Desarrollo de un sistema automático de análisis de expresiones faciales para la detección de la mentira en adultos utilizando técnicas de aprendizaje automático

PRESENTADO POR:

Natalia Zartha Suarez

Universidad Tecnológica de Pereira

Facultad de Ingenierías

Pereira, Colombia

2021

Desarrollo de un sistema automático de análisis de expresiones faciales para la detección de la mentira en adultos utilizando técnicas de aprendizaje automático

PRESENTADO POR

Natalia Zartha Suarez

Proyecto de grado presentado como requisito parcial para optar al título de:

Magister en Ingeniería de Sistemas y Computación

Director:

Hernán Felipe García Arias

Ph.D. en Ingeniería

Universidad Tecnológica de Pereira

Facultad de Ingenierías

Pereira, Colombia

15 Octubre 2021

Dedicatoria

Dedico este trabajo de grado a mi maravillosa hija Sara Lucia Rojas Zartha, mi mayor tesoro en mi vida, que este trabajo de grado sea una inspiración para seguir estudiando y nunca rendirse en la vida. Dedico y agradezco este trabajo de grado a mis grandiosos padres, mi mamá Gloria Beatriz Suarez Fernández y a mi papá Gustavo Sarta Sossa; los cuales siempre me han apoyado, inspirado, enseñado, guiado y estado a mi lado toda mi vida, gracias por tanta entrega, amor y por rezar por mi todos los días para que Dios guie mis pasos. También dedico este trabajo de grado a mi valiosa hermana Beatriz Eugenia Zartha Suarez por su ayuda, respaldo y cariño para toda la vida, a mis hermosos sobrinos Juan Esteban y Juan Pablo Trujillo Zartha; para que nunca dejen de luchar por sueños, siempre estudien y se preparen para la vida. A todos mil gracias y los quiero mucho.

Agradecimientos

Agradezco a Dios por todas las bendiciones que ha puesto en mi camino y por la maravillosa familia que tengo.

Agradezco a mis padres y a mi hermana, por su apoyo incondicional, porque sin la ayuda de ellos no hubiera podido realizar mi Maestría.

Agradezco a mi director de Tesis el Doctor Hernán Felipe García Arias quien me oriento, me apoyo y dio las pautas para llevar a cabo este trabajo de grado exitosamente, Muchas gracias.

Agradezco a la Universidad Tecnológica de Pereira por ofrecer tan excelente programas de formación para todos, es una Universidad grandiosa donde termíneme mi Pregrado y ahora mi Postgrado, Muchas gracias.

Agradezco a profesores y amigos que han compartido conmigo nuevos conocimientos y enseñanzas académicas para seguir valorando lo que hago día a día.

Contenido

Lista de Tablas

Lista de Figuras

Resumen

Existen 7 tipos de expresiones faciales universales, las cuales son: enfado, disgusto, miedo, felicidad, tristeza, sorpresa y desprecio. Estas expresiones faciales son indiferentes a la raza o la cultura de las regiones del mundo. Estas expresiones pueden ser falsificadas y son los pequeños movimientos los que nos pueden decir si una expresión está siendo real o es una mentira. Estos pequeños movimientos se llaman microexpresiones faciales, los cuales ocurren entre 1/15 y 1/25 segundos y son imperceptibles al ojo humano. Este trabajo de grado tiene como objetivo reconocer las microexpresiones faciales mediante un modelo profundo de aprendizaje automático. Para este fin, se desarrollan 3 modelos cada uno para dos bases de datos de microexpresiones faciales SMIC (X. Li, T. Pfister, X. Huang, G. Zhao & M. Pietikäinen, 2013) y CASME II (Yan WJ, Li X, Wang SJ, Zhao G, Liu YJ, Chen YH & Fu X., 2014). El primer modelo implementado fue MicroExpSTCNN el cual fue propuesto por (S. P. Teja Reddy, S. Teja Karri, S. R. Dubey & S. Mukherjee, 2019) utilizando sobre las mismas bases de datos de microexpresiones faciales, este trabajo de grado logró obtener un accuracy mayor para ambas bases de datos (90 % para CASME II y 91.6 % para SMIC); que el reportado por la referencia, el cual fue de 87.80 % para la base de datos CASME II. El segundo modelo implementado fue un CNN 3D con data augmentation rotando las imágenes con cierto número de grados escogidos aleatoriamente, para este modelo se logró mejorar el acurracy para la base de datos CASME II (94.2 %). El tercer modelo se construyó con una CNN 2D temporal y una

capa de LSTM, lo cual logró mejorar notablemente la predicción para ambas bases de datos de microexpresiones faciales, ya que tuvo en cuenta la característica temporal de los 18 frames. También se desarrolló una aplicación donde se creó el modelo de la red neuronal y se le cargaron los pesos entrenados previamente para ambas bases de datos de SMIC (X. Li, et al., 2013) y CASME II (Yan WJ, et al., 2014). Se usó el framework Flask para visualizar el video y mostrar la microexpresión facial que predice el modelo.

Palabras clave: Aprendizaje profundo, Reconocimiento de microexpresiones faciales, Redes Neuronales Convolucionales, Redes de memoria a corto/largo plazo LSTM.

Abstract

There are 7 types of universal facial expressions, which are: anger, disgust, fear, happiness, sadness, surprise and contempt. These facial expressions are indifferent to the race or culture of the world regions. These expressions can be faked and it is the small movements that can tell us if an expression is being real or a lie. These small movements are called facial micro-expressions, which occur between 1/15 and 1/25 seconds and are imperceptible to the human eye. This degree work aims to recognize facial microexpressions using a deep machine learning model. For this purpose, 3 models each are developed for two databases of SMIC facial microexpressions (X. Li, T. Pfister, X. Huang, G. Zhao & M. Pietikäinen, 2013) and CASME II (Yan WJ, Li X, Wang SJ, Zhao G, Liu YJ, Chen YH & Fu X., 2014). The first model implemented was MicroExpSTCNN which was proposed by (SP Teja Reddy, S. Teja Karri, SR Dubey & S. Mukherjee, 2019) using the same databases of facial microexpressions, this degree work managed to obtain a higher accuracy for both databases (90% for CASME II and 91.6% for SMIC); than that reported by the reference, which was 87.80% for the CASME II database. The second model implemented was a CNN 3D with data augmentation rotating the images with a certain number of degrees chosen randomly, for this model it was possible to improve the acurracy for the CASME II database (94.2%). The third model was built with a temporal 2D CNN and an LSTM layer, which managed to significantly improve the prediction for both databases of facial microexpressions, since it took into account the temporal characteristic of the 18 frames. An application was also developed where the neural network model was

created and the previously trained weights were loaded for both databases of SMIC (X. Li, et al., 2013) and CASME II (Yan WJ, et al., 2014). The Flask framework was used to visualize the video and show the facial microexpression predicted by the model.

Keywords: Deep learning, Facial microexpression recognition, Convolutional Neural Networks, LSTM short/long-term memory networks.

Notaciones

Abreviaturas

Adam	Adaptive Moment Estimation
ANN	Artificial Neural Network
AU	Action Units (AUs)
AUC	Area under the ROC Curve
CA	Channel Attention
CNNs	Convolutional Neural Networks
3DCNN	3D Convolutional Neural Networks
DL	Deep Learning
EVM	Eulerian Video Magnification
EVS	Embedded Vision System
FACS	Facial Action Coding System
FD	Feature Difference
FME	Facial Micro-Expressions
FER	Facial Expression Recognition
SGD	Stochastic Gradient Descent
HIGO	Histograms of Image Gradient Orientation
HOG	Histograms of Oriented Gradients
IIR	Filtro Infinite impulse response
LFM	Landmark Feature Map
LBP	Local Binary Pattern
LSTM	Long Short-Term Memory Networks
MEGC	Facial Micro-Expression Grand Challenge

ML	Machine Learning
MLP	Multilayer perceptron
RGB	Espacio de color (Red-Green-Blue)
ROC	Receiver operating characteristic
RNN	Recurrent Neural Network
ResNet	Deep Residual Network
SA	Spatial Attention
SCA	Spatio-Channel Attention
SOTA	State-Of-The-Art

Capítulo I

1. Introducción

Las expresiones faciales son una de las más potentes, naturales y señales universales por medio del cual los seres humanos transmiten sus emociones, estados e intenciones. Varios Sistemas de Reconocimiento de Expresiones Faciales (FER) han sido explorado para codificar información de expresión a partir de representaciones faciales. Los sistemas FER se pueden dividir en dos categorías principales de acuerdo con las representaciones de características: imagen estática FER e imagen dinámica FER. A pesar de la poderosa capacidad de aprendizaje del deep learning, los problemas persisten cuando se aplica a FER, primero Deep Neuronal Network requiere una gran cantidad de datos de entrenamiento para evitar el sobreajuste, adicionalmente las variaciones en la pose del sujeto, iluminación y oclusión son los problemas más comunes en escenarios de expresiones faciales. (S. Li & W. Deng, 2020)

Con el descubrimiento de las micro expresiones faciales descubiertas por primera vez por los investigadores Haggard and Isaacs, tiempo después el Dr. Paul Ekman popularizó el término "Microexpresión" y amplió enormemente la investigación. Ekman descubrió que en todos estos países la gente expresaba e identificaba las 7 emociones universales de la misma manera. Incluso se aventuró a una tribu remota y primitiva llamada Fore en Papau Nueva Guinea y descubrió que expresaban las mismas emociones que nosotros. El rostro es el mejor indicador de las emociones de una

persona. Las 7 expresiones faciales son (Van Edwards, V., 2014): Sorpresa, Miedo, Disgusto, Enfado, Felicidad, Tristeza y Desprecio. Estudiar las emociones desde las microexpresiones faciales también funciona como una herramienta para descifrar el estado emocional actual de una persona con facilidad. En cualquier momento en particular, la emoción que siente una persona sólo puede descifrarse a través de la expresión expresada por esa persona. Las 80 contracciones musculares faciales de una persona y sus combinaciones dan lugar a miles de expresiones (Chavali, G. K., 2014).

Según Darwin, hay pocos músculos que sean imposibles de activar voluntariamente y estos músculos revelan las verdaderas intenciones de los demás (P. Ekman, 2009). Hay 7 características como la duración, la simetría, la velocidad de aparición, etc., que distinguen las expresiones faciales voluntarias de las involuntarias (P. Ekman, 2003). Un mentiroso puede provocar una señal emocional que delata la plausibilidad de la mentira en sí (P. Ekman, 1989, p.71–81). Las microexpresiones de emociones están típicamente representadas en todas estas clases involuntarias (Chavali, G. K., 2014).

1.1. Planteamiento del problema

Revelar a un criminal y demostrar que está mintiendo se convierte en un reto al momento de ser juzgado. Hoy en día la conducta criminal no se puede sesgar solamente a los aspectos físicos de una persona, esta depende de múltiples factores como: político, cultural, económico, psicológico, académico, conductual, entre otros.

Pero detectar cuando una persona está mintiendo puede ser la primera aproximación para descubrir a un criminal; la mentira se puede detectar mediante reconocimiento de expresiones faciales FER por sus siglas en inglés (Facial Expression Recognition); asimismo se pueden reconocer las emociones mediante microexpresiones faciales. Las microexpresiones faciales son movimientos del rostro muy breves, generadas entre 1/15 y 1/25 segundos; por ser tan breves pueden revelar verdades que el ojo humano no logra percibir.

Cuando una persona se da cuenta conscientemente de que está ocurriendo una expresión facial, la persona puede tratar de suprimir la expresión porque mostrarla puede no ser apropiado o podría deberse a una regla de exhibición cultural. (Merghani, W., 2018)

El reconocimiento de expresiones faciales a pesar de ser un sistema muy potente aún presenta algunos obstáculos al momento de ser analizados los datos, como: cambios de iluminación, posturas no frontales de la cabeza y oclusiones. También en los sistemas Deep FER (Deep Facial Expression Recognition) presenta el desafío de la falta de datos en términos de cantidad y calidad.

Tanto en una entrevista como en la recogida de información de testimonios, siempre existe la posibilidad que la persona mienta. Algunos métodos y técnicas para detectar la mentira son: dejar que hable, cambiar de tema y preguntar de nuevo, lenguaje corporal, el mito de la dirección de los ojos, control de realidad, SCAN, polígrafo y otros

detectores de mentiras; (Agencyworld, 2019) en todos estos, existe un porcentaje insatisfactorio para verificar la verdad. El polígrafo mide el ritmo cardíaco, la presión sanguínea y la respiración. El análisis del contenido basado en criterios, la técnica del control de la realidad, ALIED (detector de Mentiras Adaptativo) y las SCAN tienen en cuenta la experiencia personal para detectar mentiras, pero fallan por el estrés, el nerviosismo, la ansiedad, la tristeza, la vergüenza y el miedo que producen las pruebas, alterando los resultados en los individuos. (Andrea, P., et al., 2016)

Está científicamente probado que ante una situación amenazante se dispara el sistema simpático y el cuerpo se pone en alerta. “Todas las mentiras generan una reacción psicofisiológica, y si el que observa está bien entrenado, lo ve”. Ni el mentiroso más experimentado puede controlar de manera indefinida lo que la ciencia ha resuelto en llamar microexpresiones faciales. La detección de microexpresiones faciales y puntos de conflicto (rascarse la nariz, la cabeza, tocarse la barbilla, entre otros) denotan que existe una incomodidad en el individuo, es decir una disputa interna. (Fita, J.,2016)

1.1.1. Pregunta de Investigación

De acuerdo con el problema planteado anteriormente, este proyecto realiza un aporte valioso en el reconocimiento de microexpresiones faciales en expresiones espontáneas para el análisis de la mentira, para lo cual, se planteó la siguiente pregunta de investigación: *¿Es posible analizar mediante modelos de aprendizaje*

automático y/o visión por computador las expresiones faciales de una persona para la detección de la mentira en adultos?

1.2 Justificación

La mentira ha sido estudiada como un fenómeno social de gran importancia por las diversas implicaciones en la sociedad. La psicología tradicional ha estudiado a la mentira como un evento mental o representaciones cognitivas que causan nuestros actos mentirosos. El estudio experimental se ha desarrollado en gran medida como la detección de la mentira a través de indicadores corporales (Ortega González, M., 2010). Otros estudios recientes, han demostrado que las microexpresiones faciales son una fuente de comportamiento importante para la detección de intenciones hostiles y conductas peligrosas. (M. Owayjan, A. Kashour, N. Al Haddad, M. Fadel & G. Al Souki, 2012)

Los investigadores están desarrollando varias técnicas para detectar individuos mentirosos. Las autoridades del aeropuerto británico están probando un sistema basado en el Sistema de Codificación de Acción Facial (FACS). El rostro humano es un vehículo de señalización que envía mensajes utilizando no sólo su estructura básica y tono muscular, sino también cambios en la cara que transmite expresiones, como sonrisas, fruncir el ceño, entre otras. El estado de ánimo y las intenciones de la persona se pueden leer en las expresiones faciales.

Esta investigación será pertinente para detectar la mentira mediante las expresiones faciales de una persona adulta. Científicos tiempo atrás, han tenido un acercamiento a estudiar la conducta criminal y los han catalogado por sus rasgos físicos. Detectar la mentira puede ser el inicio de muchas otras investigaciones en las áreas de criminalidad, psicología, administración, humanística, judicial, entre otras.

1.2.1. Pertinencia

La mentira ha sido estudiada como un fenómeno social de gran importancia por las diversas implicaciones en la sociedad. La psicología tradicional ha estudiado a la mentira como un evento mental o representaciones cognitivas que causan nuestros actos mentirosos. El estudio experimental se ha desarrollado en gran medida como la detección de la mentira a través de indicadores corporales (González, 2010). Otros estudios recientes, han demostrado que las microexpresiones faciales son una fuente de comportamiento importante para la detección de intenciones hostiles y conductas peligrosas. (M. Owayjan, et al., 2012)

1.2.2. Viabilidad

Los investigadores están desarrollando varias técnicas para detectar individuos mentirosos. Las autoridades del aeropuerto británico están probando un sistema basado en el Sistema de Codificación de Acción Facial (FACS). El rostro humano es un

vehículo de señalización que envía mensajes utilizando no sólo su estructura básica y tono muscular, sino también cambios en la cara que transmite expresiones, como sonrisas, fruncir el ceño, entre otras. El estado de ánimo y las intenciones de la persona se pueden leer en las expresiones faciales. (M. Owayjan, et al., 2012)

1.2.3. Impacto

Esta investigación será pertinente para detectar la mentira mediante las expresiones faciales de una persona adulta. Científicos tiempo atrás, han tenido un acercamiento a estudiar la conducta criminal y los han catalogado por sus rasgos físicos. Detectar la mentira puede ser el inicio de muchas otras investigaciones en las áreas de criminalidad, psicología, administración, humanística, judicial, entre otras. Además, se espera que con el desarrollo de este trabajo se contribuya al estudio de técnicas de aprendizaje profundo para el análisis de microexpresiones faciales y reconocimientos de eventos.

1.3. Objetivos de la investigación

1.3.1. Objetivo general

Desarrollar un sistema de análisis de expresiones faciales para la detección de la mentira en los adultos mediante técnicas de aprendizaje automático.

1.3.2. Objetivos específicos

1. Desarrollar un modelo de reconocimiento de expresiones faciales utilizando un enfoque de Machine Learning y/o visión por computador.

2. Caracterizar los rasgos de las expresiones faciales para reconocer de manera automática la mentira en los adultos.

3. Integrar el modelo desarrollado para construir un prototipo de un sistema automático de detección de la mentira basado en las expresiones faciales y/o microexpresiones.

Capítulo II

2. Estado del Arte

Las microexpresiones faciales son expresiones faciales muy breves y espontáneas que aparecen en el rostro de los humanos cuando ocultan deliberada o inconscientemente una emoción. La microexpresión tiene una duración más corta que la macro-expresión, lo que la hace más desafiante para humanos y máquinas. Durante los últimos diez años, el reconocimiento automático de microexpresiones ha atraído cada vez más la atención de investigadores en psicología, informática, seguridad, neurociencia y otras disciplinas relacionadas (Merghani, W., 2018). El artículo A Review on Facial Micro-Expressions Analysis: Datasets, Features and Metrics (Merghani, W., 2018) se revisaron los conjuntos de datos, las características y las métricas de rendimiento implementadas en la literatura. Se discuten los desafíos relevantes, como la configuración espacial temporal durante la recopilación de datos, las clases emocionales frente a las clases objetivas en el etiquetado de datos, las regiones faciales en el análisis de datos, la estandarización de métricas y los requisitos para la implementación en el mundo real. (Merghani, W., 2018)

El desarrollo de conjuntos de datos espontáneos de microexpresión es uno de los mayores desafíos que enfrenta esta área de investigación. Es difícil provocar microexpresiones porque son difíciles de fingir, por lo que necesitamos obtener la verdadera emoción mientras la persona intenta ocultarla (Merghani, W., 2018). Algunos

conjuntos de datos espontáneos hasta la fecha incluyen: SMIC (X. Li, et al., 2013), CASME (Wen-Jing Yan, Q. Wu, Yong-Jin Liu, Su-Jing Wang & X. Fu, 2013), CASME II (Yan WJ, et al., 2014), SAMM (A. K. Davison, C. Lansley, N. Costen, K. Tan and M. H. Yap, 2018) y CAS (ME) 2 (F. Qu, S.-J. Wang, W.-J. Yan, H. Li, S. Wu, & X. Fu, 2017).

En el artículo (Merghani, W., 2018) explican cada base de datos, la cantidad de frames, los pixeles, los movimientos faciales capturados, como estaba distribuida la población de muestra, la calidad de la cámara, la cantidad de macro-expresiones y microexpresiones faciales espontáneas obtenidas en el análisis. De igual manera se explican los tipos de características, clasificadores y métricas utilizados durante la última década para el reconocimiento de microexpresiones por año y autores.

Como el reconocimiento de microexpresiones está todavía en sus inicios para el año 2018; como lo plantean los autores de este artículo, en comparación con la macro expresión, requiere esfuerzos combinados de multidisciplinarios (incluidos psicología, informática, fisiología, ingenieros y diseñadores de políticas) para lograr resultados confiables para la aplicación práctica en el mundo real. Un punto controvertido es si se debe permitir o no detectar estas microexpresiones, ya que la teoría detrás de ellas establece que la persona que intenta ocultar su emoción experimenta estos movimientos de manera involuntaria y probablemente sin saberlo. Si somos capaces de detectarlos con alta precisión, entonces efectivamente le estamos robando a una persona el poder ocultar algo que es privado para ellos. Desde un punto de vista ético,

saber cuándo alguien está engañando sería ventajoso, pero te quita la libertad que tenías en tus emociones. (Merghani, W., 2018)

En FACS, AU se define como el movimiento facial básico, que funciona como los bloques de construcción para formular múltiples expresiones faciales. FACS indica que la detección exitosa de AU facilita enormemente el análisis de las acciones o expresiones faciales complicadas (W. Li, F. Abtahi, Z. Zhu, 2017). En otras palabras, es muy esencial explorar AU para interpretar profundamente el comportamiento facial de las expresiones (Li, Y., Huang, X., & Zhao, G., 2021). Dado que las AU de microexpresión corresponden a activaciones musculares específicas de la cara y tienen baja intensidad, las AU de microexpresión están más directamente relacionadas con la distorsión regional (D. Acharya, Z. Huang, D. Pani Paudel, L. Van Gool, 2018).

En el artículo Micro-expression action unit detection with spatial and channel attention (D. Acharya, et al., 2018). se propone un módulo de atención espacial (SA) para codificar la correlación de covarianza de todas las regiones en la cara, combinaron SA y CA para mejorar aún más el rendimiento de la detección de AU de microexpresión. Finalmente, se propone un framework deep end-to-end llamado SCA para detectar ACs en microexpresiones faciales a traves de second-order spatial y channel attentions. SCA (Spatio-Channel Attention) propuesto puede extraer los cambios regionales locales y la relación de las regiones faciales locales

simultáneamente para obtener una representación AU de microexpresión discriminativa y robusta (Li, Y., et al., G., 2021).

Evaluando el costo computacional de este artículo (Li, Y., et al., G., 2021); para realizar el análisis usaron un GPU NVIDIA Tesla K80c con 12 GB de memoria, el cual tardo 2.5h procesando 260 M parámetros para las 3 bases de datos CASME (Wen-Jing Yan, et al., 2013), CASME II (Yan WJ, et al., 2014) y SAMM (A. K. Davison, et al., 2018). Además, SCA no necesita una red troncal adicional para conocer la función local y la relación entre las funciones locales. Por lo tanto, se cree que el SCA propuesto en este artículo tiene un costo de cálculo eficiente y un costo de almacenamiento pequeño (Li, Y., et al., G., 2021).

En el artículo Facial Micro-Expression Recognition Using Two-Dimensional Landmark Feature Maps (D. Y. Choi & B. C. Song, 2020) se propone un algoritmo de reconocimiento FME (Facial micro-expression) basado en LFM (landmark feature map) que se basa en CNN (Convolultional Neuronal Network) y LSTM (long short-term memory) y la clasificación la hacen con deep learning-based (D. Y. Choi, et al., 2020).

Durante algunos años, se ha realizado un gran desafío de microexpresión (MEGC) para promover el desarrollo competitivo de técnicas de reconocimiento de FME (J. See, M. H. Yap, J. Li, X. Hong & S.-J. Wang, 2019). En MEGC, muchas técnicas de FER se evalúan utilizando conjuntos de datos específicos de microexpresión como SMIC (X. Li,

et al., 2013), CASME II (Yan WJ, et al., 2014), SAMM (A. K. Davison, et al., 2018) y su conjunto de datos compuestos (D. Y. Choi, et al., 2020).

El LFM propuesto genera un patrón único según la clase de emoción, independientemente de la fuerza de la emoción facial, lo que permite un reconocimiento de emociones efectivo. Para conjuntos de datos de microexpresión como SMIC (X. Li, et al., 2013) y CASME II (Yan WJ, et al., 2014), el método propuesto mostró superioridad en precisión y puntuación F1-score sobre SOTA (state-of-the-art) (D. Y. Choi, et al., 2020).

En el trabajo de grado *"Micro-Expression Extraction For Lie Detection Using Eulerian Video (Motion and Color) Magnification"* (Chavali, G. K., 2014) para la tesis de Maestría en Ingeniería Eléctrica del Blekinge Institute of Technology August en el 2014, usan el análisis Eulerian Video Magnification (EVM) para ampliar el movimiento y el color de los frames de los videos y así calcular la emoción y el pulso de la persona para detectar la mentira.

Después de todo el análisis, de montar los videos sobre una GUI en Matlab y extraer las microexpresiones faciales, este trabajo de grado no se puede llamar un detector de mentiras, ya que no detecta explícitamente ninguna mentira, pero extrae una emoción que sería útil en el proceso de detección de mentiras. De esta tesis se concluye que observar, analizar y comprender estas microexpresiones puede intensificar el proceso de detección de una mentira, pero comprender tales expresiones impulsivas es una

tarea tediosa y difícil para una persona promedio. En general, un contacto físico con un sujeto bajo prueba puede inducir una sensación de conciencia en esa persona. Utilizando el aumento de video euleriano y las redes neuronales, se extrae la emoción a través del aumento de movimiento y el pulso a través del aumento de color sin ningún contacto físico con el sujeto bajo prueba. La principal ventaja de no tener contacto físico resultaría en una fuga de emociones inconscientes que se capturan y descifran (Chavali, G. K., 2014).

Este de grado plantea la ventaja de analizar el color en los frames de video, ya que proporciona información del pulso de la persona, tal medida ayuda a la detección de la mentira; pero de igual manera no es una tarea fácil.

En EVM, se seleccionan ciertas ubicaciones espaciales para amplificar las variaciones en las bandas de frecuencia temporal. El filtrado temporal amplifica tanto el color como el movimiento. La descomposición de los frames de video se realiza utilizando la estructura piramidal de Laplacian, debido a dos razones diferentes, tal como se especifica (Hao-Yu Wu, M. Rubinstein, E. Shih, J. Guttag, F. Durand, & W. Freeman, 2012). Después de la descomposición de estas bandas espaciales, se aplica un filtrado temporal a cada banda. En el filtrado temporal se está considerando un valor de píxel en la banda de frecuencia con la correspondiente serie temporal y posteriormente se aplica un filtro pasabanda (Hao-Yu Wu, et. al., 2012). La señal de paso de banda extraída se multiplica luego por un factor de amplificación de α, donde α es específico del usuario y de la aplicación. Esta señal amplificada se agrega a la

original. Todos estos fotogramas descompuestos espacialmente se reconstruyen nuevamente para formar la salida final, donde se magnifica el movimiento o el color.

Las microexpresiones de emociones son cambios no periódicos en el rostro. EVM magnifica los movimientos no periódicos de la cara, cuando estos cambios están dentro de la banda de paso del filtro de banda de paso temporal (Hao-Yu Wu, et. al., 2012). Así, EVM magnifica los movimientos no periódicos de menor magnitud que exhibe el rostro. EVM no solo funciona para videos de larga duración, sino que también funciona para videos con duraciones muy cortas (Chavali, G. K., 2014).

En la ampliación del movimiento transformaron los movimientos sutiles que son invisibles a simple vista para ampliarlos mediante la ampliación de movimiento de EVM. Este aumento de movimiento ayuda a observar esos movimientos sutiles y espontáneos en la cara (microexpresiones) que pueden pasar fácilmente desapercibidos (Chavali, G. K., 2014).

En la ampliación de movimiento, se exagera el movimiento amplificando los cambios en el color temporal en valores de píxeles fijos, en lugar de utilizar los algoritmos tradicionales de estimación de movimiento. Las estructuras piramidales laplacianas se utilizan para la descomposición espacial del movimiento. Los cambios temporales que ocurren en la ampliación de movimiento se analizan utilizando el primer orden.

Expansión de la serie de Taylor (Hao-Yu Wu, et. al., 2012). La ampliación de movimiento se demuestra tanto para movimientos pequeños como grandes. Para movimientos grandes, se utilizan frecuencias más altas y un factor de amplificación grande (Chavali, G. K., 2014). La ampliación de movimiento utiliza un filtrado temporal con la banda de paso amplia. Además, a veces se utiliza un filtro IIR de orden bajo (del orden de 1, 2) (Hao-Yu Wu, et. al., 2012).

Para la ampliación del color se utiliza para averiguar el flujo sanguíneo en la cara, que es invisible a simple vista. Así, sin ningún contacto físico, se calcula la frecuencia del pulso. El proceso de aumento de color es el mismo que el de aumento de movimiento. La ampliación del color varía con la ampliación del movimiento con la elección del filtro temporal y la estructura piramidal que utiliza para la descomposición espacial. La ampliación de color utiliza estructuras piramidales de Gauss para la descomposición espacial. El uso de pirámides de Gauss en la ampliación del color se realiza ya que reducen el ruido de cuantificación y potencian los cambios de color en el pulso (Hao-Yu Wu, et. al., 2012). En general, se utilizó un filtro de banda de paso estrecho (Hao-Yu Wu, et. al., 2012).

Los aumentos de movimiento y color de los videos brindan una plataforma para observar los cambios que se están produciendo en el rostro sin ningún contacto físico. El aumento de movimiento ayuda a observar los cambios que ocurren en las microexpresiones y el aumento de color ayuda a observar la frecuencia del pulso del sujeto bajo prueba (Chavali, G. K., 2014).

En el artículo The design and development of a Lie Detection System using facial micro-expressions (M. Owayjan, et al., 2012) presenta el diseño y desarrollo de un sistema de detección de mentiras utilizando microexpresiones faciales. Es un sistema de visión automatizado diseñado e implementado usando LabVIEW. Se utiliza un sistema de visión integrado (EVS) para capturar la entrevista del sujeto. (M. Owayjan, et al., 2012)

Los seres humanos transmiten mensajes de forma voluntaria e involuntaria utilizando sus rostros. Hay ocho expresiones faciales básicas, la cuales son: ira, desprecio, disgusto, miedo, felicidad, alegría, tristeza y sorpresa; y estas expresiones están codificadas como combinaciones de Unidades de Acción (AU) de diferentes músculos de la cara de acuerdo con el Sistema de Codificación de Acción Facial (FACS) desarrollado por Ekman (P. Ekman & W. V. Friesen, 1977) (M. Owayjan, et al., 2012).

En este artículo (M. Owayjan, et al., 2012) se implementó un sistema de visión por computadora capaz de analizar detalladamente las expresiones faciales dentro de un marco activo y dinámico. El sistema analiza las expresiones faciales mediante la observación de articulaciones significativas del rostro del sujeto en una secuencia de fotogramas extraídos de un video. La emoción humana basada en microexpresiones faciales es un tema importante de investigación en psicología. Se cree que el sistema que se desarrollado en este artículo puede ser útil en muchas áreas donde se necesita

interpretación psicológica, como en interrogatorios policiales, seguridad del aeropuerto y del territorio nacional, empleo y pruebas clínicas. (M. Owayjan, et al., 2012)

En el artículo Towards Reading Hidden Emotions: A Comparative Study of Spontaneous Micro-expression Spotting and Recognition Method (X. Li *et al*., 2018) usaron un modelo para el reconocimiento de microexpresiones basado en el contraste de diferencia de características (FD) y la detección de picos, el cual no requiere entrenamiento. Desarrollaron un framework avanzado para el reconocimiento de ME. Este nuevo framework logra un rendimiento mucho mejor que los trabajos anteriores porque: primero, emplearon el método de aumento de video Euleriano para el aumento de movimiento para contrarrestar la baja intensidad de los ME; y en segundo lugar, investigaron tres descriptores de características diferentes (Local Binary Pattern (LBP), histogramas de gradientes orientados (HOG) e histogramas de orientación de gradiente de imagen (HIGO)) y sus combinaciones en tres planos ortogonales para esta tarea (X. Li *et al*., 2018).

Los estudios muestran (P. Ekman & W. V. Friesen, 1969) que los EM espontáneos ocurren involuntariamente, y que los productores de los EM normalmente ni siquiera se dan cuenta de que han presentado tal emoción. Al pedir a las personas que realicen expresiones lo más rápido posible, se pueden obtener MEs posadas, pero son diferentes de las espontáneas tanto en propiedades espaciales como temporales (Wen-Jing Yan, Q. Wu, J. Liang, Y.-H. Chen & X. Fu, 2013) (S. Porter, 2008) (X. Li *et al*., 2018).

Los autores usaron las bases de datos de microexpresiones faciales espontáneas SMIC (X. Li, et al., 2013) y CASME II (Yan WJ, et al., 2014), y demostraron que el método propuesto por ellos supera el estado del arte en ambas bases de datos. Propusieron el primer sistema automático de análisis ME que primero detecta y luego reconoce ME. Supera a los humanos en el reconocimiento de ME por un margen significativo y se desempeña de manera comparable a los humanos en la tarea combinada de detección y reconocimiento de ME. Este método tiene muchas aplicaciones potenciales, como la detección de mentiras, la aplicación de la ley y la psicoterapia. Una limitación del método actual es que utiliza un intervalo fijo para detectar los puntos de tiempo pico, otra limitación del método de detección de ME es que los movimientos que no son de ME, como el parpadeo de los ojos, deben descartarse de los casos de ME reales, y una posible solución es combinar la detección de AU con el proceso de FD y la tercera limitación es que los conjuntos de bases de datos ME disponibles actualmente son aún limitados considerando el tamaño de los datos y el contenido del video (X. Li *et al*., 2018).

Capítulo III

3. Marco Teórico

3.1. Teoría de las emociones

“Emotions are a process, a particular kind of automatic appraisal influenced by our evolutionary and personal past, in which we sense that something important to our welfare is occurring, and a set of psychological changes and emotional behaviors begins to deal with the situation." - Paul Ekman, PhD (P. Ekman, 2021)

En otras palabras, las emociones nos preparan para afrontar eventos importantes sin tener que pensar en ellos. Estas respuestas emocionales ocurren de forma espontánea, lo que significa que no elegimos sentirlas, simplemente nos suceden automáticamente. De todas las emociones humanas que experimentamos, hay siete emociones universales que todos sentimos (Felicidad, Tristeza, Enfado, Disgusto, Sopresa, Miedo, Desprecio), que trascienden las diferencias lingüísticas, regionales, culturales y étnicas (P. Ekman, 2021).

