \sum_{i=1}^{c} N_i (\overline{X_i} - \overline{X})(\overline{X_i} - \overline{X})^T \tag{4.1}$$

y

$$S_w = \frac{1}{N} \sum_{i=1}^{c} \sum_{j=1}^{N_i} (X_{ij} - \overline{X_i})(X_{ij} - \overline{X_i})^T \tag{4.2}$$

El objetivo a optimizar está dado por:

$$\max_{W} \frac{tr(W^T S_b W)}{tr(W^T S_w W)} \tag{4.3}$$

La solución a este problema se obtiene calculando los autovectores asociados a los primeros autovalores mayores del problema de autovalores generalizado $S_b w = \lambda S_w w$. Una vez obtenido W, para proyectar una muestra original o nueva, sólo se debe multiplicar a izquierda por W^T. Así, $X' = W^T X$.

4.1.2. 2DLDA

2DLDA [17] es una extensión de LDA a dos dimensiones, introducida principalmente para tratar con imágenes. En vez de vectores de entrada, tomamos matrices $[X_1, ..., X_N]$, con $X_l \in \mathbb{R}^{d1 \times d2}$. Salvo por ese detalle, el método es análogo a LDA.

Dependiendo de cómo se armen las matrices de separación inter e intraclase, se operará en las filas o columnas de la matriz [62]. Las fórmulas de S_b y S_w mostradas en 4.4 y 4.5 operan sobre las filas. Para operar sobre las columnas, se deben modificar estas matrices por

$$S_b = \frac{1}{N} \sum_{i=1}^{c} N_i (\overline{X_i} - \overline{X})^T (\overline{X_i} - \overline{X}) \tag{4.4}$$

y

$$S_w = \frac{1}{N} \sum_{i=1}^{c} \sum_{j=1}^{N_i} (X_{ij} - \overline{X_i})^T (X_{ij} - \overline{X_i}) \tag{4.5}$$

En nuestro trabajo, cuando mencionamos 2DLDA a secas, hacemos referencia a esta versión alternativa que opera sobre las columnas.

4.1.3. (2D)²LDA

Si aplicamos ambas versiones de 2DLDA (la que opera con filas y la de columnas), obtendríamos dos matrices de proyección U y V. Entonces, podríamos proyectar una imagen como $X' = U^T XV$, reduciendo dimensionalmente la imagen tanto en ancho como en alto. A este método se lo denomina (2D)²LDA [62].

Definimos:

$$S_b = \frac{1}{N}\sum_{i=1}^{c} N_i(\overline{X_i} - \overline{X})(\overline{X_i} - \overline{X})^T \tag{4.6}$$

$$S_w = \frac{1}{N}\sum_{i=1}^{c}\sum_{j=1}^{N_i} (X_{ij} - \overline{X_i})(X_{ij} - \overline{X_i})^T \tag{4.7}$$

$$S_{b2} = \frac{1}{N}\sum_{i=1}^{c} N_i(\overline{X_i} - \overline{X})^T(\overline{X_i} - \overline{X}) \tag{4.8}$$

$$S_{w2} = \frac{1}{N}\sum_{i=1}^{c}\sum_{j=1}^{N_i} (X_{ij} - \overline{X_i})^T(X_{ij} - \overline{X_i}) \tag{4.9}$$

Luego, obtenemos U y V maximizando

$$\max_V \frac{tr(V^T S_b V)}{tr(V^T S_w V)} \tag{4.10}$$

$$\max_U \frac{tr(U^T S_{b2} U)}{tr(U^T S_{w2} U)} \tag{4.11}$$

4.1.4. G2DLDA

Existen muchas otras variantes de 2DLDA. G2DLDA [61] es una generalización de 2DLDA, utilizando un método iterativo e incluyendo regularización. Siendo $V_i = \overline{X_i} - \overline{X} \in \mathbb{R}^{d1 \times d2}$, $Z_{ij} = X_{ij} - \overline{X_i} \in \mathbb{R}^{d1 \times d2}$, $i = 1, ..., c$, $j = 1, ..., N_i$, y $\|\cdot\|_F$ la norma Frobenius, se redefine el objetivo a optimizar como

$$\min_W \frac{\sum_{i=1}^{c}\sum_{j=1}^{N_i} \|W^T Z_{ij}\|_p^p + \sigma\|W\|_p^p}{\sum_{i=1}^{c} \|W^T V_i\|_p^p} \tag{4.12}$$

Sujeto a que $W^T W = I$ (garantizando ortogonalidad de los vectores discriminantes), agregando el término regularizador $\sigma\|W\|_p^p$, y tomando una norma arbitraria.

4.2. Citation-kNN

Para la clasificación utilizamos Citation-kNN [14], un método que combina la clasificación de los k-vecinos más cercanos (kNN [16]) junto con una estrategia de citas y referencias aplicado al problema de *Multiple Instance Learning* (MIL).

El enfoque tradicional de kNN determina la clase de una instancia a través de una votación de las instancias vecinas en el set de entrenamiento. En Citation-kNN, se toman en cuenta no sólo los vecinos de esa instancia al momento de votación, sino a todas las instancias que tienen como vecina a la que se va a clasificar. Así, las $R - referencias$ se definen como los R vecinos de la instancia. Las $C - citas$ de una instancia corresponden a todas las instancias que contengan a la primera entre sus C vecinos más cercanos. En el caso de clasificación binaria, se calculan los votos positivos p como $p = R_p + C_p$ y los negativos como $n = R_n + C_n$ y luego se determina que la clase es positiva si $p > n$.

En el paper original de Citation-kNN [14], el método es utilizado para resolver el problema de MIL, el cual consiste en clasificar conjuntos (o "bolsas") de instancias a las que se les asigna una sola clase, independientemente de si las instancias dentro del mismo pertenecen o no a la clase determinada. Para esto, fue necesario definir una función de distancia entre "bolsas" distinta a la euclidiana.

4.3. Distancia Hausdorff

En Citation-kNN [14], la métrica seleccionada fue la distancia Hausdorff, definida como:

$$H_{SL}(A, B) = \text{máx}(h_S(A, B), h_L(B, A))$$

Siendo A y B "bolsas" de instancias, y h_F la distancia Hausdorff dirigida, definida como:

$$h_F = F^{th}_{a \in A} \min_{b \in B} \|a - b\|$$

Es decir, para cada punto de a, se calcula su vecino más cercano en b (con una norma que podría ser la euclidiana). Estos se ordenan y el valor F-rankeado se toma como la distancia Hausdorff dirigida final.

En las figuras 4.2 y 4.3 se muestran ejemplos de distancia Hausdorff entre secuencias de imágenes faciales[1]. La figura 4.2 es un ejemplo entre secuencias de sujetos distintos usando el descriptor geométrico. Las flechas indican el cuadro de menor distancia euclidiana. Las flechas rojas son de A a B, mientras que las azules de B a A. Aquí la distancia Hausdorff dirigida para ambos lados es la misma (4.61), pero esto no necesariamente es así en toda comparación de secuencias. En la figura 4.3 se aprecia un ejemplo de distancia de un cuadro de la secuencia A a los otros en la secuencia B usando el descriptor geométrico.

4.4. Citation-kNN para FER

Acevedo et al. [15], aplican este método de clasificación al problema de clasificación de expresiones faciales en secuencias. En principio, siendo que el método de clasificación

[1] En realidad, la distancia en estos casos se calcula entre secuencias de descriptorees geométricos obtenidos a partir de estas imágenes de caras. Siendo que esto se explica en la siguiente sección, decidimos simplificar por razones pedagógicas.

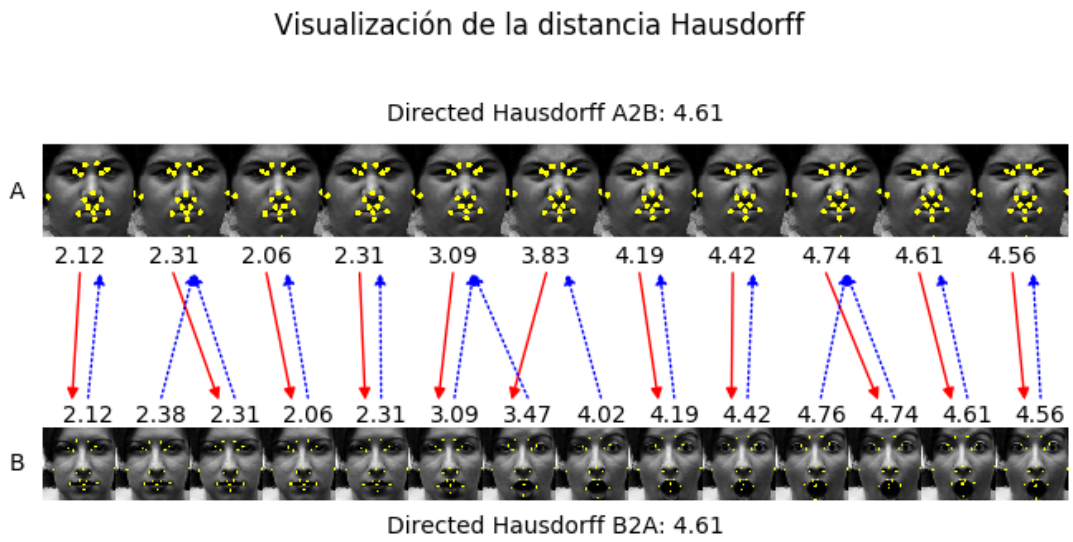


Fig. 4.2: Ejemplo de distancia Hausdorff entre secuencias de sujetos distintos usando el descriptor geométrico.

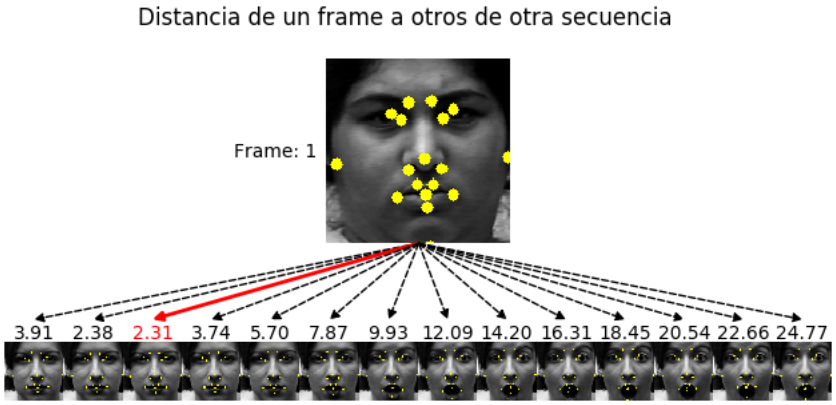


Fig. 4.3: Distancia de un cuadro de A a los otros en la secuencia B usando el descriptor geométrico. La flecha roja indica el cuadro con menor distancia euclidiana.

original es binario [14], expanden la votación de Citation-kNN para considerar múltiples clases. Ahora cada referencia y cita vale un voto, que es pesado por la distancia Hausdorff de la instancia a su cita/referencia (y tomando $1/D$, para que sea una función decreciente de la distancia.).

Sea L el conjunto de todas las posibles etiquetas. Para cada "bolsa" de entrada a clasificar X, $\{r_1, ..., r_{n_r}\}$ y $\{c_1, ..., c_{n_c}\}$ son las etiquetas de las clases de sus n_r referencias más cercanas y n_c citas más cercanas. La clase predicha l^* se obtiene sumando los votos de las citas y referencias:

$$l^* = argmax_{l \in L} \left(\sum_{i=1}^{n_c} \alpha_i \delta(l, c_i) + \sum_{i=1}^{n_c} \beta_i \delta(l, r_i) \right) \tag{4.13}$$

donde $\delta(a, b) = 1$ si y sólo si $a = b$ y $\delta(a, b) = 0$ de otra manera. Y α_i y β_i son los

pesos para la cita C_i y referencia R_i respectivamente. En el paper definen $\alpha_i = \frac{1}{H(X,C_i)}$ y $\beta_i = \frac{1}{H(X,R_i)}$, siendo $H(\cdot)$ la distancia Hausdorff descripta anteriormente.

Para cada cuadro en las secuencias, se extraen los landmarks de la cara. De un set de 68, toman 18 específicos y con estos construyen 816 triángulos que describen la forma geométrica de la cara y su dinámica. A partir de estos triángulos, construyen un descriptor basado en los ángulos y áreas de estos triángulos, al cual se le agrega un *timestamp* para darle información temporal al cuadro. Los ángulos seleccionados en estos descriptores corresponden a los que mayor variación tienen a través de la secuencia, para todas las secuencias en la base de datos. En este caso entonces, las "bolsas" que toma Citation-kNN están conformadas por estos descriptores geométricos con timestamps agregados para cada cuadro.

En el paper original, obtienen 89.30 % de precisión en pruebas de "leave-one-out subject cross validation" utilizando sets de validación y entrenamiento que consideran secuencias que van de la expresión neutra a otra de máxima intensidad. Estas últimas provienen del dataset Cohn-Kanade extendido [63, 64].

Method	Train: OA Test: OA	Train: OAO Test: OA	Train: OA Test: OAO	Train: OAO Test: OAO
Accuracy (%)	89.30	83.18	83.79	89.30

Tab. 4.1: Precisión con los distintos datasets de validación y entrenamiento en el paper de *Acevedo et al.* [15] OA: Onset-Apex, OAO: Onset-Apex-Offset.

Además, proveen resultados del método aplicado a la clasificación de cada expresión particular (Anger, Contempt, Disgusted, Fearful, Happy, Sad, Surprised)

	An	Co	Di	Fe	Ha	Sa	Su	Promedio de las clases
Cit-kNN	84.40	77.80	96.60	44.00	98.60	75.00	98.80	82.10

Tab. 4.2: Precisión por clase en el paper de *Acevedo et al.* [15]. Representa la cantidad de secuencias correctamente clasificadas sobre el total de secuencias para una clase particular. El último valor representa el promedio de estos valores.