Estudiar las emociones desde las microexpresiones faciales también funciona como una herramienta para descifrar el estado emocional actual de una persona con facilidad. En cualquier momento en particular, la emoción que siente una persona solo puede descifrarse a través de la expresión expresada por esa persona. Las 80

contracciones musculares faciales de una persona y sus combinaciones dan lugar a miles de expresiones (Chavali, G. K., 2014).

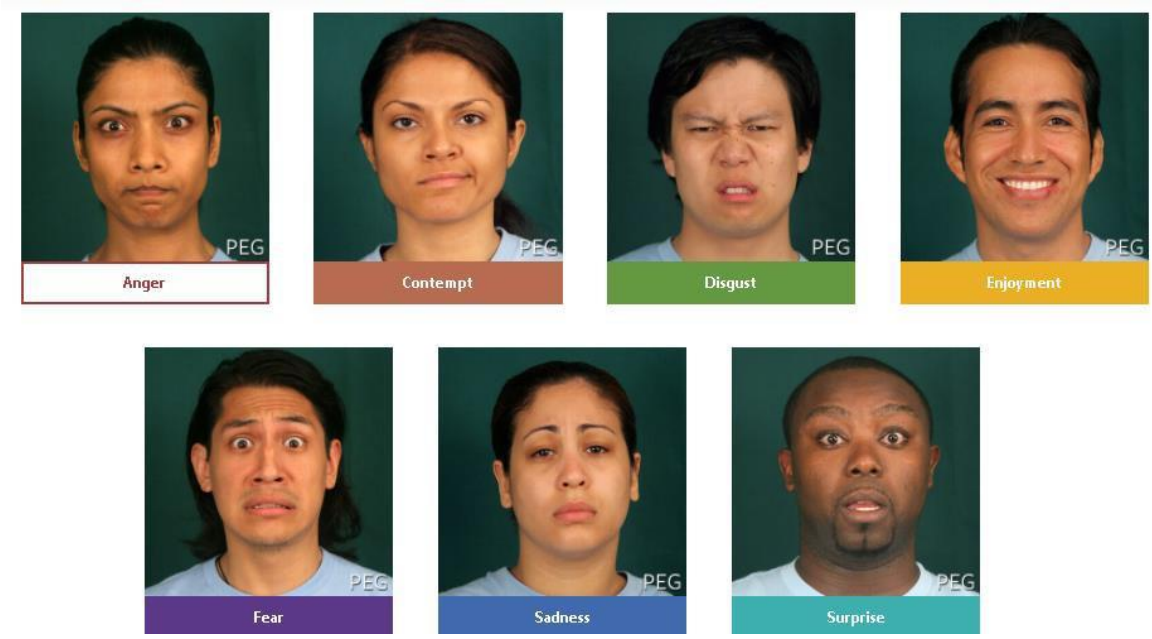


Figura 1 Las 7 Emociones Universales. Fuente (P. Ekman, 2021)

Cada una de las emociones universales tiene señales, fisiologías y líneas de tiempo distintivas (Chavali, G. K., 2014).

3.1.1. Las 7 Emociones Universales

Cada emoción o familia de emociones tiene una variedad de expresiones asociadas, pero visualmente no idénticas. Por ejemplo, la ira tiene 60 expresiones visualmente no idénticas con propiedades centrales iguales (P. Ekman, 1993). Esta propiedad central diferencia a la familia de la ira de la familia del miedo. Una familia de emociones se distingue de otra familia de emociones basándose en 8 características diferentes (P. Ekman, 1992). Las emociones universales también se denominan emociones básicas.

Todo investigador de las emociones coincidió en la universalidad de seis emociones básicas: ira, disgusto, tristeza, alegría, miedo y sorpresa. En los últimos años, hay una adición más de emoción universal llamada desprecio (P. Ekman, 1992). (Chavali, G. K., 2014)

3.1.1.1. Enfado

La respuesta cuando una persona se siente molesta o la respuesta cuando una persona es atacada o lastimada por alguien. La ira también puede ser una respuesta desarrollada a partir del odio extremo (D. Cordaro & P. Ekman, 2011). La emoción de ira está representada en la Figura 1.

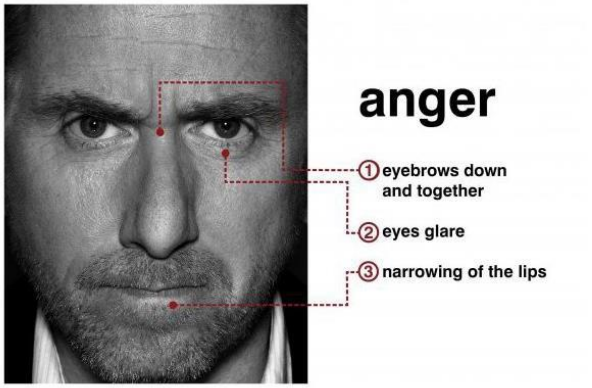


Figura 2 Microexpresión facial de enfado. Fuente (Babich, N., 2016)

3.1.1.2. Disgusto

Es la respuesta cuando una persona se siente repulsivamente provocada por algo ofensivo o repugnante (D. Cordaro & P. Ekman, 2011). La emoción de disgusto está representada en la Figura 3.

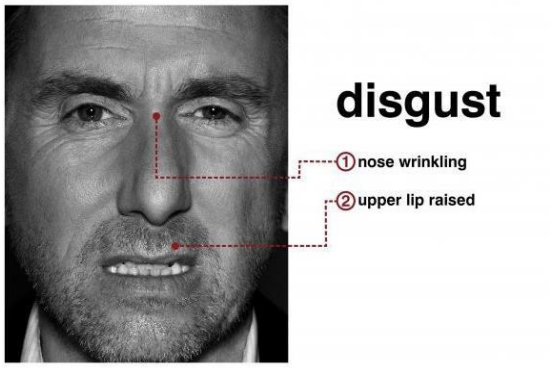


Figura 3 Microexpresión facial de disgusto. Fuente (Babich, N., 2016)

3.1.1.3. Miedo

La respuesta cuando una persona siente una amenaza, daño o dolor (D. Cordaro & P. Ekman, 2011). La emoción del miedo está representada en la Figura 4.

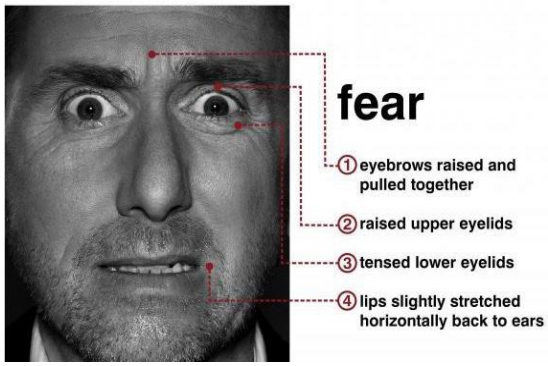


Figura 4 Microexpresión facial de miedo. Fuente (Babich, N., 2016)

3.1.1.4. Felicidad

La respuesta cuando una persona se siente contenta, feliz o placentera (D. Cordaro & P. Ekman, 2011). La emoción de la felicidad está representada en la Figura 5.

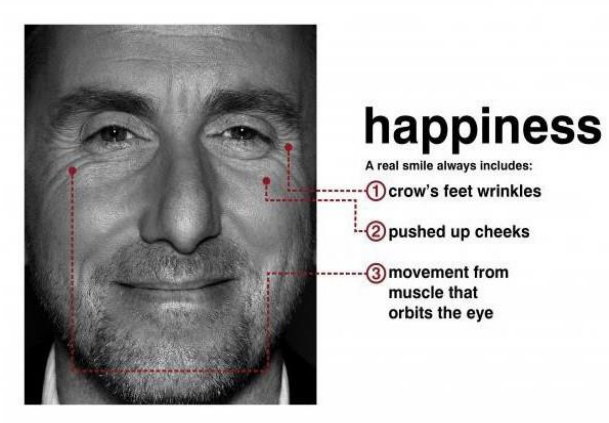
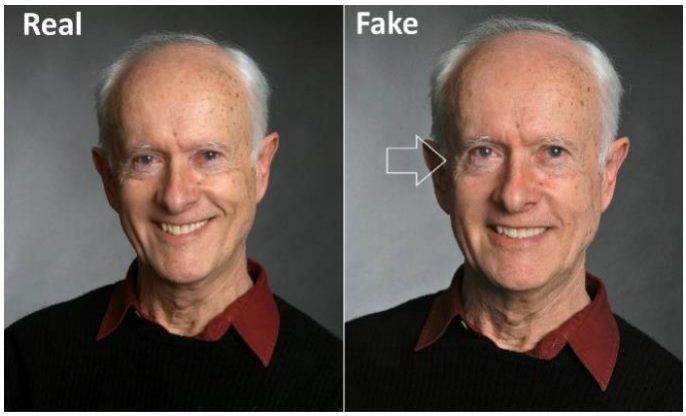


Figura 5 Microexpresión facial de felicidad. Fuente (Babich, N., 2016)

Los músculos de los ojos laterales pueden detectar la felicidad falsa. La siguiente imagen muestra la felicidad real y otra falsa. (Babich, N., 2016)



Fuente (Babich, N., 2016)

3.1.1.5. Tristeza

La respuesta cuando una persona se siente infeliz o pierde a alguien o algo (D. Cordaro & P. Ekman, 2011). La emoción de la tristeza está representada en la Figura 6.

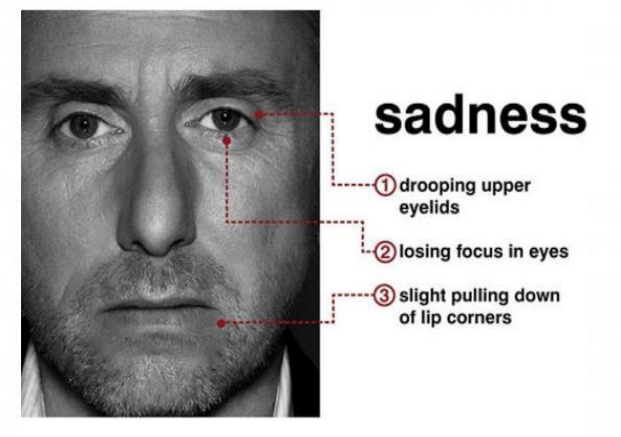


Figura 6 Microexpresión facial de tristeza. Fuente (Babich, N., 2016)

3.1.1.6. Sorpresa

La respuesta cuando una persona siente algo repentino e inesperado (D. Cordaro & P. Ekman, 2011). La emoción de la sorpresa está representada en la Figura 7.

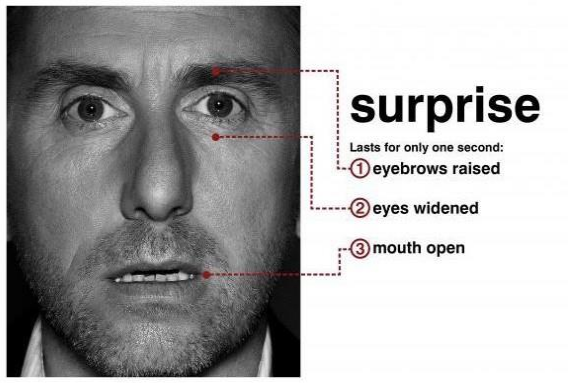


Figura 7 Microexpresión facial de sorpresa. Fuente (Babich, N., 2016)

3.1.1.7. Desprecio

La respuesta cuando una persona se siente superior a otra (D. Cordaro & P. Ekman, 2011). La emoción del desprecio está representada en la Figura 8.

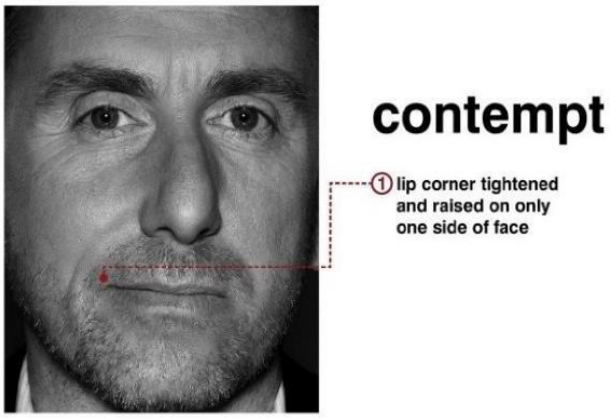


Figura 8 Microexpresión facial de desprecio. Fuente (Babich, N., 2016)

Cada familia de emociones contiene una gran cantidad de expresiones que las representan, pero el desprecio es la única emoción que tiene solo dos expresiones. Así, la familia del desprecio es muy limitada (D. Matsumoto & P. Ekman, 2004). Los datos necesarios para trabajar el desprecio también son bastante insuficientes, ya que la evolución de esta emoción es muy reciente. A partir de ahora la emoción de desprecio no se considera desde hace algún tiempo (Chavali, G. K., 2014).

3.2. Expresiones faciales

Las expresiones son manifestaciones externas de los cambios internos que están ocurriendo u ocurrieron en la mente (P. Ekman, 1997). Estas expresiones son firmas de los cambios en la mente. Identificar una expresión es muy fácil ya que implica los

correspondientes cambios musculares en una región particular de la cara (Chavali, G. K., 2014).

.

Las expresiones son generalmente categorizadas en tres tipos:

1. **Macro Expressions**: El tipo general de expresiones que ocurren en 4 a 5 segundos de tiempo (Chavali, G. K., 2014).
2. **Micro Expressions**: Expresiones involuntarias que ocurren en un abrir y cerrar de ojos. La duración de este tipo de expresiones es de 1/5 de segundo a 1/25 de segundo (Chavali, G. K., 2014).
3. **Subtle Expressions:** Expresiones involuntarias que solo dependen de la intensidad de la expresión más que de la duración (Chavali, G. K., 2014).

3.3. Micro expresiones faciales

Las microexpresiones fueron descubiertas por primera vez por los investigadores Haggard and Isaacs. Dr. Paul Ekman popularizó el término "**microexpresión**" y amplió enormemente la investigación. Ekman descubrió que en todos estos países la gente expresaba e identificaba las 7 emociones universales de la misma manera. Incluso se aventuró a una tribu remota y primitiva llamada Fore en Papau Nueva Guinea y descubrió que expresaban las mismas emociones que nosotros. El rostro es el mejor indicador de las emociones de una persona. (Van Edwards, V., 2014)

Las microexpresiones faciales son movimientos del rostro muy breves, generadas entre 1/15 y 1/25 segundos; por ser tan breves pueden revelar verdades que el ojo humano no logra percibir.

3.4. Emoción espontánea

Cuando se trata de emociones espontáneas naturales, la variación en la expresión facial a lo largo de la emoción puede ser impredecible. Estas respuestas emocionales ocurren de forma espontánea, lo que significa que no elegimos sentirlas, simplemente nos suceden automáticamente (P. Ekman, 2021). Las microexpresiones espontáneas son muy sutiles.

3.5. La mentira

Actualmente vivimos en la sociedad de la post verdad, los límites entre la verdad y la mentira están diluidos. Estamos rodeados de fakes news, opiniones que se disfrazan de verdad y noticias sin contrastar. Por tanto, entendemos que las mentiras son expresiones que se emiten voluntaria y deliberadamente. Por contra una persona que “falta a la verdad”, pero que no es consciente de estar haciéndolo, no estaría mintiendo, sino que está exponiendo su propio error (Montejano, S., 2020).

La forma obvia de percibir a una persona mientras está mintiendo es mirar más de cerca para que el mentiroso falle. Evidentemente, si se captan las razones por las cuales una mentira puede fallar, la gente puede acercarse más para atrapar a un

mentiroso. Un mentiroso o una mentira pueden fallar por dos razones (P. Ekman, 1989, p.71–81).

Pensamiento: Porque el mentiroso no se ha preparado lo suficiente o porque un mentiroso no pudo resolver la situación.

Sentimientos: Porque el mentiroso no pudo administrar sus emociones.

Por lo tanto, cuando las personas son lo suficientemente cautelosas y comprenden las razones del fracaso del mentiroso, pueden atrapar fácilmente a un mentiroso**.** Paul Ekman, un científico famoso y pionero en los estudios de las expresiones faciales y las emociones, encontró tres técnicas más para detectar una mentira (P. Ekman, 1997, p.93). En general, las señales de comportamiento no están bajo el control consciente de ninguna persona. Observar y comprender las señales de comportamiento de un mentiroso, que se filtran sin su propio conocimiento, determina y dice mucho sobre esa persona. Este estudio se centra en unos pocos casos en los que el propio comportamiento no verbal de una persona, como **las microexpresiones**, revela la emoción subyacente a pesar de que el mentiroso trata verbalmente de ocultar o enmascarar la verdad (Chavali, G. K., 2014).

Según Darwin, hay pocos músculos que sean imposibles de activar voluntariamente y estos músculos revelan las verdaderas intenciones de los demás (P. Ekman, 2009). Hay 7 características como la duración, la simetría, la velocidad de aparición, etc., que distinguen las expresiones faciales voluntarias de las involuntarias (P. Ekman, 2003).

Un mentiroso puede provocar una señal emocional que delata la plausibilidad de la mentira en sí (P. Ekman, 1989, p.71–81). Las microexpresiones de emociones están típicamente representadas en todas estas clases involuntarias (Chavali, G. K., 2014).

La detección de mentiras ha sido un tema imperecedero y en evolución. Las técnicas de polígrafo han sido la técnica más popular y exitosa hasta la fecha. El principal inconveniente del polígrafo es que no se pueden obtener buenos resultados sin mantener un contacto físico con el sujeto bajo prueba. En general, este contacto físico induciría una conciencia extra en el sujeto. Además, cualquier tipo de excitación en el sujeto desencadena falsos positivos al realizar las pruebas tradicionales basadas en el polígrafo. Con todos estos inconvenientes en el polígrafo, también, debido a los rápidos desarrollos en los campos de la visión por computadora y la inteligencia artificial, con algoritmos más nuevos y rápidos, han obligado a la humanidad a buscar y adaptarse a los métodos contemporáneos en la detección de mentiras.

Observar las expresiones faciales de las emociones en una persona sin ningún contacto físico e implementar estas técnicas utilizando inteligencia artificial es uno de esos métodos. El concepto de magnificar una microexpresión e intentar descifrarlas es bastante prematuro en esta etapa, pero evolucionaría en el futuro (Chavali, G. K., 2014).

3.6. Deep learning

Las expresiones faciales son una de las más potentes, naturales y señales universales por medio del cual los seres humanos transmiten sus emociones, estados e intenciones. Varios Sistemas de Reconocimiento de Expresiones Faciales (FER) han sido explorado para codificar información de expresión a partir de representaciones faciales. Los sistemas FER se pueden dividir en dos categorías principales de acuerdo con las representaciones de características: imagen estática FER e imagen dinámica FER. A pesar de la poderosa capacidad de aprendizaje del deep learning, los problemas persisten cuando se aplica a FER, primero Deep Neuronal Network requiere una gran cantidad de datos de entrenamiento para evitar el sobreajuste, adicionalmente las variaciones en la pose del sujeto, iluminación y oclusión son los problemas más comunes en escenarios de expresiones faciales. (S. Li & W. Deng, 2020)

3.6.1. Convolutional Neural Networks (CNNs)

Una red neuronal convolucional puede disponer de decenas o cientos de capas que aprenden a detectar diferentes características de una imagen. Se aplican filtros a cada imagen de entrenamiento con distintas resoluciones, y la salida de cada imagen convolucionada se emplea como entrada para la siguiente capa. Los filtros pueden variar desde características muy simples, tales como el brillo y los bordes, hasta más complejas, como las características que definen el objeto de manera única (Matlab, 2021).

La red neuronal de convolución (CNN), a menudo llamada ConvNet, tiene una arquitectura de alimentación directa profunda y tiene una capacidad asombrosa para generalizar de una mejor manera en comparación con las redes con capas completamente conectadas (Nebauer, C. 1998). (Fieres, J., Schemmel, J., & Meier, K., 2006) describe a CNN como el concepto de detectores de características jerárquicas de una manera inspirada biológicamente. Puede aprender características muy abstractas y puede identificar objetos de manera eficiente (Zhang, Z., 2016). Las razones considerables por las que CNN se considera por encima de otros modelos clásicos son las siguientes. En primer lugar, el interés clave para la aplicación de CNN radica en la idea de utilizar el concepto de reparto de peso, por lo que el número de parámetros que necesitan entrenamiento se reduce sustancialmente, lo que se traduce en una mejor generalización (Arel, I., Rose, D. C., & Karnowski, T. P., 2010). Debido a los parámetros menores, la CNN se puede entrenar sin problemas y no sufre un sobreajuste (Timoshenko, D., & Grishkin, V., 2013). En segundo lugar, la etapa de clasificación se incorpora con la etapa de extracción de características (LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P., 1998), ambas utilizan el proceso de aprendizaje. En tercer lugar, es mucho más difícil implementar grandes redes usando modelos generales de redes neuronales artificiales (ANN) que implementando en CNN (Tivive, F. H. C., & Bouzerdoum, A., 2005). Las CNN se utilizan ampliamente en varios dominios debido a su notable rendimiento (Wang, J., Lin, J., & Wang, Z., 2016), como la clasificación de imágenes ((Krizhevsky, A., Sutskever, I., & Hinton, G. E., 2012); (Zeiler, M. D., & Fergus, R., 2013); (Donahue, J., Jia, Y., Vinyals, O., Hoffman, J.,

Zhang, N., Tzeng, E., & Darrell, T., 2014), la detección de objetos (Szegedy, C., Toshev, A., & Erhan, D., 2013), la detección de rostros (Timoshenko, D., & Grishkin, V., 2013)., el reconocimiento de voz (Sainath, T. N., Kingsbury, B., Mohamed, A. R., Dahl, G. E., Saon, G., Soltau, H., Ramabhadran, B., 2013), reconocimiento de vehículos (Luo, X., Shen, R., Hu, J., Deng, J., Hu, L., & Guan, Q., 2017), retinopatía diabética (Pratt, H., Coenen, F., Broadbent, D. M., Harding, S. P., & Zheng, Y. , 2016), reconocimiento de expresiones faciales (Uçar, A., 2017) y muchos más. (Indolia, S., Goswami, A. K., Mishra, S. P., & Asopa, P., 2018).

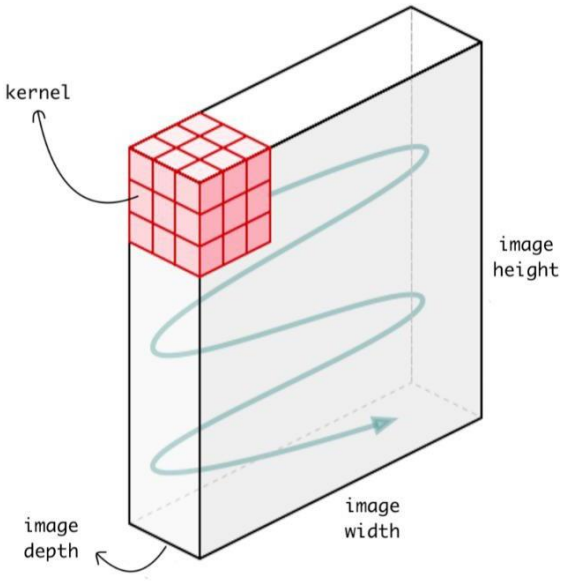


Figura 9 Kernel deslizándose sobre la imagen. Fuente (Verma, S., 2019)

La Figura 9 es una red neuronal de convolución estándar que se introdujo por primera vez en la arquitectura Lenet-5. Conv2D se utiliza generalmente en datos de imagen. Se llama CNN bidimensional porque el núcleo se desliza a lo largo de dos

dimensiones en los datos. Toda la ventaja de usar CNN es que puede extraer las características espaciales de los datos usando su kernel, lo que otras redes no pueden hacer. Por ejemplo, CNN puede detectar bordes, distribución de colores, etc. en la imagen, lo que hace que estas redes sean muy sólidas en la clasificación de imágenes y otros datos similares que contienen propiedades espaciales. (Verma, S., 2019)

3.6.1.1. Arquitecturas de Redes Neuronales Convolucionales CNN

Las CNN se han utilizado ampliamente en una variedad de aplicaciones de visión por computadora, incluido FER. A principios del siglo XXI, varios estudios de la literatura sobre FER (B. Fasel, 2002), (F. Beat, 2002) determinaron que las CNN funcionan bien en los cambios de ubicación de la cara, así como en las variaciones de escala. También se descubrió que funcionan mejor que el perceptrón multicapa (MLP) cuando se observan variaciones de la pose de la cara que no se habían visto anteriormente. Los investigadores utilizaron CNN para ayudar a resolver varios problemas de reconocimiento de expresiones faciales, como la traducción, la rotación, la independencia del sujeto y la invariancia de escala (M. Matsugu, K. Mori, Y. Mitari & Y. Kaneda, 2003)(S. Singh & F. Nasoz, 2020).

Las redes neuronales convolucionales aprovechan el hecho de que la entrada consiste en imágenes y restringen la arquitectura de una manera más sensible. En particular, a diferencia de una red neuronal normal, las capas de una ConvNet tienen neuronas dispuestas en 3 dimensiones: ancho, alto, profundidad. (Tenga en cuenta que la palabra profundidad aquí se refiere a la tercera dimensión de un volumen de

activación, no a la profundidad de una red neuronal completa, que puede referirse al número total de capas en una red) (cs231n, 2021).

Al igual que otras redes neuronales, una CNN está compuesta por una capa de entrada, una capa de salida y muchas capas intermedias ocultas (cs231n, 2021).

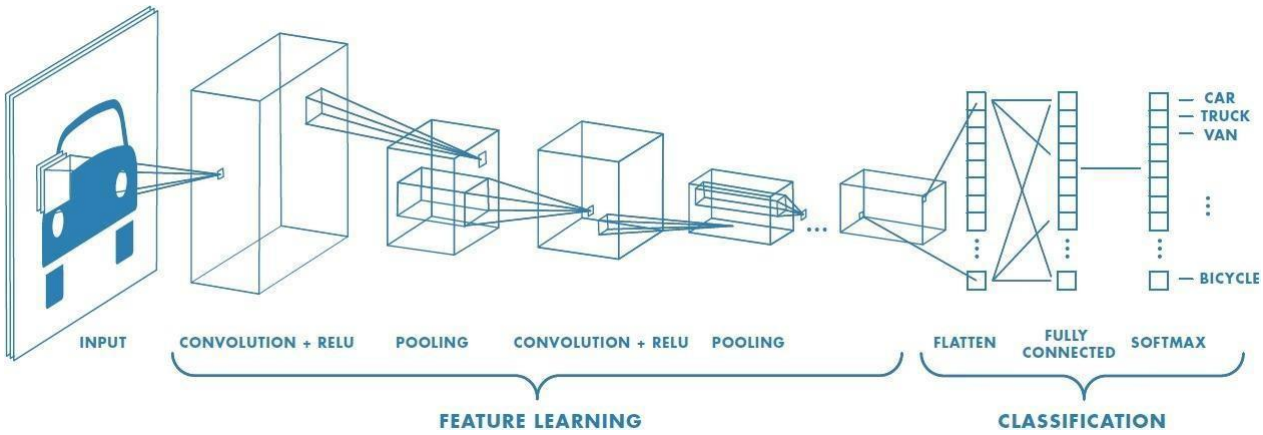


Figura 10 Las redes neuronales convolucionales. Fuente: (Matlab, 2021).

Estas capas realizan operaciones que alteran los datos con el objetivo de aprender características específicas de dichos datos. Las 3 capas más frecuentes son: convolución, activación o ReLU, y pooling (Matlab, 2021).

Convolución: somete las imágenes de entrada a un conjunto de filtros convolucionales, cada uno de los cuales activa ciertas características de las imágenes (Matlab, 2021).

3.6.1.1.1. Layer Patterns

La forma más común de una arquitectura ConvNet apila algunas capas CONV-RELU, las sigue con capas POOL y repite este patrón hasta que la imagen se ha fusionado espacialmente a un tamaño pequeño. En algún momento, es común realizar la transición a capas completamente conectadas. La última capa completamente conectada contiene la salida, como las puntuaciones de la clase. En otras palabras, la arquitectura de ConvNet más común sigue el patrón: (cs231n, 2021).

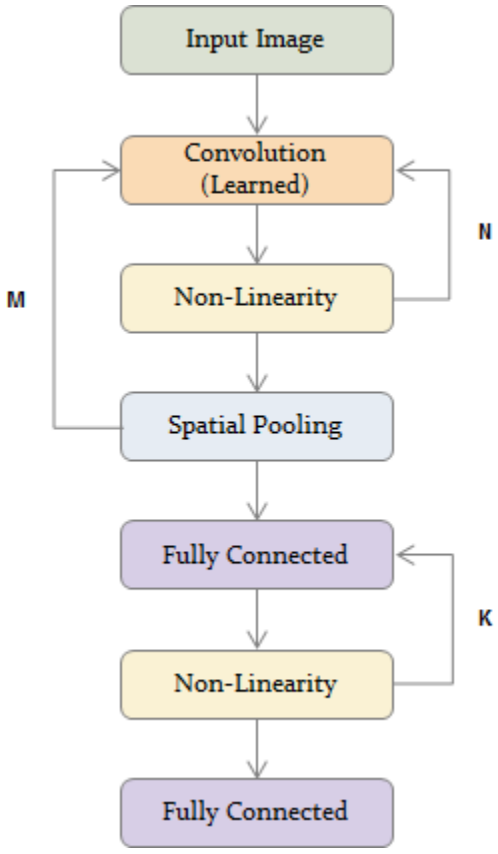


Figura 11 Layer patterns. Fuente: (R., 2017)

3.6.1.1.2. Dropout

La capa Dropout es una máscara que anula la contribución de algunas neuronas hacia la siguiente capa y deja sin modificar todas las demás. Podemos aplicar una capa Dropout al vector de entrada, en cuyo caso anula algunas de sus características; pero también podemos aplicarlo a una capa oculta, en cuyo caso anula algunas neuronas ocultas.

Las capas dropouts son importantes en el entrenamiento de CNN porque evitan el sobreajuste de los datos de entrenamiento. Si no están presentes, el primer lote de muestras de entrenamiento influye en el aprendizaje de una manera desproporcionadamente alta. Esto, a su vez, evitaría el aprendizaje de características que aparecen solo en muestras o lotes posteriores (Baeldung, 2020).

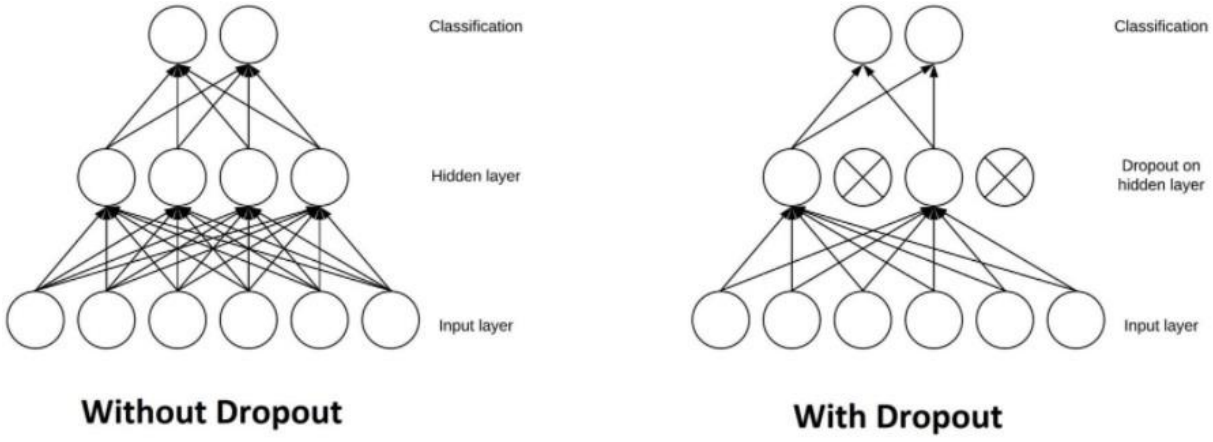


Figura 12 Dropout. Fuente: (Baeldung, 2020).

3.6.1.1.3. Max pooling

Pooling simplifica la salida al disminuir la tasa de muestreo no lineal, reduciendo así el número de parámetros que la red necesita aprender (Matlab, 2021). La agrupación es una característica comúnmente incorporada en las arquitecturas de redes neuronales convolucionales (CNN). La idea principal detrás de una capa de pooling es "acumular" características de mapas generados mediante la convolución de un filtro sobre una imagen. Formalmente, su función es reducir progresivamente el tamaño espacial de la representación para reducir la cantidad de parámetros y computación en la red. La forma más común de agrupación es la agrupación máxima (DeepAI., 2020).

Max pooling se realiza en parte para ayudar a sobreajustar al proporcionar una forma abstracta de la representación. Además, reduce el costo computacional al reducir el número de parámetros a aprender y proporciona invariancia de traducción básica a la representación interna. La agrupación máxima se realiza aplicando un filtro máximo a (normalmente) subregiones no superpuestas de la representación inicial. Las otras formas de agrupación son: promedio, general (DeepAI., 2020).

Figura 13 Max pooling. Fuente: (DeepAI., 2020).

3.6.1.1.4. Data Augmentation

El aumento de datos es uno de los métodos de regularización. El propósito de la regularización es evitar sobreajustar el modelo creado. La regularización mejorará el rendimiento del modelo en datos que nunca se han visto. El aumento de datos es una técnica de manipulación de datos sin perder el núcleo o la esencia de los datos. El aumento de datos se realiza para obtener más datos que los datos originales (datos sintéticos) (A. Budhiman, S. Suyanto & A. Arifianto, 2019). En la Figura 14 se puede ver un rostro rotado aleatoriamente 20 ° grados para una imagen tomada de la base de datos de microexpresiones faciales CASME II.

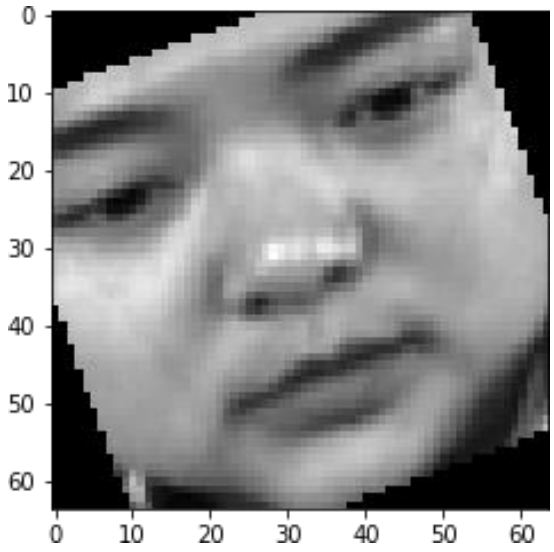


Figura 14 Imagen rotada con data augmentation para la base de datos CASME II

3.6.2. Redes Neuronales Convolucionales 3D CNN

Las redes 3D pueden aprender el espacio y características temporales en videos simultáneamente, lo que supera las limitaciones de la arquitectura de dos flujos. Las CNN 3D han producido resultados prometedores en Reconocimiento de acciones basado en video. Sin embargo, todos estos métodos basados en 3D CNN solo utilizan datos RGB para el reconocimiento de acciones (H. Wu, X. Ma & Y. Li, 2021).

En las CNN 2D, las convoluciones se aplican en los mapas de características 2D para calcular las características de las dimensiones espaciales únicamente. Cuando se aplica a problemas de análisis de video, es deseable capturar la información de movimiento codificada en múltiples cuadros contiguos, por esto las redes neuronales convolucionales 3D se pueden usar para calcular características de las dimensiones espaciales y temporales. En la Figura 15 se muestra la comparación de convoluciones 2D (a) y 3D (b). En (b) el tamaño del núcleo de convolución en el temporal la dimensión es 3, y los conjuntos de conexiones están codificados por colores para que los pesos compartidos sean del mismo color (S. Ji, W. Xu, M. Yang & K. Yu, 2013).

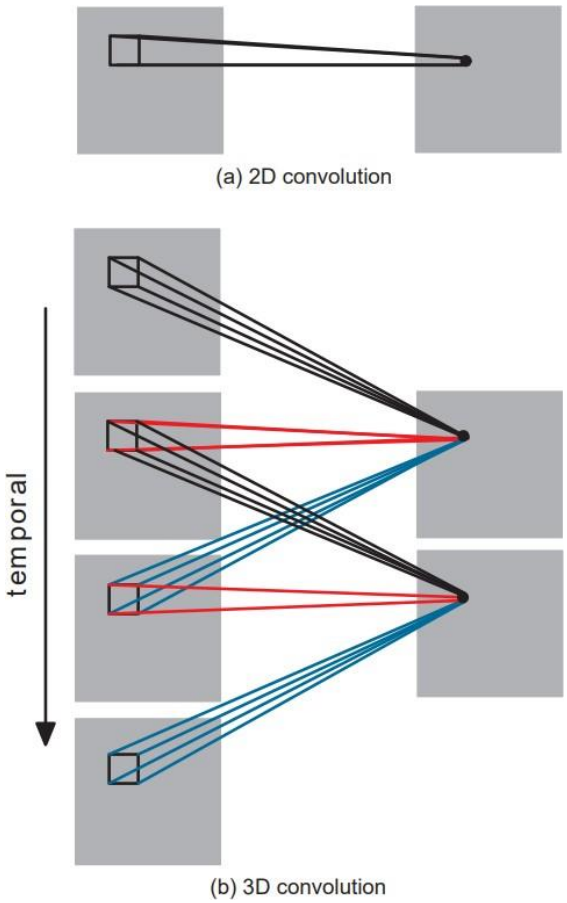


Figura 15 Comparación de convoluciones 2D (a) y 3D (b). En (b) el tamaño del núcleo de convolución en el temporal la dimensión es 3, y los conjuntos de conexiones están codificados por colores para que los pesos compartidos sean del mismo color. En 3D convolución, se aplica el mismo kernel 3D a la superposición Cubos 3D en el video de entrada para extraer características de movimiento. Fuente: (S. Ji, et al., 2013)

En Conv3D, el núcleo se desliza en 3 dimensiones como se muestra en Figura 16. Kernel deslizándose sobre datos 3D Conv3D se utiliza principalmente con datos de imágenes 3D. Como datos de imágenes de resonancia magnética (IRM). Los datos de resonancia magnética se utilizan ampliamente para examinar el cerebro, la médula espinal, los órganos internos y muchos más. Una tomografía computarizada (TC)

también es un ejemplo de datos 3D, que se crea combinando una serie de imágenes de rayos X tomadas desde diferentes ángulos alrededor del cuerpo. Podemos usar Conv3D para clasificar estos datos médicos o extraer características de ellos. (Verma, S., 2019)

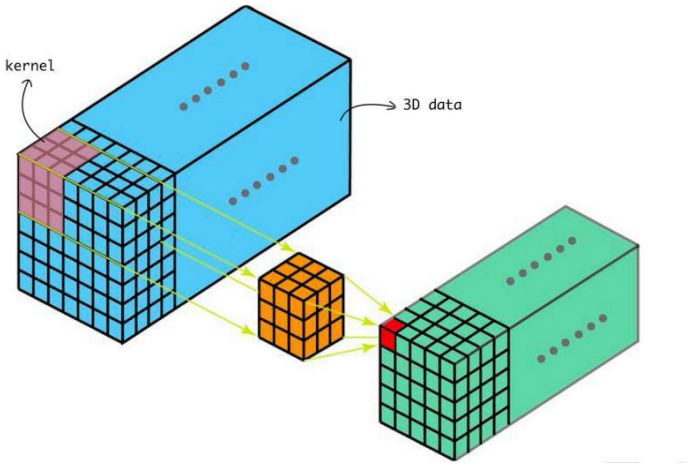


Figura 16 Kernel deslizándose sobre datos 3D. Fuente (Verma, S., 2019)

3.6.3. Redes de memoria a corto/largo plazo LSTM

Las redes neuronales LSTM son una extensión de las Redes Neuronales Recurrentes RNN. Las RNN son un tipo de redes capaces de reconocer y predecir secuencias de datos a lo largo del tiempo, como textos, genomas, discurso hablado o series numéricas. Este tipo de redes se fundamentan en bucles que permiten que la salida de la red o de una parte de ella en un momento dado sirva como entrada de la propia red en el siguiente momento (Mañas, A. M., 2021).

Las RNNs tienen un contratiempo importante conocido como **problema del desvanecimiento del gradiente**; es decir, tienen dificultades para aprender dependencias de largo alcance. Cuando se realiza la propagación hacia atrás, es decir, nos movemos hacia atrás en la red y calculamos los gradientes de pérdida (error) con respecto a los pesos, los gradientes tienden a ser cada vez más pequeños a medida que nos movemos hacia atrás en la red. Esto significa que las neuronas en las capas anteriores aprenden muy lentamente en comparación con las neuronas en las capas posteriores en la jerarquía. Las capas anteriores de la red son las más lentas de entrenar. Este es un problema en todos los tipos de redes neuronales, pero particularmente es nocivo para redes en dónde lo que se pretende es tener la componente de memoria necesaria para pronóstico de series temporales. Afortunadamente, este problema fue resuelto por Hochreiter y Schmidhuber en 1997 (Hochreiter, Sepp, and Jürgen Schmidhuber, 1997) mediante la creación de las **LSTM**. Las redes de memoria a corto/largo plazo, LSTM, son un tipo especial de RNN, capaz de aprender dependencias a largo plazo (Mañas, A. M., 2021).

Las LSTM están diseñados explícitamente para evitar el problema de dependencia a largo plazo. Recordar información durante largos períodos de tiempo es prácticamente su comportamiento predeterminado. Todas las redes neuronales recurrentes tienen la forma de una cadena de módulos repetitivos de la red neuronal. En las RNN estándar, este módulo de repetición tendrá una estructura muy simple, como una sola capa de activación. Los LSTM también tienen esta estructura tipo cadena, pero el módulo de repetición tiene una estructura diferente. En lugar de tener una sola capa de red

neuronal, hay cuatro que interactúan de una manera muy especial Figura 17 (Mañas, A. M., 2021).

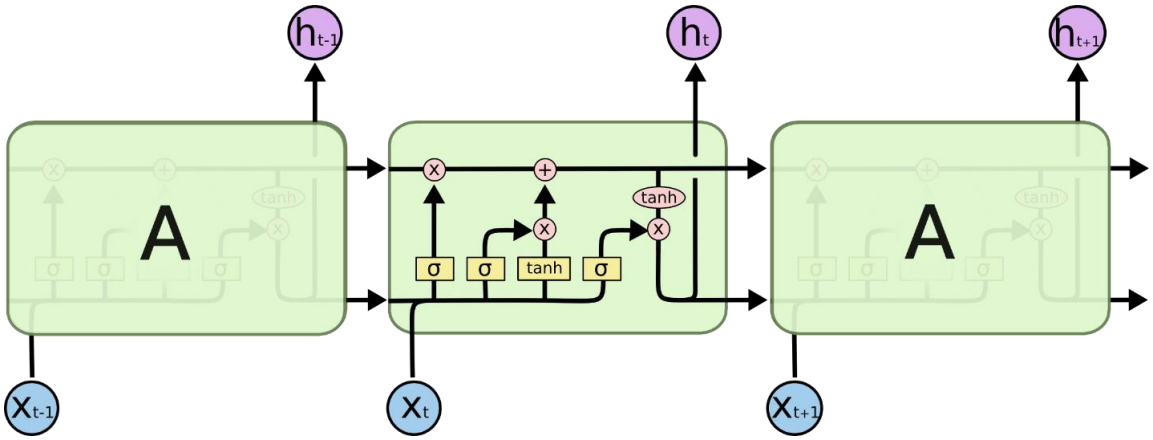


Figura 17 Capas de las celdas LSTM. Fuente (Mañas, A. M., 2021)

La clave de las redes LSTM es el estado de la célula, la línea horizontal que recorre la parte superior del diagrama. El estado de la célula es algo así como una cinta transportadora. Corre hacia abajo por toda la cadena, con solo algunas interacciones lineales menores. Es muy fácil que la información fluya sin cambios. El LSTM tiene la capacidad de eliminar o agregar información al estado de la célula, cuidadosamente regulado por estructuras llamadas compuertas (Mañas, A. M., 2021).

Las puertas son una forma de permitir que la información fluya. Se componen de una capa de red neuronal sigmoidea y una operación de multiplicación puntual. La capa sigmoide produce números entre cero y uno, que describen la cantidad de cada componente que debe dejarse pasar. Un valor de cero significa “no dejar pasar nada”,

mientras que un valor de uno significa “dejar pasar todo”. Un LSTM tiene tres de estas compuertas, para proteger y controlar el estado de la celda (Mañas, A. M., 2021).

Afortunadamente, para utilizar este tipo de red no es necesario implementarlas desde cero, sino que se pueden utilizar distintos frameworks. Como se explicará más adelante, los experimentos que hemos realizado se han basado en Keras sobre TensorFlow.

3.6.4. Redes residuales ResNet

ResNet fue propuesta por Kaiming et al en el año 2015 que introdujo una arquitectura llamada Red residual con amplio uso en visión computacional.

Deep Residual Network (ResNet) es una red neuronal artificial que se crea con el objetivo de superar el problema de menor precisión al crear una ANN simple con una capa más profunda que una ANN menos profunda (K. O’Shea & R. Nash, 2015). En otras palabras, el propósito de la Red Residual Profunda es hacer ANN con capas más profundas con alta precisión (K. He, X. Zhang, S. Ren & J. Sun, 2016). El concepto de la red residual profunda es hacer ANN que pueda actualizar el peso a una capa menos profunda (reducir el gradiente de degradación). El concepto se implementa mediante una "conexión de acceso directo". El concepto de red residual se muestra en la Figura 18 (A. Budhiman, et al., 2019).

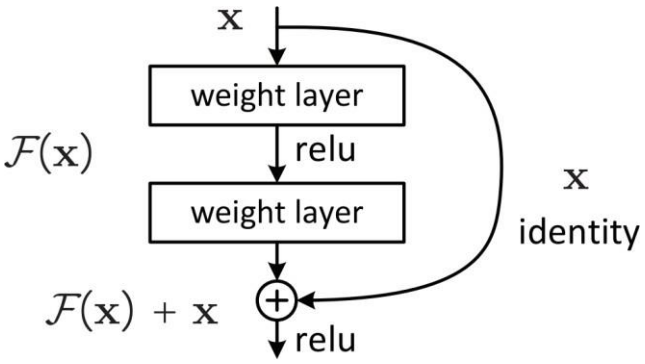


Figura 18 Residual learning: a building block. Fuente (K. He, et al., 2016)

3.6.5. Funciones de activación

Las funciones de activación transforman la suma ponderada de las entradas que se dedica a las neuronas artificiales. Estas funciones deben ser no lineal para codificar patrones complejos de los datos. Las funciones de activación más populares son sigmoide, tanh y RELU. RELU es la función de activación más popular en las redes neuronales profundas. En la Figura 19 se muestran las funciones de activación Sigmoid, Tanh y ReLU con sus respectivas formulas (Carchenilla., 2016).

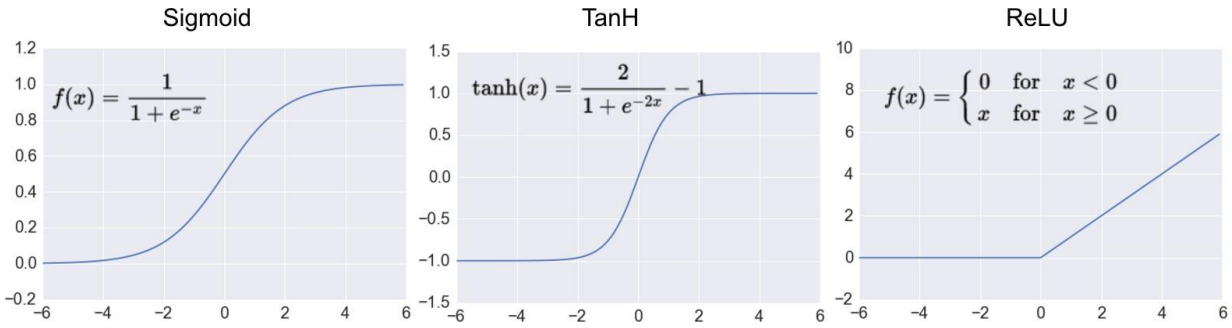


Figura 19 Funciones de activación. Fuente (Carchenilla., 2016)

3.6.5.1. Softmax

$$\sigma(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \quad para \quad i = 1, \dots, K \quad y \quad z = (z_1, \dots, z_K) \quad \in R^{K}$$

Ecuación 1 función de activación Softmax

Muchas redes neuronales multicapa terminan en una penúltima capa que genera puntuaciones de valor real que no se escalan convenientemente y con las que puede ser difícil trabajar. Aquí, el softmax es muy útil porque convierte las puntuaciones en una distribución de probabilidad normalizada, que puede mostrarse a un usuario o usarse como entrada para otros sistemas. Por esta razón, es habitual añadir una función softmax como capa final de la red neuronal. La Ecuación 1 muestra la fórmula para la función de activación Softmax, donde todos los valores z_i son los elementos del vector de entrada y pueden tomar cualquier valor real. El término en la parte inferior de la fórmula es el término de normalización que asegura que todos los valores de salida

de la función sumen 1, constituyendo así una distribución de probabilidad válida (Wood, T., 2020).

3.6.5.2. Función de perdida *Cross Entropy*

$$H(p,q) = -\sum_{x \in X} p(x)\ log\ log\ q\ (x)$$

Ecuación 2 Función de perdida Cross Entropy

La entropía cruzada indica la distancia entre lo que el modelo cree que debería ser la distribución de salida y cuál es realmente la distribución original. Se define como la Ecuación 2. La medida de entropía cruzada es una alternativa ampliamente utilizada del error al cuadrado. Se utiliza cuando se puede entender que las activaciones de nodos representan la probabilidad de que cada hipótesis sea cierta, es decir, cuando el resultado es una distribución de probabilidad. Por lo tanto, se utiliza como función de pérdida en redes neuronales que tienen activaciones S*oftmax* en la capa de salida (Dahal, P. (2021).

3.6.5.3. ReLu: Unidad lineal rectificada

Permite un entrenamiento más rápido y eficaz al asignar los valores negativos a cero y mantener los valores positivos. También se lo denomina *activación*, dado que solo las características activadas pasan a la siguiente capa (Matlab, 2021).

Estas operaciones se repiten en decenas o cientos de capas, de modo que cada capa aprende a identificar diferentes características (Matlab, 2021).

3.6.6. Algoritmos de optimización

Los optimizadores son algoritmos que se utilizan para reducir la función de pérdida y actualizar los pesos en retropropagación. A continuación, se explican los optimizadores que se implementaron para este trabajo de grado. (Choudhary, 2021)

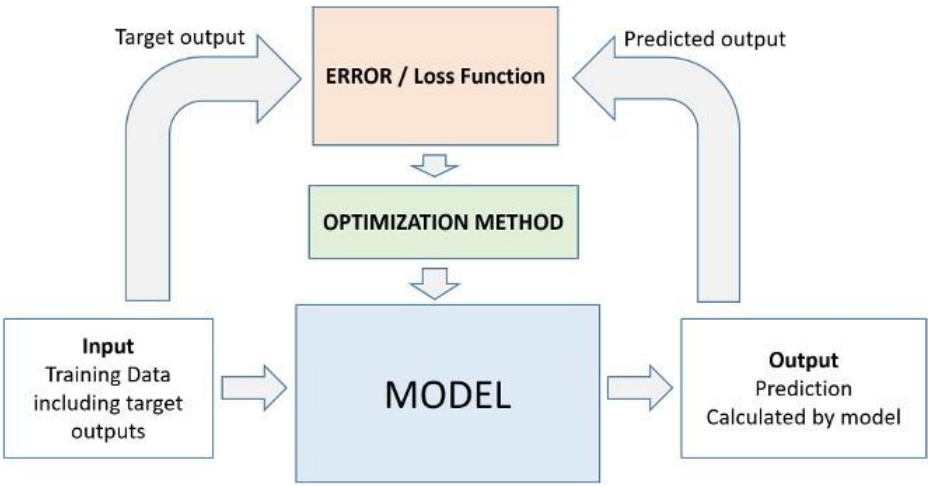


Figura 20 Optimizadores en Deep Learning. Fuente (Choudhary, 2021)

3.6.6.1. Adam

La Estimación del Momento Adaptativo or Adaptive moment estimation por sus siglas en inglés (Kingma, D. P., & Ba, J. L., 2015) es un método que calcula las tasas de aprendizaje adaptativo para cada parámetro. Además de almacenar un promedio en

declive exponencial de gradientes cuadrados v_t como Adadelta y RMSprop, Adam también mantiene un promedio exponencialmente decreciente de gradientes m_t similar al momentum. Mientras que el momentum puede verse como una bola que baja por una pendiente, Adam se comporta como una bola pesada con fricción, que por lo tanto prefiere mínimos planos en la superficie de error. Calculamos los promedios decrecientes y los gradientes cuadrados m_t y v_t respectivamente como sigue (Ruder, S., 2016):

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) gt$$

Ecuación 3

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g^2 t$$

Ecuación 4

m_t y v_t son estimaciones del primer momento (la media) y el segundo momento (la varianza no centrada) de los gradientes respectivamente, de ahí el nombre del método. Como m_t y v_t se inicializan como vectores de ceros, los autores de Adam observan que están sesgados hacia cero, especialmente durante los pasos de tiempo iniciales, y especialmente cuando las tasas de caída son pequeñas (es decir, β_1 y β_2 están cerca de 1) (Ruder, S., 2016).

Contrarrestan estos sesgos calculando estimaciones de primer y segundo momento corregidas por sesgo:

$$m_t = \frac{m_t}{1 - \beta_1^t}$$

Ecuación 5

$$v_t = \frac{v_t}{1 - \beta_2^t}$$

Ecuación 6

Luego, se utilizan para actualizar los parámetros de la siguiente manera: lo que produce la regla de actualización de Adam:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{v_t} + \epsilon} mt$$

Ecuación 7

Los autores proponen valores predeterminados de 0,9 para β_1, 0.999 para β_2, y 10^{-8} para ϵ (Ruder, S., 2016).

3.6.6.2. SGD

El descenso de gradiente estocástico (SGD) es un enfoque simple pero muy eficiente para ajustar clasificadores lineales y regresores bajo funciones de pérdida convexa como máquinas de vectores de soporte (lineales) y regresión logística. Aunque SGD ha existido en la comunidad de aprendizaje automático durante mucho tiempo, recientemente ha recibido una atención considerable en el contexto del aprendizaje a gran escala. SGD se ha aplicado con éxito a problemas de aprendizaje automático dispersos y a gran escala que se encuentran a menudo en la clasificación de texto y el procesamiento del lenguaje natural. Dado que los datos son escasos, los

clasificadores de este módulo se adaptan fácilmente a problemas con más de 10 ^ 5 ejemplos de entrenamiento y más de 10 ^ 5 características. (Scribbr, 2021)

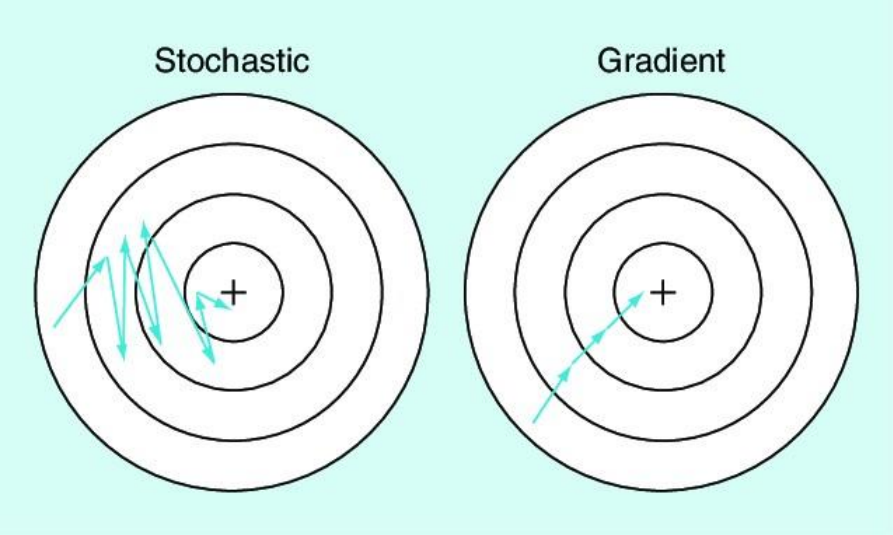


Figura 21 El descenso de gradiente estocástico. Fuente (Choudhary, 2021)

En lugar de tomar datos completos de una vez, en SGD se toma un registro a la vez para alimentar la red neuronal y actualizar los pesos. SGD se actualiza solo una vez, no hay redundancia, es más rápido que GD y menos costoso computacionalmente. SGD se actualiza con mayor frecuencia, la función de costos tendrá oscilaciones severas como podemos ver en la Figura 21. La oscilación de SGD puede saltar a un mejor mínimo local. (Choudhary, 2021)

Capítulo IV

4. Materiales y Métodos

4.1. Preparación de los datos

Existe varias bases de datos de microexpresiones faciales, entre las más reconocidas se encuentran SMIC (X. Li, et al., 2013), USF-HD (M. Shreve, S. Godavarthy, D. Goldgof S. & Sarkar, 2011), CASME (Wen-Jing Yan, et al., 2013), CASME II (Yan WJ, et al., 2014) y SAMM (A. K. Davison, et al., 2018).

	Polikovsky et al. [13]	SMIC [14]	USF-HD [15]	CASME [16]	CASME II [17]	SAMM
Micro-Movements	42	164	100	195	247	159
Participants	10	16	N/A	35	35	32
Resolution	640×480	640×480	720×1280	640×480/720×1280	640×480	2040×1088
Facial Resolution	N/A	190×230	N/A	150×190	280×340	400×400
FPS	200	100	29.7	60	200	200
Spontaneous/Posed	Posed	Spontaneous	Posed	Spontaneous	Spontaneous	Spontaneous
FACS Coded	No	No	No	Yes	Yes	Yes
Emotion Classes	6	3	6	7	5	7
Mean Age (SD)	N/A	26.7 (N/A)	N/A	22.03 (SD = 1.60)	22.03 (SD = 1.60)	33.24 (SD = 11.32)
Ethnicities	3	3	N/A	1	1	13

Figura 22 Resumen de conjuntos de datos disponibles públicamente que contienen microexpresiones faciales.

Fuente: (A. K. Davison, et al., 2018)

Para este trabajo de grado se tuvo acceso a la base de datos de CASME II y SMIC, las cuales se explican a continuación.

4.1.1. Base de datos de microexpresiones faciales CASME II

CASME II (Yan WJ, Li X, Wang SJ, Zhao G, Liu YJ, Chen YH & Fu X., 2014) es una base de datos de microexpresiones faciales con una velocidad de fotogramas de 200 fps y 247 microexpresiones codificadas con FACS de 26 participantes. El área facial utilizada para el análisis es más grande que CASME y SMIC a 280 × 340 píxeles. Este conjunto de datos incluye solo participantes chinos y categoriza de la misma manera. CASME II utiliza participantes en su mayoría estudiantes con una edad media de 22,03 (DE = 1,60). Además de usar solo una etnia, este conjunto de datos usa solo participantes jóvenes, restringiendo el conjunto de datos al análisis de participantes de apariencia similar (según las características de la edad) (A. K. Davison, et al., 2018).

La base de datos CASME II se obtiene mediante un FTP, el cual tiene una carpeta llamada “CASME2_preprocessed_small_Li Xiaobai” que contienen las imágenes pre-procesadas y los rostros en una resolución de 64 x 64 pixeles. Esta base de datos se obtiene de (Prof. Xiaolan Fu’s Group., 2008).

La base de datos CASME II tiene las expresiones faciales de enojo, felicidad y disgusto.

4.1.2. Base de datos de microexpresiones faciales SMIC

El conjunto de datos SMIC (X. Li, T. Pfister, X. Huang, G. Zhao & M. Pietikäinen, 2013) usa una configuración de interrogación para causar un alto estrés en los sujetos. Dado que las microexpresiones a menudo se muestran en situaciones de alto estrés

donde se reprimen las emociones, el entorno de interrogación permitió que ocurrieran microexpresiones naturales y espontáneas (ichi.pro, 2021). Esta base de datos consta de 164 microexpresiones espontáneas filmadas a 100 fps y fue una de las primeras en incluir microexpresiones espontáneas obtenidas a través de experimentos de inducción emocional. Sin embargo, este conjunto de datos no se codificó mediante el Sistema de codificación de acciones faciales (FACS) (P. Ekman & W. V. Friesen, 1987) y no proporciona información sobre secuencias neutrales (la cara del participante no se mueve antes del inicio). Los protocolos para el experimento de inducción consistieron en mostrar a los participantes videos para reaccionar y pedirles que reprimieran sus emociones, sin embargo, sin la codificación FACS, la categorización de las etiquetas de las emociones se dejó a la propia auto información del participante. Dejar la categorización a los participantes permite introducir la subjetividad sobre los estímulos emocionales. La calidad de grabación también se redujo debido al parpadeo de la luz y el área facial fue de 190 × 230 píxeles. El SMIC incluyó un grupo demográfico más amplio de participantes, 6 mujeres y 14 hombres. La etnia era más diversa que los conjuntos de datos anteriores con diez asiáticos, nueve caucásicos y un participante africano, sin embargo, esto todavía solo incluye tres etnias y no proporciona una buena descripción general de una población (A. K. Davison, et al., 2018).

La base de datos SMIC se accede mediante una carpeta de Google Drive y tiene una carpeta llamada “SMIC_all_cropped” que contienen las imágenes pre-procesadas y los rostros en una resolución de 64 x 64 pixeles. Esta base de datos se obtiene de (University of oulu., 2018).

La base de datos SMIC cuenta con las expresiones faciales de Negativo, Positivo y Sorpresa.

4.2. Caja de herramientas

4.2.1. Google Colaboratory

Colaboratory, también llamado Colab, es un producto de Google Research. Colab permite que todos puedan escribir y ejecutar código arbitrario de Python en el navegador. Es ideal para aplicarlo en proyectos de aprendizaje automático, análisis de datos y educación. Más técnicamente, Colab es un servicio de notebook alojado de Jupyter que no requiere configuración para usarlo y brinda acceso gratuito a recursos computacionales, incluidas GPU. Los recursos de Colab no están garantizados ni son ilimitados y, en ocasiones, los límites de uso fluctúan. Esto es necesario para que Colab pueda ofrecer los recursos en forma gratuita. Los notebooks de Colab se almacenan en Google Drive, pero también se pueden cargar desde GitHub. Los notebooks de Colab se pueden compartir del mismo modo que las Hojas de cálculo o los Documentos de Google (Google, 2021).

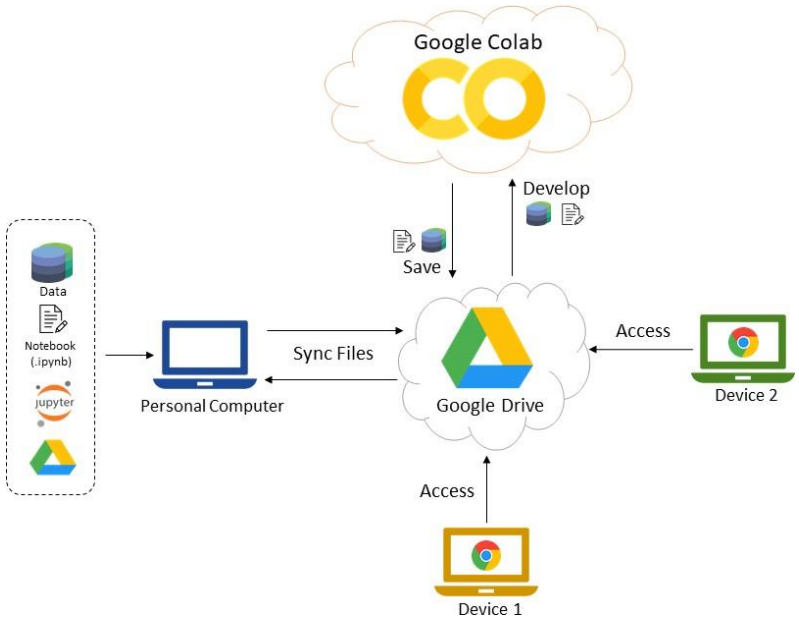


Figura 23 Diagrama de Conexión de Google Colaboratory. Fuente: (Elhawary, M. M., 2020)

4.2.2. Tensorflow

TensorFlow es una plataforma de código abierto de extremo a extremo para el aprendizaje automático. Tiene un ecosistema integral y flexible de herramientas, bibliotecas y recursos de la comunidad que permite a los investigadores impulsar el estado del arte en ML y a los desarrolladores crear e implementar fácilmente aplicaciones impulsadas por ML (TensorFlow, 2021).

4.2.3. Keras

Keras es una API de aprendizaje profundo escrita en Python, que se ejecuta sobre la plataforma de aprendizaje automático TensorFlow. Fue desarrollado con un enfoque en

permitir una experimentación rápida. Ser capaz de pasar de la idea al resultado lo más rápido posible es clave para hacer una buena investigación. Keras reduce la carga cognitiva del desarrollador para que pueda concentrarse en las partes del problema que realmente importan. Keras adopta el principio de divulgación progresiva de la complejidad: los flujos de trabajo simples deben ser rápidos y fáciles, mientras que los flujos de trabajo arbitrariamente avanzados deben ser posibles a través de una ruta clara que se base en lo que ya ha aprendido. Keras proporciona un rendimiento y una escalabilidad de la fuerza de la industria: es utilizado por organizaciones y empresas como la NASA, YouTube o Waymo (Team, K., 2021).

4.2.4. Sklearn

Scikit-Learn es una de estas librerías gratuitas para Python. Cuenta con algoritmos de clasificación, regresión, clustering y reducción de dimensionalidad. Además, presenta la compatibilidad con otras librerías de Python como NumPy, SciPy y matplotlib. La gran variedad de algoritmos y utilidades de Scikit-learn la convierten en la herramienta básica para empezar a programar y estructurar los sistemas de análisis datos y modelado estadístico. Los algoritmos de Scikit-Learn se combinan y depuran con otras estructuras de datos y aplicaciones externas como Pandas o PyBrain. La ventaja de la programación en Python, y Scikit-Learn en concreto, es la variedad de módulos y algoritmos que facilitan el aprendizaje y trabajo del científico de datos en las primeras fases de su desarrollo. La formación de un Máster en Data Science hace hincapié en estas ventajas, pero también prepara a sus alumnos para trabajar en otros

lenguajes. La versatilidad y formación es la clave en el campo tecnológico (Scikit-Learn, 2019).

4.2.5. Flask

En la actualidad existen muchas opciones para crear páginas web y muchos lenguajes (PHP, JAVA), y en este caso Flask nos permite crear de una manera muy sencilla aplicaciones web con Python. Flask es un *"micro"* Framework escrito en Python y concebido para facilitar el desarrollo de Aplicaciones Web bajo el patrón MVC. La palabra *"micro"* no designa a que sea un proyecto pequeño o que nos permita hacer páginas web pequeñas, sino que al instalar Flask tenemos las herramientas necesarias para crear una aplicación web funcional, pero si se necesita en algún momento una nueva funcionalidad hay un conjunto muy grande extensiones (plugins) que se pueden instalar con Flask que le van dotando de funcionalidad. De principio en la instalación no se tienen todas las funcionalidades que se pueden necesitar, pero de una manera muy sencilla se pueden extender el proyecto con nuevas funcionalidades por medio de plugins. El patrón MVC es una manera o una forma de trabajar que permite diferenciar y separar lo que es el modelo de datos (los datos que van a tener la App que normalmente están guardados en BD), la vista (página HTML) y el controlador (donde se gestiona las peticiones de la app web) (Muñoz, J. D., 2020).

4.2.6. Anaconda

Anaconda fue construida por científicos de datos, para científicos de datos. Más de 20 millones de personas utilizan nuestra tecnología para resolver los problemas más difíciles. Anaconda cuenta un repositorio basado en la nube para encontrar e instalar más de 7500 paquetes de ciencia de datos y aprendizaje automático como Tensorflow y Keras (Anaconda, 2021).

En Figura 24 se puede ver la aplicación de Anaconda instalada en un sistema operativo Windows 10 y en la Figura *25* las variables de interno creadas para posteriormente instalar los frameworks o librerías de aprendizaje automático.