4.5. Citation-kNN para FER aplicado a la tarea *One-Shot*

Nuestro problema particular concierne, al igual que *Acevedo et al.* [15], la clasificación de una secuencia de imágenes faciales correspondiente a una expresión particular. Estas secuencias parten de una cara neutra a una que denota la expresión con máxima intensidad.

La diferencia con el paper original consiste en que la clasificación no depende de otras secuencias del dataset, sino únicamente de la secuencia original. Con este propósito, generamos secuencias para cada expresión utilizando StyleGAN2. Luego, el proceso de clasificación es equivalente: Se calcula la distancia Hausdorff de la secuencia original a cada una de las secuencias sintéticas. Con estas distancias se obtienen tanto las citas como referencias y se calcula la etiqueta a partir de la ecuación 4.13.

Sin embargo, una adaptación directa de Citation-kNN con el descriptor geométrico utilizado en el paper, no obtuvo resultados convincentes. En las sección 5.3 se detallan los experimentos para esta tarea y sus resultados.

5. EXPERIMENTOS Y RESULTADOS

En esta sección describimos los experimentos realizados con sus respectivos resultados. Comenzamos detallando cómo preparamos los datos para su uso en los mismos. Luego, presentamos el caso *baseline* y continuamos con las pruebas con el esquema *One-Shot* para distintas variantes de 2DLDA. Exploramos también el uso de múltiples secuencias por gesto, el impacto de HyperStyle en la generación de cuadros sintéticos, y la utilización de embeddings de la red generadora como representación de las imágenes faciales. Adicionamos un pequeño experimento sobre la rotación de las expresiones Angry y describimos algunos errores de clasificación comunes a través de ejemplos.

5.1. Preparación de datos

El dataset utilizado para la experimentación fue Cohn-Kanade extendido (CK+) [64] para todos los experimentos salvo expreso contrario. Utilizamos 309 (327 cuando incluimos Contempt) secuencias etiquetadas obtenidas de 118 sujetos. La distribución de las instancias se observa en la tabla 5.1

Expresión	An	Di	Fe	Ha	Sa	Su	Total
#Secuencias	45	59	25	69	28	83	309

Tab. 5.1: Distribución de las instancias por clase para el dataset Cohn-Kanade extendido

Para la detección de landmarks utilizamos MTCNN [65][1], y a partir de estos alineamos y recortamos las imágenes. La alineación se realiza a partir de una transformación afín, que asigna los landmarks a puntos de referencia de una imagen de entrada de StyleGAN2. El *crop* incial es de 1024x1024 centrado en la cara.

Aplicamos esta transformación tanto a las imágenes del dataset *Compound facial expressions of emotion* (para la obtención de vectores latentes de las expresiones) como las de Cohn-Kanade extendido. Para las de CK+, utilizamos el crop de 1024x1024 para obtener los vectores latentes base de las caras neutras. Luego, para la clasificación utilizamos imágenes escaladas de una resolución de 128x128 píxeles en escala de grises. Tanto las imágenes de la secuencia de entrada como las imágenes generadas tienen ese tamaño. Un ejemplo de todo el proceso de transformación de las imágenes se puede apreciar en la figura 5.1.

5.2. Baseline: *Leave-One-Out* con CkNN sobre el descriptor geométrico

Como base, partimos de la replicación del trabajo de *Acevedo et al.* [15]. Nos remitimos a la tarea de OA (Onset-Apex) en el contexto de MIL (Multiple Instance Learning). Esto es, las secuencias clasificadas corresponden a series de cuadros de un mismo sujeto realizando una expresión particular partiendo desde la expresión neutra, hasta una de 7 prototípicas[2] de máxima intensidad. De la misma forma, la metodología de testeo fue con

[1] Específicamente, usamos esta implementación: `https://github.com/timesler/facenet-pytorch`

[2] En este experimento agregamos la expresión de Contempt con el objeto de mantener la comparación con los números del trabajo original.

Fig. 5.1: Ejemplo de transformación de la imagen original del dataset Cohn-Kanade extendido al formato que utilizamos en los experimentos. De izquierda a derecha: Imagen original, detección de landmarks, transformación y crop, crop final centrado en la cara.

validación cruzada usando un esquema de "leave-one-out" para sujetos. Es decir, se itera por cada sujeto y se utilizan todas las secuencias del dataset como base de comparación del modelo, salvo aquellas que consideren al sujeto actual. Las secuencias del sujeto actual se clasifican una por una comparando contra este conjunto. Luego se repite la misma tarea con los demás sujetos.

Si bien en el trabajo original se obtiene 89.30 % de precisión para esta tarea con ciertos hiperparámetros, con esta reimplementación logramos un 85.32 % de precisión. Estos números se alinean más a otras tareas que se detallan en el paper (como OAO vs OA) presentadas en la tabla 4.1. Esto se deba probablemente a diferencias en la implementación, como podría ser la selección de "mejores" ángulos, el procesado de las secuencias del dataset, o la obtención de landmarks, entre otras. En la tabla 5.2 podemos visualizar el *recall* del método por cada expresión. En cada expresión, salvo miedo (Fearful), obtenemos menor o igual *recall* en las clasificaciones.

Los parámetros utilizados fueron $\beta = 30$, $s = 3{,}6$, $R = 8$, $C = 20$, y $K = 0{,}9$. Siendo β el peso que se le da a los timestamps, s la escala de las áreas, R el número de referencias a considerar, C el número de referencias que poseen las citas, y K un parámetro de la distancia Hausdorff que indica el elemento a tomar en el ranking.

5.2.1. Diferencias entre usar mejores ángulos o no

El descriptor geométrico está constituído por los ángulos y áreas de los triángulos conformados por los landmarks. Cada triángulo está conformado por 3 landmarks, por lo que es necesario realizar una elección del ángulo a considerar. Una opción simple es tomar el primer ángulo de cada triángulo. Una alternativa un poco más sofisticada consiste en tomar los ángulos que maximicen en promedio las diferencias entre cada cuadro en la secuencia. Realizamos una comparación entre estas dos formas de armar el descriptor geométrico tomando los mismos parámetros en cada caso.

Si es que sólo tomamos los ángulos "default", obtenemos una *accuracy* de 84.40 %, por lo que hay una reducción pequeña en la misma. En la tabla 5.2 se comparan el *recall* por cada expresión entre los dos métodos. Podemos observar que existe un impacto negativo en usar ángulos "default" en el *recall* de la mayoría de las clases salvo Angry y Disgusted que mejoran sus valores y Contempt que lo mantiene.

Como obtenemos mejores resultados en general, decidimos utilizar el método de "mejores" ángulos para todos los experimentos con el descriptor geométrico.

	An	Di	Fe	Ha	Sa	Su	Co	Total
Citation-kNN	84.40	96.60	44.00	98.60	75.00	98.80	77.80	89.30
Reproducción ("mejores" ángulos)	68.89	96.60	56.00	98.55	53.57	98.80	66.67	85.32
Reproducción (ángulos "default")	73.33	98.31	52.00	97.10	46.43	96.39	66.67	84.40

Tab. 5.2: *Recall* por clase en la tarea de Leave-One-Out usando Citation-kNN sobre el descriptor geométrico. Comparación entre la versión de [15] y su reproducción, y diferencia entre usar «mejores» ángulos (los que maximicen las diferencias entre cada cuadro de la secuencia en promedio) vs los «default» (el primero de cada triángulo).

5.3. *One-Shot* con CkNN

Para cada sujeto se generan las secuencias correspondientes a las expresiones prototípicas (una secuencia por expresión, salvo se exprese lo contrario), y se itera por cada secuencia del sujeto, comparando la secuencia actual contra las generadas. Para esta comparación se utiliza Citation-kNN con distancia Hausdorff.

Para representar las secuencias, tomamos como base el descriptor geométrico usado en el paper original. Detallamos resultados de experimentos y las limitaciones de este descriptor en esta tarea particular. Realizamos distintas pruebas con el objetivo de determinar la efectividad del clasificador y confirmar que fuera el descriptor geométrico el causante de la deficiencia en la precisión de las predicciones. Consideramos landmarks espejados, agregar ruido gaussiano y añadir la secuencia "ground-truth" al conjunto a comparar.

Luego, introducimos una alternativa al descriptor geométrico: la familia de métodos de 2DLDA. Realizamos varias pruebas que se detallan a continuación. Aunque nos enfocamos en esta representación de las secuencias, en la sección 6.2 se describen otras opciones probadas superficialmente.

Para todos los experimentos en la tarea de *One-Shot* utilizamos los siguientes parámetros: $\beta = 1{,}0$, $s = 3{,}6$, $R = 3$, $C = 5$, $K = 0{,}8$.

5.3.1. *One-Shot* con el descriptor geométrico

Para el caso base, utilizando la misma representación del experimento Baseline: *Leave-One-Out* con CkNN sobre el descriptor geométrico, obtenemos en esta tarea un *accuracy* del 75.08 %. En la tabla 5.3 se puede observar el *recall* por clase. Las clases con menor *recall* son Sad y Fearful que corresponden a las clases con menor cantidad de instancias.

An	Di	Fe	Ha	Sa	Su	Total
51.11	69.49	48.00	100.00	21.43	97.59	75.08

Tab. 5.3: *Recall* por clase en la tarea de *One-Shot* usando Citation-kNN sobre el descriptor geométrico.

Realizamos algunos experimentos básicos para evaluar la calidad de las clasificaciones con el método. Si bien resulta obvio, reemplazar la secuencia generada de la expresión a clasificar por la original resulta en *accuracy* del 100 %, tanto promedio por sujeto, como *recall* perfecto por cada clase de expresión. Esto corrobora que el clasificador funciona correctamente y que el impacto en la calidad de las predicciones se debe tanto la calidad de la generación como la representación de las secuencias de cuadros sintéticos.

El descriptor geométrico es robusto a espejado de las imágenes. Como prueba, espejamos los landmarks en el experimento anterior (mismas secuencias) y nos devuelve los mismos resultados (100 %).

Para confirmar la sensibilidad a ruido del descriptor, aplicamos ruido uniforme (u) a los landmarks de las imágenes antes de normalizarlos. Las imágenes son de 128x128 pixeles y los landmarks están en ese mismo rango. Repetimos el experimento de las mismas secuencias, pero aplicando ruido a los landmarks de las secuencias en el conjunto de entrenamiento. Con $u = 1{,}0$, no hay cambios y obtenemos 100 % de *accuracy*. Con valores de $u = 2{,}0$ baja a 99.03 %, con $u = 3{,}0$ es del 81.55 %, y ya con $u = 4{,}0$ baja al 56.96 %. Esto indicaría que el descriptor geométrico no es afectado por pequeñas perturbaciones de ruido. Podemos observar también en la tabla 5.4 cómo afecta el ruido a la clasificación de cada expresión, siendo las expresiones de Happy y Surprised las más robustas a las perturbaciones con ruido uniforme.

	An	Di	Fe	Ha	Sa	Su	Total
u = 2.0	93.33	100.00	100.00	100.00	100.00	100.00	99.03
u = 3.0	60.00	71.19	68.00	100.0	53.57	98.8	81.55
u = 4.0	17.78	27.12	24.0	98.55	10.71	90.36	56.96

Tab. 5.4: *Recall* por clase cuando se le agrega ruido uniforme $[-u, u]$ a los landmarks de las imágenes de las secuencias.

A modo de comparación con el descriptor geométrico, decidimos probar la clasificación en la tarea de *One-Shot* usando Citation-kNN con las imágenes de los cuadros en escalas de grises. La distancia utilizada fue la Hausdorff entre secuencias, con distancia euclidiana entre cuadros. Obtenemos un *accuracy* de 71.52 %, que es menor a cuando se utiliza el descriptor. Sin embargo, el *recall* por clase es mejor en Disgusted y Sad, como se puede observar en la tabla 5.5.

	An	Di	Fe	Ha	Sa	Su	Total
Sin transformaciones	20.00	89.83	48.00	85.51	35.71	93.98	71.52
Descriptor geométrico	51.11	69.49	48.00	100.00	21.43	97.59	75.08

Tab. 5.5: *Recall* por clase en la tarea de *One-Shot* usando Citation-kNN sobre las imágenes sin ninguna transformación vs descriptor geométrico.

5.4. *One-Shot* con 2DLDA

Realizamos experimentos para la tarea de *One-Shot* con LDA, 2DLDA, $(2D)^2$LDA y G2DLDA. Indicamos en cada caso el *recall* general y por expresión. A modo de referencia, se presenta un cuadro comparativo de los resultados en la tabla 5.6.

A partir del primer cuadro de la secuencia a clasificar se generan las secuencias sintéticas para cada expresión. Luego, de los cuadros de la secuencia generada se obtiene una matriz de transformación con el método seleccionado (2DLDA), la cual se utiliza para proyectar los cuadros de las secuencias sintéticas y la original. Finalmente, estos vectores son pasados a Citation-kNN, para determinar a qué clase pertenece la expresión.

En la figura 5.2 se muestran distintas vistas de los vectores obtenidos por 2DLDA para cada cuadro y un sujeto particular ubicados espacialmente, reducidos con PCA a 3 dimensiones (a). Los círculos corresponden a cuadros generados mientras que las estrellas a los originales. Cada color mapea a una expresión. Notar que existe una separación apreciable entre las secuencias sintéticas de cada expresión, y aunque las secuencias originales están también separadas, no se corresponden en posición con las generadas. En (b) se muestra la distancia Hausdorff de una secuencia original Happy a una generada de la misma expresión. Esta comparación se realiza entre la secuencia original y todas las sintéticas.