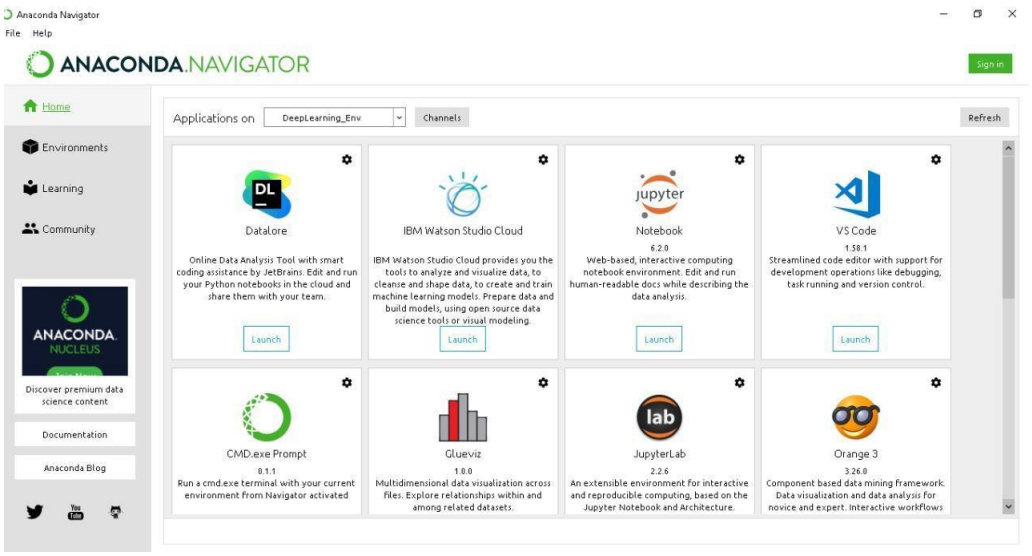


Figura 24 Anaconda.Navigator Instalado en computador con un sistema operativo Windows 10. Fuente: Captura de pantalla de ANACONDA instalado en el equipo donde se desarrolló este trabajo de grado.

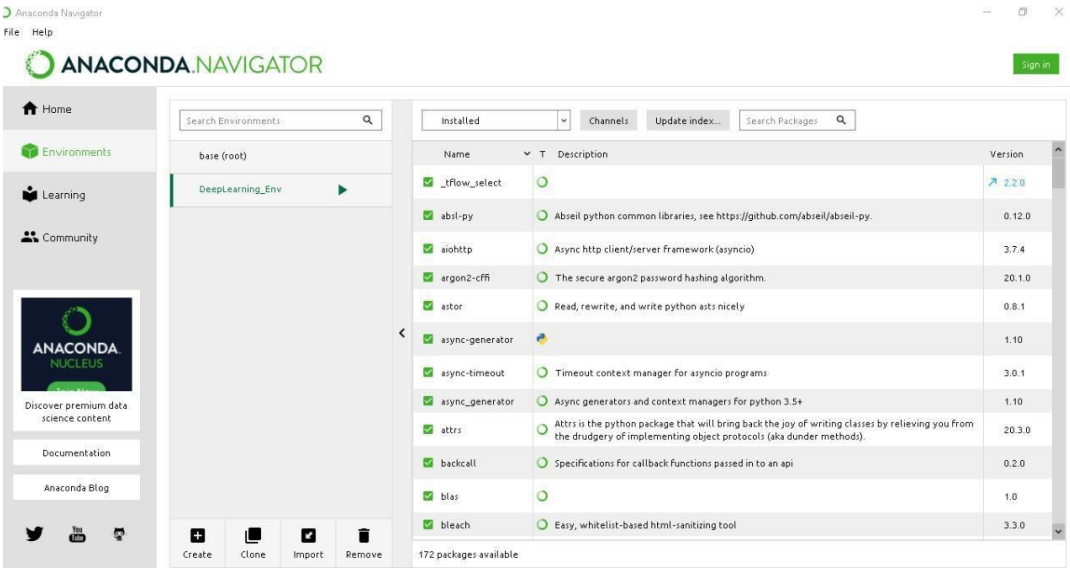


Figura 25 Anaconda.Navigator con las variables de entorno instaladas para correr Tensorflow y Keras. Fuente: Captura de pantalla de ANACONDA instalado en el equipo donde se desarrolló este trabajo de grado.

4.2.7. TensorBoard

El panel de **Graphs** de TensorBoard es una herramienta poderosa para examinar su modelo de TensorFlow. Puede ver rápidamente un gráfico conceptual de la estructura de su modelo y asegurarse de que coincida con su diseño previsto. También puede ver un gráfico de nivel de operación para comprender cómo TensorFlow entiende su programa. Examinar el gráfico de nivel de operaciones puede darle una idea de cómo cambiar su modelo. Por ejemplo, puede rediseñar su modelo si el entrenamiento avanza más lento de lo esperado (Scikit-Learn, 2019).

TensorBoard proporciona la visualización y las herramientas necesarias para experimentar con el aprendizaje automático, las principales características son: seguir y visualizar métricas tales como la pérdida y la exactitud, visualizar el grafo del modelo

(operaciones y capas), ver histogramas de pesos, sesgos y otros tensores a medida que cambian con el tiempo, proyectar incorporaciones en un espacio de dimensiones más bajas, mostrar imágenes, texto y datos de audio, crear perfiles de programas de TensorFlow y mucho más. En la Figura 26 se muestra TensorFlow ejecutado desde Google Colab donde se ejecutó el primero modelo CNN creado para este trabajo de grado y en la Figura 27 se muestra la url de TensorBoard.Dev que Goole Colab generó para seguir trabajando desde ese ambiente el modelo.

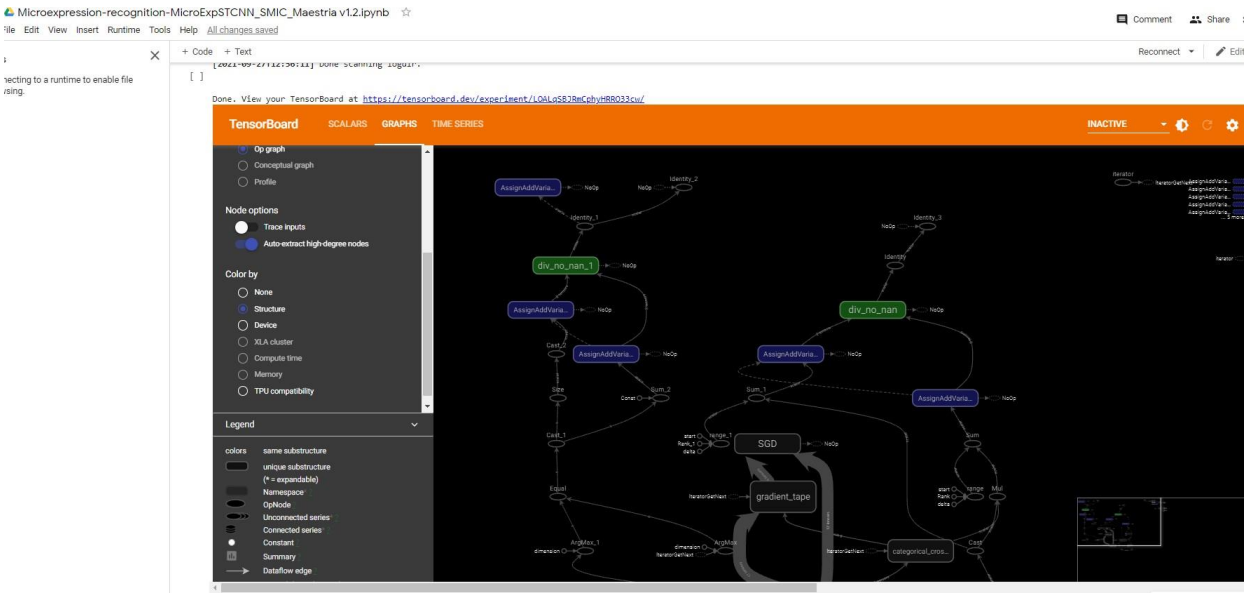


Figura 26 TensorBoard, ejecutado desde Google Colab sobre el primer modelo ejecutado para este trabajo de grado. Fuente: Imagen tomada de TensorBoard con el resultado de uno de los modelos montados.

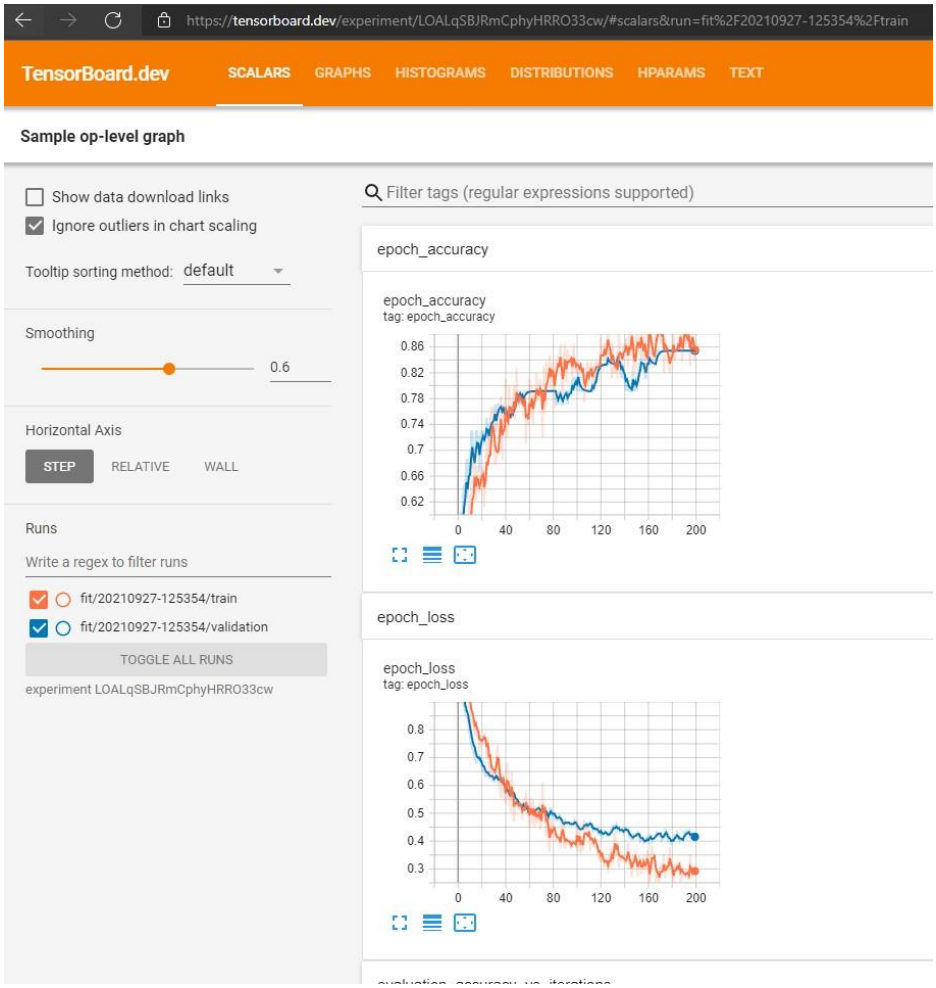


Figura 27 TensorBoard.dev generado para un modelo CNN ejecutado desde Google Colab. Fuente: Imagen tomada de TensorBoard con el resultado de uno de los modelos montados.

4.2.8. OpenCV

OpenCV (Open Source Computer Vision Library) es una biblioteca de software de visión artificial y aprendizaje automático de código abierto. OpenCV se creó para proporcionar una infraestructura común para aplicaciones de visión por computadora y para acelerar el uso de la percepción de la máquina en los productos comerciales. Al

ser un producto con licencia BSD, OpenCV facilita que las empresas utilicen y modifiquen el código.

La biblioteca tiene más de 2500 algoritmos optimizados, que incluyen un conjunto completo de algoritmos de aprendizaje automático y visión por computadora clásicos y de última generación. Estos algoritmos se pueden utilizar para detectar y reconocer rostros, identificar objetos, clasificar acciones humanas en videos, rastrear los movimientos de la cámara, rastrear objetos en movimiento, extraer modelos 3D de objetos, producir nubes de puntos 3D a partir de cámaras estéreo, unir imágenes para producir una alta resolución, imagen de una escena completa, buscar imágenes similares de una base de datos de imágenes, eliminar ojos rojos de imágenes tomadas con flash, seguir los movimientos oculares, reconocer paisajes y establecer marcadores para superponerlos con realidad aumentada, etc. OpenCV tiene más de 47 mil personas de usuarios comunidad y un número estimado de descargas superior a 18 millones. La biblioteca se utiliza ampliamente en empresas, grupos de investigación y organismos gubernamentales. (OpenCV, 2020)

4.3. Reconocimiento de expresiones faciales utilizando modelos profundos

Se implementaron 3 proyectos en python para el reconocer expresiones faciales y microexpresiones faciales.

4.3.1. PyEmotionRecognition

Se clonó el proyecto del repositorio de github PyEmotionRecognition (Kamath, D., 2017) se configuró el ambiente local para ejecutarlo. El link donde se referencia la base de datos de las imágenes nunca cargo (Consortium, 2020), así que se descargó una base de datos de imágenes de kaggle (Sharma, G., 2020).

El proyecto tiene la estructura que se muestra en la Figura 28 y en la Figura 29 se muestran los resultados después de ejecutarlo, se muestra el kernel, el accuracy y el accuracy promedio.

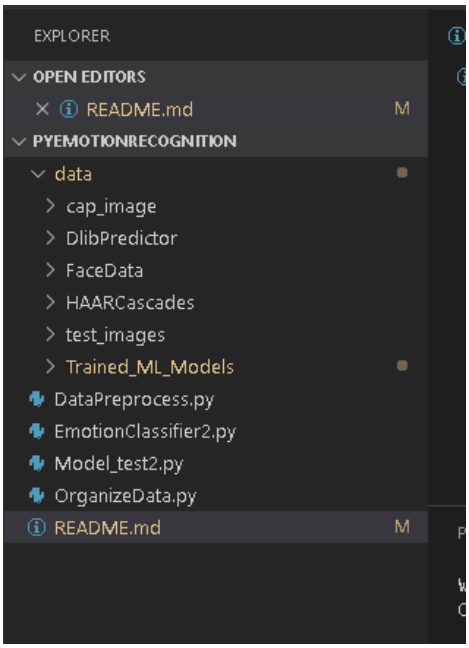


Figura 28 Estructura proyecto PyEmotionRecognition. Fuente: Captura de pantalla del equipo donde se desarrolló este trabajo de grado.

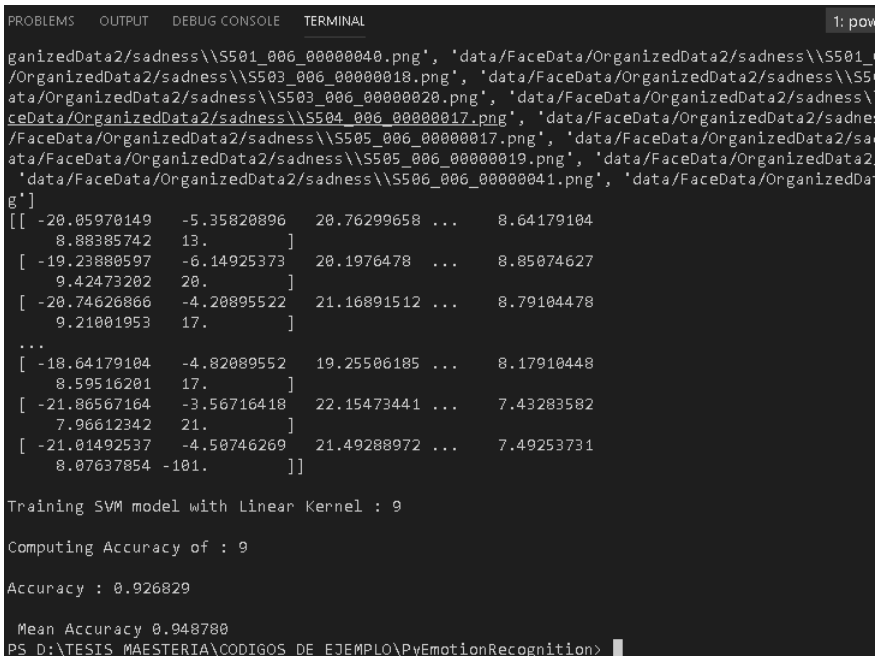


Figura 29 Ejecución de proyecto. Fuente: Captura de pantalla del equipo donde se desarrolló este trabajo de grado.

4.3.2. Landmarks

Con base a la documentación para Landmark Detector de la referencia (Paulvangent, 2021) se montó un proyecto para reconocimiento de puntos en el rostro, el cual se encuentra en el repositorio de Github (Zartha Suarez, N., 2021). Los resultados se pueden ver en la Figura 30.

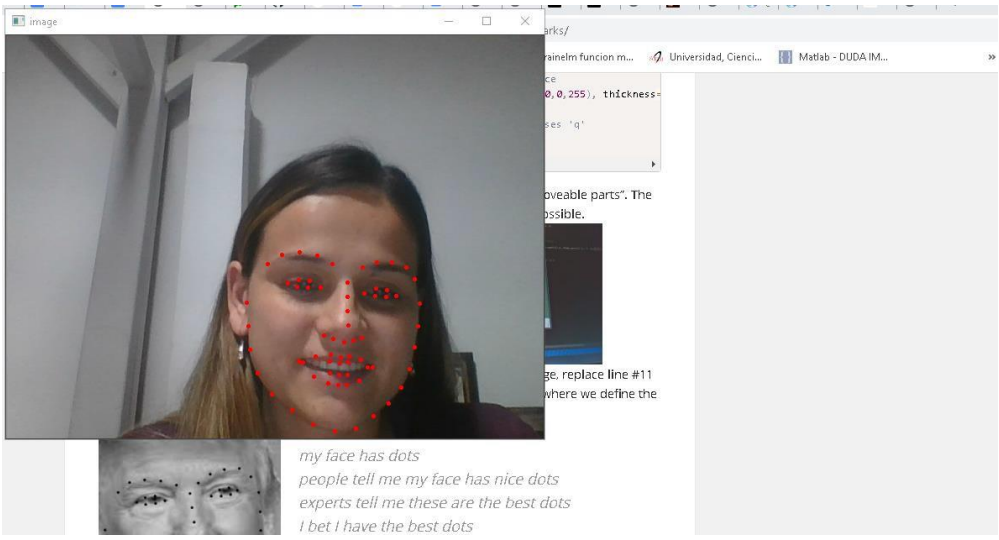


Figura 30 Resultados de montar el proyecto Landmark detector. Fuente: Captura de pantalla del equipo donde se desarrolló este trabajo de grado.

4.3.3. Reconocimiento de expresiones faciales con Keras

Se realizó el tutorial y la certificación para el curso *facial expressionrecognition with keras* de Coursera (Kekre, S., 2020). El cual consta de 9 tareas a seguir para llegar a un producto final que detecta las expresiones faciales de los videos. El curso cuenta de 9 tareas a seguir, en las cuales se crea una red neuronal convolucional CNN, se entrena el modelo, se convierte el modelo a formato json, y se crea una aplicación usando Flask para validar el modelo con el reconocimiento de expresiones faciales de videos para verla en el navegador en una ip especifica.

Se instalo el software Anaconda para correr el proyecto con las librerías de Tensorflow y keras en el sistema operativo Windowns. En la Figura 31 se muestra la variable de entorno “DeepLearning_Dev” que se creó con el software Anaconda para instalar las librerías tensorflow y keras, y asi poder correr el proyecto por medio del

terminal. En la Figura 28 se muestra el terminar corriendo el archivo principal para ejecutar el proyecto y abrir la IP http://127.0.0.1:4996/, en la Figura 33 se muestra el video que se eligió como input para detectar las expresiones faciales.

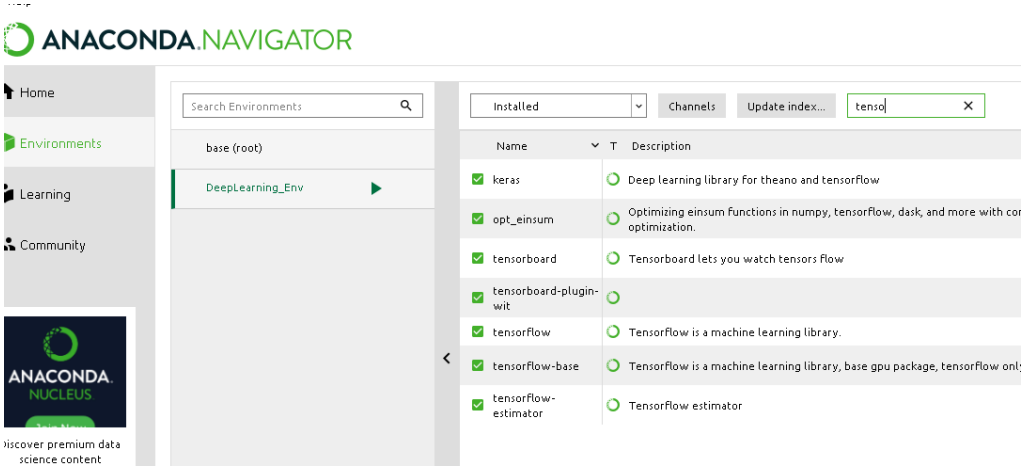


Figura 31 Configuración de la variable de entorno en Anaconda para correr las librerías de tensorflow y keras del proyecto en el sistema operativo windows. Fuente: Captura de pantalla del equipo donde se desarrolló este trabajo de grado.

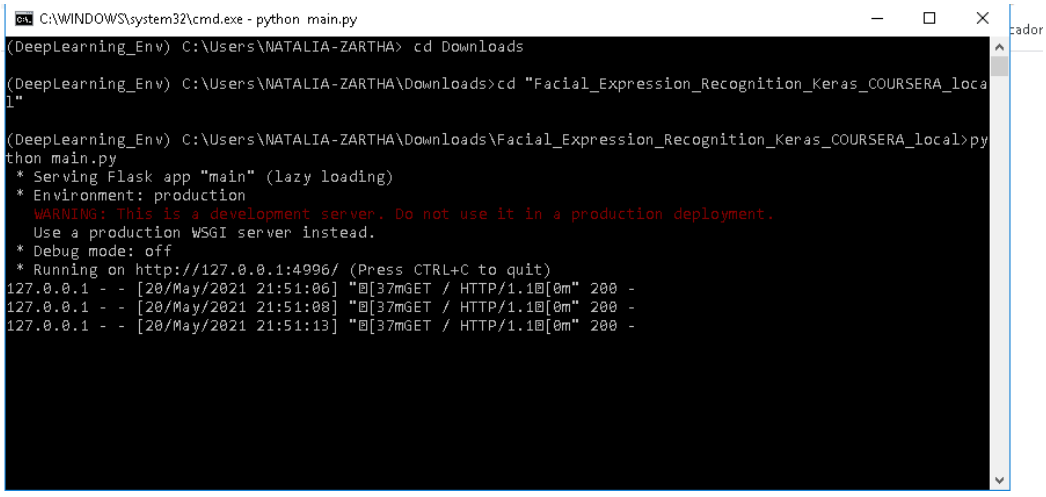


Figura 32 Terminal donde se corre el archivo main.py de pyhon. Fuente: Captura de pantalla del equipo donde se desarrolló este trabajo de grado.

Figura 33 Detección de expresiones faciales con keras. Fuente: Captura de pantalla del equipo donde se desarrolló este trabajo de grado.

Vamos a entrenar un modelo de microexpresiones faciales para saber si la emoción es actuada o verdadera.

4.4. Evaluación de los modelos

4.4.1. F1 score

$$F_1 = \frac{2 * Precisión\ * Sensibilidad\ (recall)}{Precisión + Sensibilidad\ (recall)}$$

Ecuación 8 F1 score

F1 es una medida general de la precisión de un modelo que combina precisión y recall, de esa extraña manera en que la suma y la multiplicación solo mezclan dos ingredientes para hacer un plato separado por completo. Es decir, una buena puntuación F1 significa que tiene pocos falsos positivos y pocos falsos negativos, por lo que está identificando correctamente las amenazas reales y no le molestan las falsas alarmas. Una puntuación F1 se considera perfecta cuando es 1, mientras que el modelo es un fracaso total cuando es 0.

Todos los modelos generarán algunos falsos negativos, algunos falsos positivos y posiblemente ambos. Si bien puede ajustar un modelo para minimizar uno u otro, a menudo se enfrenta a una compensación, en la que una disminución de los falsos negativos conduce a un aumento de los falsos positivos, o viceversa. Deberá optimizar las métricas de rendimiento que sean más útiles para su problema específico (Nicholson, C).

4.4.2. Accuracy

La métrica accuracy representa el porcentaje total de valores correctamente clasificados, tanto positivos como negativos, por lo que corresponde a las predicciones correctas realizadas por el modelo sobre el número total de ejemplos. También se le conoce como

exactitud. Es recomendable utilizar esta métrica en problemas en los que los datos están

balanceados, es decir, que haya la misma cantidad de valores de cada etiqueta (Cagatay C., 2012). Esta métrica se define a partir de la relación dada en la siguiente ecuación:

$$\frac{TP + TN}{TP + \ TN + FP + FN}$$

Ecuación 9 Accuracy

4.4.3. Precisión

La métrica de precisión es utilizada para poder saber qué porcentaje de valores que se han clasificado como positivos son realmente positivos, es decir que evalúa los datos por

el rendimiento de predicciones positivas (Cagatay C., 2012).

$$\frac{TP}{(TP + \ FP)}$$

Ecuación 10 Precisión

La Ecuación 10 nos muestra la forma de calcular la precisión, donde:

$TP = True\ Positive$, son los valores que el algoritmo clasifica como positivos y que realmente son positivos.

$FP = False\ Positive$, son los valores que el algoritmo clasifica como positivo cuando realmente son negativos

4.4.4. Sensibilidad (recall)

La sensibilidad y la especificidad son dos valores que nos indican la capacidad de nuestro estimador para discriminar los casos positivos, de los negativos. La sensibilidad se representa como la fracción de verdaderos positivos, mientras que la especificidad, es la fracción de verdaderos negativos.

La sensibilidad también se conoce como Tasa de Verdaderos Positivos (True Positive Rate) ó TP. Es la proporción de casos positivos que fueron correctamente identificadas por el algoritmo. Se calcula así: (Arce, J., 2020)

$$\frac{TP}{TP + FN}$$

Ecuación 11 Sensibilidad

La Ecuación 11 nos muestra la forma en que se calcula la sensibilidad donde:

$TP = True\ Positive,$ son los valores que el algoritmo clasifica como positivos y que realmente son positivos.

$FN = False\ Negative,$ son los valores que el algoritmo clasifica como negativo cuando realmente son positivos

4.4.5. Especificidad

También conocida como la Tasa de Verdaderos Negativos, (“true negative rate”) o TN. Se trata de los casos negativos que el algoritmo ha clasificado correctamente. Expresa cuan bien puede el modelo detectar esa clase. Se calcula: (Arce, J., 2020)

$$\frac{TN}{TN + FP}$$

Ecuación 12 Especificidad

La especificidad se calcula con la Ecuación 12, donde:

$TN = True\ negatives,$ Son valores que el algoritmo clasifica como negativos y que realmente son negativos.

$FP = False\ positives,$ son los valores que el algoritmo clasifica como positivo cuando realmente son negativos.

4.4.6. Curva ROC

Una curva característica de funcionamiento del receptor, o curva ROC, es un diagrama gráfico que ilustra la capacidad de diagnóstico de un sistema clasificador binario a medida que varía su umbral de discriminación. El método fue desarrollado originalmente para operadores de receptores de radar militares a partir de 1941, lo que llevó a su nombre.

La curva ROC se crea trazando la tasa de verdaderos positivos (TPR) frente a la tasa de falsos positivos (FPR) en varios valores de umbral. La tasa de verdaderos positivos también se conoce como sensibilidad, recuerdo o probabilidad de detección (M. Owayjan, A. Kashour, N. Al Haddad, M. Fadel & G. Al Souki, 2012) en el aprendizaje automático. La tasa de falsos positivos también se conoce como probabilidad de falsa alarma (M. Owayjan, A. Kashour, N. Al Haddad, M. Fadel & G. Al Souki, 2012) y se puede calcular como (1 - especificidad). También se puede considerar como una gráfica de la potencia en función del error de tipo I de la regla de decisión (cuando el rendimiento se calcula a partir de una muestra de la población, se puede considerar como estimadores de estas cantidades). La curva ROC es, por tanto, la sensibilidad o recuperación en función de la caída (Wikipedia, 2021).

True Positive Rate (TPR) es sinónimo de recuperación y, por lo tanto, se define de la siguiente manera:

$$TPR = \frac{TP}{TP + FN}$$

Ecuación 13 True Positive Rate (TPR)

La tasa de falsos positivos (FPR) se define de la siguiente manera:

$$FPR = \frac{FP}{FP + TN}$$

Ecuación 14 False Positive Rate (FPR)

Una curva ROC traza TPR frente a FPR en diferentes umbrales de clasificación. Si se reduce el umbral de clasificación, se clasifican más elementos como positivos, lo que aumenta tanto los falsos positivos como los verdaderos positivos. La Figura 34 muestra una curva ROC típica (Google Developers, 2021).

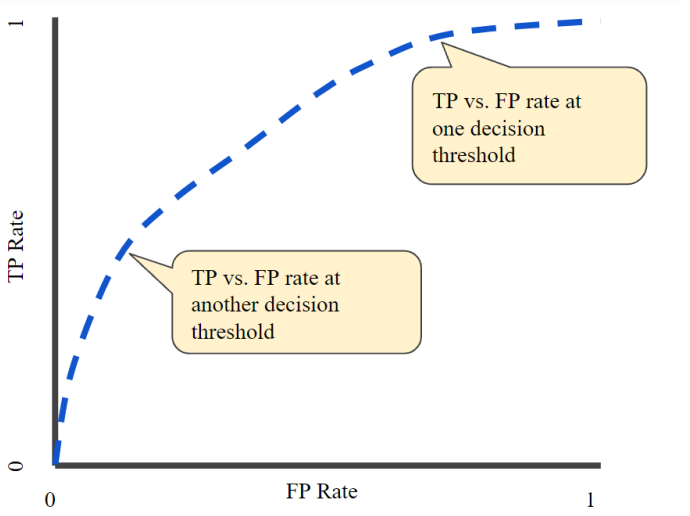


Figura 34 Curva ROC. Fuente: (Google Developers, 2021)

Para calcular los puntos en una curva ROC, podríamos evaluar un modelo de regresión logística muchas veces con diferentes umbrales de clasificación, pero esto sería ineficiente. Afortunadamente, existe un algoritmo eficiente basado en clasificación que puede proporcionarnos esta información, llamado AUC (Google Developers, 2021).

AUC significa "Área bajo la curva ROC". Es decir, AUC mide el área bidimensional completa debajo de la curva ROC completa (piense en el cálculo integral) de (0,0) a (1,1) (Google Developers, 2021). En la Figura 35 se muestra el Area bajo la curva ROC.

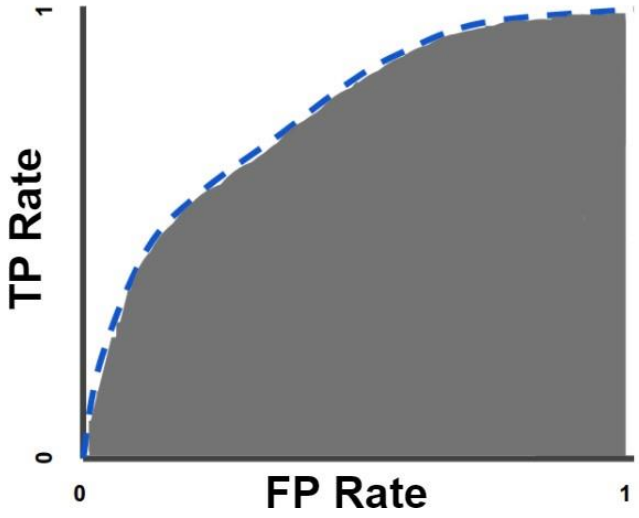


Figura 35 AUC (Área bajo la curva ROC). Fuente: (Google Developers, 2021)

4.4.7. Matriz de Confusión

La matriz de confusión es una matriz que describe el desempeño de un algoritmo, contiene información sobre la clase real y las predicciones del modelo (A. Budhiman, et al., 2019). Esta matriz es fundamental a la hora de evaluar el desempeño del modelo ya que realiza un conteo de los aciertos y errores de cada una de las clases en la clasificación. En la Tabla 1 se da un ejemplo de que valores muestra una matriz de confusión para cada clase que se esté evaluando.

Tabla 1 Ejemplo de una Matriz de Confusión. Fuente: Elaboración propia

	NO (Predicción)	**YES (Predicción)**
NO (Actual)	True Negative (TN) A	False Positive (FP) B
YES (Actual)	False Negative (FN) C	True Positive (TP) D

Capítulo V

5. Resultados y Discusiones

5.1. Modelo convolucional MicroExpSTCNN 3D para el reconocimiento de microexpresiones faciales

Este fue el primer modelo que se montó para el reconocimiento de microexpresiones faciales tomado del artículo “Spontaneous Facial Micro-Expression Recognition using 3D Spatiotemporal Convolutional Neural Networks” (S. P. Teja Reddy, et al., 2019). El cual plantea dos tipos de modelos el **MicroExpSTCNN** y el MicroExpFuseNe sobre las bases de datos SMIC y CAS(ME)2. En el artículo se obtuvo un mejor resultado para el modelo MicroExpSTCNN sobre las dos bases de datos que los demás, es por eso que para este primer modelo que se montó se tomó el codigo fuente (Prasanna Teja Reddy, B. S., 2019) del artículo (S. P. Teja Reddy, et al., 2019) y se trabajó sobre el modelo MicroExpSTCNN, el cual trabaja con 18 frames.

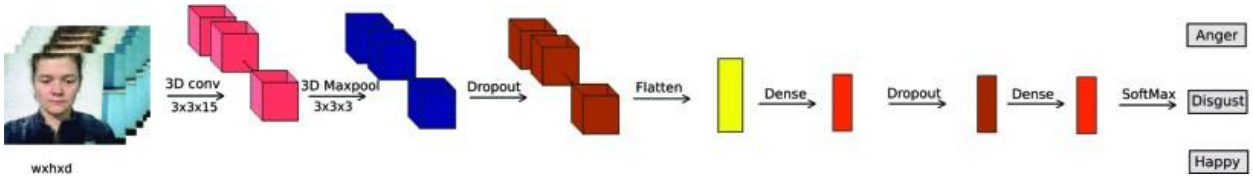


Figura 36 La arquitectura MicroExpSTCNN propuesta para el reconocimiento de microexpresiones utilizando 3D-CNN. Fuente (S. P. Teja Reddy, et al., 2019)

En la Figura 36 se muestra la arquitectura propuesta para el modelo denominado MicroExpSTCNN, el cual se basa en las regiones de la cara completa, utiliza el 3DCNN para explotar la relación espaciotemporal conjunta. El modelo propuesto es básicamente los 3D-CNN, está diseñado con el debido cuidado del número de capas y tamaños de filtro para el problema de reconocimiento de microexpresiones faciales (S. P. Teja Reddy, et al., 2019).

El modelo original usa 96 frames, para este trabajo de grado usamos 18 frames para ambas bases de datos CASME II y SMIC.

En la Figura 37 se muestra la arquitectura para el modelo propuesto MicroExpSTCNN en (S. P. Teja Reddy, et al., 2019), esta misma Arquitectura de Red Neuronal Convolucional 3D se usó para las dos bases de datos de microexpresiones.

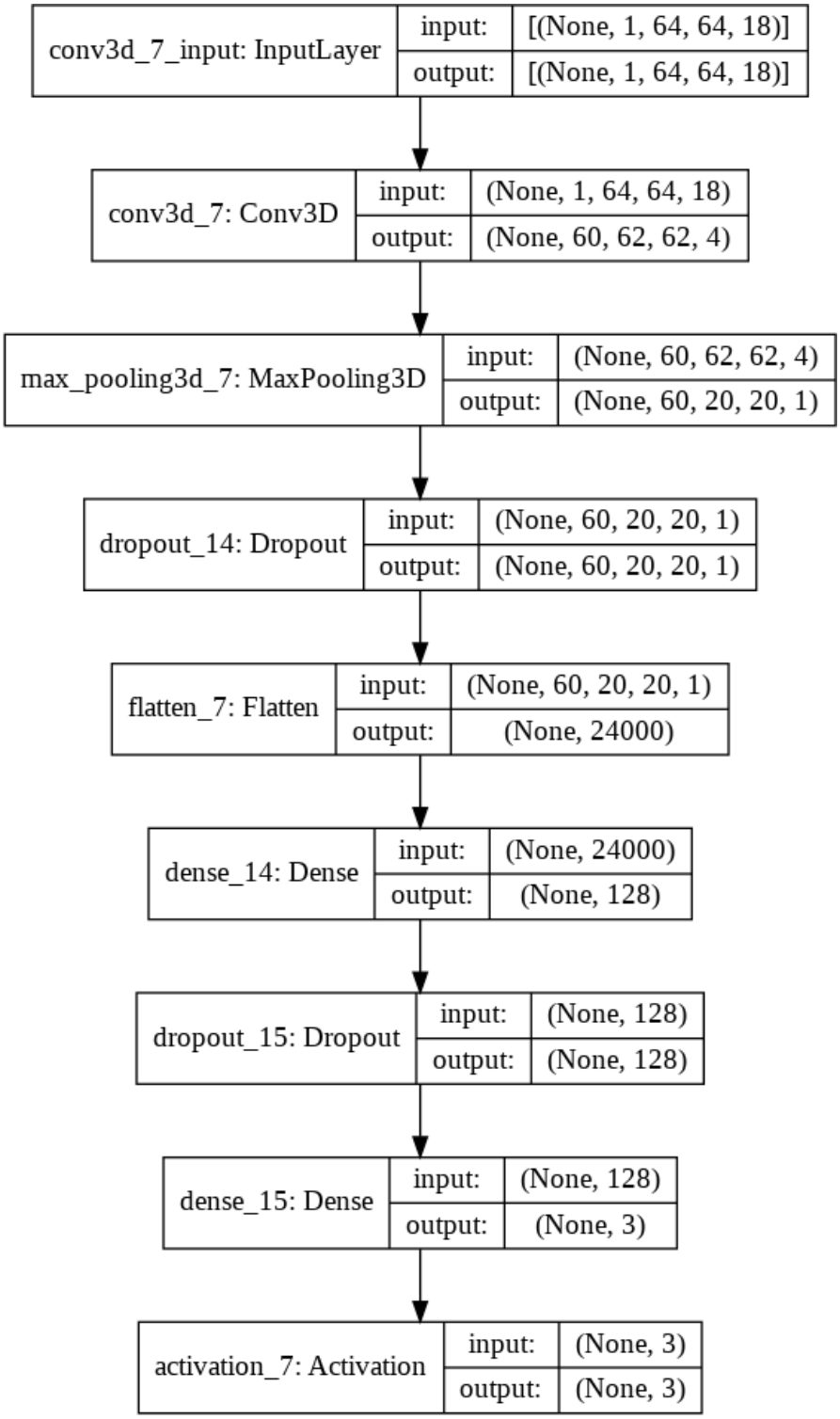


Figura 37 Arquitectura propuesta para el modelo MicroExpSTCNN. Fuente: Modelo generado con Google Colab

5.1.1. Base de datos CASME II

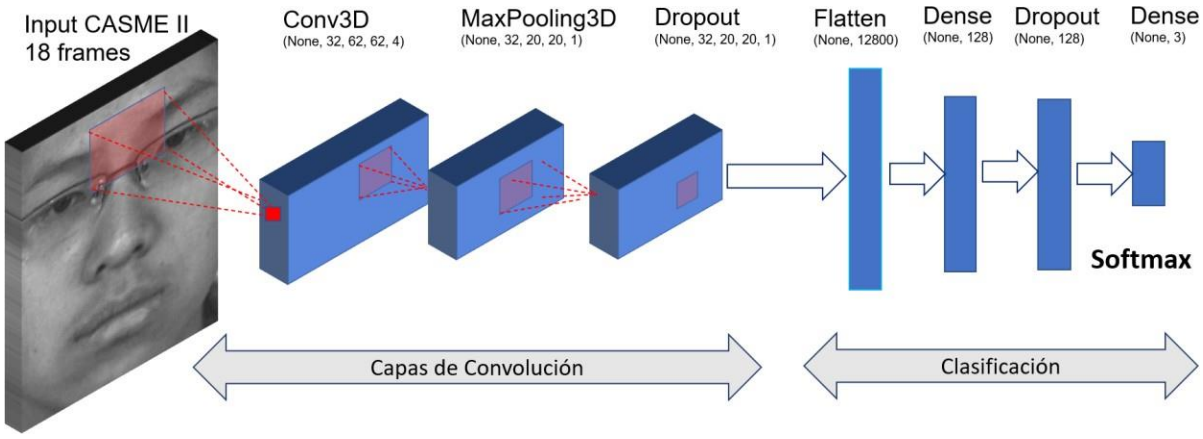


Figura 38 Arquitectura modelo MicroExpSTCNN Red Neuronal Convolucional 3D, para la base de datos CASME II. Fuente: Elaboración propia

En la Figura 38 se muestra la arquitectura que se montó para el modelo MicroExpSTCNN y la base de datos CASME II (Yan WJ, Li X, Wang SJ, Zhao G, Liu YJ, Chen YH & Fu X., 2014). En la Tabla 2 se muestran los parámetros de las capas utilizadas. MicroExpSTCNN es un modelo CNN 3D secuencial. El set de imágenes se dividió entre entrenamiento y validación con un train_test_split y un 35 % del tamaño de prueba, se utilizaron 150 épocas, 34 filtros, un kernel de (3, 3, 15), la función de activación Softmax, una función de perdida categorical_crossentropy y un optimizador SGD. Para la base de datos CASME II se obtuvo un accuracy del 0.9 con un total de parámetros del modelo de 1643267.

Tabla 2 Parametros del modelo de la Red Neuronal Convolucional 3D, modelo MicroExpSTCNN para la base de datos CASME II

Layer (Type)	**Output Shape**	**Param #**
Convolución 3D (Conv3D)	(None, 32, 62, 62, 4)	4352
Max Pooling (MaxPooling3D)	(None, 32, 20, 20, 1)	0
Dropout	(None, 32, 20, 20, 1)	0
Flatten	(None, 12800)	0
Dense	(None, 128)	1638528
Dropout	(None, 128)	0
Dense	(None, 3)	387
Activación	(None, 3)	0

En la Figura 39 se puede observar la gráfica del accuracy en rojo se encuentran los datos de entrenamiento y en naranja los datos de pruebas, se puede observar que en la época 60 el modelo se vuelve más estable, mejora la exactitud con la que predice las etiquetas y aumenta su desempeño. En la Figura 40 se encuentran la gráfica de la pérdida, en azul estan los datos de entrenamiento y en naranja los datos de pruebas, en esta gráfica podemos observar que la pérdida va disminuyendo a medida que se aumentan las épocas.

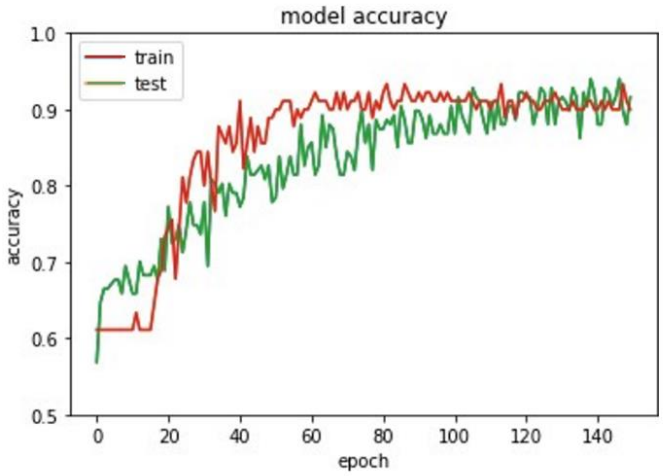


Figura 39 Gráfica del Accuracy modelo de la Red Neuronal Convolucional 3D, modelo MicroExpSTCNN para la base de datos CASME II. En rojo los datos de entrenamiento y en verde los datos de pruebas. Accuracy de 0.9. Fuente: Gráfica generada con Google Colab

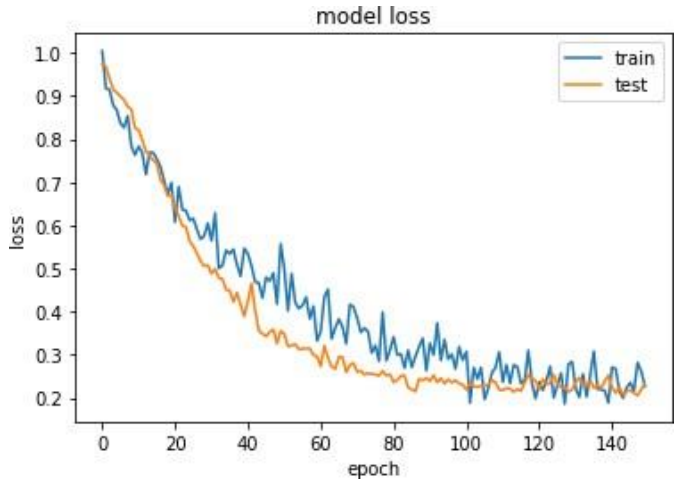


Figura 40 Gráfica de la pérdida modelo de la Red Neuronal Convolucional 3D, modelo MicroExpSTCNN para la base de datos CASME II. En azul los datos de entrenamiento y en naranja los de pruebas. Fuente: Gráfica generada con Google Colab

En la Figura 41 y Figura 42 se muestra el resultado entregado por TensorBoard después del entramiento del modelo para la base de datos CASME II. Estos gráficos

ayudan a depurar el modelo y encontrar cómo hacerlo mejor, nos ayuda a comprender las dependencias entre las operaciones, cómo se calculan los pesos, muestra la función de pérdida, entre otros.

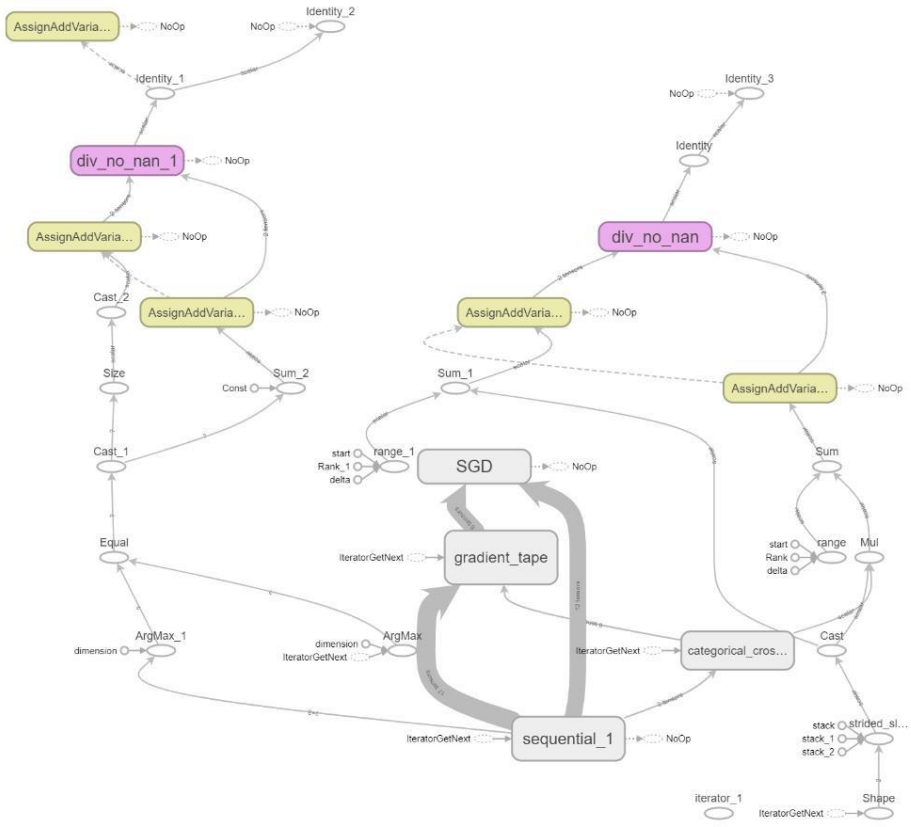


Figura 41 Diagrama (a) entregado por TensorBoard después del entrenamiento del modelo MicroExpSTCNN.
Fuente: Gráfica generada con TensorBoard

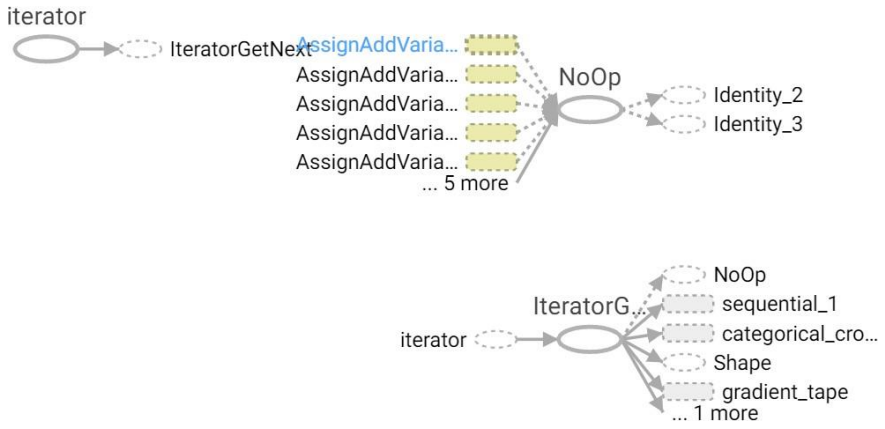


Figura 42 Diagrama (a) entregado por TensorBoard después del entrenamiento del modelo MicroExpSTCNN. Fuente: Gráfica generada con TensorBoard

5.1.1.1. Matriz de Confusión

En la Tabla 3 se encuentra la Matriz de Confusión, con la cual evaluando el desempeño del algoritmo de clasificación del modelo. En la Tabla 3 podemos ver la matriz de confusión del modelo donde cada columna representa las predicciones de cada clase, y en cada fila tenemos el número real de instancias de cada clase. Para este modelo las clases son nuestras etiquetas: "Angry", "Happy" y "Disgust" y la matriz de confusión nos muestra que para estas clases el modelo tiene más cantidad de aciertos que de errores; para Angry es 51, para Happy ed 19 y para Disgust es 11.

Tabla 3 Matriz de confusión base de datos CASME II para el modelo propuesto MicroExpSTCNN. Fuente: Elaboración propia con la matriz de confusión entregada por Google Colab.

True label	**Angry**	51	0	4
	Happy	3	19	0

	Disgust	0	2	11
		Angry	**Happy**	**Disgust**
		Predicted label		

Después del proceso de entrenamiento del modelo usando los datos de entrenamiento y validación, donde se ajustaron los parámetros e hiperparámetros del mismo para obtener el mejor rendimiento con respecto a la exactitud (Accuracy ver Figura 39 y el menor error de la función de pérdida (Loss ver Figura 40); el modelo se evaluó a través de las métricas f1 score, sensibilidad, precisión, curva ROC y AUC que permitieron evidenciar el rendimiento real del modelo. Un modelo presenta un buen desempeño cuando estas métricas dan cercanas a 1 y por el contrario es un mal indicador si estas métricas dan cercanas a cero. Para el modelo MicroExpSTCNN que se montó con base a la referencia (S. P. Teja Reddy, et al., 2019) en el cual se ajustaron los parámetros modelo del obteniendo mejores resultados que la referencia. El f1 score macro dio 0.868, el f1 score micro 0.9, la sensibilidad dio 0.879 y la precisión 0.860; estas métricas cercanas a 1 indican que el modelo es bueno para predecir las microexpresiones faciales para las etiquetas "Angry", "Happy" y "Disgust". Los resueltos de las métricas de rendimiento de modelo se pueden ver en la Tabla 4.

Tabla 4 Resultado de las métricas de rendimiento del modelo MicroExpSTCNN para la base de datos CASME II. Fuente: Elaboración propia

Modelo	Base de Datos	Precisión	Sensibilidad	Accuracy	F1 Score Macro	F1 Score Micro	AUC
CNN 3D MicroExpSTCNN	CASME II	0.860	0.879	0.9	0.868	0.9	0.984

5.1.1.2. Curva ROC y AUROC

En la Figura 43 se encuentra el resultado de la curva ROC con un AUC (Area bajo la curva) de 0.984, lo cual nos indica que el modelo está clasificando bien las etiquetas y está teniendo un buen resultado de aprendizaje automático.

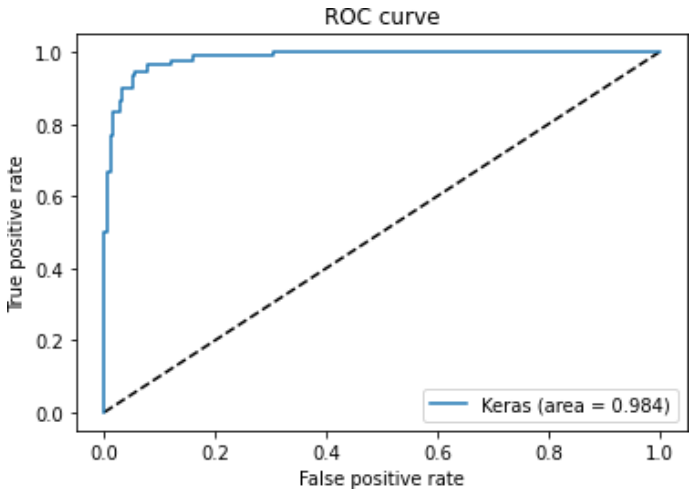


Figura 43 Curva ROC modelo MicroExpSTCNN para la base de datos CASME II. Fuente: Gráfica generada con Google Colab

El resultado de este modelo se encuentra en Google Colab y en Github.

Ver Anexo a - Modelo 1 - Base de datos CASME II.

5.1.2. Base de datos SMIC

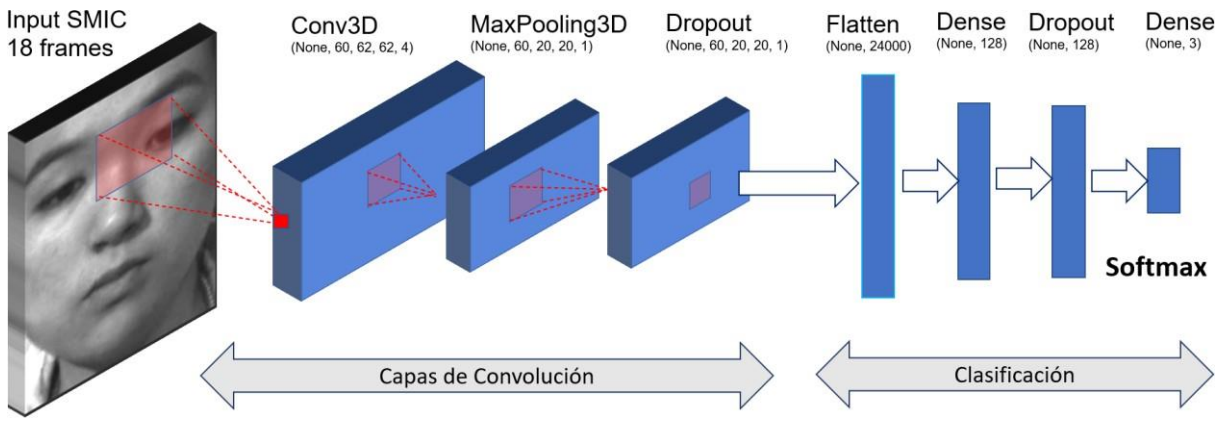


Figura 44 Arquitectura modelo MicroExpSTCNN Red Neuronal Convolucional 3D, para la base de datos SMIC.
Fuente: Elaboración propia

En la Figura 44 se encuentra la arquitectura del modelo MicroExpSTCNN que se montó para la base de datos SMIC y en la Tabla 5 se muestran los parámetros para la Red Neuronal Convolucional 3D, se creó el modelo 3D secuencial, el set de imágenes se dividió entre imágenes de pruebas y de entrenamiento con la función train_test_split y con un 30 % del tamaño de prueba, se usaron 200 épocas, 60 filtros para la convolución 3D, un kernel de (3,3,15) y se usó la función de activación Softmax. Se obtuvo un accuracy del 0.917 para un total de 3080675 del modelo.

Tabla 5 Parámetros del modelo de la Red Neuronal Convolucional 3D, modelo MicroExpSTCNN para la base de datos SMIC. Fuente: Elaboración propia

Layer (Type)	**Output Shape**	**Param #**
Convolución 3D (Conv3D)	(None, 60, 62, 62, 4)	8160
Max Pooling (MaxPooling3D)	(None, 60, 20, 20, 1)	0
Dropout	(None, 60, 20, 20, 1)	0
Flatten	(None, 24000)	0
Dense	(None, 128)	3072128
Dropout	(None, 128)	0
Dense	(None, 3)	387
Activación	(None, 3)	0

En la Figura 45 se puede observar la gráfica del accuracy en rojo se encuentran los datos de entrenamiento y en naranja los datos de pruebas, se puede observar que en la época 50 el modelo se vuelve más estable, mejora la exactitud con la que predice las etiquetas y aumenta su desempeño. En la Figura 46 se encuentran la gráfica de la pérdida, en azul estan los datos de entrenamiento y en naranja los datosde pruebas, en esta gráfica podemos observar que la pérdida va disminuyendo a medida que se aumentan las épocas, esto indica que el modelo está aprendiendo.

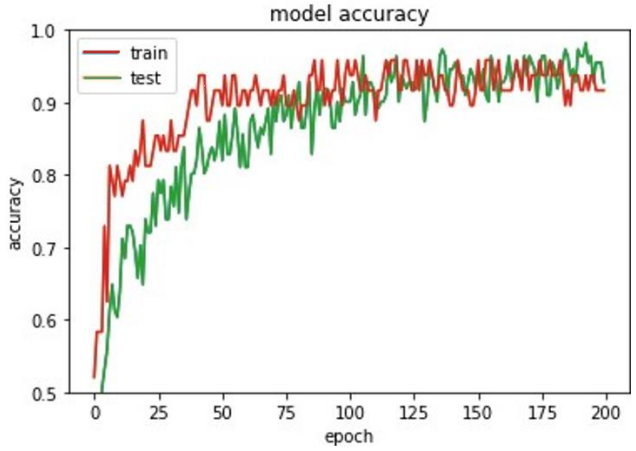


Figura 45 Gráfica del Accuracy modelo de la Red Neuronal Convolucional 3D, modelo MicroExpSTCNN para la base de datos SMIC. En rojo los datos de entrenamiento y en verde las pruebas. Accuracy de 0.917. Fuente: Gráfica generada con Google Colab

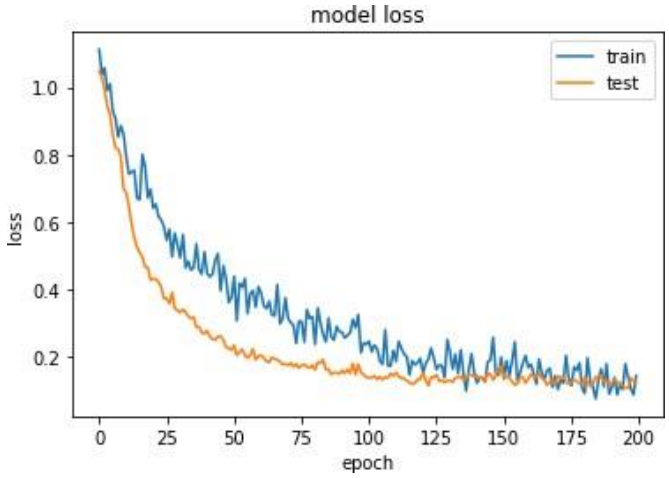


Figura 46 Gráfica de la pérdida modelo de la Red Neuronal Convolucional 3D, modelo MicroExpSTCNN para la base de datos SMIC. En azul los datos de entrenamiento y en naranja los de pruebas. Fuente: Gráfica generada con Google Colab

5.1.2.1. Matriz de Confusión

En la Tabla 6 se encuentra la Matriz de Confusión del modelo MicroExpSTCNN para la base de datos SMIC, con las etiquetas Negativo, Positivo y Sorpresa. En la matriz de confusión cada columna representa las predicciones de cada clase, y en cada fila tenemos el número real de instancias de cada clase; las clases son las etiquetas. La matriz de confusión nos muestra que para estas clases el modelo tiene más cantidad de aciertos que de errores; para la clase Negativo el conteo dio 23, para la clase Positivo el conteo dio 9 y para para la clase sorpresa dio 12. Esto nos permite visualizar el desempeño del algoritmo de aprendizaje del modelo.

Tabla 6 Matriz de confusión modelo MicroExpSTCNN para la base de datos SMIC. Fuente: Elaboración propia a partir de la matriz de confusión entregada por Google Colab.

True label	**Negativo**	23	3	0
	Positivo	0	9	1
	Sorpresa	0	0	12
		Negativo	**Positivo**	**Sorpresa**
		Predicted label		

Después del proceso de entrenamiento del modelo usando los datos de entrenamiento y validación, donde se ajustaron los parámetros e hiperparámetros del mismo para obtener el mejor rendimiento con respecto a la exactitud (Accuracy ver Figura *45* y el menor error de la función de pérdida (Loss ver Figura *46*); el

modelo se evaluó a través de las métricas f1 score, sensibilidad, precisión, curva ROC y AUC que permitieron evidenciar el rendimiento real del modelo. Un modelo presenta un buen desempeño cuando estas métricas dan cercanas a 1 y por el contrario es un mal indicador si estas métricas dan cercanas a cero. Para el modelo MicroExpSTCNN y la base de datos SMIC que se montó con base a la referencia (S. P. Teja Reddy, et al., 2019) en el cual se ajustaron los parámetros del obteniendo mejores resultados que la referencia.

El f1 score macro dio 0.906, el f1 score micro 0.916, la sensibilidad dio 0.928 y la precisión 0.891; estas métricas cercanas a 1 indican que el modelo es bueno para predecir las microexpresiones faciales para las etiquetas Negativo, Positivo y Sorpresa. En la Tabla 7 se puede ver el resultado de los valores de accuracy, f1, precisión, AUC y sensibilidad.

Tabla 7 Resultado de las métricas de rendimiento del modelo MicroExpSTCNN para la base de datos SMIC. Fuente: Elaboración propia

Modelo	Base de Datos	Precisión	Sensibilidad	Accuracy	F1 Score Macro	F1 Score Micro	AUC
CNN 3D MicroExpSTCNN	SMIC	0.891	0.928	0.917	0.906	0.917	0.993

5.1.2.2. Curva ROC y AUROC

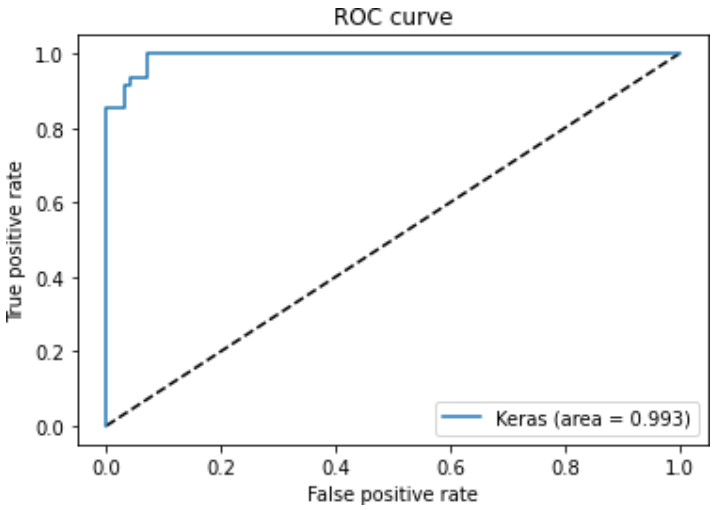


Figura 47 Curva ROC para el modelo MicroExpSTCNN para la base de datos SMIC. Fuente: Gráfica generada con Google Colab

En la Figura 47 se encuentra el resultado de la gráfica ROC con un área de 0.984. Un area cercana a uno nos indica que el modelo está clasificando correctamente la mayoría de las etiquetas y está teniendo un buen resultado de aprendizaje automático. El resultado de este modelo se puede ver en el Anexo a - Modelo 1 - Bases de datos SMIC.

5.2. Modelo convolucional 3D para el reconocimiento de microexpresiones faciales con data augmentation

Para este modelo se cargaron las imágenes en formato numpy array para las dos bases de datos de microexpresiones faciales SMIC (X. Li, et al., 2013) y CASME II (Yan WJ, Li X, Wang SJ, Zhao G, Liu YJ, Chen YH & Fu X., 2014), se agregó una

función de rotación para aumentar el número de imágenes, basados en el modelo creado por Hasib Zunair llamado “3D image classification from CT scans” (Zunair, H., 2020) y adaptado a nuestra necesidad. La arquitectura de la Red Neuronal Convolucional 3D con Data Augmentation para este modelo se muestra en la Figura 48. Con TensorBoard se generó un diagrama del entramiento del modelo, ver Figura 49 y Figura 50.