	An	Di	Fe	Ha	Sa	Su	Total
LDA	13.33	28.81	72.00	27.54	17.86	33.73	30.10
2DLDA	26.67	77.97	72.00	42.03	10.71	31.33	43.37
$(2D)^2$LDA	31.11	86.44	32.00	89.86	32.14	79.52	67.96
G2DLDA	20.00	84.75	40.00	68.12	17.86	95.18	64.72

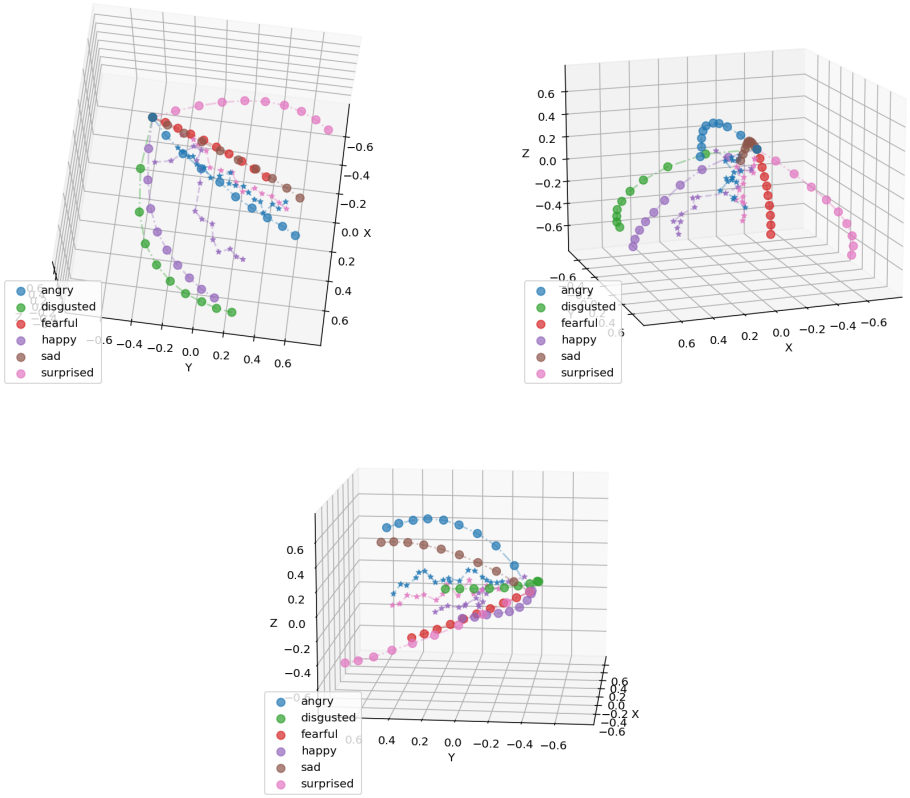
Tab. 5.6: *Recall* por clase en la tarea de *One-Shot*. Comparación entre 2DLDA y sus variantes.

La versión unidimensional, LDA, obtiene resultados pobres para la tarea de *One-Shot*. El *accuracy* obtenido es del 30.10 %. Mucho menor que con el descriptor geométrico (75.08 %). En la única expresión que supera al descriptor geométrico es en Fearful (72.00 % vs 48.00 %), pero esto ocurre porque hay una mayor proporción de instancias predichas de esa clase, como se puede observar en la matriz de confusión en la figura 5.3.

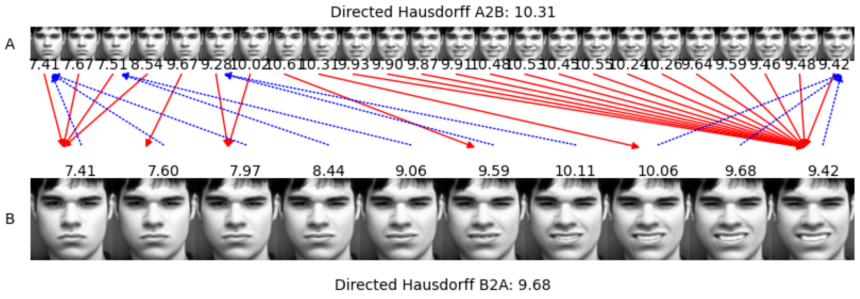
2DLDA se introduce para aprovechar la estructura bidimensional de las imágenes. Sin embargo, predecir utilizando las transformaciones de 2DLDA para la representación de las imágenes también obtiene malos resultados, aunque marginalmente mejores a usar LDA. Obtenemos 43.37 % de *accuracy*, con mejoras en el *recall* en todas las clases salvo Surprised y Sad que pasan de 17.86 % a 10.71 % y de 33.73 % a 31.33 % contra LDA respectivamente. La clase con mayor mejora es Disgusted, que pasa de 28.81 % a 77.97 %. Comparando contra el descriptor geométrico, 2DLDA obtiene mejores resultados en Disgusted y Fearful, pero con baja precisión, y por lo tanto muchos falsos positivos.

$(2D)^2$LDA opera tanto en las filas como en las columnas de la imagen, lo que se traduce en una mejora en los números. Obtenemos una *accuracy* total de 67.96 % que sigue siendo menor que la del descriptor geométrico (75.08 %). Comparando contra este, las clases en la que $(2D)^2$LDA supera en *recall* al mismo son en Sad, con 34.14 % vs 21.43 %, y en Disgusted con 86.44 % vs 69.49 %. Contra LDA y 2DLDA, hay mejoras en todas las expresiones, salvo Fearful, que baja del 72.00 % a 32.00 %, pero con una mayor precisión en la clasificación.

G2DLDA incluye un parámetro de reducción dimensional r_1, que refleja el tamaño de W (la matriz de transformación del método). Para este experimento fijamos $r_1 = 64$, pero mantuvimos el tamaño de las imágenes y los demás parámetros de Citation-kNN igual a los demás experimentos. Fijamos la norma del método como $p = 1{,}5$, $\sigma = 0{,}001$, $\epsilon = 1 \times 10^{-6}$ y el número de iteraciones a 100. Obtenemos peores resultados que con el descriptor geométrico en general y en todas las clases, salvo en Disgusted con 84.75 % de *recall*. Sin embargo, esto ocurre con una menor precisión en esa clase, habiendo mayor cantidad de predicciones para la misma. La hipótesis consiste en que agregando regularización y otras propiedades (como ortonormalización de los autovectores de la matriz W) obtendríamos mejores resultados que con 2DLDA tradicional. Y aunque esto ocurre comparado contra



(a)



(b)

Fig. 5.2: (a) Distintas vistas de los vectores obtenidos por 2DLDA para cada cuadro y un sujeto particular ubicados espacialmente, reducidos con PCA a 3 dimensiones. (b) Distancia Hausdorff entre una secuencia Happy y su contraparte sintética.

2DLDA (64.72 % de *accuracy* vs 43.37 %), no supera los números de $(2D)^2$LDA (64.72 % vs 67.96 %).

5.5. Uso de múltiples secuencias por gesto

La hipótesis de este experimento es que introducir múltiples secuencias por gesto provee de más ejemplos a Citation-kNN y resulta en una mejor clasificación de los datos. Sin embargo, al testear utilizando $(2D)^2$LDA obtenemos peores resultados, como se presenta

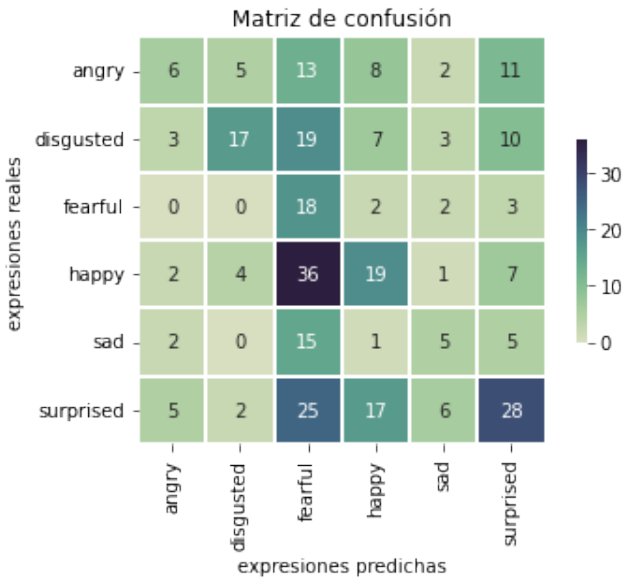


Fig. 5.3: Matriz de Confusión para el experimento *One-Shot* usando LDA y Citation-kNN.

en la tabla 5.7. Podemos observar que hay una mejora pequeña en las expresiones Angry y Sad que pasan del 31.11 % al 37.78 %, y del 32.14 % al 39.29 %, pero esto se debe a que hay una mayor proporción de secuencias clasificadas como estos gestos, indicando poca precisión, como se muestra en las matrices de confusión 5.4.

	An	Di	Fe	Ha	Sa	Su	Total
Una secuencia	31.11	86.44	32.00	89.86	32.14	79.52	67.96
Múltiples secuencias	37.78	55.93	20.00	82.61	39.29	77.11	60.52

Tab. 5.7: *Recall* por clase con $(2D)^2$LDA en la tarea de *One-Shot*. Comparación entre usar una o múltiples secuencias por clase.

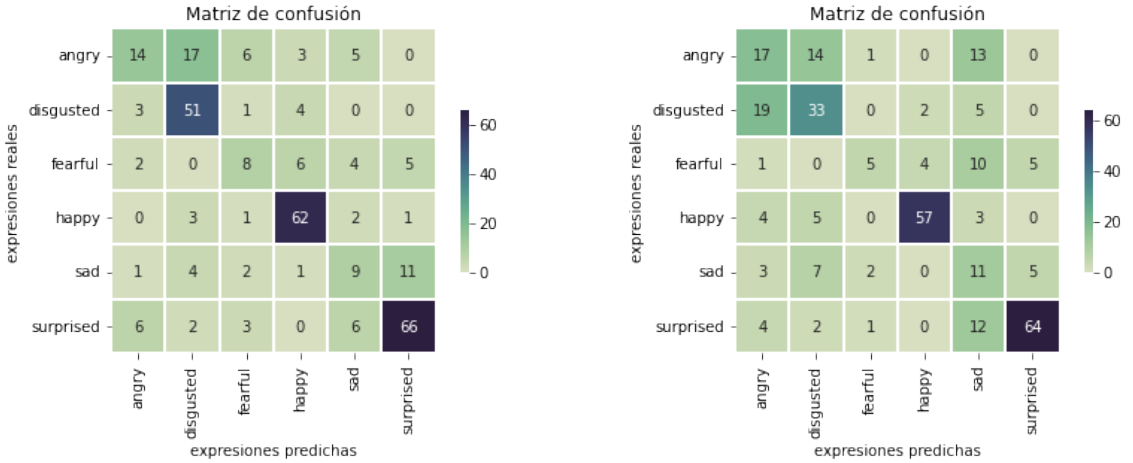
5.6. Diferencias entre StyleGAN y HyperStyle

En los experimentos utilizamos las imágenes generadas con el método de HyperStyle [12], pero ¿existe una mejora al usar este método comparado a utilizar las imágenes generadas con StyleGAN2 sin cambios en los pesos de la red generadora?

Para corroborar que efectivamente es así, medimos el error de generar las imágenes del dataset CK+ sin modificaciones. Es decir, para cada imagen en el dataset, obtenemos el vector latente y generamos una imagen correspondiente con el generador de StyleGAN2 y otra con el mismo modificado con los pesos obtenidos con el método de HyperStyle. Medimos el error entre estas imágenes generadas y sus originales, y comparamos.

Por otro lado, nos importa saber cómo impacta esta elección en la clasificación, así que también realizamos un experimento contrastando ambas bases de comparación. Nuestra hipótesis es que mejorar la calidad de las imágenes generadas (en este caso teniendo un mayor parecido al sujeto), mejoraría la clasificación de las secuencias.

Tomando las imágenes neutras de todos los sujetos y comparándolas con sus versiones



(a) Una secuencia (b) Múltiples secuencias

Fig. 5.4: Matrices de Confusión para el experimento *One-Shot* usando (2D)^{2}LDA y Citation-kNN. Comparación entre usar una o múltiples secuencias por clase.

generadas tanto usando StyleGAN2 y HyperStyle, podemos apreciar que el error es menor utilizando estas últimas. En la tabla 5.8 se pueden observar los distintos valores de error comparados. Esto indicaría que las imágenes generadas con HyperStyle son más similares a las originales que las generadas usando StyleGAN2 sin modificaciones a los pesos del generador.

En la figura 5.5 presentamos algunos ejemplos de cómo se expresan estas diferencias al utilizar el generador sin modificaciones. Incluyen tanto modificaciones a pose, iluminación y contraste como cambios en la expresión, mirada y pequeños elementos de la identidad del sujeto como serían el pelo o aros.

	ME	MAE	MSE	RMSE
StyleGAN2	-0.004	0.070	0.010	0.101
HyperStyle	0.002	0.035	0.003	0.055

Tab. 5.8: Errores promedio entre imágenes originales y las generadas con StyleGAN2 sin modificaciones en los pesos, e imágenes originales y las generadas usando HyperStyle. Se presentan error medio (ME), error absoluto medio (MAE), error cuadrático medio (MSE) y raíz cuadrada del error cuadrático medio (RMSE).

En cuanto a la calidad de las predicciones, en general mejora utilizando el método de HyperStyle. El *accuracy* general pasa de 63.52 % a 67.96 %. Las expresiones que obtienen peores resultados comparados con StyleGAN2 sin modificaciones son Happy y Surprised, con cambios de 91.30 % a 89.86 % y 86.59 % a 79.52 % respectivamente. Es interesante que en estos gestos no sólo empeora el *recall*, sino que también la precisión, como puede observarse en la matriz de confusión 5.4. En general, las mejoras no son muy grandes, salvo para Disgusted que pasa de 61.02 % a 86.44 %, a costo de precisión. Los valores por expresión se muestran en la tabla 5.9.