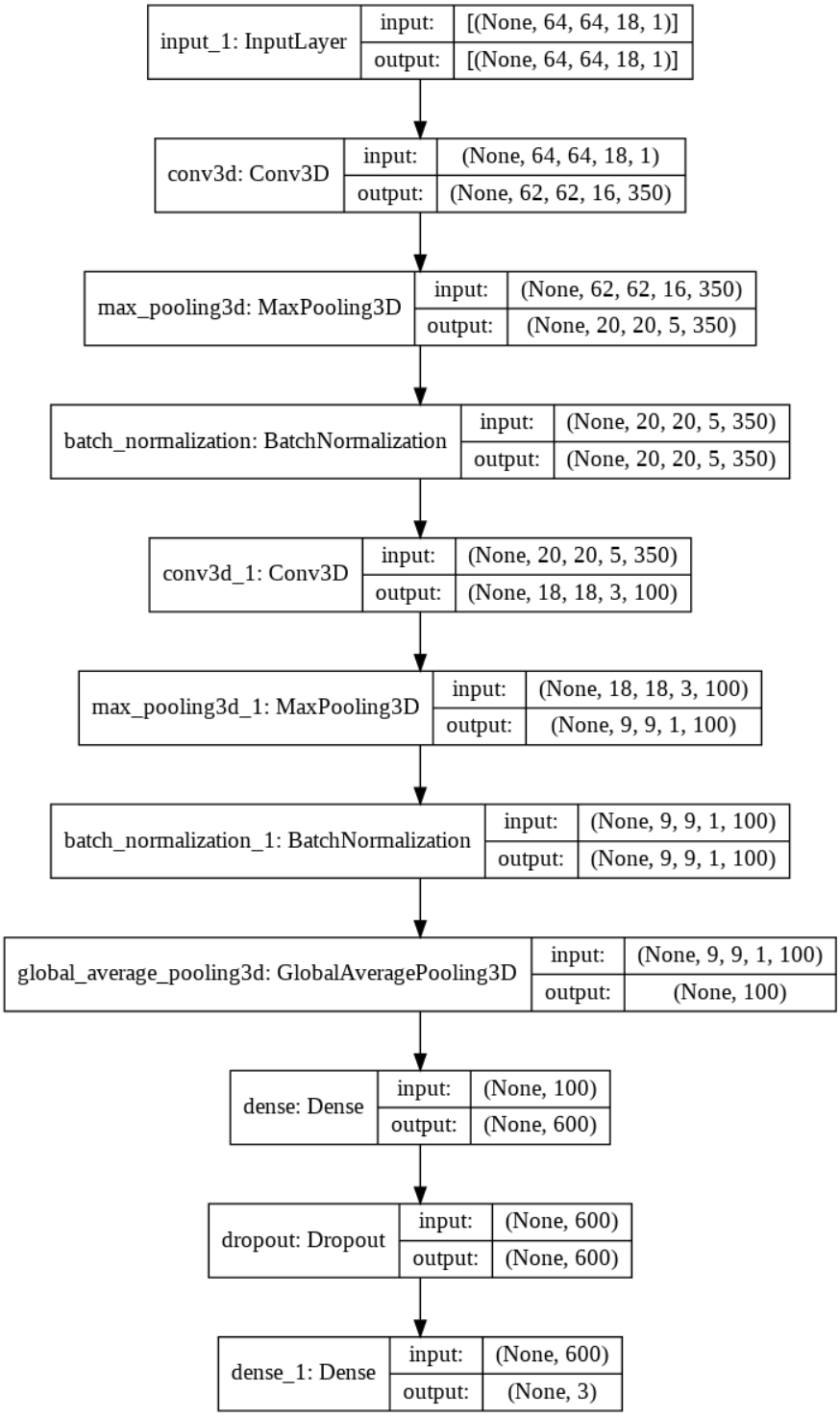


Figura 48 Arquitectura Red Neuronal Convolucional 3D con Data Augmentation.
Fuente: Modelo generada con Google Colab

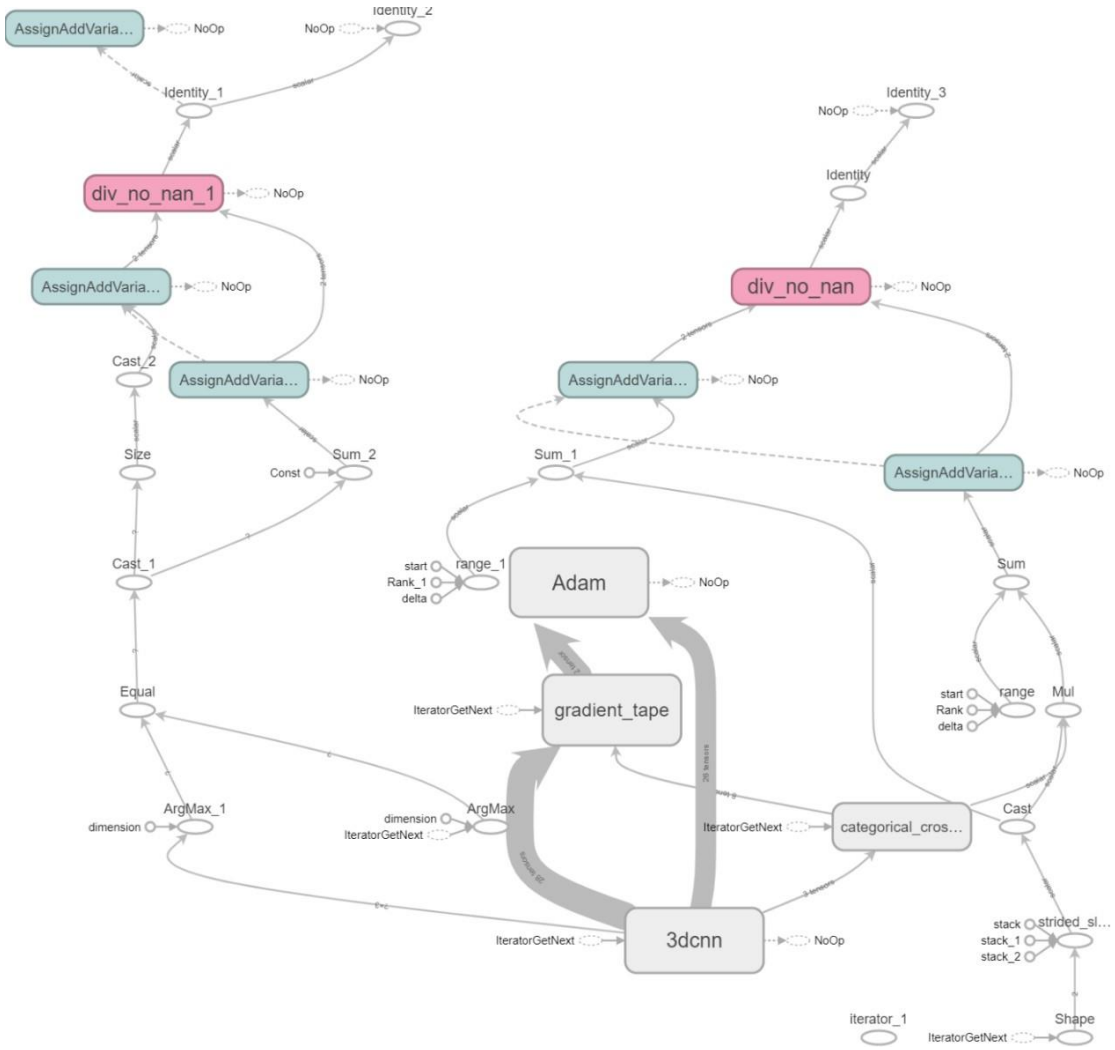


Figura 49 Diagrama (a) de entrenamiento del modelo generado con TensorBoard para el modelo CNN 3D con Data Augmentation para las dos bases de datos (SMIC y CASME II). Fuente: Imagen generada con TensorBoard

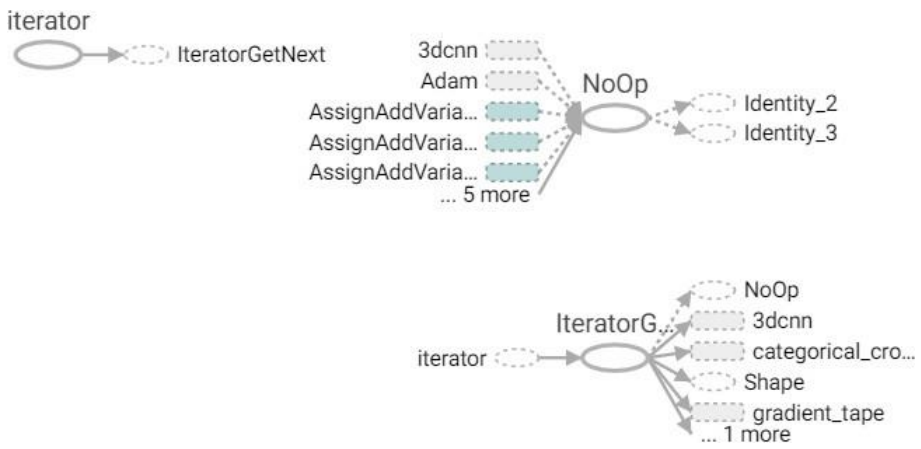


Figura 50 Diagrama (b) de entrenamiento del modelo generado con TensorBoard para el modelo CNN 3D con Data Augmentation para las dos bases de datos (SMIC y CASME II). Fuente: Imagen generada con TensorBoard

5.2.1. Base de datos CASME II

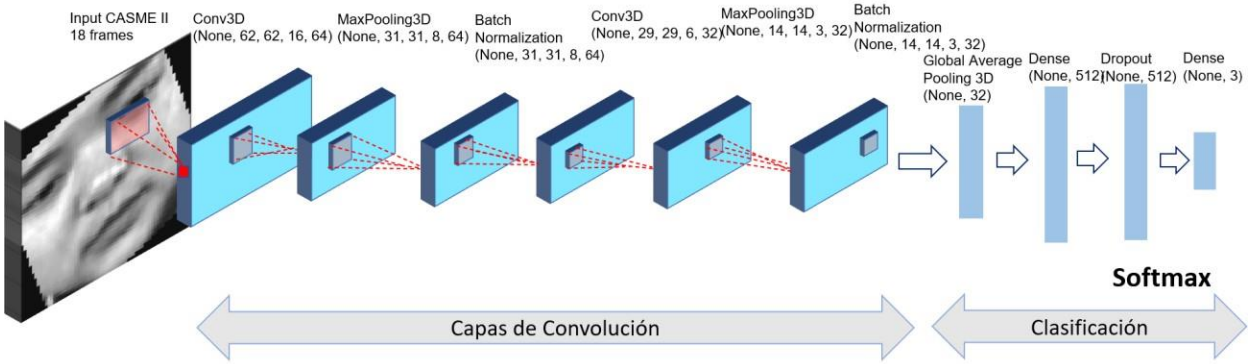


Figura 51 Arquitectura Red Neuronal Convolucional 3D con Data Augmentation para la base de datos CASME II. Fuente: Elaboración propia

En la Figura 51 se muestra la arquitectura que se creó para un Red Neuronal Convolucional 3D con Data Augmentation. Para este modelo se rotaron las imágenes

escogiendo un ángulo aleatorio dentro de este rango [-20, -10, -5, 5, 10, 20]. En la Figura 52 se muestra la imagen de un rostro rotado aleatoriamente y en la Figura 53 se muestra los 18 frames rotados. Para este modelo tambien se trabajó con 18 frames para ambas bases de datos CASME II (Yan WJ, Li X, Wang SJ, Zhao G, Liu YJ, Chen YH & Fu X., 2014) y SMIC (X. Li, et al., 2013).

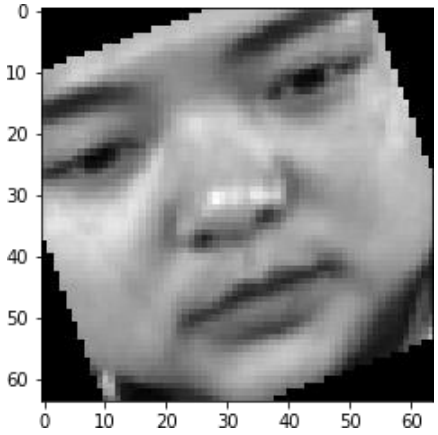


Figura 52 Rostro rotado aleatoriamente para la base de datos CASME II. Fuente: Imagen generada con Google Colab

Figura 53. 18 frames rotados aleatoriamente para la base de datos CASME II. Fuente: Imagen generada con Google Colab

En la Figura 54 se muestra el codigo de la función para generar el data augmentation, se define un train_dataset y un validation_dataset, el cual está tomando las imágenes

rotadas y aumenta la cantidad de imágenes para después enviarlas al modelo de la Red Neuronal Convolucional.

```
# Define data loaders.
train_loader = tf.data.Dataset.from_tensor_slices((x_train, y_train))
validation_loader = tf.data.Dataset.from_tensor_slices((x_val, y_val))

batch_size = 2
# Augment the on the fly during training.
train_dataset = (
    train_loader.shuffle(len(x_train))
    .map(train_preprocessing)
    .batch(batch_size)
    .prefetch(2)
)
# Only rescale.
validation_dataset = (
    validation_loader.shuffle(len(x_val))
    .map(validation_preprocessing)
    .batch(batch_size)
    .prefetch(2)
)
```

Figura 54 Codigo para realizar el data augmentation. Fuente: Elaboración propia tomado de un fragmento de codigo de Google Colab.

En la Tabla 8 se muestra los parámetros y las capas de la Red Neuronal Convolucional 3D con Data augmentation para la base de datos CASME II (Yan WJ, Li X, Wang SJ, Zhao G, Liu YJ, Chen YH & Fu X., 2014). Para este modelo se usuario 250 epocas con un EarlyStopping, 64 filtros para la mira capa Convolucional 3D y 32 para la segunda, con un tamaño del kernel de 3 y una función de activación Softmax. En la Figura 55 se encuentra la tabla del accurracy en el cual se puede observar que en la época 200 el modelo se vuelve más estable entregando un acurracy del 0.942. En la Figura 56 se encuentra la gráfica de la perdida y lo cual indica que a medida que van pasando las épocas le modelo aprende mejor. Para este modelo se obtuvo un total de parámetros de 75939.

En la Figura 57 el rostro que se le entrego al modelo para predecir la microexpresión, el modelo predijo “Angry” con un 98.02 %.

Tabla 8 Parámetros del modelo de la Red Neuronal Convolucional con Data augmentation para la base de datos CASME II. Fuente: Elaboración propia

Layer (Type)	**Output Shape**	**Param #**
InputLayer	[(None, 64, 64, 18, 1)]	0
Convolución 3D (Conv3D)	(None, 62, 62, 16, 64)	1792
Max Pooling (MaxPooling3D)	(None, 31, 31, 8, 64)	0
Batch Normalization	(None, 31, 31, 8, 64)	256
Convolución 3D (Conv3D)	(None, 29, 29, 6, 32)	55328
Max Pooling (MaxPooling3D)	(None, 14, 14, 3, 32)	0
Batch Normalization	(None, 14, 14, 3, 32)	128
Global Average Pooling 3D	(None, 32)	0
Dense	(None, 512)	16896
Dropout	(None, 512)	0
Dense	(None, 3)	1539

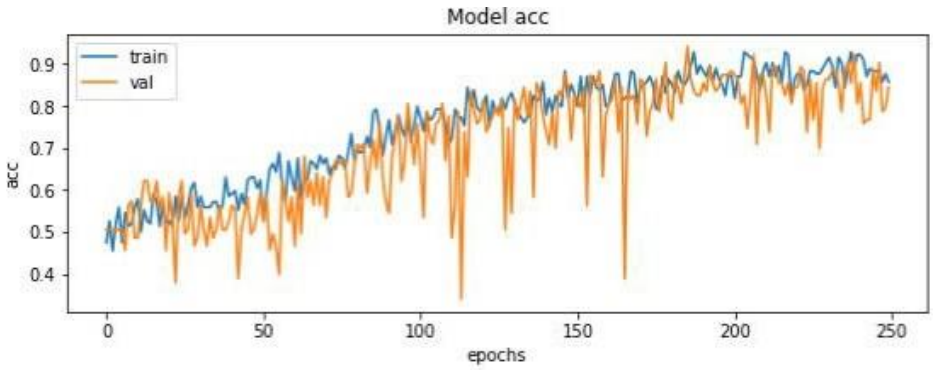


Figura 55 Gráfica del accuracy modelo de la Red Neuronal Convolucional con Data augmentation para la base de datos CASME II. En azul los datos de entrenamiento y con naranja los de validación. Accuracy de 0.942. Fuente: Gráfica generada con Google Colab

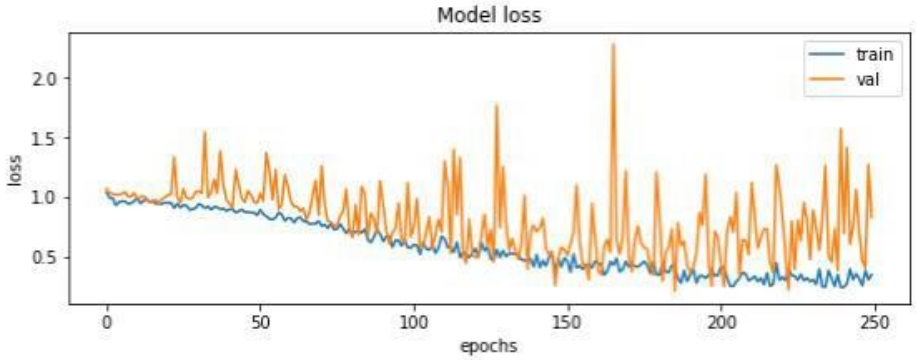


Figura 56 Gráfica de las perdidas modelo de la Red Neuronal Convolucional con Data augmentation para la base de datos CASME II. Con azul los datos de entrenamiento y con naranja los de validación. Fuente: Gráfica generada con Google Colab

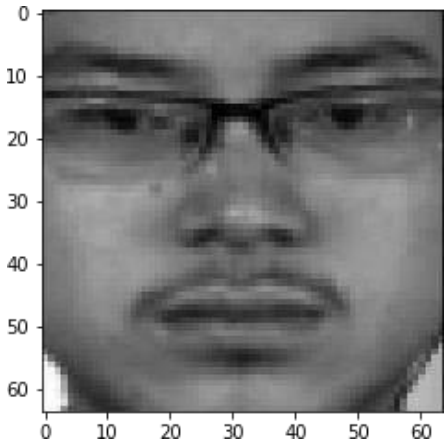


Figura 57 Rostro que se le entrego al modelo para predecir. Fuente: Imagen generada con Google Colab

Tabla 9 Resultado de la predicción de modelo con Data Augmentation para la base de datos CASME II. Fuente: Elaboración propia

	Angry [%]	Happy [%]	Disgust [%]
Prediction percentage	98.02	0.74 %	1.24 %

5.2.1.1. Matriz de Confusión

En la Figura 58 se puede ver la matriz de confusión con un buen conteo de verdaderos positivos para las etiquetas “Angry”, “Happy” y “Disgust”; son pocos los datos que quedan como falsos positivos y falsos negativos. Se puede evidenciar que la matriz de confusión en comparación con el modelo anterior, entrega mayor número de conteos verdaderos evidenciando un mejor desempeño en el modelo. Para la clase Angry el conteo dio 48, para la clase Happy dio 33 y para para la clase Disgust dio 16. Esto nos permite visualizar el desempeño del algoritmo de aprendizaje del modelo.

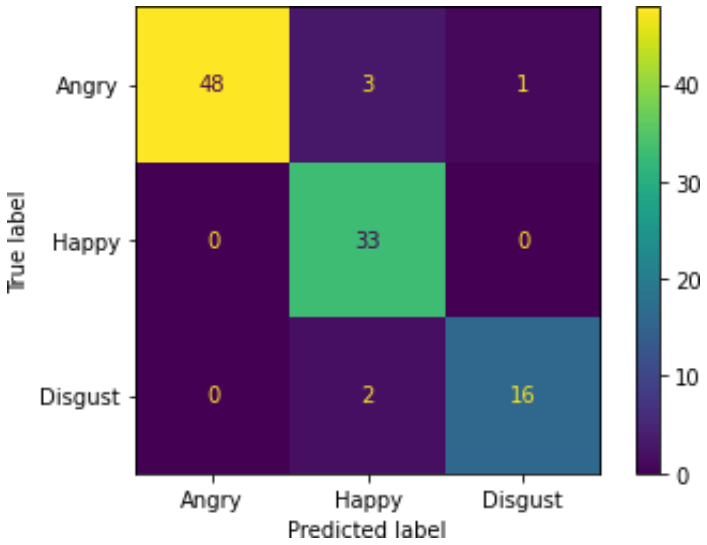


Figura 58 Matriz de Confusión modelo de la Red Neuronal Convolucional con Data augmentation para la base de datos CASME II. Fuente: Imagen generada con Google Colab

Después del proceso de entrenamiento del modelo usando los datos de entrenamiento y validación, donde se ajustaron los parámetros e hiperparámetros del mismo para obtener el mejor rendimiento con respecto a la exactitud (Accuracy ver Figura 55 y el menor error de la función de pérdida (Loss ver Figura 56); el modelo se evaluó a través de las métricas f1 score, sensibilidad, precisión, curva ROC y AUC que permitieron evidenciar el rendimiento real del modelo. Para la base de datos CASME II el f1 score macro dio 0.935, el f1 score micro dio 0.942, la sensibilidad dio 0.937 y la precisión 0.936. Las métricas dan un número muy cercanas a uno, esto indica el buen desempeño de modelo.

Tabla 10 Resultado de las métricas de rendimiento del modelo CNN 3D con Data Augmentation para la base de datos CASME II. Fuente: Elaboración propia

Modelo	Base de Datos	Precisión	Sensibilidad	Accuracy	F1 Score Macro	F1 Score Micro	AUC
CNN 3D con Data Augmentation	CASME II	0.936	0.937	0.942	0.935	0.942	0.989

5.2.1.2. Curva ROC y AUROC

En la Figura 59 se puede ver la curva ROC con un AUC de 0.989. Esto quiere decir que el modelo está prediciendo bien la microexpresión facial para las etiquetas “Angry”, “Happy” y “Disgust”.

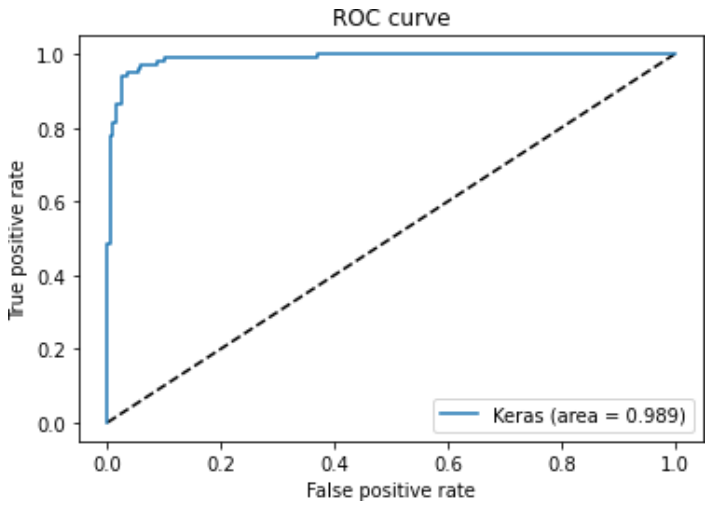


Figura 59 Curva ROC del modelo CNN 3D con Data Augmentation para la base de datos CASME II. Fuente: Gráfica generada con Google Colab

Los modelos se encuentran en en Google Colab y Github.

Ver Anexo a - Modelo 2 - Base de datos CASME II.

5.2.2. Base de datos SMIC

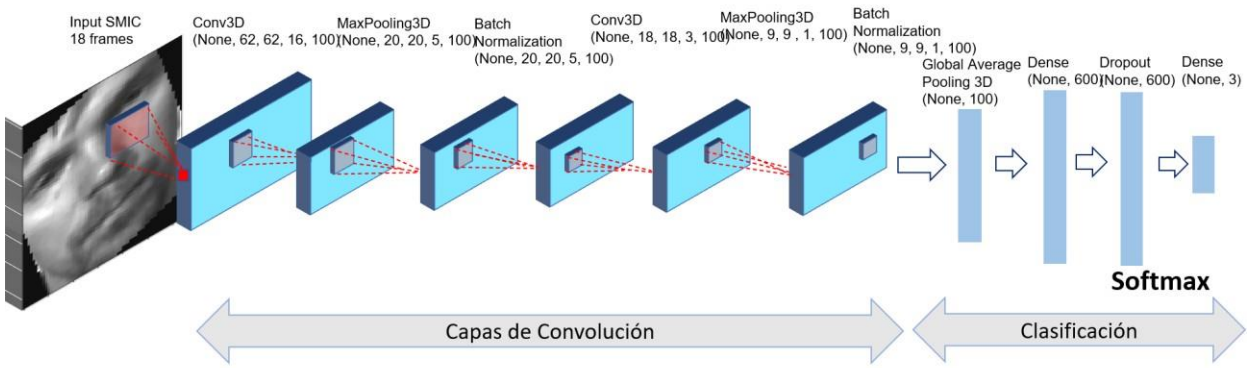


Figura 60 Arquitectura Red Neuronal Convolucional 3D con Data Augmentation para la base de datos SMIC.
Fuente: Elaboración propia

Para este modelo se rotaron las imágenes escogiendo un ángulo aleatorio dentro de este rango [-20, -10, -5, 5, 10, 20]. En la Figura 61 se muestra la imagen de un rostro rotado aleatoriamente y en la Figura 62 se muestra los 18 frames rotados. Para este modelo tambien se trabajó con 18 frames para la base de datos SMIC.

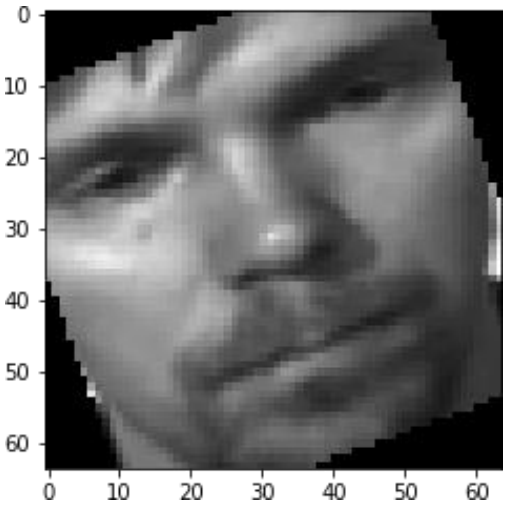


Figura 61 Rostro rotado aleatoriamente de la base de datos SMIC. Fuente: Imagen generada con Google Colab

Figura 62. 18 Frames rotados. Fuente: Imagen generada con Google Colab

En la Tabla 11 se muestran los parámetros de la Red Neuronal Convolucional con Data augmentation para la base de datos SMIC. Se usaron 350 filtros para la primera capa de Convolucion 3D y para la segunda 100, se usó un kernel de tamaño 3 y 300 épocas con EarlyStopping. El total de parámetros para este modelo fue de 336103.

En la Figura 63 se puede observar la gráfica del accuracy en rojo se encuentran los datos de entrenamiento y en naranja los datos de pruebas, se puede observar que en la época 150 el modelo se vuelve más estable, mejora la exactitud con la que predice las etiquetas y aumenta su desempeño, el accurracy para este modelo dio 0.843. En la Figura 64 se encuentran la gráfica de la pérdida, en azul estan los datos de entrenamiento y en naranja los datos de pruebas, en esta gráfica podemos observar que la pérdida va disminuyendo a medida que se aumentan las épocas, esto indica que el modelo está aprendiendo para hacer una mejor predicción.

Tabla 11 Parámetros del modelo de la Red Neuronal Convolucional con Data augmentation para la base de datos SMIC. Fuente: Elaboración propia

Layer (Type)	**Output Shape**	**Param #**
InputLayer	[(None, 64, 64, 18, 1)]	0
Convolución 3D (Conv3D)	(None, 62, 62, 16, 100)	2800
Max Pooling (MaxPooling3D)	(None, 20, 20, 5, 100)	0
Batch Normalization	(None, 20, 20, 5, 100)	400
Convolución 3D (Conv3D)	(None, 18, 18, 3, 100)	270100
Max Pooling (MaxPooling3D)	(None, 9, 9, 1, 100)	0
Batch Normalization	(None, 9, 9, 1, 100)	400
Global Average Pooling 3D	(None, 100)	0
Dense	(None, 600)	60600
Dropout	(None, 600)	0
Dense	(None, 3)	1803

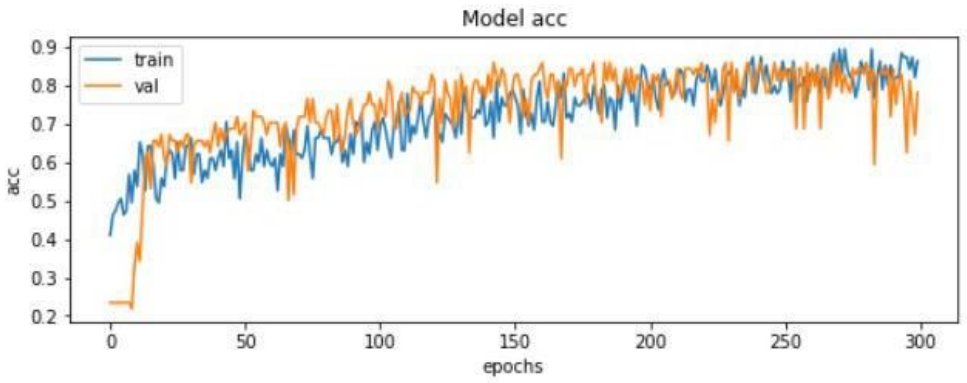


Figura 63 Gráfica del accuracy modelo de la Red Neuronal Convolucional con Data augmentation para la base de datos SMIC. En azul los datos de entrenamiento y en naranja los de validación. Accuracy de 0.843. Fuente: Gráfica generada con Google Colab

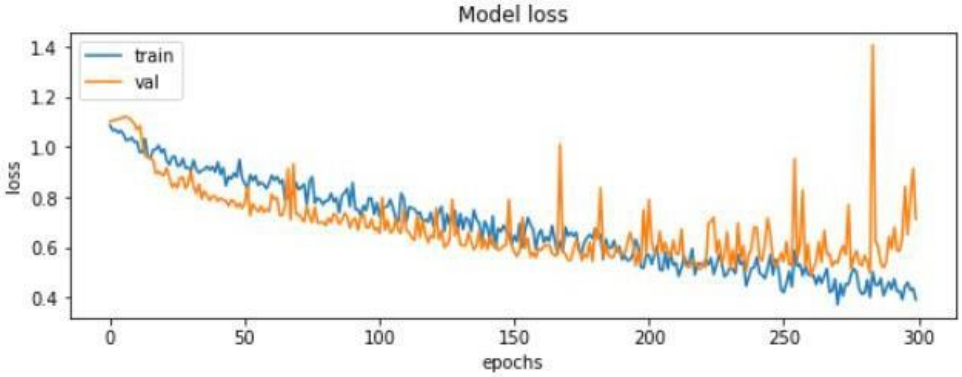


Figura 64 Gráfica de la perdida modelo de la Red Neuronal Convolucional con Data augmentation para la base de datos SMIC. En azul los datos de entrenamiento y en naranja los de validación. Fuente: Gráfica generada con Google Colab

5.2.2.1. Matriz de Confusión

En la Figura 64 está la matriz de confusión para las etiquetas “Negativo”, “Positivo” y Sorpresa”. En la matriz de confusión cada columna representa las predicciones de

cada clase, y en cada fila tenemos el número real de instancias de cada clase; las clases son las etiquetas. La matriz de confusión nos muestra que para estas clases el modelo tiene más cantidad de aciertos que de errores; para la clase Negativo el conteo dio 24, para la clase Positivo el conteo dio 13 y para para la clase sorpresa dio 17. Estos números son mayores que el entregado por la matriz de confusión del anterior modelo. Esto nos permite visualizar que este algoritmo de aprendizaje del modelo está aprendiendo mejor.

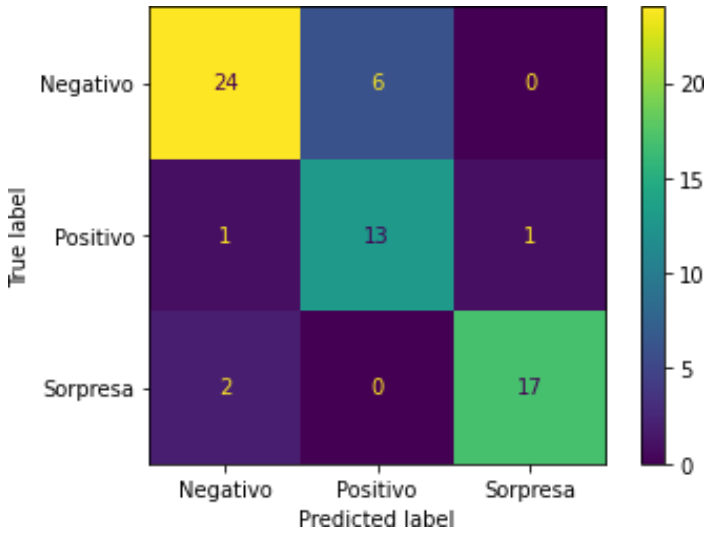


Figura 65 Matriz de Confusión del modelo de la Red Neuronal Convolucional con Data augmentation para la base de datos SMIC. Fuente: Imagen generada con Google Colab

Después del proceso de entrenamiento del modelo usando los datos de entrenamiento y validación, donde se ajustaron los parámetros e hiperparámetros del mismo para obtener el mejor rendimiento con respecto a la exactitud (Accuracy ver

Figura 63 y el menor error de la función de pérdida (Loss ver Figura 64); el modelo se evaluó a través de las métricas f1 score, sensibilidad, precisión, curva ROC y AUC que permitieron evidenciar el rendimiento real del modelo. Un modelo presenta un buen desempeño cuando estas métricas dan cercanas a 1 y por el contrario es un mal indicador si estas métricas dan cercanas a cero. Para este modelo el f1 score macro dio 0.842, el f1 micro dio 0.843, la sensibilidad dio 0.85 y la precisión dio 0.84; se puede observar que las métricas dan un número cercano a uno.

Tabla 12 Resultado de las métricas de rendimiento del modelo CNN 3D con Data Augmentation para la base de datos SMIC. Fuente: Elaboración propia

Modelo	Base de Datos	Precisión	Sensibilidad	Accuracy	F1 Score Macro	F1 Score Micro	AUC
CNN 3D con Data Augmentation	SMIC	0.839	0.854	0.844	0.842	0.844	0.947

5.2.2.2. Curva ROC y AUROC

En la Figura 66 se puede ver la curva ROC con un AUC de 0.947. lo cual indica que el modelo está prediciendo la mayoría de las expresiones correctamente, pero no tanto como lo hace para la base de datos CASME II.

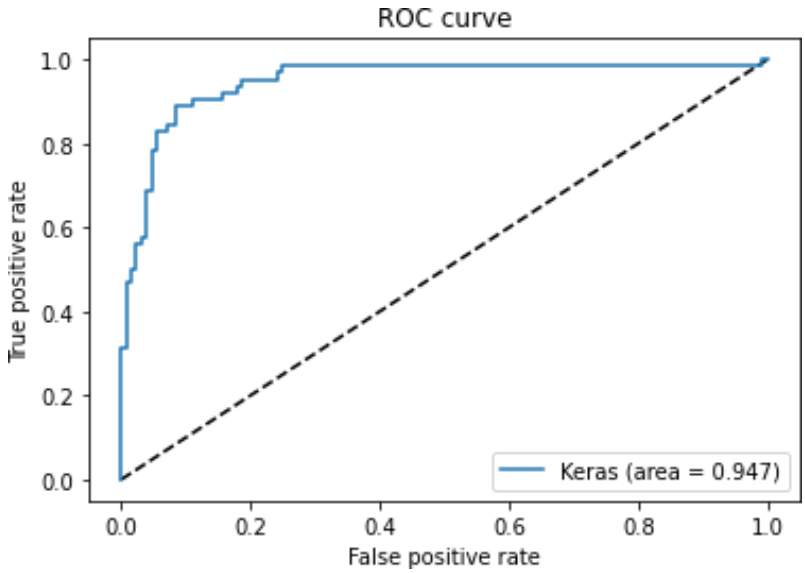


Figura 66 Curva ROC del modelo de la Red Neuronal Convolucional con Data augmentation para la base de datos SMIC. Fuente: Gráfica generada con Google Colab

El modelo se encuentra en Google Colab y Github. Ver Anexo a – Modelo 2 - Base de datos SMIC.

Los dos modelos implementados anteriormente, entrega un buen resultado en las métricas para la base de datos CASME II (Yan WJ, Li X, Wang SJ, Zhao G, Liu YJ, Chen YH & Fu X., 2014) y SMIC (X. Li, T. Pfister, X. Huang, G. Zhao & M. Pietikäinen, 2013). Par el primer modelo implementado tomado de la referencia (S. P. Teja Reddy, et al., 2019) se mejoraron los parámetros de la red y entro mejores resultados de accuray, f1 score, sensibilidad, precisión y curva ROC dieron más cercanas a uno. Para el segundo modelo con data augmentation la matriz de confusión dio mejor para ambas

bases de datos al igual que las métricas de rendimiento del modelo. Pero, aun así, no se estaba teniendo en cuenda la característica temporal de las imágenes ya que son 18 frames de una secuencia de imágenes que cambian dinámicamente. Para superar este inconveniente se montó un modelo temporal con una capa LSTM que permite modelar la variabilidad temporal de un dataset de imágenes a partir de unidades de memoria.

5.3. Modelo temporal profundo para el reconocimiento de expresiones faciales

Para este modelo se implementó una Red Neuronal Convolucional 2D con una distribución de tiempo y una capa Long short term memory LSTM entregando mejores resultados que los dos modelos anteriormente mencionados para ambas bases de datos de microexpresiones, ya que tuvo en cuenta la característica temporal de la secuencia de imágenes de los 18 frames. Este modelo temporal predice mejor la microexpresión facial. En la Figura 67 se muestra la arquitectura del modelo que se construyó de la Red Neuronal Convolucional temporal, este modelo se usó para ambas bases de datos CASME II (Yan WJ, et al., 2014) y para SMIC (X. Li, et al., 2013).

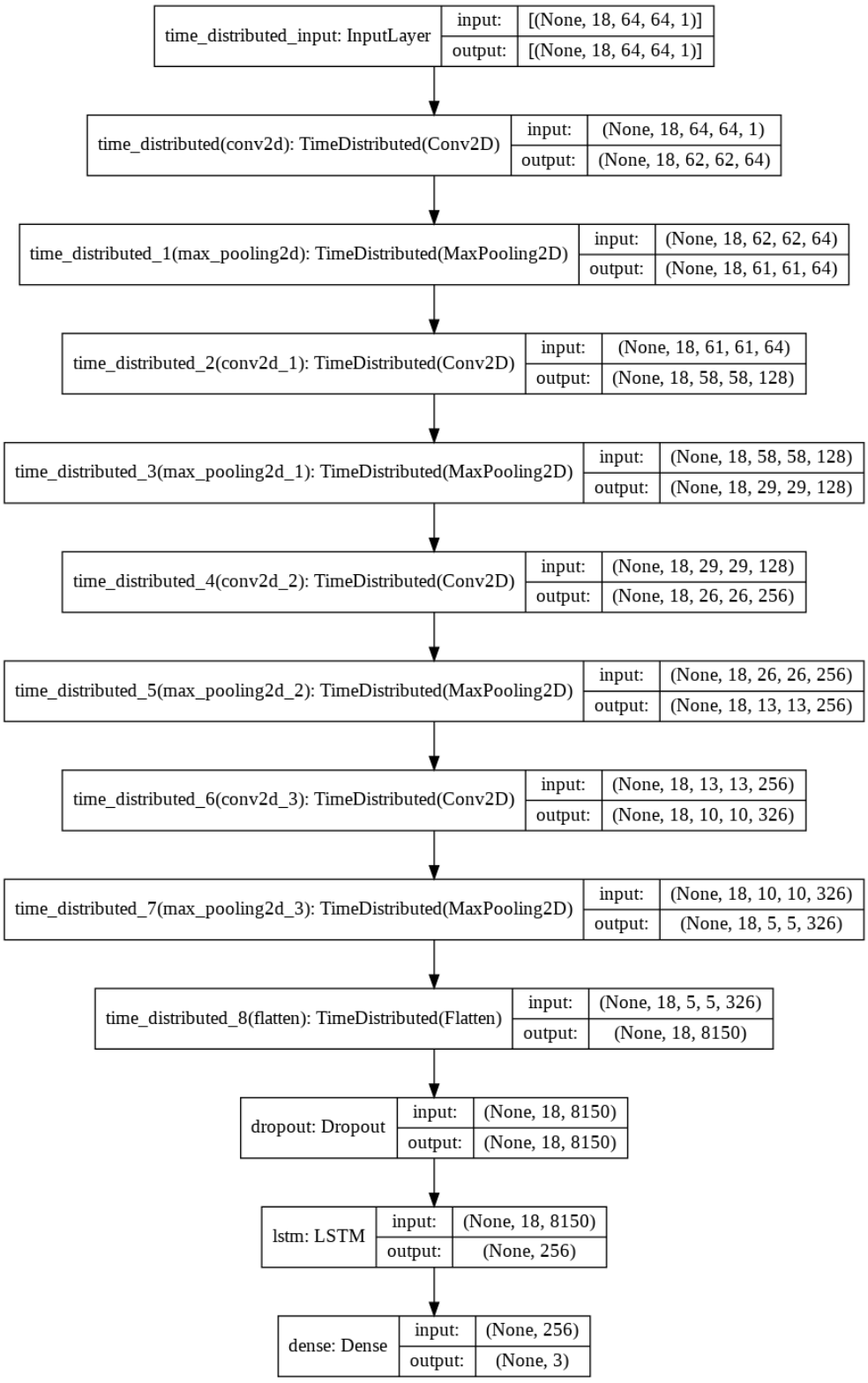


Figura 67 Arquitectura del modelo temporal profundo para el reconocimiento de expresiones faciales.
Fuente: Modelo generado con Google Colab

5.3.1. Base de datos CASME II

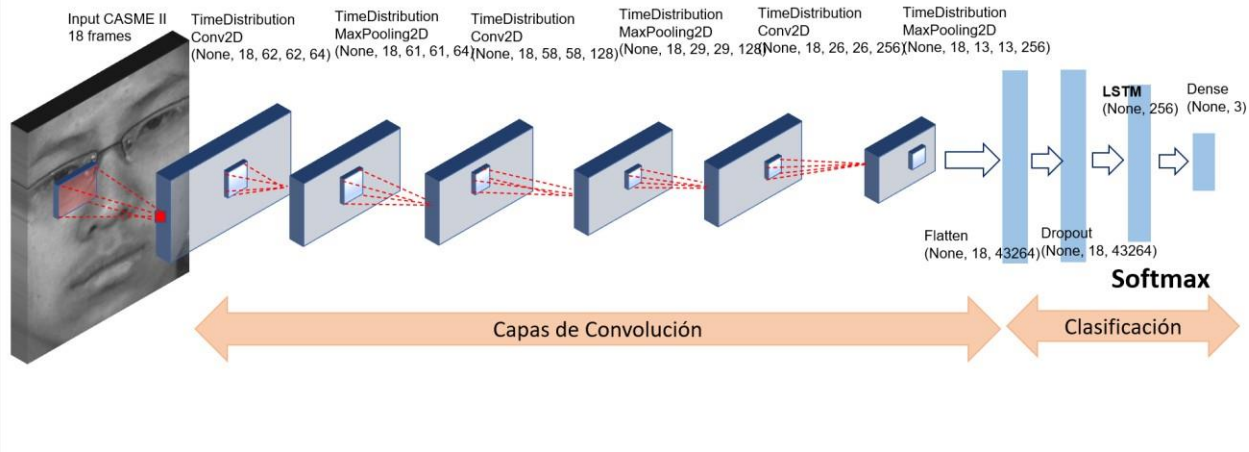


Figura 68 Arquitectura Red Neuronal Convolucional 2D con distribución de tiempo y LSTM para la base de datos CASME II. Fuente: Elaboración propia

Se Construyó una Red Neuronal Convolucional 2D con una distribución de tiempo, un filtro de 64 Para la primera capa Convolucional 2D, 128 filtros para la segunda y 256 para la tercera, tambien se le agregó capa LSTM (Long Short Term Memory) con 256 filtros, una función de activación Softmax y Adam para la optimización. En la Tabla 13 se muestra los parámetros del modelo creado. Este modelo generó 45222659 parámetros.

Tabla 13 Parámetros del modelo temporal con una capa de LSTM para la base de datos CASME II. Fuente: Elaboración propia

Layer (Type)	**Output Shape**	**Param #**
TimeDistribution Conv2D	(None, 18, 62, 62, 64)	640
TimeDistribution MaxPooling2D	(None, 18, 61, 61, 64)]	0
TimeDistribution Convolution2D	(None, 18, 58, 58, 128)	131200
TimeDistribution MaxPooling2D	(None, 18, 29, 29, 128)	0
TimeDistribution Convolution2D	(None, 18, 26, 26, 256)	524544
TimeDistribution MaxPooling2D	(None, 18, 13, 13, 256)	0
TimeDistribution Flatten	(None, 18, 43264)	0
Dropout	(None, 18, 43264)	0
LSTM	(None, 256)	44565504
Dense	(None, 3)	771

En la Figura 69 se muestra la gráfica del accuracy, para este modelo temporal con la base de datos CASME II, se obtuvo un accuracy del 0.980. En la Figura 70 se encuentra las perdidas.

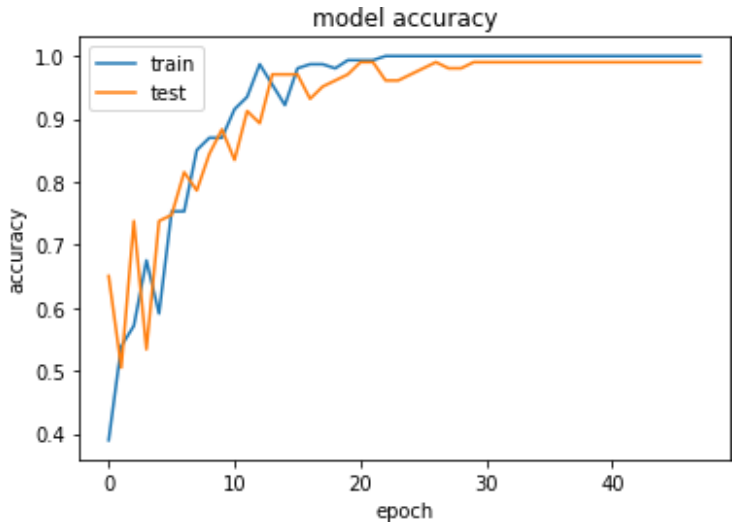


Figura 69 Gráfica del accuracy modelo temporal con una capa de LSTM para la base de datos CASME II. En azul estan los datos de entrenamiento y en naranja de pruebas. Accuracy de 0.980. Fuente: Gráfica generada con Google Colab

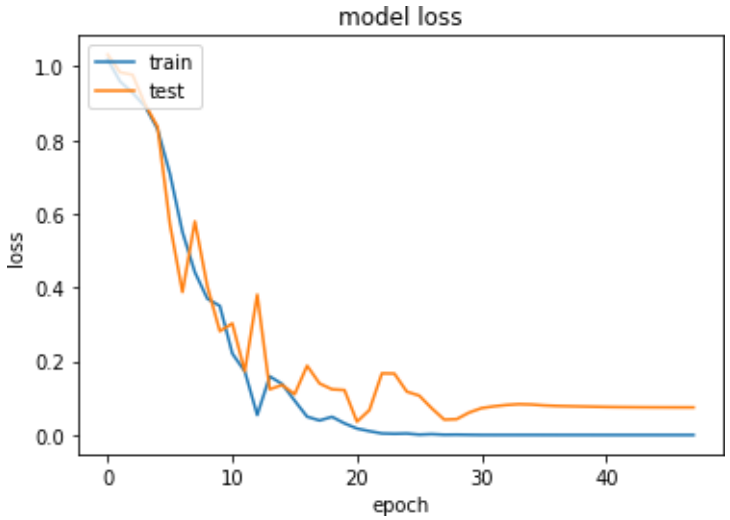


Figura 70 Gráfica de las perdidas modelo temporal con una capa de LSTM para la base de datos CASME II. En azul estan los datos de entrenamiento y en naranja los de pruebas. Fuente: Gráfica generada con Google Colab

5.3.1.1. Matriz de Confusión

En la Figura 71 se encuentra la matriz de confusión para las etiquetas “Angry”, “Happy” y “Disgust”. En la matriz de confusión cada columna representa las predicciones de cada clase, y en cada fila tenemos el número real de instancias de cada clase; las clases son las etiquetas. La matriz de confusión nos muestra que para estas clases el modelo tiene más cantidad de aciertos que de errores; para la clase Angry el conteo dio 52, para la clase Happy el conteo dio 33 y para para la clase Disgust dio 17. Estos números son mayores que el entregado por la matriz de confusión de los dos modelos implantados anteriormente. Esto nos permite visualizar que este algoritmo de aprendizaje del modelo temporal está aprendiendo mejor que el modelo MicroExpSTCNN y el CNN 3D con data augmentation.

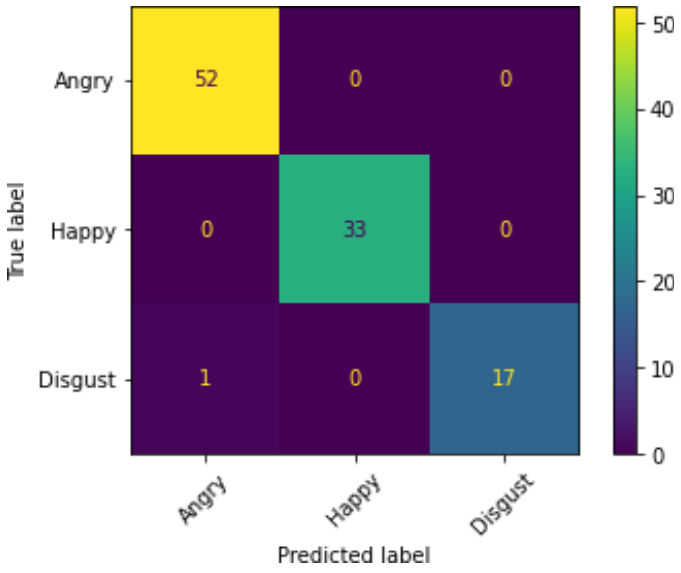


Figura 71 Matriz de Confusión para el modelo temporal y la base de datos de microexpresiones faciales CASME II. Fuente: Imagen generada con Google Colab

Después del proceso de entrenamiento del modelo usando los datos de entrenamiento y validación, donde se ajustaron los parámetros e hiperparámetros del mismo para obtener el mejor rendimiento con respecto a la exactitud (Accuracy ver Figura 69 Figura *63*Figura *45*Figura *39*y el menor error de la función de pérdida (Loss ver Figura 70Figura *69*); el modelo se evaluó a través de las métricas f1 score, sensibilidad, precisión, curva ROC y AUC que permitieron evidenciar el rendimiento real del modelo. El f1 score macro dio 0.972, el f1 score micro dio 0.980, la sensibilidad dio 0.962 y la precisión dio 0.984. Resultados muy cercanos a uno y mayores comparados con los dos modelos anteriores. En la Tabla 14 se muestran los resultados de accuracy,

precisión, f1 y sensibilidad para el modelo temporal LSTM para la base de datos CASME II.

Tabla 14 Resultado de las métricas de rendimiento del modelo CNN temporal con LSTM para la base de datos CASME II. Fuente: Elaboración propia gráfica

Modelo	Base de Datos	Precisión	Sensibilidad	Accuracy	F1 Score Macro	F1 Score Micro	AUC
CNN Temporal con LSTM	CASME II	0.984	0.962	0.980	0.972	0.980	0.999

5.3.1.2. Curva ROC y AUROC

En la Figura 72 la curva ROC con un AUC de 0.999; es un número muy cercano a uno, lo cual indica un excelente comportamiento del modelo, ya que el modelo está aprendiendo a reconocer las microexpresiones faciales de las etiquetas“Angry”, “Happy” y “Disgust”. El AUC es mayor comparado con los dos modelos implementados anteriormente. La curva ROC es una representación gráfica que ilustra la relación entre la sensibilidad y la especificidad de un sistema clasificador para diferentes puntos de corte.

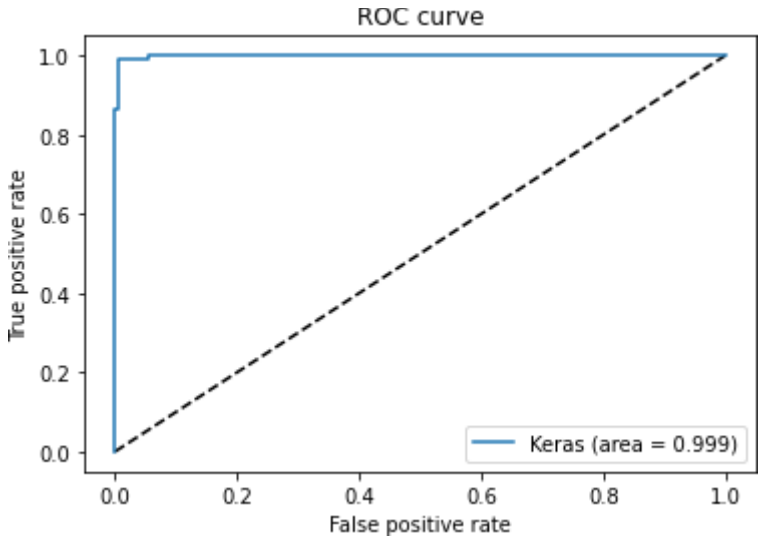


Figura 72 Curva ROC para el modelo temporal y la base de datos de microexpresiones faciales CASME II.

Fuente: Gráfica generada con Google Colab

El modelo se encuentra en Google Colab y en Github. Ver Anexo a - Modelo 3 - Base de datos CASME II.

5.3.2. Base de datos SMIC

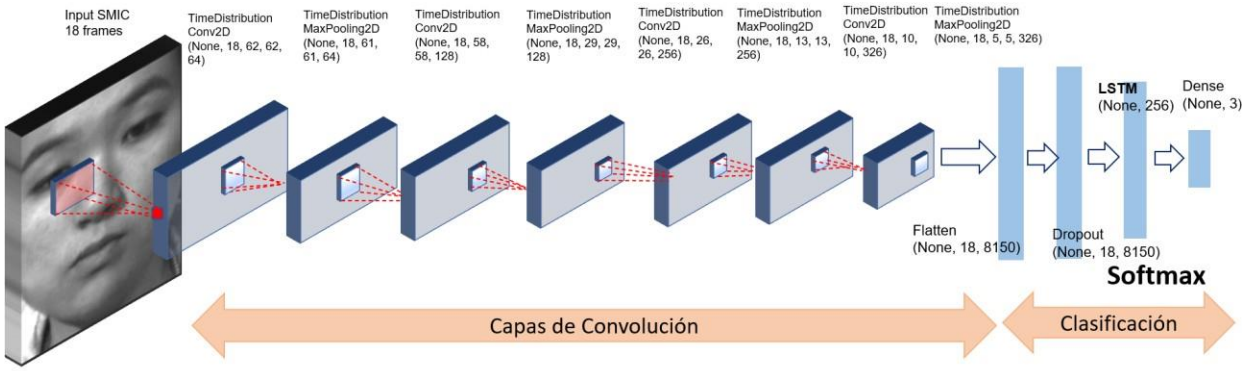


Figura 73 Arquitectura Red Neuronal Convolucional 2D con distribución de tiempo y LSTM para la base de datos SMIC. Fuente: Elaboración propia

Se Construyó una Red Neuronal Convolucional 2D con un modelo Secuencial, una distribución de tiempo, un filtro de 128 para la primera capa de Convolución, 256 para la segunda y 326 para la tercera, a diferencia de la base de datos CASME II, a esta se le agregó una capa de Convolución más. También se agregó una capa LSTM (Long Short Term Memory) con un filtro de 256. En la Tabla 15 se muestran los parámetros de la Red Neuronal Convolucional temporal. Este modelo generó 10601545 parámetros.

Tabla 15 Parámetros del modelo temporal con una capa de LSTM para la base de datos SMIC. Fuente: Elaboración propia

Layer (Type)	**Output Shape**	**Param #**
TimeDistribution Conv2D	(None, 18, 62, 62, 64)	640
TimeDistribution MaxPooling2D	(None, 18, 61, 61, 64)]	0
TimeDistribution Convolution2D	(None, 18, 58, 58, 128)	131200
TimeDistribution MaxPooling2D	(None, 18, 29, 29, 128)	0
TimeDistribution Convolution2D	(None, 18, 26, 26, 256)	524544
TimeDistribution MaxPooling2D	(None, 18, 13, 13, 256)	0
TimeDistribution Convolution2D	(None, 18, 10, 10, 326)	1335622
TimeDistribution MaxPooling2D	(None, 18, 5, 5, 326)	0
TimeDistribution Flatten	(None, 18, 8150)	0
Dropout	(None, 18, 8150)	0
LSTM	(None, 256)	8608768
Dense	(None, 3)	771

En la Figura 74 se encuentra el resultado del accuracy, para este modelo dio 0.922 y en la Figura 75 se encuentra la gráfica de las perdidas. El accuracy nos muestra que después de la época 60 el modelo se volvió más estable, y la gráfica de perdida nos

indica que el modelo está aprendiendo cada vez más a medida que transcurren las épocas.

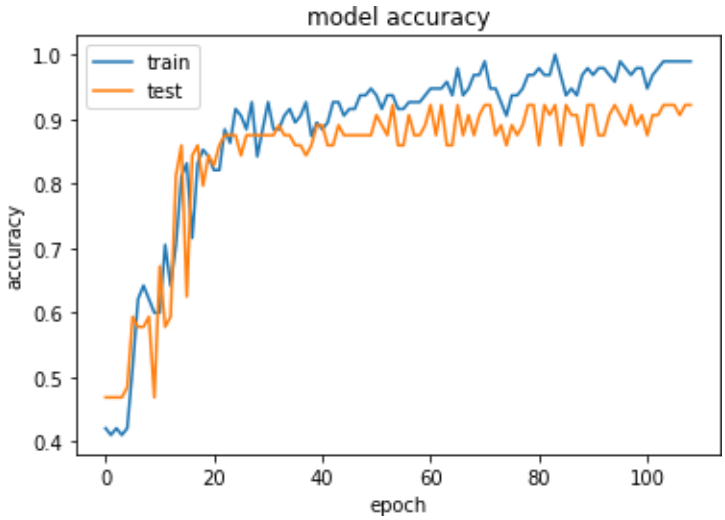


Figura 74 Gráfica del accuracy modelo temporal y la base de datos de microexpresiones faciales SMIC. En azul los datos de entrenamiento y en naranja los de pruebas. Accuracy de 0.922. Fuente: Gráfica generada con Google Colab

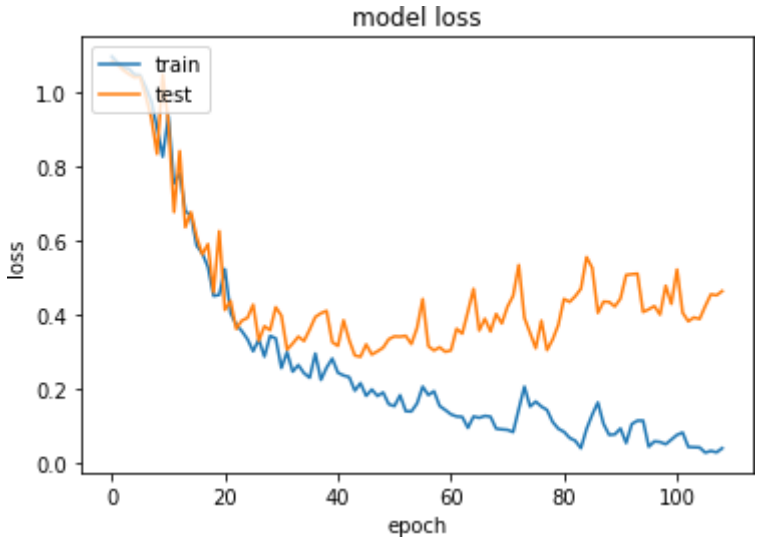


Figura 75 Gráfica de las perdidas modelo temporal y la base de datos de microexpresiones faciales SMIC. En azul los datos de entrenamiento y en naranja los de pruebas. Fuente: Gráfica generada con Google Colab

5.3.2.1. Matriz de Confusión

En la Figura 76 se muestra la matriz de confusión para las etiquetas Negativo, Positivo y Neutro. En la matriz de confusión cada columna representa las predicciones de cada clase, y en cada fila tenemos el número real de instancias de cada clase; las clases son las etiquetas. La matriz de confusión nos muestra que tiene muy poco conteo de errores y más de aciertos; para la clase Negativo el conteo dio 27, para la clase Positivo el conteo dio 13 y para para la clase Neutro dio 19. Esto nos permite visualizar que este algoritmo de aprendizaje del modelo temporal está aprendiendo mejor que el modelo MicroExpSTCNN y el CNN 3D con data augmentation.

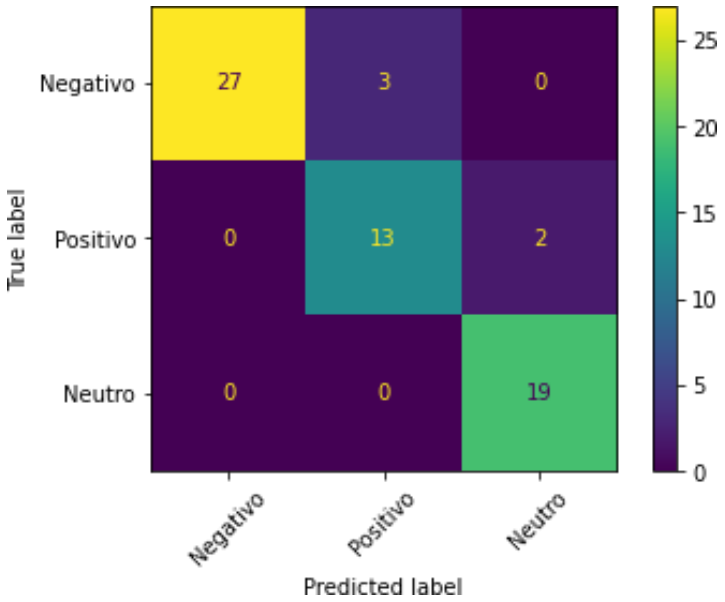


Figura 76 Matriz de Confusión para el modelo temporal y la base de datos de microexpresiones faciales SMIC.
Fuente: Imagen generada con Google Colab

Después del proceso de entrenamiento del modelo usando los datos de entrenamiento y validación, donde se ajustaron los parámetros e hiperparámetros del mismo para obtener el mejor rendimiento con respecto a la exactitud (Accuracy ver Figura 74 Figura *69*Figura *63*Figura *45*Figura *39*y el menor error de la función de pérdida (Loss ver Figura 75Figura 70Figura *69*); el modelo se evaluó a través de las métricas f1 score, sensibilidad, precisión, curva ROC y AUC que permitieron evidenciar el rendimiento real del modelo. El f1 score macro para este modelo temporal dio 0.912, el f1 micro 0.922, la sensibilidad dio 0.922 y una precisión de 0.906; estas métricas al ser un número muy cercano a uno nos indican un buen desempeño del modelo y del algoritmo de aprendizaje. En la Tabla 16 muestra los resultados de accuracy, f1, precisión y sensibilidad para el modelo temporal para la base de datos SMIC.

Tabla 16 Resultado de las métricas de rendimiento del modelo CNN temporal con LSTM para la base de datos SMIC. Fuente: Elaboración propia

Modelo	Base de Datos	Precisión	Sensibilidad	Accuracy	F1 Score Macro	F1 Score Micro	AUC
CNN Temporal con LSTM	SMIC	0.906	0.922	0.922	0.912	0.922	0.974

5.3.2.2. Curva ROC y AUROC

En la Figura 77 la curva ROC con un área de 0.974, un número muy cercano a uno, esto nos indica que el modelo está aprendiendo a reconocer las microexpresiones faciales de las etiquetas del set de imágenes.

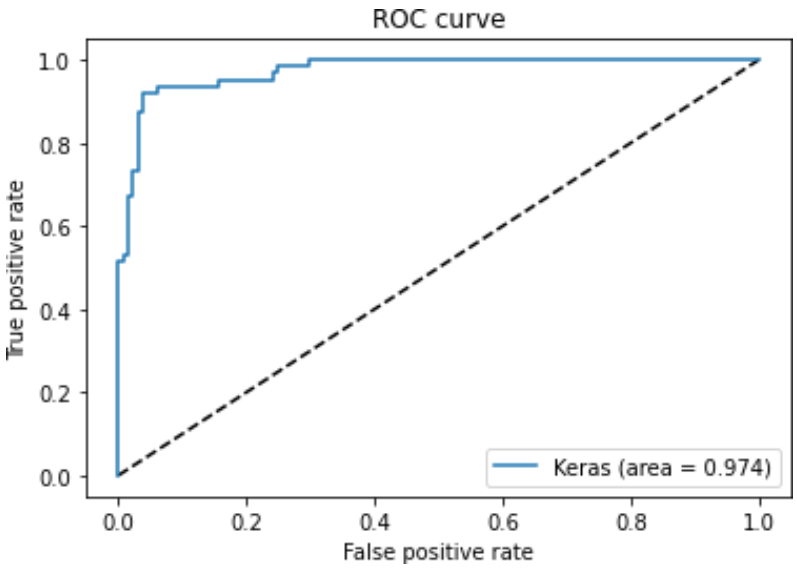


Figura 77 Curva ROC para el modelo temporal y la base de datos de microexpresiones faciales SMIC. Fuente: Imagen generada con Google Colab

El modelo se encuentra en Google Colab y en Github. Ver Anexo a - Modelo 3 - Base de datos SMIC.

5.4. Evaluación cuantitativa de los modelos propuestos para diferentes métricas

En la Tabla 17 se puede encontrar los resultados para los tres modelos implementados para las dos bases de datos de microexpresiones faciales CASME II (Yan WJ, et al., 2014) y SMIC (X. Li, et al., 2013). El primer modelo implementado fue el MicroExpSTCNN tomando como referencia el artículo de S. P. Teja Reddy, S. Teja Karri, S. R. Dubey and S. Mukherjee, "Spontaneous Facial Micro-Expression Recognition using 3D Spatiotemporal Convolutional Neural Networks" (S. P. Teja Reddy, et al., 2019). El accurracy que se logró obtener para la base de datos SMIC fue 91.6 % y para la base de datos CASME II 90 %, estos datos superan las métricas obtenidas del modelo de referencia. Con el segundo modelo implementado con Data Augmentation se puedo mejorar las métricas para la base de datos CASME II obteniendo un accuracy de 94.2 %, por otro lado, para la base de datos SMIC el acurracy disminuyo con un valor de 84.4 %. Por último, se creó un modelo temporal con LSTM donde se pudo mejorar las métricas para ambas bases de datos SMIC y CASME II; para CASME II se obtuvo un accuracy de 98 % y para SMIC de 92.2 %. Al agregar la capa LSTM al modelo permito tener en cuenta el dinamismo de las imágenes y adaptarse de forma única a la entrada; ya que la entrada era 18 frames de la misma microexpresión facial; eso mejoró mucho el rendimiento del modelo, ya que se puedo obtener métricas muy cercanas a 1 y superiores a los dos modelos desarrollados anteriormente.

Tabla 17 Resultado de las métricas para los tres modelos implementados. Fuente: Elaboración propia

Modelo	Base de Datos	Precisión	Sensibilidad	Accuracy	F1 Score	AUC

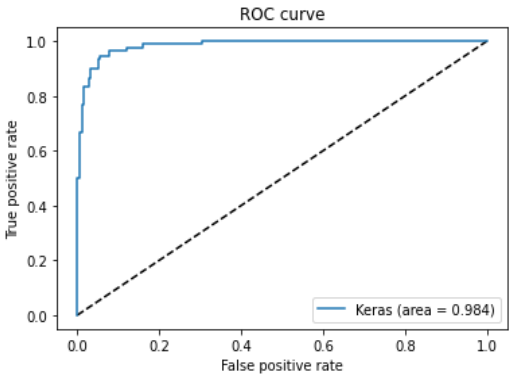
CNN 3D MicroExpSTCNN	CASME II	0.860	0.879	0.9	0.868	0.984
CNN 3D MicroExpSTCNN	SMIC	0.891	0.928	0.916	0.906	0.993
CNN 3D con DataAugmentation	CASME II	0.936	0.937	0.942	0.935	0.989
CNN 3D con DataAugmentation	SMIC	0.839	0.854	0.844	0.842	0.947
CNN Temporal conLSTM	**CASME II**	**0.984**	**0.962**	**0.980**	**0.972**	**0.999**
CNN Temporal conLSTM	**SMIC**	**0.906**	**0.922**	**0.922**	**0.912**	**0.974**

En la Tabla 18, Tabla 19 y Tabla 20 se puede ver el consolidado de los tres modelos para las dos bases de datos de microexpresiones faciales, esto con el fin de tener una visión general de los resultados.