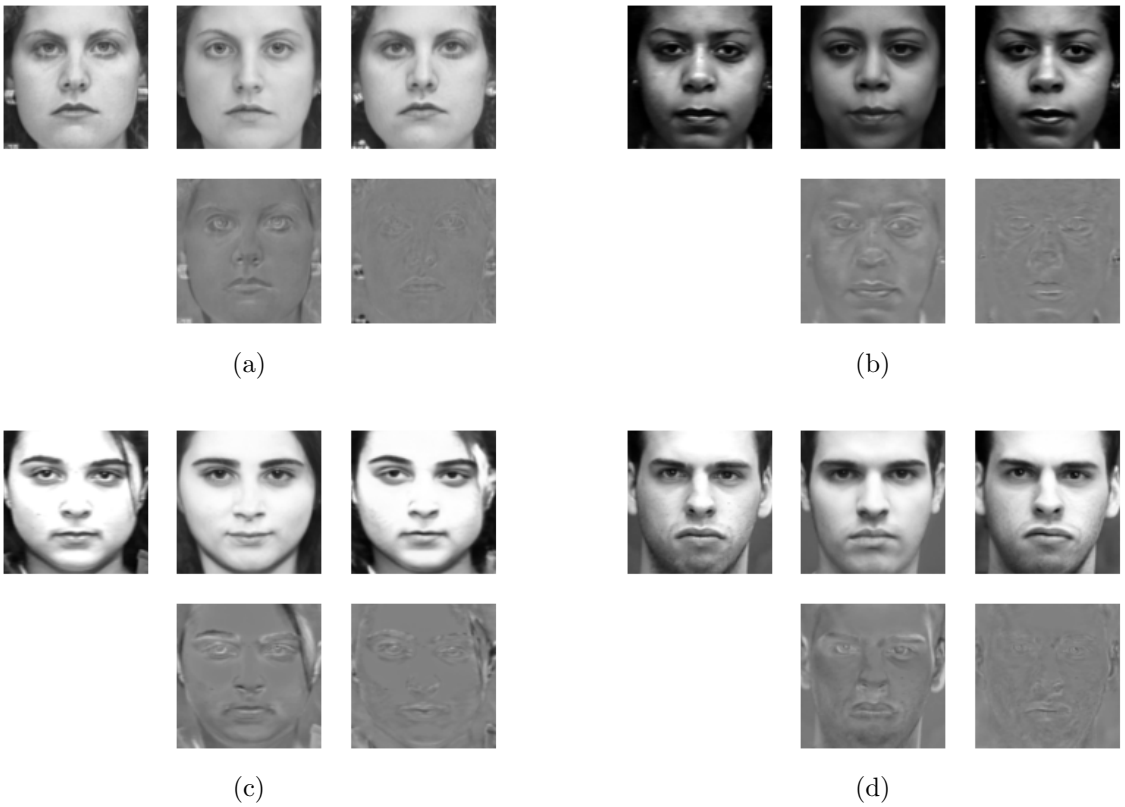


Fig. 5.5: Ejemplos de sujetos con caras neutras generadas. De izquierda a derecha: imagen original, imagen generada con StyleGAN2 sin modificar los pesos, imagen generada usando el método de HyperStyle. La segunda fila de cada ejemplo muestra la diferencia contra la imagen original.

	An	Di	Fe	Ha	Sa	Su	Total
StyleGAN2	25.00	61.02	28.00	91.30	25.00	86.59	63.52
HyperStyle	31.11	86.44	32.00	89.86	32.14	79.52	67.96

Tab. 5.9: *Recall* por clase con $(2D)^2$LDA en la tarea de *One-Shot*. Comparación entre usar las imágenes generadas por StyleGAN2 sin modificaciones a los pesos versus las obtenidas con el método de HyperStyle.

5.7. Clasificación usando embeddings únicamente

En este experimento, evaluamos la utilización de los embeddings de StyleGAN2 únicamente en la clasificación. Esto es, obtener los vectores latentes de las imágenes originales y las generadas, y pasar estos como entrada a Citation-kNN en vez de las imágenes transformadas.

Utilizando únicamente los embeddings y Citation-kNN, obtenemos un 86.41 % de *accuracy*, superando al baseline del descriptor geométrico con 75.08 %. El método de los embeddings obtiene grandes mejoras en todas las expresiones, salvo Happy y Surprised en donde pasan de 100 % a 89.86 % y de 97.59 % a 92.77 %. En la tabla 5.10 se presenta esta comparación.

	An	Di	Fe	Ha	Sa	Su	Total
Embeddings	75.56	94.92	76.00	89.86	67.86	92.77	86.41
Descriptor geométrico	51.11	69.49	48.00	100.00	21.43	97.59	75.08

Tab. 5.10: *Recall* por clase usando los embeddings y Citation-kNN en la tarea de *One-Shot*. Comparación contra usar el descriptor geométrico

5.8. Modificación de la rotación de expresión Angry

Notamos que los vectores latentes obtenidos para las expresiones llevan consigo los sesgos del dataset del cual fueron extraídos. Así, los cuadros generados de la expresión Angry modifican también la rotación de la cara del sujeto. Decidimos intentar reducir ese artefacto, restando un vector latente correspondiente a la rotación al vector de la expresión Angry. El vector de rotación es obtenido de la misma manera que los de las expresiones, pero usando el dataset BIWI [66].

Con esta modificación, en el experimento de *One-Shot* con el descriptor geométrico, obtenemos 79,28 % de *accuracy*, superando al experimento sin modificaciones. En la tabla 5.11 mostramos el *recall* por clase. La mejora en promedio se debe a un aumento en el *recall* para las expresiones Disgusted, Sad y (en menor medida) Fearful. Las expresiones Angry y Surprised sufren bajas en sus valores.

	An	Di	Fe	Ha	Sa	Su	Total
Corrigiendo rotación de Angry	48.89	98.31	52.00	100.00	50.00	83.13	79.29
Sin modificaciones	51.11	69.49	48.00	100.00	21.43	97.59	75.08

Tab. 5.11: *Recall* por clase usando el descriptor geométrico y Citation-kNN en la tarea de *One-Shot*. Comparación entre usar las caras Angry generadas originalmente vs las corregidas por rotación.

5.9. Errores de clasificación comunes

En la figura 5.6 se presentan las últimas imágenes en las secuencias en las que todos los métodos fallaron en la clasificación para distintas combinaciones de sujetos y expresiones. Hay distintas razones posibles por las que estas secuencias fueron mal clasificadas.

Por lo general, notamos que las expresiones son muy sutiles, ya sea en las imágenes originales, las generadas o ambas, como se puede observar en los sujetos 37, 94, 101 y 103. Expresiones con gestos de poca intensidad son más difíciles de clasificar. Mostramos ejemplos de expresiones sutiles en la figura 5.7. En el sujeto 118, las expresiones faciales originales son muy exageradas, mientras que las generadas no llegan a ese nivel de intensidad. Existen ciertos sujetos para los cuales la generación de expresiones particulares no es exitosa y la intensidad de los gestos generados es muy baja.

El sujeto 117 tiene el pelo distinto en la imagen original que en la generada. A veces ocurre que las imágenes neutras, de las que parte una secuencia original y a partir de la cual se generan las nuevas secuencias, tienen detalles distintos a los de la imagen final. De la misma forma, pueden diferir en escala o rotación. También es posible que la imagen generada difiera lo suficiente debido a que el embedding obtenido para la expresión no aísla perfectamente la información de la expresión facial y, por lo tanto, modifica otros

factores de la imagen, como podría ser pose, apertura y cierre de los ojos, detalles del pelo y demás. Ejemplos de este comportamiento pueden observarse en la figura 5.8.

La imagen del sujeto 9 tiene mucho brillo, al punto de que no se distinguen ciertos rasgos faciales. La expresión facial del gesto generado es también muy sutil.

La secuencia del sujeto 94 está mal etiquetada. Probablemente debería ser Disgusted en vez de Surprised. Es la única secuencia mal etiquetada del dataset.

Algunos cuadros de las secuencias originales difieren de las generadas en rotación, escala u otras transformaciones, como se observa en la figura 5.9. Esto puede deberse a un mal procesamiento de los datos en estos ejemplos.

(a) Sujeto 9 Angry

(b) Sujeto 37 Angry

(c) Sujeto 94 Angry

(d) Sujeto 94 Surprised

(e) Sujeto 101 Angry

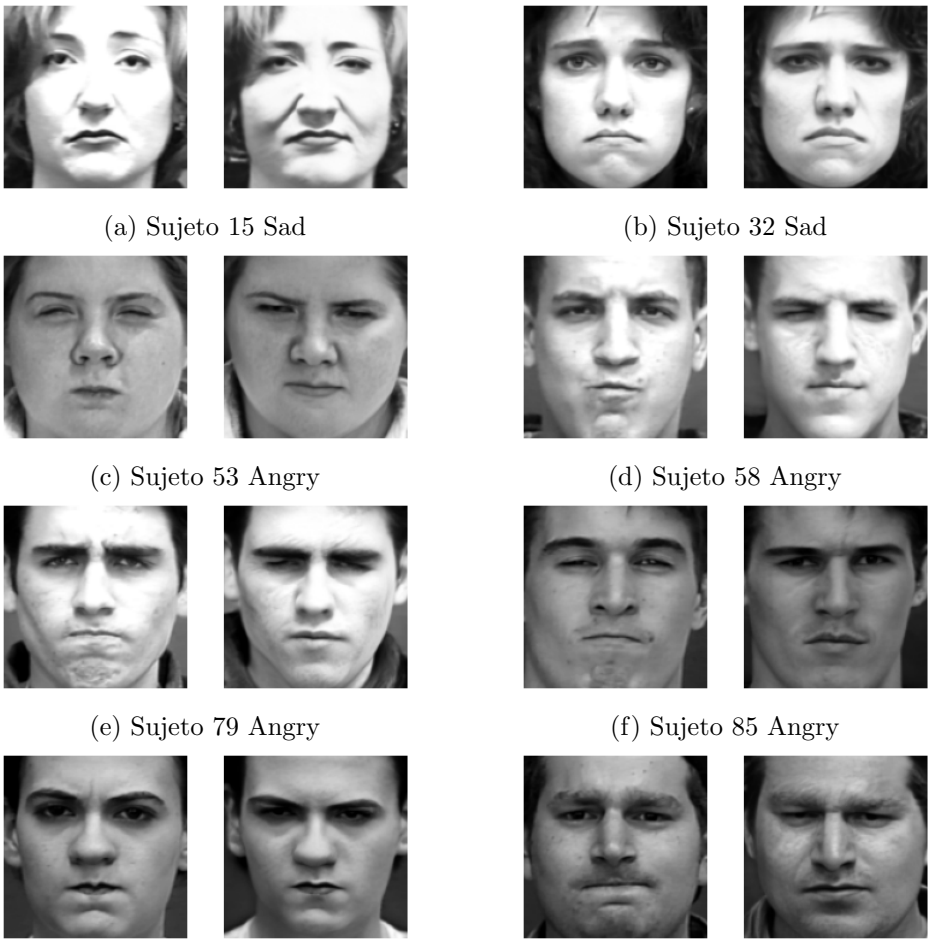
(f) Sujeto 103 Sad

(g) Sujeto 117 Sad

(h) Sujeto 118 Angry

(i) Sujeto 118 Fearful

Fig. 5.6: Todos los métodos fallaron en clasificar estos sujetos y expresiones. A la izquierda las imágenes originales y a la derecha las generadas.



(a) Sujeto 15 Sad (b) Sujeto 32 Sad

(c) Sujeto 53 Angry (d) Sujeto 58 Angry

(e) Sujeto 79 Angry (f) Sujeto 85 Angry

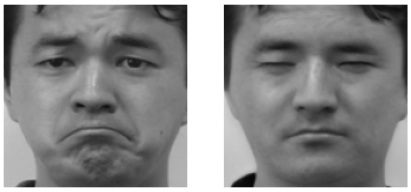
(g) Sujeto 95 Angry (h) Sujeto 99 Angry

Fig. 5.7: Ejemplos de imágenes frecuentemente mal clasificadas: Expresiones originales (lado izquierdo) muy sutiles. De la misma forma, las generadas (lado derecho) son de baja intensidad.

(a) Sujeto 3 Angry

(b) Sujeto 12 Angry

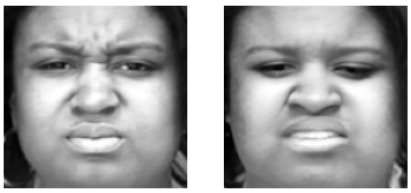
(c) Sujeto 46 Sad



(d) Sujeto 119 Sad

Fig. 5.8: Ejemplos de imágenes frecuentemente mal clasificadas: Imágenes generadas (lado derecho) con cambios que no corresponden a la expresión, como podrían ser pose u ojos.

(a) Sujeto 2 Sad



(b) Sujeto 90 Disgusted

(c) Sujeto 95 Angry

(d) Sujeto 100 Happy

Fig. 5.9: Ejemplos de imágenes frecuentemente mal clasificadas: Imágenes originales (lado izquierdo) difieren a las generadas (lado derecho) en rotación, escala, etc.

6. EXPERIMENTOS ADICIONALES

Antes de llegar a la solución presentada, exploramos distintos métodos para la generación de cuadros sintéticos y otras representaciones de imágenes faciales. En esta sección presentamos un generador sintético inicial, GANimation, y otras representaciones de imágenes, los *Active Appeareance Models*, *Active Shape Models*, y *Shape-Free Texture Models*. Presentamos estos métodos por completitud, pero no ahondamos en los resultados de los experimentos realizados con los mismos.

6.1. Generador sintético inicial: GANimation

GANimation [20] es un método de generación de animación facial desde una sóla imagen como input. La contribución de este paper es utilizar el FACS[1] para condicionar la red, permitiendo controlar la magnitud de activación de cada *Action Unit*(AU) de una manera continua, y combinarlas para la formulación de nuevas expresiones. Además, su método es no supervisado, si bien se necesitan las imágenes anotadas con las AU activadas como entrada al entrenar. Adicionalmente, integran varios mecanismos para agregar robustez y precisión a la generación.

Inicialmente decidimos usar esta red (entrenada sobre Aff-Wild2 [67][2]) para la generación de secuencias. El hecho de que se use un vector de AUs para determinar los gestos de las imágenes generadas tiene tanto ventajas como desventajas. La principal ventaja es la edición de cualquier AU individual, pudiendo generar cualquier tipo de expresión facial. Sin embargo, esto implicaba también un proceso de búsqueda para encontrar las AUs correspondientes a cada expresión. Por otro lado, aunque en teoría se pudiera generar todo tipo de expresiones, en la práctica modificar varias AUs a la vez podía devenir en artefactos, debido a que las AUs no son independientes entre sí. Otra ventaja es que las modificaciones a la imagen original sólo ocurren en las secciones de la cara relevantes a la AU activada, debido a que la red utiliza una máscara de atención.

Finalmente decidimos utilizar StyleGAN2, debido a que la calidad de las imágenes generadas por GANimation era inferior y por lo tanto daba peores resultados en los experimentos.

6.2. Otras representaciones de imágenes

Inicialmente para representar las imágenes no utilizamos 2DLDA. Los *Active Appeareance Models* [76] (AAM), introducidos por Edwards, Cootes y Taylor, son un método iterativo para encontrar correspondencias entre modelos estadísticos de la "apariencia" de imágenes.

En nuestro contexto, nos interesa medir la distancia entre imágenes de expresiones faciales. Por lo tanto, hacíamos uso de los modelos y no del método iterativo para encontrar correspondencias entre ellos.

Entonces, cuando hablamos de un *Active Appearance Model*, nos referimos a un modelo que combina tanto información de la forma de la imagen facial como de la textura de la

[1] Facial Action Coding System.

[2] Los autores insisten en que citemos todos estos trabajos [68] [69] [70] [71] [72] [73] [74] [75].

misma, caracterizados de una manera particular. A partir de esto, obtenemos un vector de parámetros del mismo para cada imagen, lo cual nos permite medir distancias en ese espacio paramétrico.

Decidimos no utilizar esta representación por la complejidad de la explicación del pipeline y porque no daba resultados muy favorables. Sin embargo, dejamos en esta sección los detalles del método por completitud.

6.2.1. Modelo de Forma Facial: *Active Shape Models* (ASM)

Los *Active Shape Models* [77] (ASM) son modelos estadísticos de la forma de un objeto particular en imágenes, basados en *Active Contour Models* [78]. Tales objetos son representados como un conjunto de puntos que determinan la forma o estructura del mismo. En nuestro caso, estos puntos son un conjunto fijo de landmarks faciales, tanto internos como del contorno de la cara. Denominamos tal conjunto como *forma facial*, o simplemente la *forma de la cara* del sujeto correspondiente.

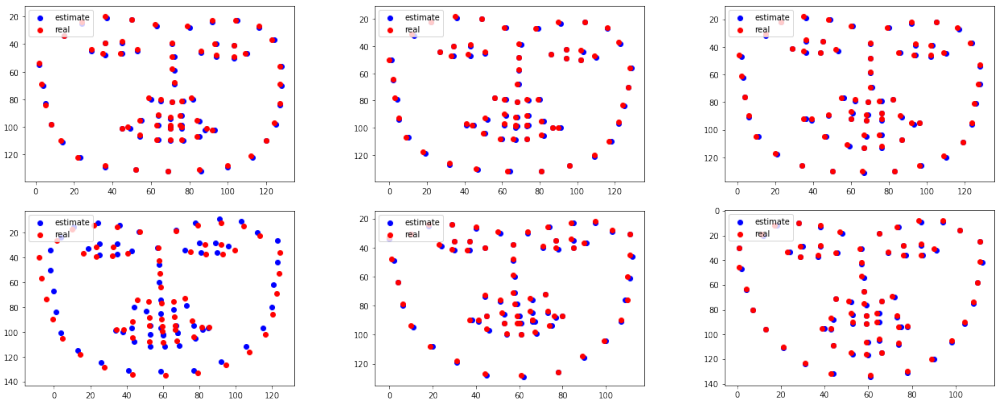


Fig. 6.1: Ejemplos de comparaciones entre landmarks originales vs estimados del ASM. En rojo se marcan los landmarks originales y en azul los que resultan de la estimación del modelo, reconstruída a partir de cada vector de parámetros b.

Las *formas faciales* del set de entrenamiento son alineadas a través del algoritmo de Procrustes. A partir de estadísticas de las posiciones de los puntos, se genera un modelo de la distribución de los mismos (*Point Distribution Model*). Este modelo contiene la distribución promedio de las posiciones de los puntos y parámetros que determinan los principales modos de variación del dataset de entrenamiento. Luego, una nueva *forma facial* puede describirse en términos de los parámetros que minimicen la distancia entre esta y formas reconstruídas a partir de estos modelos. Un ejemplo de landmarks estimados para distintas formas faciales puede observarse en la figura 6.1.

En términos de la implementación, el modelo puede obtenerse fácilmente utilizando PCA sobre el conjunto de entrenamiento de formas faciales alineadas. Así, puede definirse a partir de la forma promedio, y los autovectores (y los autovalores[3]) de la matriz de covarianza obtenidos.

Cualquier landmark del dominio del modelo puede ser expresado como la suma del promedio de la posición del mismo y una combinación lineal de los autovectores.

[3] Guardamos estos para facilitar el proceso de *fitting* de las formas faciales como se describe más adelante.

Luego, una forma facial x cualquiera del dataset puede ser aproximada a través de la suma de la forma promedio $\bar{x}$ y una suma pesada por los desvíos obtenidos por los primeros t modos de variación (autovectores obtenidos por PCA).

$$x = \bar{x} + P\mathbf{b} \tag{6.1}$$

Siendo P la matriz de los primeros t autovectores y $\mathbf{b}$ un vector de pesos. Este vector $\mathbf{b}$ corresponde a los parámetros de nuestro modelo.

La varianza de cada componente de $\mathbf{b}$, $\mathbf{b}_k$, sobre el dataset de entrenamiento corresponde al autovector λ_k calculado anteriormente con PCA. Por lo tanto, se proponen límites para cada $\mathbf{b}_k$ de la siguiente manera:

$$-3\sqrt{\lambda_k} \leq \mathbf{b}_k \leq 3\sqrt{\lambda_k} \tag{6.2}$$

Los cuales usamos también nosotros.

Para obtener los parámetros que mejor aproximen una *forma facial* para un modelo entrenado se utiliza un método iterativo. La idea general es que cada landmark individual es perturbado localmente hacia una mejor posición. Estas deformaciones son transformadas al espacio paramétrico del modelo, aplicando restricciones a los parámetros obtenidos para que se mantengan en el dominio de formas posibles de representar. El proceso se repite hasta que no haya cambios apreciables[4]. Un ejemplo de este proceso puede observarse en la figura 6.2.

Si bien el paper describe varios métodos para encontrar los desplazamientos de los puntos que podrían hacerse cada iteración, una implementación "*naïve*" que simplemente los mueve en la dirección de los de la forma a ajustarse pareciera funcionar en la práctica, por lo que decidimos realizar estas perturbaciones de esa manera.

En cada iteración, se obtiene una aproximación de la *forma facial* a partir del vector de parámetros b y el modelo entrenado, con la equación 6.1. Luego, se alínea a través de Procrustes esta aproximación a la *forma facial* objetivo. Acto siguiente, conseguimos el desplazamiento:

$$d\mathbf{x} = x - \tilde{x}$$

Y lo normalizamos. A partir de $d\mathbf{x}$, podemos obtener el desplazamiento en el espacio paramétrico del modelo ($d\mathbf{b}$) de la siguiente manera:

$$d\mathbf{b} = P^T d\mathbf{x}$$

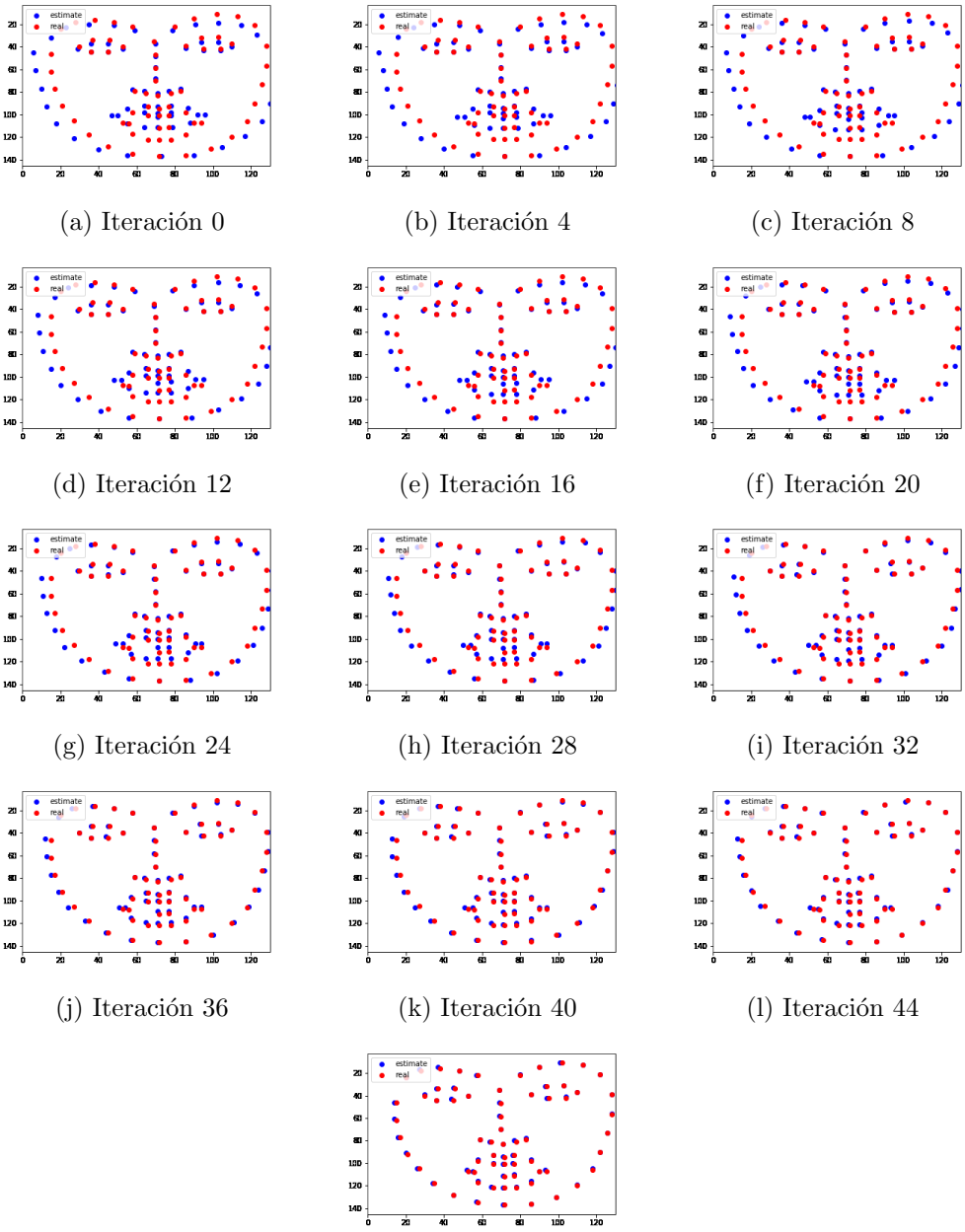
Y con este, se puede actualizar el vector de parámetros:

$$\tilde{\mathbf{b}} = \mathbf{b} + W_b \; d\mathbf{b}$$

donde W_b es una matriz diagonal de pesos que indican la magnitud de desplazamiento para cada parámetro. Por simplicidad, fijamos esta matriz en la identidad.

Finalmente, para garantizar que el vector de parámetros corresponde a una forma consistente con en el modelo entrenado, se fijan límites para cada $\mathbf{b}_k$, como se detalló en la ecuación 6.2

[4] En términos implementativos, elegimos que el criterio de corte sea que la suma absoluta de las diferencias entre formas sea menor a un ϵ, o la distancia (norma 2) entre el b de la iteración actual y el b de la anterior sea menor a ϵ. También decidimos cortar a las 100 iteraciones si no hay convergencia.



(a) Iteración 0 (b) Iteración 4 (c) Iteración 8

(d) Iteración 12 (e) Iteración 16 (f) Iteración 20

(g) Iteración 24 (h) Iteración 28 (i) Iteración 32

(j) Iteración 36 (k) Iteración 40 (l) Iteración 44

(m) Iteración 48

Fig. 6.2: Iteraciones de la estimación de una *forma facial* a partir de un modelo de ASM. En rojo se marcan los landmarks originales y en azul los que resultan de la estimación del modelo, reconstruída a partir de cada vector de parámetros **b**.

6.2.2. Modelo de Textura Facial (sin Forma)

La idea general de los *Shape-Free Texture Models* (SFTMs) [79] es la misma que la de los ASMs, sólo que esta vez siendo aplicada a la textura de la imagen facial, independiente de la *forma facial*.

Para lograr esta última condición, las imágenes faciales son deformadas a partir de un

algoritmo de *warping* hacia la cara promedio del dataset de entrenamiento. En nuestro caso, decidimos realizar este "warp" utilizando los landmarks faciales. Obtenemos triángulos a partir de estos con Delaunay, y realizamos transformaciones afines para cada uno de los polígonos con su correspondiente en la cara promedio. Pueden observarse ejemplos de este proceso en la figura 6.3.

La imagen obtenida es desglosada en un vector unidimensional y normalizado[5]. De todos los vectores obtenidos de las imágenes de entrenamiento, se calcula la textura sin forma promedio. Para conseguir el modelo, se aplica PCA de la misma forma que para los ASM, sobre vectores de textura sin forma restándoles la textura sin forma promedio.

El vector de parámetros b para una aproximación de textura sin forma de una imagen facial se obtiene de una manera similar. Como antes, se realiza un *warp* de la imagen objetivo a la forma promedio. Luego, se desglosa a un vector unidimensional, el cual es normalizado y al que se le resta la textura sin forma promedio. Llamamos a este vector normalizado $\tilde{\mathbf{x}}$ y $\mathbf{b}$ se calcula como:

$$b = P^T \, \tilde{\mathbf{x}} \tag{6.3}$$

Se limita a $\mathbf{b}$ para que esté en el rango $[-3\sqrt{\lambda_k},\ 3\sqrt{\lambda_k}]$, como se muestra en la ecuación 6.2 para ASMs. A diferencia de los ASMs, este proceso no es iterativo y realizar esto una única vez es suficiente en la práctica.

Un nuevo vector de textura sin forma a partir de un vector de parámetros $\mathbf{b}$ se obtiene con la siguiente fórmula:

$$x = \bar{\mathbf{X}} + P\mathbf{b}$$

Siendo $\bar{\mathbf{X}}$ el vector de textura sin forma promedio.

Si se quisiera reconstruir la imagen facial a partir de esta textura, se tendría que hacer el proceso de *warp* inverso. Es decir, desde la forma promedio a la forma original de la cara a aproximar.

6.2.3. Modelo Combinado: Active Appearance Models (AAM)

Los SFTMs son ideados para ser combinados con el modelo de landmarks (ASM), con la intención de agregar información de la textura a los modelos. Estos nuevos modelos combinados se denominan *Active Appearance Models* (AAMs).

Un "modelo combinado" en este caso se refiere a un modelo que usa para su entrenamiento vectores que son la concatenación de los vectores de parámetros normalizados que se obtienen de un modelo de forma (ASM) y uno de textura (SFTM).

Otra vez, para obtener este modelo, se aplica PCA sobre los vectores del conjunto de entrenamiento. Así, cualquier otro vector nuevo puede aproximarse multiplicando la matriz de los primeros autovectores por el vector a aproximar, de una manera similar a la fórmula 6.3, y tomando límites como en 6.2.

Para obtener los vectores de los modelos de forma y textura a partir de un vector b del modelo combinado, realizamos la operación inversa:

$$x = \mathbf{P}b$$

[5] En este caso, dividiendo por 255.

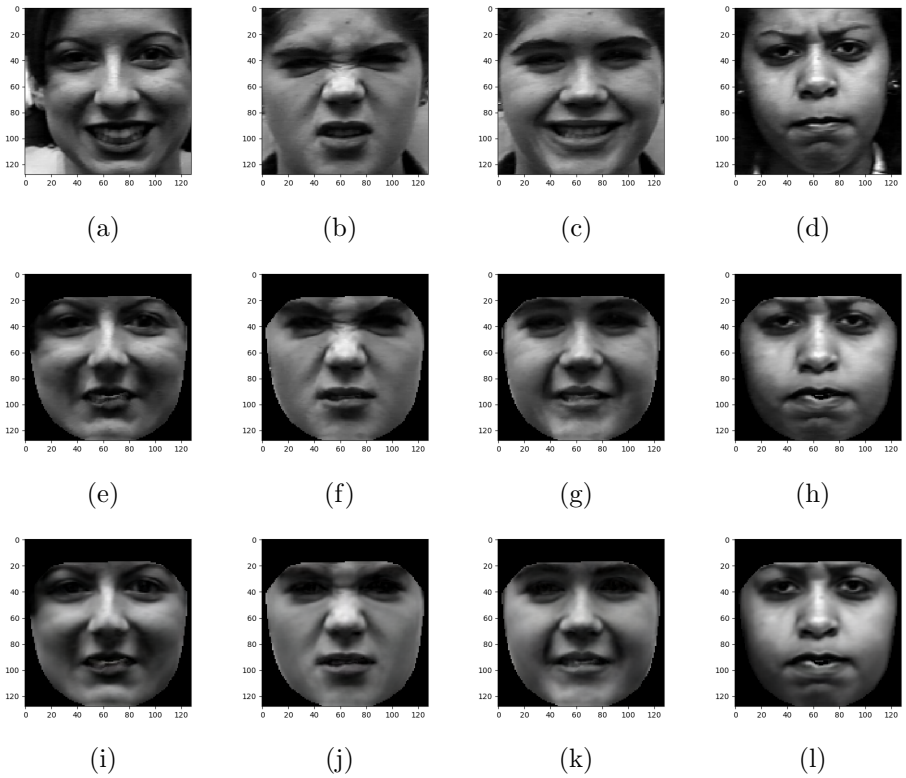


Fig. 6.3: SFTMs: En la primera hilera (a, b, c, d) se muestran las imágenes faciales originales. La segunda fila (e, f, g, h) corresponde a las imágenes resultantes luego de hacer el *warp* a la *forma facial* promedio. En la última fila (i, j, k, l) se muestran las estimaciones reconstruidas del SFTM.

Y dividimos en las partes correspondientes el vector resultante para cada modelo (ASM y SFTM), siendo que el vector resultante era la concatenación de ambos vectores de parámetros. En la figura 6.4 podemos ver un ejemplo de este proceso.

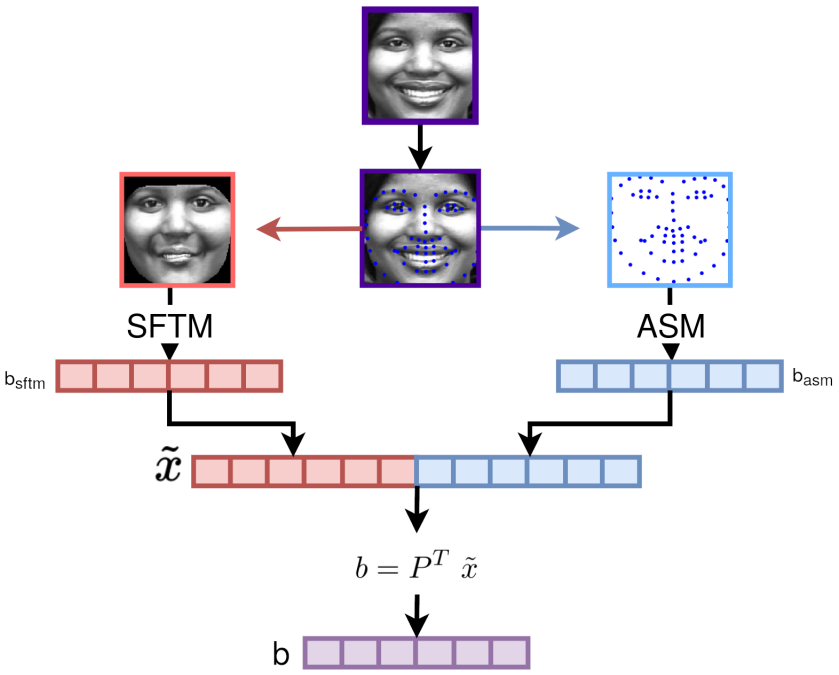


Fig. 6.4: Active Appearance Model: Diagrama de la construcción de un vector de parámetros **b** de una imagen particular. A partir de la imagen facial se obtienen los landmarks y la textura facial sin forma, que son estimados a partir de modelos de ASM y SFTM respectivamente. Los $\mathbf{b}_{sftm}$ y $\mathbf{b}_{asm}$ se concatenan para formar el vector $\tilde{\mathbf{x}}$. Luego, a través de la matriz de autovectores P obtenida del modelo combinado de AAM entrenado, se transforma $\tilde{\mathbf{x}}$ para obtener **b**.

7. CONCLUSIONES

7.1. Discusión

En la búsqueda de una representación de las imágenes de expresiones faciales optamos por 2DLDA y sus variantes, porque constituyen un método que por definición intenta maximizar la distancia interclase y minimizar la intraclase. Sin embargo, no logramos obtener los resultados deseados y, comparados con el descriptor geométrico, las métricas no logran superar el *baseline*. Compilamos los resultados para la tarea de *One-Shot* en la tabla 7.1.

Un subconjunto de expresiones resultó más sencillo de clasificar que las otras. Sad y Fearful por lo general obtienen los peores resultados. Esto puede deberse a que hay una menor cantidad de secuencias con esas expresiones en el dataset. Pero también, Angry, que posee más secuencias que estas dos expresiones obtiene malos resultados consistentemente. Es posible que las expresiones de Angry, Sad, Fearful y Disgusted sean visualmente muy similares, tanto en el dataset original como el de comparación generado.

Hay una pequeña mejora entre usar las imágenes de StyleGAN2 sin modificar los pesos en contraste con usar el método de HyperStyle. Con HyperStyle obtenemos imágenes más similares a las del dataset original.

En un enfoque inicial, utilizamos otra GAN (GANimation [20], ver sección 6), que generaba imágenes menos realistas y con mayores artefactos. No registramos los valores de los experimentos con esa red en este trabajo, pero eran peores que los obtenidos con StyleGAN2. Esto implica que para obtener mejores resultados, importa la similitud entre las imágenes generadas y las reales. Esto puede inferirse también del experimento de la corrección de la rotación de las imágenes de la expresión Angry. Mejorar la calidad de las imágenes generadas y su similitud a las originales, impacta positivamente en los resultados.

Sin embargo, no sólo importa que haya invarianza del sujeto en la generación, sino que la representación de los cuadros esté en un espacio que generalice bien las expresiones. Esto es porque sólo elegimos una expresión facial por "emoción", mientras que en la realidad distintos sujetos expresan la misma "emoción" con distintas expresiones faciales.

Pensamos que el error se podía mitigar introduciendo una mayor cantidad de secuencias por expresión para comparar. Pero sin un buen método que encuentre expresiones representativas y una representación de los cuadros que logre una buena generalización de las expresiones, garantizando separación entre las mismas, es difícil mejorar los resultados obtenidos.

No es sorpresa que usar los embeddings de la red directamente dé muy buenos resultados. Sin embargo, es difícil explicarlos. Nuestro objetivo era encontrar una representación que tenga una explicación matemática sencilla, a la vez que efectiva. Si bien 2DLDA no obtuvo los resultados esperados, es una aproximación interesante debido a sus propiedades intrínsecas de separación por clase.

Existen otras variantes de 2DLDA que buscan subsanar algunos de los defectos del método. G2DLDA introduce regularización y ortonormalización de los vectores que conforman la matriz de transformación. A pesar de eso, el método es muy lento para clasificación en tiempo real, y tampoco obtuvo los resultados esperados.

	An	Di	Fe	Ha	Sa	Su	Total
Descriptor geométrico	51.11	69.49	48.00	**100.00**	21.43	**97.59**	75.08
LDA	13.33	28.81	72.00	27.54	17.86	33.73	30.10
2DLDA	26.67	77.97	72.00	42.03	10.71	31.33	43.37
$(2D)^2$LDA	31.11	86.44	32.00	89.86	32.14	79.52	67.96
G2DLDA	20.00	84.75	40.00	68.12	17.86	95.18	64.72
Embeddings	**75.56**	**94.92**	**76.00**	89.86	**67.86**	92.77	**86.41**

Tab. 7.1: Recall por clase en la tarea de *One-Shot*. Comparación entre todos los métodos utilizados.

7.2. Trabajo Futuro

Logramos definir un método para la clasificación de secuencias de expresiones faciales en la tarea de *One-Shot*. Aunque los resultados pueden ser mejorables, es importante notar que los módulos del pipeline pueden ser intercambiados independientemente.

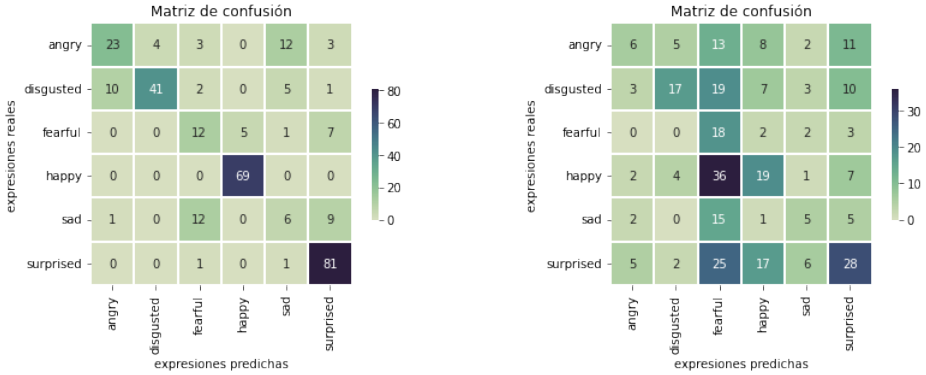
Si bien hay varias ramas posibles de exploración, el enfoque en un trabajo futuro podría darse en la generación de expresiones representativas. Sobre todo en la selección de los vectores latentes para cada expresión. De la misma forma, resulta de interés explorar en mayor profundidad la utilización de múltiples secuencias por "emoción". También, la búsqueda de un modelo de representación de los cuadros que generalice correctamente estas expresiones.

Además, es importante lograr aislar y eliminar los artefactos en la generación de cuadros lo más posible. En particular, en las variaciones de alto nivel como iluminación, pose, etc., más que píxeles aislados que podrían ser considerados ruido.

El método es efectivo en la clasificación de un subconjunto de expresiones. Obtener un dataset más balanceado y con una mayor cantidad de secuencias para las expresiones con peores resultados también podría ayudar a obtener una mayor calidad en las predicciones.

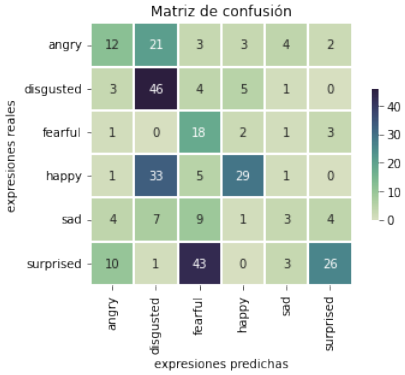
8. ANEXO

8.1. Gráficos adicionales

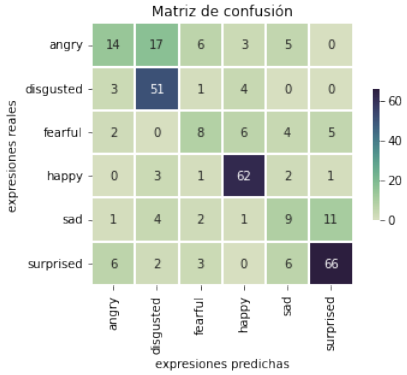


(a) Descriptor geométrico

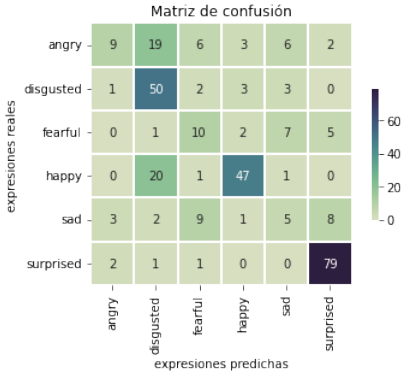
(b) LDA



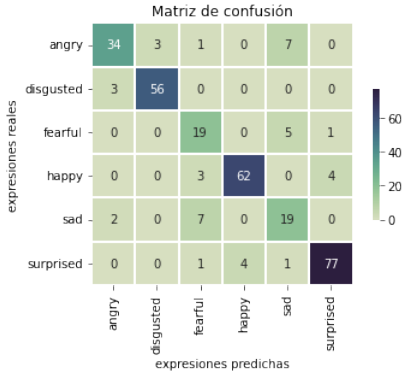
(c) 2DLDA



(d) $(2D)^2$LDA

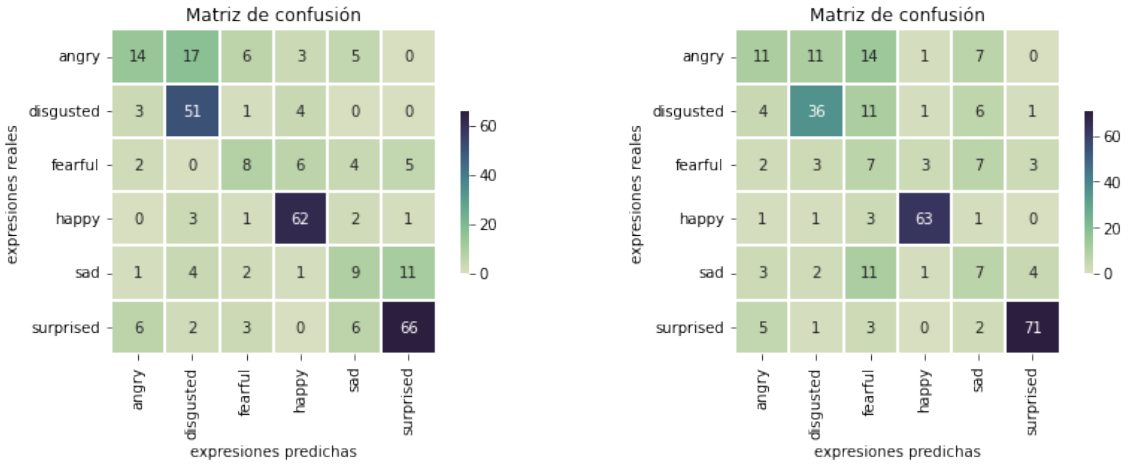


(e) G2DLDA



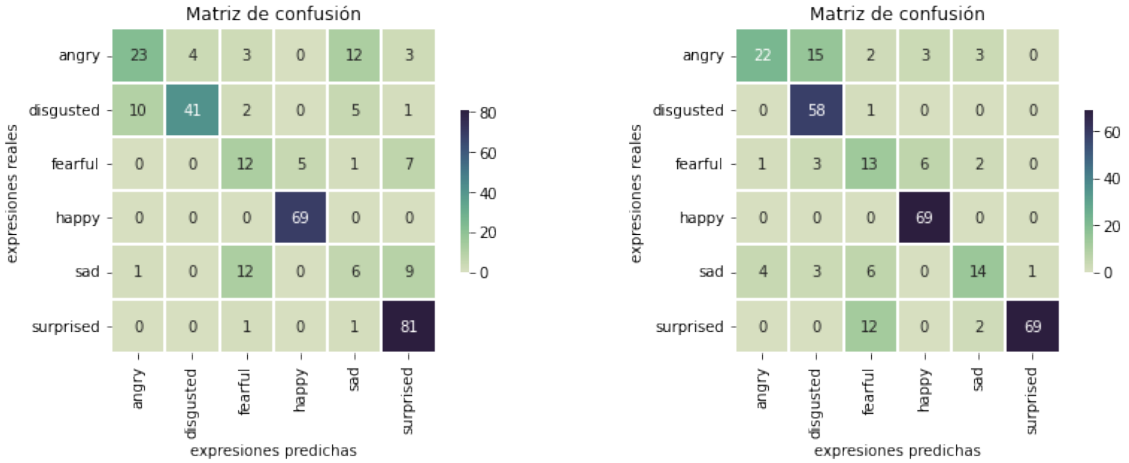
(f) Embeddings

Fig. 8.1: Matrices de confusion para el experimento One-Shot con Citation-kNN y todos los métodos de representación propuestos.



(a) HyperStyle (b) StyleGAN

Fig. 8.2: Matrices de Confusión para el experimento One-Shot usando $(2D)^2$LDA y Citation-kNN. Comparación entre usar las imágenes generadas por StyleGAN2 sin modificaciones a los pesos versus las obtenidas con el método de HyperStyle.

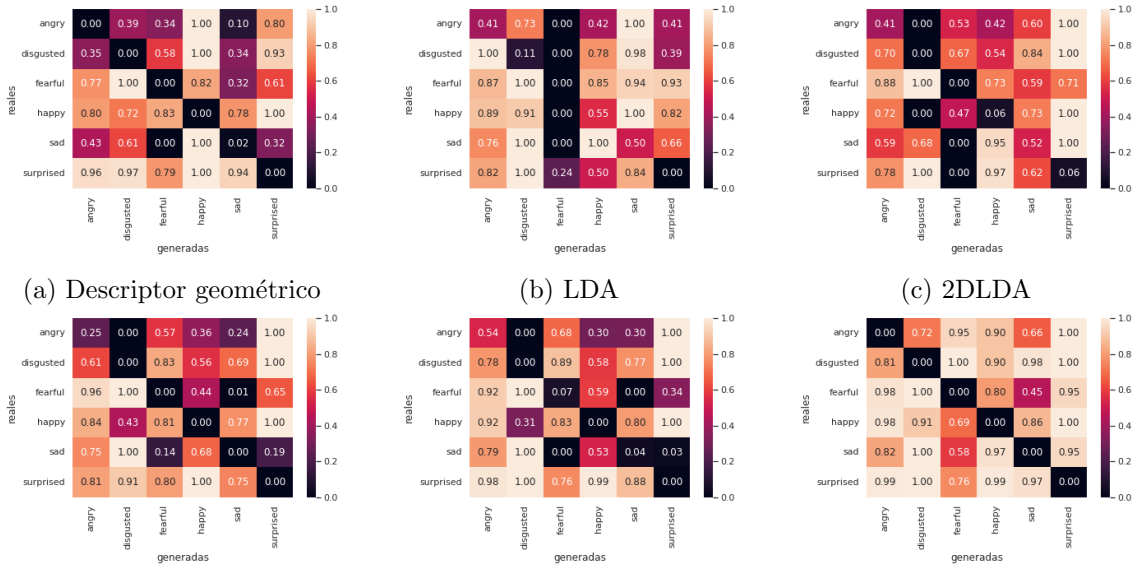


(a) Sin modificaciones (b) Corrigiendo rotación de **An**gry

Fig. 8.3: Matrices de confusión para el experimento One-Shot usando descriptor geométrico y Citation-kNN. Comparación entre modificar la rotación de los cuadros con la expresión Angry vs no hacerlo.

	An	Di	Fe	Ha	Sa	Su	Total
Descriptor geométrico	58.23	78.85	43.64	96.50	22.64	88.04	64.65
LDA	19.05	39.08	23.84	30.89	21.28	38.10	28.71
2DLDA	31.58	55.09	33.64	53.21	14.63	44.07	38.70
$(2D)^2$LDA	39.44	75.00	34.78	85.52	33.33	79.52	57.93
G2DLDA	30.00	65.79	37.04	75.20	20.00	89.27	52.88
Embeddings	80.95	94.92	67.86	91.85	63.33	93.33	82.04

Tab. 8.1: *F-1 score* por clase en la tarea de *One-Shot*. Comparación entre 2DLDA y sus variantes.



(a) Descriptor geométrico (b) LDA (c) 2DLDA

(d) $(2D)^2$LDA (e) G2DLDA (f) Embeddings

Fig. 8.4: Distancias Hausdorff promedio normalizadas entre secuencias generadas y reales para cada método.

BIBLIOGRAFíA

[1] C. A. Corneanu y col. «Survey on RGB, 3D, Thermal, and Multimodal Approaches for Facial Expression Recognition: History, Trends, and Affect-Related Applications». En: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 38.8 (2016), págs. 1548-1568.

[2] Paul Ekman y Wallace V. Friesen. «Constants across cultures in the face and emotion». En: *Journal of Personality and Social Psychology, 17(2), 124–129*. 1971.

[3] D. Matsumoto. «More evidence for the universality of a contempt expression». En: *Motivation and Emotion, vol. 16, no. 4, pp. 363–368*. 1992.

[4] Rachael E. Jack y col. «Facial expressions of emotion are not culturally universal». En: *Proceedings of the National Academy of Sciences* 109.19 (2012), págs. 7241-7244. ISSN: 0027-8424. DOI: 10.1073/pnas.1200155109. eprint: https://www.pnas.org/content/109/19/7241.full.pdf. URL: https://www.pnas.org/content/109/19/7241.

[5] Paul Ekman y Wallace V. Friesen. «Facial action coding system: a technique for the measurement of facial movement». En: *Consulting Psychologists Press, Palo Alto, CA*. 1978.

[6] Shichuan Du y Aleix Martinez. «Compound facial expressions of emotion: From basic research to clinical applications». En: *Dialogues in Clinical Neuroscience* 17 (dic. de 2015), págs. 443-455.

[7] Evangelos Sariyanidi, Hatice Gunes y Andrea Cavallaro. «Automatic Analysis of Facial Affect: A Survey of Registration, Representation, and Recognition». En: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 37.6 (2015), págs. 1113-1133. DOI: 10.1109/TPAMI.2014.2366127.

[8] Yaqing Wang y col. «Generalizing from a Few Examples: A Survey on Few-shot Learning». En: *ACM Comput. Surv.* 53.3 (jun. de 2020). ISSN: 0360-0300. DOI: 10.1145/3386252. URL: https://doi.org/10.1145/3386252.

[9] Ian J. Goodfellow y col. «Generative Adversarial Nets». En: *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*. NIPS'14. Cambridge, MA, USA: MIT Press, 2014, 2672–2680.

[10] Tero Karras, Samuli Laine y Timo Aila. *A Style-Based Generator Architecture for Generative Adversarial Networks*. 2019. arXiv: 1812.04948 [cs.NE].

[11] Xun Huang y Serge Belongie. *Arbitrary Style Transfer in Real-time with Adaptive Instance Normalization*. 2017. arXiv: 1703.06868 [cs.CV]. URL: https://arxiv.org/abs/1703.06868.

[12] Yuval Alaluf y col. *HyperStyle: StyleGAN Inversion with HyperNetworks for Real Image Editing*. 2022. eprint: 2111.15666.

[13] Tero Karras y col. *Analyzing and Improving the Image Quality of StyleGAN*. 2020. arXiv: 1912.04958 [cs.CV].

[14] Jun Wang y Jean-daniel Zucker. «Solving the multiple-instance problem: A lazy learning approach». En: *Proc. 17th International Con. on Machine Learning*. Ene. de 2000, págs. 1119-1126.

[15] Daniel Acevedo y col. «A Citation k-NN Approach for Facial Expression Recognition». En: *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications*. Ed. por Marcelo Mendoza y Sergio Velastín. Cham: Springer International Publishing, 2018, págs. 1-9. ISBN: 978-3-319-75193-1.

[16] Evelyn Fix y J. L. Hodges. «Discriminatory Analysis. Nonparametric Discrimination: Consistency Properties». En: *International Statistical Review / Revue Internationale de Statistique* 57.3 (1989), págs. 238-247. ISSN: 03067734, 17515823. (Visitado 28-11-2023).

[17] Ming Li y Baozong Yuan. «2D-LDA: A statistical linear discriminant analysis for image matrix». En: *Pattern Recognition Letters* 26.5 (2005), págs. 527-532. ISSN: 0167-8655. DOI: https://doi.org/10.1016/j.patrec.2004.09.007.

[18] Hatice Gunes y BjöRn Schuller. «Categorical and Dimensional Affect Analysis in Continuous Input: Current Trends and Future Directions». En: *Image Vision Comput.* 31.2 (feb. de 2013), 120–136. ISSN: 0262-8856. DOI: 10.1016/j.imavis.2012.06.016. URL: https://doi.org/10.1016/j.imavis.2012.06.016.

[19] Mehdi Mirza y Simon Osindero. *Conditional Generative Adversarial Nets*. 2014. arXiv: 1411.1784 [cs.LG]. URL: https://arxiv.org/abs/1411.1784.

[20] Albert Pumarola y col. «GANimation: Anatomically-aware Facial Animation from a Single Image». En: *CoRR* abs/1807.09251 (2018). arXiv: 1807.09251. URL: http://arxiv.org/abs/1807.09251.

[21] Diederik P Kingma y Max Welling. *Auto-Encoding Variational Bayes*. 2022. arXiv: 1312.6114 [stat.ML].

[22] Raymond Yeh y col. *Semantic Facial Expression Editing using Autoencoded Flow*. 2016. arXiv: 1611.09961 [cs.CV]. URL: https://arxiv.org/abs/1611.09961.

[23] Ziyi Chang, George Alex Koulieris y Hubert P. H. Shum. *On the Design Fundamentals of Diffusion Models: A Survey*. 2023. arXiv: 2306.04542 [cs.LG].

[24] Di Chang y col. *MagicPose: Realistic Human Poses and Facial Expressions Retargeting with Identity-aware Diffusion*. 2024. arXiv: 2311.12052 [cs.CV]. URL: https://arxiv.org/abs/2311.12052.

[25] Robin Rombach y col. *High-Resolution Image Synthesis with Latent Diffusion Models*. 2022. arXiv: 2112.10752 [cs.CV]. URL: https://arxiv.org/abs/2112.10752.

[26] T. Ojala, M. Pietikainen y D. Harwood. «Performance evaluation of texture measures with classification based on Kullback discrimination of distributions». En: *Proceedings of 12th International Conference on Pattern Recognition*. Vol. 1. 1994, 582-585 vol.1. DOI: 10.1109/ICPR.1994.576366.

[27] Esa Rahtu y col. «Local phase quantization for blur-insensitive image analysis». En: *Image and Vision Computing* 30.8 (2012). Special Section: Opinion Papers, págs. 501-512. ISSN: 0262-8856. DOI: https://doi.org/10.1016/j.imavis.2012.04.001. URL: https://www.sciencedirect.com/science/article/pii/S0262885612000510.

[28] N. Dalal y B. Triggs. «Histograms of oriented gradients for human detection». En: *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*. Vol. 1. 2005, 886-893 vol. 1. DOI: 10.1109/CVPR.2005.177.

[29] Evangelos Sariyanidi y col. «Local Zernike Moment Representation for Facial Affect Recognition». En: sep. de 2013. DOI: 10.5244/C.27.108.

[30] N. Ahmed, T. Natarajan y K.R. Rao. «Discrete Cosine Transform». En: *IEEE Transactions on Computers* C-23.1 (1974), págs. 90-93. DOI: 10.1109/T-C.1974.223784.

[31] Karl Pearson. *LIII. On lines and planes of closest fit to systems of points in space*. Nov. de 1901. DOI: 10.1080/14786440109462720. URL: https://doi.org/10.1080/14786440109462720.

[32] Archana Kumari Sharma y col. «A Survey on Feature Extraction Technique for Facial Expression Recognition System». En: *2018 4th International Conference on Computing Communication and Automation (ICCCA)*. 2018, págs. 1-6. DOI: 10.1109/CCAA.2018.8777550.

[33] Matthew Turk y Alex Pentland. «Eigenfaces for Recognition». En: *Journal of Cognitive Neuroscience* 3.1 (ene. de 1991), págs. 71-86. ISSN: 0898-929X. DOI: 10.1162/jocn.1991.3.1.71. eprint: https://direct.mit.edu/jocn/article-pdf/3/1/71/1932018/jocn.1991.3.1.71.pdf. URL: https://doi.org/10.1162/jocn.1991.3.1.71.

[34] Jérémie Nicolle y col. «Robust continuous prediction of human emotions using multiscale dynamic cues». En: *Proceedings of the 14th ACM International Conference on Multimodal Interaction*. ICMI '12. New York, NY, USA: Association for Computing Machinery, 2012, 501–508. ISBN: 9781450314671. DOI: 10.1145/2388676.2388783. URL: https://doi.org/10.1145/2388676.2388783.

[35] Hong-Bo Deng y col. «A new facial expression recognition method based on local Gabor filter bank and PCA plus LDA». En: *International Journal of Information Technology* 11.11 (2005), págs. 86-96.

[36] Raviteja Vemulapalli y Aseem Agarwala. «A Compact Embedding for Facial Expression Similarity». En: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Jun. de 2019.

[37] C. Cortes y V. Vapnik. «Support-vector networks». En: *Mach Learn* 20 (1995), 273–297. DOI: 10.1007/BF00994018.

[38] Ira Cohen y col. «Facial expression recognition from video sequences: temporal and static modeling». En: *Computer Vision and Image Understanding* 91.1 (2003). Special Issue on Face Recognition, págs. 160-187. ISSN: 1077-3142. DOI: https://doi.org/10.1016/S1077-3142(03)00081-X. URL: https://www.sciencedirect.com/science/article/pii/S107731420300081X.

[39] Hivi Ismat Dino y Maiwan Bahjat Abdulrazzaq. «Facial Expression Classification Based on SVM, KNN and MLP Classifiers». En: *2019 International Conference on Advanced Science and Engineering (ICOASE)*. 2019, págs. 70-75. DOI: 10.1109/ICOASE.2019.8723728.

[40] P. Viola y M. Jones. «Rapid object detection using a boosted cascade of simple features». En: *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001.* Vol. 1. 2001, págs. I-I. DOI: 10.1109/CVPR.2001.990517.

[41] Shan Li y Weihong Deng. «Deep Facial Expression Recognition: A Survey». En: *IEEE Transactions on Affective Computing* 13.3 (2022), págs. 1195-1215. DOI: 10.1109/TAFFC.2020.2981446.

[42] Alex Krizhevsky, Ilya Sutskever y Geoffrey E. Hinton. «ImageNet Classification with Deep Convolutional Neural Networks». En: *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1.* NIPS'12. Lake Tahoe, Nevada: Curran Associates Inc., 2012, págs. 1097-1105. URL: http://dl.acm.org/citation.cfm?id=2999134.2999257.

[43] Sébastien Ouellet. «Real-time emotion recognition for gaming using deep convolutional network features». En: *CoRR* abs/1408.3750 (2014). arXiv: 1408.3750. URL: http://arxiv.org/abs/1408.3750.

[44] Karen Simonyan y Andrew Zisserman. *Very Deep Convolutional Networks for Large-Scale Image Recognition.* 2014. arXiv: 1409.1556 [cs.CV].

[45] Gil Levi y Tal Hassner. «Emotion Recognition in the Wild via Convolutional Neural Networks and Mapped Binary Patterns». En: nov. de 2015, págs. 503-510. DOI: 10.1145/2818346.2830587.

[46] Hui Ding, Shaohua Kevin Zhou y Rama Chellappa. *FaceNet2ExpNet: Regularizing a Deep Face Recognition Net for Expression Recognition.* 2016. arXiv: 1609.06591 [cs.CV].

[47] Christian Szegedy y col. *Going Deeper with Convolutions.* 2014. arXiv: 1409.4842 [cs.CV].

[48] Xiangyun Zhao y col. *Peak-Piloted Deep Network for Facial Expression Recognition.* 2016. arXiv: 1607.06997 [cs.CV].

[49] Kaiming He y col. *Deep Residual Learning for Image Recognition.* 2015. arXiv: 1512.03385 [cs.CV].

[50] Ping Hu y col. «Learning Supervised Scoring Ensemble for Emotion Recognition in the Wild». En: *Proceedings of the 19th ACM International Conference on Multimodal Interaction.* ICMI '17. New York, NY, USA: ACM, 2017, págs. 553-560. ISBN: 978-1-4503-5543-8. DOI: 10.1145/3136755.3143009. URL: http://doi.acm.org/10.1145/3136755.3143009.

[51] Behzad Hasani y Mohammad H. Mahoor. *Facial Expression Recognition Using Enhanced Deep 3D Convolutional Neural Networks.* 2017. arXiv: 1705.07871 [cs.CV].

[52] Hinton G. LeCun Y Bengio Y. «Deep learning». En: *Nature* (2015), págs. 436-44. DOI: 10.1038/nature14539.

[53] Yu Zhang y col. «A Survey on Neural Network Interpretability». En: *IEEE Transactions on Emerging Topics in Computational Intelligence* 5.5 (2021), págs. 726-742. DOI: 10.1109/TETCI.2021.3100641.

[54] Omer Tov y col. *Designing an Encoder for StyleGAN Image Manipulation.* 2021. eprint: 2102.02766.

[55] Martin Arjovsky, Soumith Chintala y Léon Bottou. *Wasserstein GAN*. 2017. arXiv: 1701.07875 [stat.ML].

[56] Tero Karras y col. *Progressive Growing of GANs for Improved Quality, Stability, and Variation*. 2018. arXiv: 1710.10196 [cs.NE]. URL: https://arxiv.org/abs/1710.10196.

[57] Animesh Karnewar y Oliver Wang. *MSG-GAN: Multi-Scale Gradients for Generative Adversarial Networks*. 2020. arXiv: 1903.06048 [cs.CV]. URL: https://arxiv.org/abs/1903.06048.

[58] David Ha, Andrew Dai y Quoc V. Le. *HyperNetworks*. 2016. arXiv: 1609.09106 [cs.LG]. URL: https://arxiv.org/abs/1609.09106.

[59] R. A. Fisher. «The Use Of Multiple Measurements In Taxonomic Problems». En: *Annals of Eugenics* 7.2 (1936), págs. 179-188. DOI: https://doi.org/10.1111/j.1469-1809.1936.tb02137.x.

[60] C. Radhakrishna Rao. «The Utilization of Multiple Measurements in Problems of Biological Classification». En: *Journal of the Royal Statistical Society. Series B (Methodological)* 10.2 (1948), págs. 159-203. ISSN: 00359246. URL: http://www.jstor.org/stable/2983775 (visitado 26-11-2023).

[61] Chun-Na Li y col. «Generalized two-dimensional linear discriminant analysis with regularization». En: *Neural Networks* 142 (2021), págs. 73-91. ISSN: 0893-6080. DOI: https://doi.org/10.1016/j.neunet.2021.04.030.

[62] Peratham Wiriyathammabhum y Boonserm Kijsirikul. «Basis Selection for 2DLDA-Based Face Recognition Using Fisher Score». En: dic. de 2009, págs. 708-715. ISBN: 978-3-642-10676-7. DOI: 10.1007/978-3-642-10677-4_81.

[63] T. Kanade, J. F. Cohn y Yingli Tian. «Comprehensive database for facial expression analysis». En: *Proceedings Fourth IEEE International Conference on Automatic Face and Gesture Recognition (Cat. No. PR00580)*. 2000, págs. 46-53.

[64] P. Lucey y col. «The Extended Cohn-Kanade Dataset (CK+): A complete dataset for action unit and emotion-specified expression». En: *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Workshops*. 2010, págs. 94-101.

[65] Kaipeng Zhang y col. «Joint Face Detection and Alignment Using Multitask Cascaded Convolutional Networks». En: *IEEE Signal Processing Letters* 23.10 (oct. de 2016), 1499–1503. ISSN: 1558-2361. DOI: 10.1109/lsp.2016.2603342. URL: http://dx.doi.org/10.1109/LSP.2016.2603342.

[66] Gabriele Fanelli y col. «Real Time Head Pose Estimation from Consumer Depth Cameras». En: *Pattern Recognition*. Ed. por Rudolf Mester y Michael Felsberg. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, págs. 101-110. ISBN: 978-3-642-23123-0.

[67] Dimitrios Kollias y col. «Abaw: Valence-arousal estimation, expression recognition, action unit detection & emotional reaction intensity estimation challenges». En: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, págs. 5888-5897.

[68] Dimitrios Kollias. «ABAW: learning from synthetic data & multi-task learning challenges». En: *European Conference on Computer Vision*. Springer. 2023, págs. 157-172.

[69] Dimitrios Kollias. «Abaw: Valence-arousal estimation, expression recognition, action unit detection & multi-task learning challenges». En: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, págs. 2328-2336.

[70] Dimitrios Kollias y Stefanos Zafeiriou. «Analysing affective behavior in the second abaw2 competition». En: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, págs. 3652-3660.

[71] D Kollias y col. «Analysing Affective Behavior in the First ABAW 2020 Competition». En: *2020 15th IEEE International Conference on Automatic Face and Gesture Recognition (FG 2020)(FG)*, págs. 794-800.

[72] Dimitrios Kollias, Viktoriia Sharmanska y Stefanos Zafeiriou. «Distribution Matching for Heterogeneous Multi-Task Learning: a Large-scale Face Study». En: *arXiv preprint arXiv:2105.03790* (2021).

[73] Dimitrios Kollias y Stefanos Zafeiriou. «Affect Analysis in-the-wild: Valence-Arousal, Expressions, Action Units and a Unified Framework». En: *arXiv preprint arXiv:2103.15792* (2021).

[74] Dimitrios Kollias y Stefanos Zafeiriou. «Expression, Affect, Action Unit Recognition: Aff-Wild2, Multi-Task Learning and ArcFace». En: *arXiv preprint arXiv:1910.04855* (2019).

[75] Dimitrios Kollias, Viktoriia Sharmanska y Stefanos Zafeiriou. «Face Behavior a la carte: Expressions, Affect and Action Units in a Single Network». En: *arXiv preprint arXiv:1910.11111* (2019).

[76] T. F. Cootes, G. J. Edwards y C. J. Taylor. «Active appearance models». En: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 23.6 (2001), págs. 681-685. DOI: 10.1109/34.927467.

[77] T.F. Cootes y col. «Active Shape Models-Their Training and Application». En: *Computer Vision and Image Understanding* 61.1 (1995), págs. 38-59. ISSN: 1077-3142. DOI: https://doi.org/10.1006/cviu.1995.1004. URL: https://www.sciencedirect.com/science/article/pii/S1077314285710041.

[78] M. Kass, A. Witkin y D. Terzopoulos. «Snakes: Active contour models». En: *Int J Comput Vision* 1 (1988), 321–331. DOI: 10.1007/BF00133570.

[79] A. Lanitis, C.J. Taylor y T.F. Cootes. «A unified approach to coding and interpreting face images». En: *Proceedings of IEEE International Conference on Computer Vision*. 1995, págs. 368-373. DOI: 10.1109/ICCV.1995.466919.