Tabla 18 Consolidado de la matriz de confusión, curva ROC y gráfica del accuracy para el modelo MicroExpSTCNN para la base de datos CASME II y SMIC. Fuente: Elaboración propia

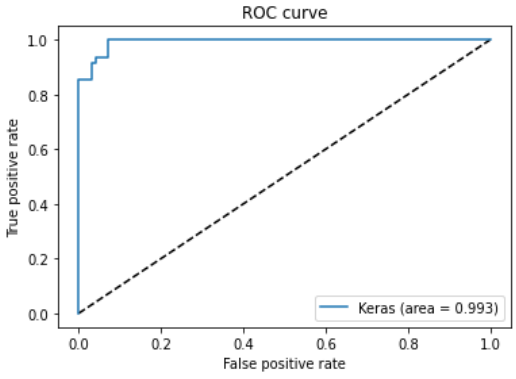
CASME II

True label	**Angry**	51	0	4
	Happy	3	19	0
	Disgust	0	2	11
		Angry	**Happy**	**Disgust**
		Predicted label		



SMIC

True label	**Negativo**	23	3	0
	Positivo	0	9	1
	Sorpresa	0	0	12
		Negativo	**Positivo**	**Sorpresa**
		Predicted label		



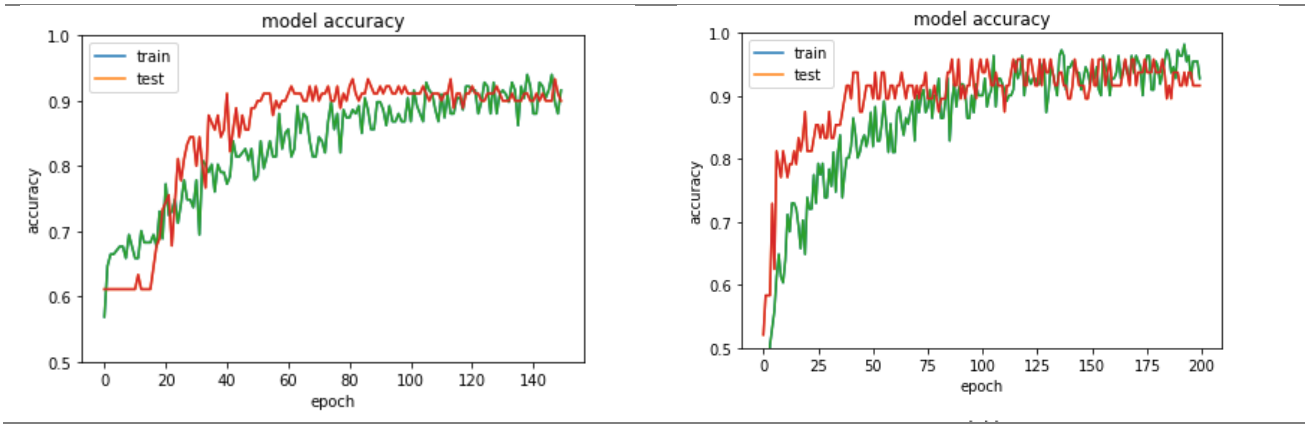


Tabla 19 Consolidado de la matriz de confusión, curva ROC y gráfica del accuracy para el modelo CNN 3D con data augmentation para la base de datos CASME II y SMIC. Fuente: Elaboración propia

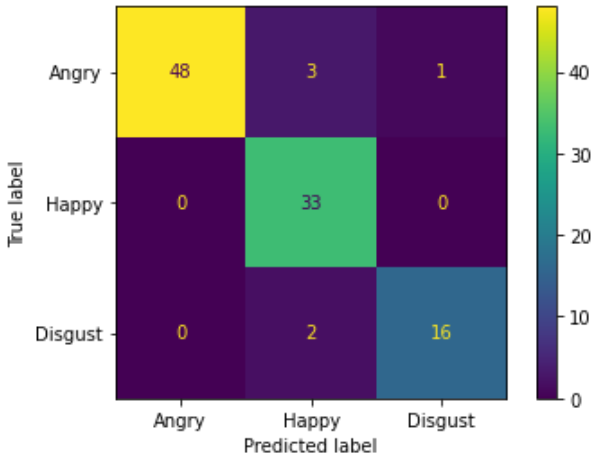
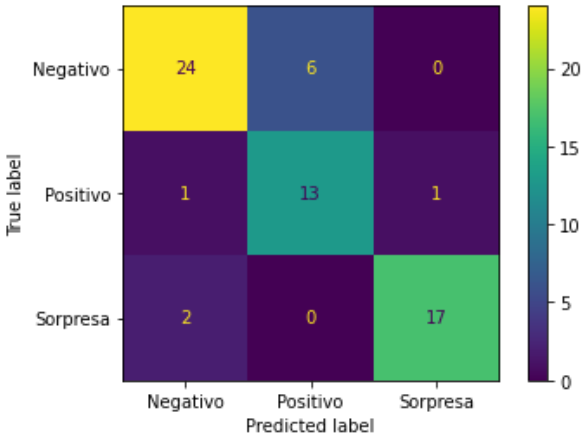
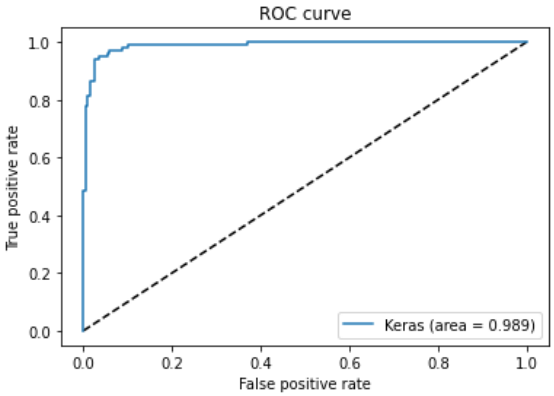
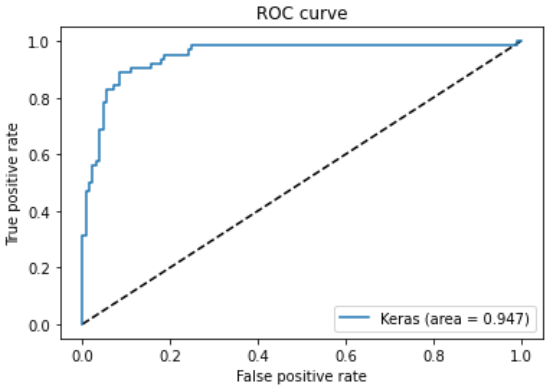
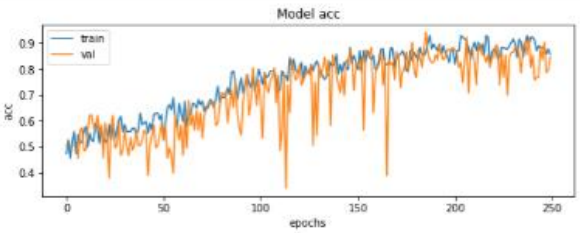
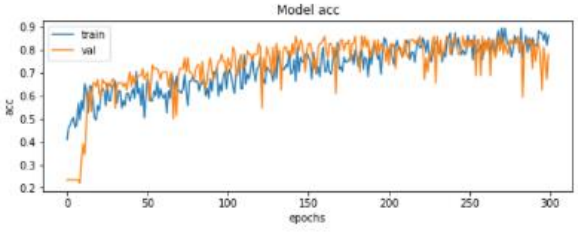
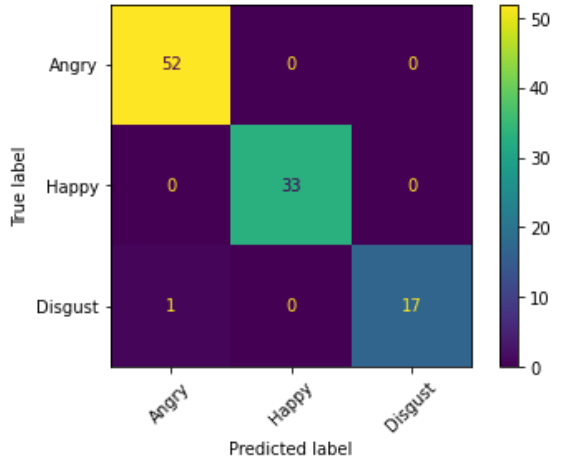
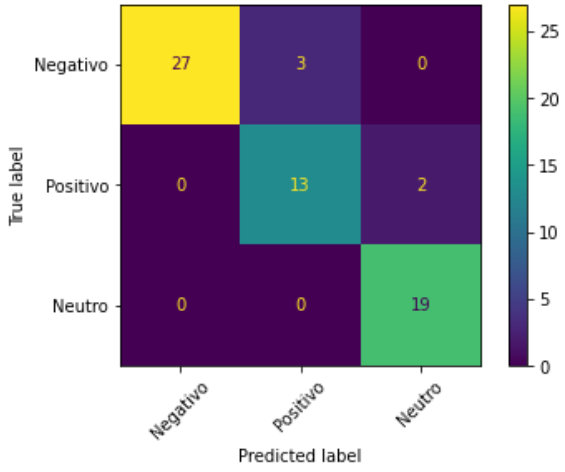
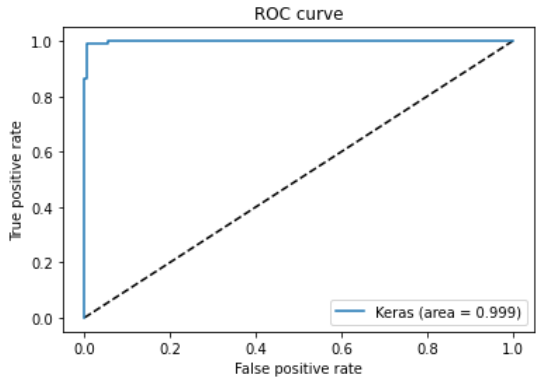
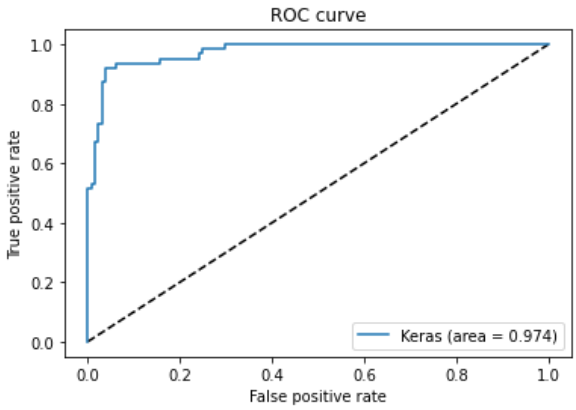
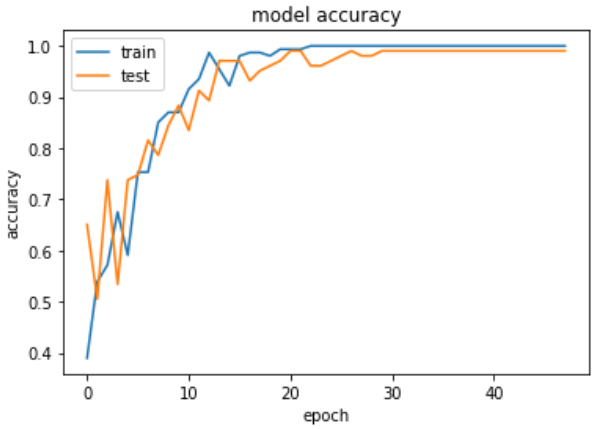
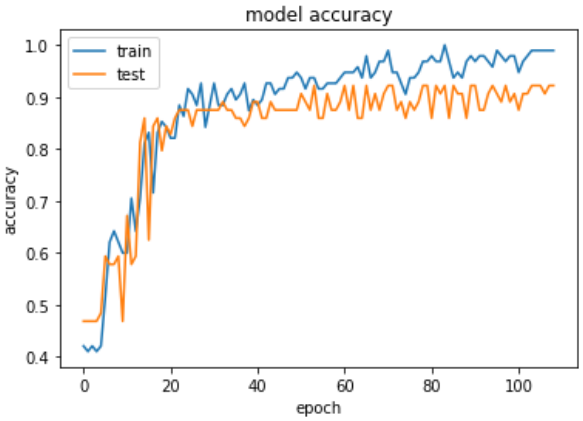
CASME II	SMIC
	
	
	

Tabla 20 Consolidado de la matriz de confusión, curva ROC y gráfica del accuracy para el modelo CNN temporal con una capa LSTM. Fuente: Elaboración propia

CASME II	SMIC
	
	
	

5.5. Comparación con el estado del arte

El primero modelo implementado MicroExpSTCNN fue implementado anteriormente por S. P. Teja Reddy, S. Teja Karri, S. R. Dubey and S. Mukherjee, en el artículo "Spontaneous Facial Micro-Expression Recognition using 3D Spatiotemporal Convolutional Neural Networks" (S. P. Teja Reddy, et al., 2019), ellos implementaron una Red Neuronal Convolucional 3D para la base de datos SMIC (X. Li, et al., 2013) y CASME II (Yan WJ, et al., 2014).

En la Tabla 22 se puede ver que el mejor accuracy para la base de datos CASME II que pudieron obtener fue del 87.80 %, en este trabajo de grado el mejor accuracy que se pudo obtener fue del 90%, mejorando los resultados de modelo notoriamente; en este trabajo de grado para el modelo MicroExpSTCNN se le trabajo con 150 épocas, se creó el modelo en Google Colab y se trabajó con GPUs, mejorando el tiempo de respuesta y entrenamiento del modelo. El modelo original (S. P. Teja Reddy, et al., 2019) trabajó con 80 % de datos de entrenamiento y 20 % de validación, en este trabajo de grado se trabajó con el 70 % de entrenamiento y 30 % de validación.

Tabla 21 La precisión media con desviación estándar para los modelos MicroExpSTCNN y MicroExpFuseNet propuestos sobre los conjuntos de datos CAS (ME) 2 y SMIC. La desviación estándar se calcula de 91 a 100 épocas para mostrar la estabilidad de los modelos en las últimas épocas. Fuente: Tomado el artículo original (S. P. Teja Reddy, et al., 2019)

Database	MicroExpSTCNN	MicroExpFuseNet	
		Intermediate Fusion	Late Fusion
CAS(ME)^2	82.20±5.02%	78.57±5.78%	73.20±3.87%
SMIC	62.50±2.55%	51.20±3.56%	59.95±3.19%

Tabla 22 El rendimiento del modelo MicroExpSTCNN en términos de precisión con diferentes tamaños de filtro sobre el conjunto de datos CAS (ME) 2. El mejor resultado se resalta en negrita. Fuente: Tomado el artículo original (S. P. Teja Reddy, et al., 2019)

Filter Size	Accuracy [%]	Filter Size	Accuracy [%]
3 x 3 x 3	82.93	5 x 5 x 15	51.22
3 x 3 x 7	80.49	5 x 5 x 19	29.27
3 x 3 x 15	**87.80**	7 x 7 x 3	70.73
3 x 3 x 19	29.27	7 x 7 x 7	51.22
5 x 5 x 3	63.30	7 x 7 x 15	29.27
5 x 5 x 7	58.54	7 x 7 x 19	51.22

Tabla 23 Matriz de confusión para el modelo MicroExpSTCNN sobre la base de datos CASME II. Fuente: Tomado el artículo original (S. P. Teja Reddy, et al., 2019)

Class	Happy	Angry	Disgust
Happy	19	1	1
Angry	0	11	1
Disgust	1	1	6

Tabla 24 Matriz de confusión para el modelo MicroExpSTCNN sobre la base de datos SMIC. Fuente: Tomado el artículo original (S. P. Teja Reddy, et al., 2019)

Class	Negative	Positive	Surprise
Negative	14	0	5
Positive	2	2	2
Surprise	1	0	6

En la Tabla 25 se encuentra la comparación entre el modelo MicroExpSTCNN propuseto por (S. P. Teja Reddy, et al., 2019) y el modelo MicroExpSTCNN modficado y mejorado en este trabajo de grado.

Tabla 25 Comparación del modelo MicroExpSTCNN propuesto por (S. P. Teja Reddy, et al., 2019) contra el modelo MicroExpSTCNN modificado en este trabajo de grado. Fuente: Elaboración propia

Fuente	Modelo	Base de Datos	Precisión	Sensibilidad	Accuracy	F1 Score	AUC
Propuesto poreste trabajo de grado	CNN 3D MicroExpSTCNN	CASME II	0.860	0.879	0.9	0.868	0.984
(S. P. Teja Reddy, et al., 2019)	CNN 3D MicroExpSTCNN	CASME II	*No reportado en el artículo*	*No reportado en el artículo*	0.878	*No reportado en el artículo*	*No reportado en el artículo*
Propuesto poreste trabajo de grado	CNN 3D MicroExpSTCNN	SMIC	0.891	0.928	0.916	0.906	0.993
(S. P. Teja Reddy, et al., 2019)	CNN 3D MicroExpSTCNN	SMIC	*No reportado en el artículo*	*No reportado en el artículo*	0.625	*No reportado en el artículo*	*No reportado en el artículo*

5.6. Evaluación de la complejidad del modelo

Para realizar una evaluacion objetiva de las librerias utilizadas en Google Colab, se reportan los tiempos de GPU, CPU y Flops. Los Flops (del inglés floating point operations per second) son una medida de rendimiento computacional, especialmente en cálculos científicos que requieren un gran uso de operaciones de coma flotante. Los Flops son una medida más precisa que medir instrucciones por segundo. Las pruebas fueron realizadas en el servidor Backend de Google, el cual cuenta con un procesador Intel(R) Xeon(R) CPU a 2.20GHz, una GPU Nvidia K80/T4 con 12 GB de memoria y un rendimiento máximo de 8.1 TFLOPs.

Tabla 26 Comparación del tiempo de cómputo en la predicción de los 3 modelos desarrollados en este trabajo de grado. Fuente: Elaboración propia

Modelo	Base de Datos	Número total de parámetros	Tiempo de CPU [ms]	Tiempo de GPU [ms]	FLOPs [G]
CNN 3D MicroExpSTCNN	CASME II	1643267	-	-	0.1366
CNN 3D MicroExpSTCNN	SMIC	3080675	-	-	0.2562
CNN 3D con Data Augmentation	CASME II	75939	3753.628	45.484	0.7748
CNN 3D con Data Augmentation	SMIC	1019103	3484.267	44.447	3.0213
CNN Temporal con LSTM	CASME II	45222659	3626.654	46.327	28.7562
CNN Temporal con LSTM	SMIC	10601545	3699.328	48.487	33.5638

En la Tabla 26 se puede ver el registro de los parámetros y Flops de los tres modelos y cada uno para las dos bases de datos. Los Flops no estan pasando de las Gigas y comparándolos con el número de parámetros se evidencia que los modelos propuestos son escalables en esquemas de computación con bajos recursos.

5.7. Aplicación para el reconocimiento de microexpresiones faciales

Como producto de este trabajo de grado se creó una aplicación web que corre sobre una url, la cual se construyó sobre el lenguaje Python, se basó en un proyecto creado con Flask para el reconocimiento de expresiones faciales (Paulvangent, 2021). Esta herramienta reconoce las microexpresiones faciales creando el modelo y cargando los pesos entrenados previamente en Google Colab. Se agregó la captura de 18 frames que usan los tres modelos desarrollados en este trabajo de grado, hace un reconocimiento del rostro, cargar el modelo, los pesos entrenados y guardar las 18 imágenes procesadas. Las librerías de TensorFlow y Keras se instalaron por medio de Anaconda para poder correrlos en Windows 10, Se creo una variable de entorno llamada “DeepLearning_Env” en Anaconda para instalar las librerías de Tensorflow y Keras como se puede ver en la Figura 78. Sobre la variable de entorno se corrió el proyecto por medio de la consola de comandos de Windows 10 como se puede ver en la Figura 79 y Figura 80.

En la Figura 81 se muestra el video que se guardó con antelación en una carpeta del proyecto, el cual lo carga, procesa 18 frames cada 0.02 segundos, predice la microexpresión con el modelo y los pesos cargados y dibuja un cuadro azul sobre el area del rostro, la cual se ver dibujada sobre el video. Flask carga el video sobre el URL local: http://127.0.0.1:4996/. Ver Figura 81.

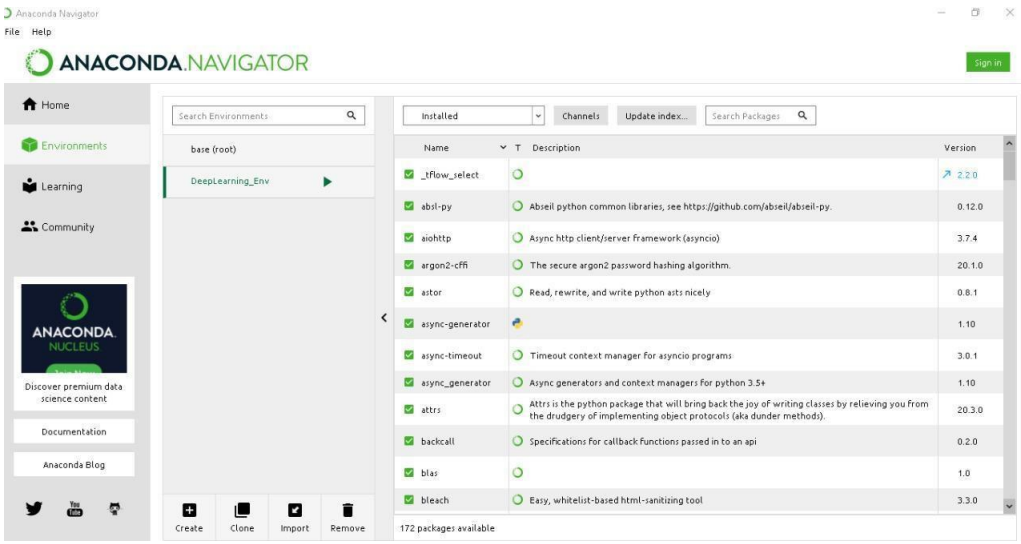


Figura 78 Ananconda.Navigator variable de entorno “DeepLearning_Env” creada. Fuente: Imagen tomada de ANACONDA instalado en el equipo donde se desarrolló este trabajo de grado.

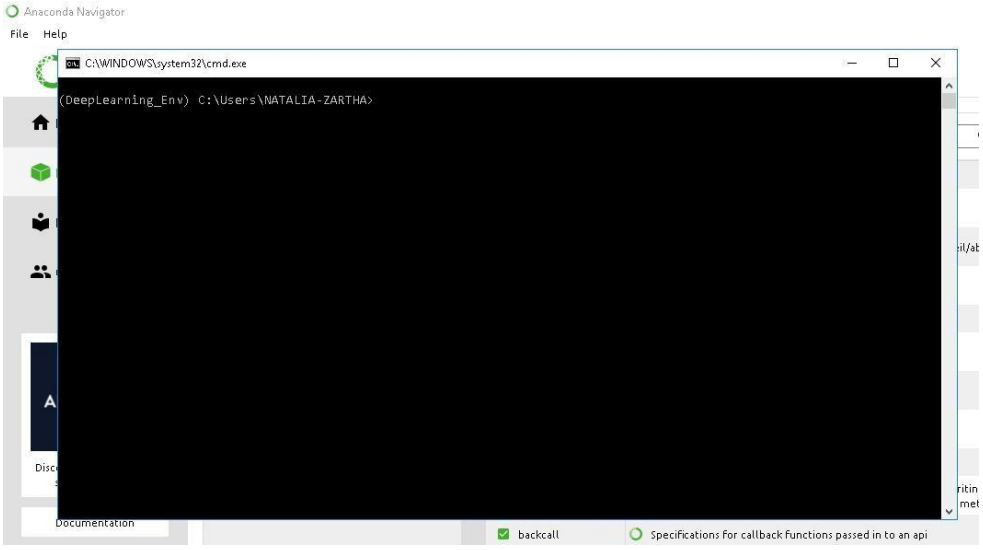


Figura 79 Consola de comandos de Windows 10 corriendo sobre la variable de entorno “DeepLearning_Env”. Fuente: Imagen tomada del equipo donde se desarrolló este trabajo de grado

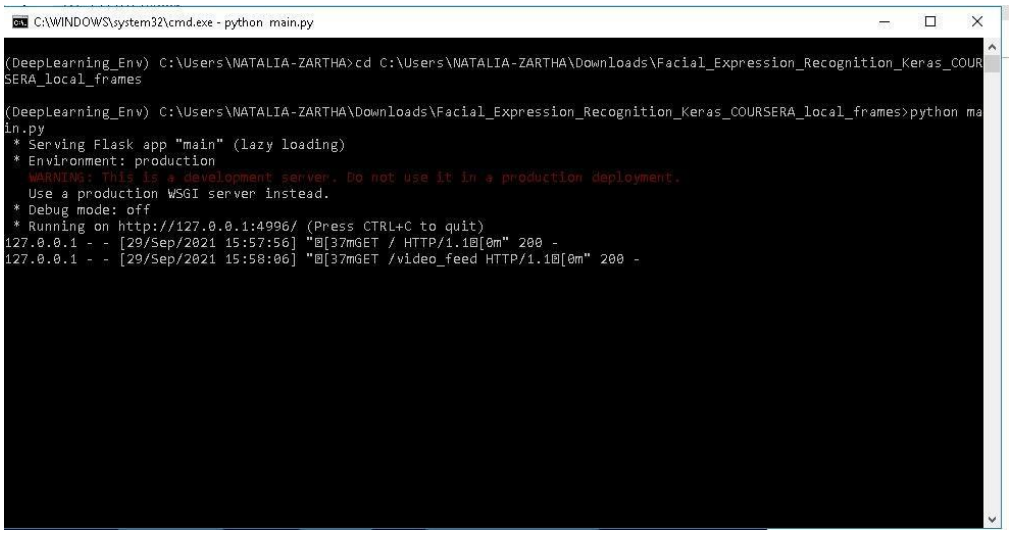


Figura 80 Corriendo la aplicación creada para el reconocimiento de microexpresiones faciales. Fuente: Imagen tomada del equipo donde se desarrolló este trabajo de grado

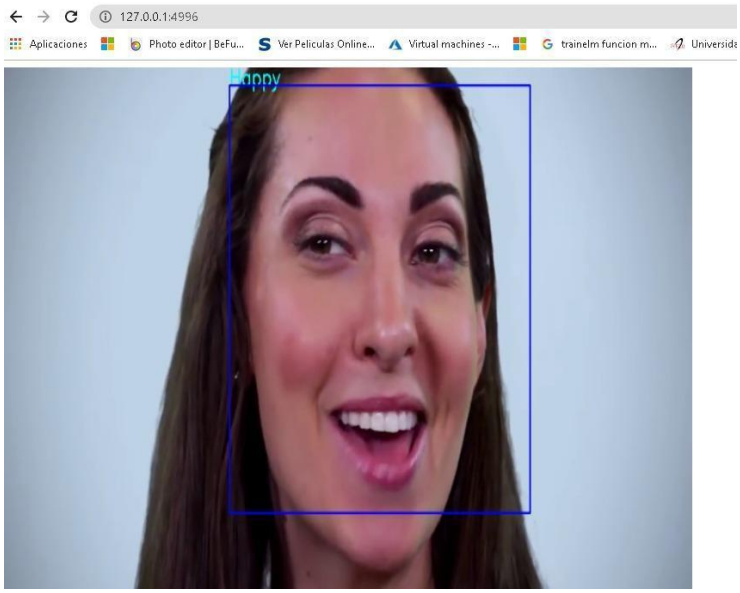


Figura 81 Aplicación creada con Flask y Python para el reconocimiento de microexpresiones faciales. Fuente: Captura de pantalla tomada del equipo donde se desarrolló este trabajo de grado

El codigo de la herramienta que se desarrolló para este trabajo de grado se encuentra en el repositorio de Github y de Google Drive. Las url de los repositorios se pueden ver en el Anexo c.

5.7.1. Herramientas

Las herramientas utilizadas para desarrollar la aplicación se encuentran a continuación acompañadas de su version.

Tabla 27 Versiones de las herramientas, lenguajes y frameworks utilizados para desarrollar la aplicación de reconocimiento de microexpresiones faciales. Fuente: Elaboración propia

Herramienta/Lenguaje/Framework	Version
Python	3.6.12
Numpy	1.19.2
OpenCV (cv2)	4.5.1.48
Flask	1.1.2
TensorFlow	2.1.0
Keras	2.3.1
Anaconda3	2020

5.7.2. Funcionalidades

La aplicación para el reconocimiento de microexpresiones faciales tiene las siguientes funcionalidades:

1. Cargar el video en la carpeta con nombre “video” dentro de proyecto para ser procesada por la aplicación.

2. Ejecutar el proyecto por medio de un ejecutable o si se trabaja sobre un sistema operativo Windows 10; instalar Anaconda, crear variable de entorno y proceder a instalar las librerías: TensorFlow y Keras.
3. Inmediatamente después de iniciar la aplicación, el proyecto carga el video y empieza a procesarlo para reconocer los rostros por medio de un clasificador de cascada “haarcascade_frontalface_default.xml”
4. Empieza a leer 18 frames cada 0.02 segundos, convertir la imagen a escala de gris, recortarlas 64 x 64 pixeles y guardarlas sobre la carpeta “frames” dentro del proyecto
5. Llamar el modelo para realizar la predicción enviando las imágenes con la estructura [conjunto de imágenes, profundidad (18 frames), ancho, alto,1 (fijo para trabajar con TensorFlow)] como el siguiente ejemplo [64, 18, 64, 64, 1].
6. Con la predicción del modelo para la microexpresión facial, se dibuja un cuadro azul (ver Figura 81 Aplicación creada con Flask y Python para el reconocimiento de microexpresiones faciales.) con el texto de la micro expresión en la parte superior.
7. Se repite el algoritmo para los siguientes 18 frames.

5.7.3. Diagrama de Flujo

Con las funcionalidades descritas anteriormente, en el siguiente diagrama (Figura 82) de flujo se puede ver el proceso completo del algoritmo de reconocimiento de microexpresiones faciales.

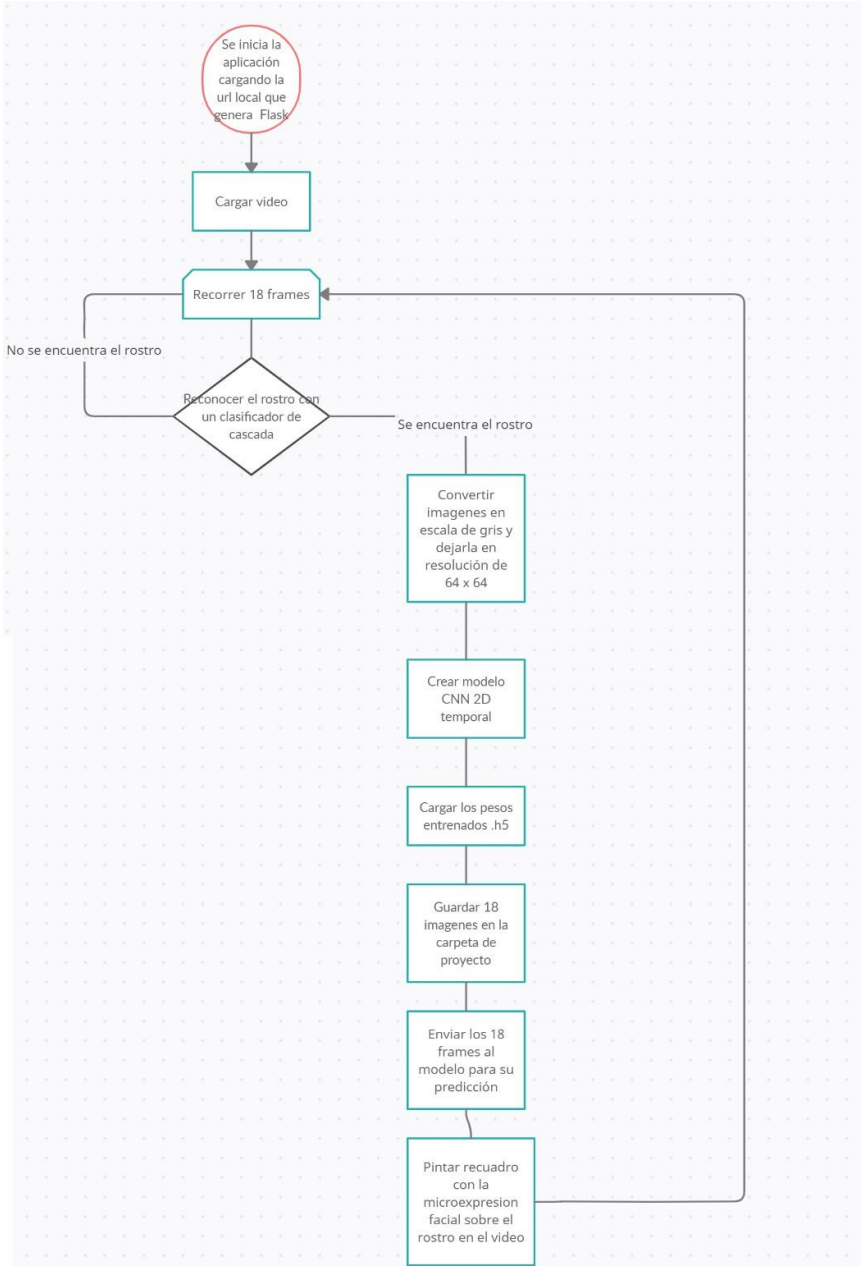


Figura 82 Diagrama de flujo de las funcionalidades de la aplicación desarrollada para el reconocimiento de microexpresión faciales. Fuente: Elaboración propia

Capítulo VI

6. Conclusiones y Trabajos futuros

6.1. Conclusiones

- Para los tres modelos propuestos obtuvieron mejores resultados de predicción para la base de datos CASME II que para la base de datos SMIC. CASME II esta codificada con FACS de 26 participantes, mientras que la base de datos SMIC no lo está. Además, la base de datos SMIC al no tener codificación FACS, la categorización de las etiquetas de las emociones se dejó a la propia auto información del participante. Esto puedo haber generado una mal etiquetada de la emoción sobre la base de datos SMIC.

- Para el modelo MicroExpSTCNN propuesto por el artículo (S. P. Teja Reddy, et al., 2019) se pudieron obtener mejores resultados para las dos bases de datos. El acurracy del modelo de referencia para la base de datos CASME II fue de 87.80 % (ver Tabla 22), mientras que el acurray para la misma base de datos fue de 90% (ver Tabla 17).

- Al agregar data augmentation al conjunto de imágenes sobre una Red Neuronal Convolucional 3D se logró mejorar las métricas para la base de datos CASME II, en cambio para la base de datos SMIC no se aprecia mejoría.

- Para el tercer modelo se creó una Red Neuronal Convolucional temporal con una capa de LSTM permitiendo tener en cuenta la característica temporal de las imágenes, ya que se trabajaron con 18 frames de una secuencia de imágenes, arrojando métricas de desempeño del modelo más cercanas a 1. Esto mejoró notablemente la predicción de las microexpresiones faciales para ambas bases de datos CASME y SMIC.

- Google Colab es una herramienta poderosa para el desarrollo de modelos profundos, TensorBoard es una herramienta fuerte en monitoreo y visualización del comportamiento de modelo, y Anaconda es una herramienta avanzada para el desarrollo de aplicaciones con modelos de aprendizaje automático.

- La certificación "*facial expressionrecognition with keras*" que se realizó con Coursera fue de mucha ayuda para el entendimiento de los modelos profundos y para el desarrollo de la herramienta de reconocimiento de microexpresiones faciales con el lenguaje de programación Python y el Framework Flask.

- Se desarrolló una aplicación en Python con Flask para el reconocimiento de microexpresiones faciales, la cual reconoció la microexpresión facial correctamente cargando el modelo previamente entrenado y los pesos.

- La mentira es un campo muy grande aun por explorar, no podemos sesgarnos solo por las expresiones faciales para identificarla, se debe tambien tener en cuenta otros parámetros como el comportamiento, el ritmo cardiaco, las respuestas al hablar y más aspectos psicológicos que ameritan un estudio más detallado.

- Este trabajo de grado logro un acercamiento importante al reconocimiento de microexpresiones faciales, llegando a unas métricas más altas que los artículos tomados como referencia. Para el modelo CNN Temporal con LSTM resultado de este trabajo de grado y la base de datos CASME se obtuvieron las siguientes métricas: Precisión 0.984, Sensibilidad 0.962, Accuracy 0.980, F1 Score 0.972 y AUC 0.999. Ahora el mismo modelo para la base de datos SMIC Precisión 0. 906, Sensibilidad 0. 922, Accuracy 0. 922, F1 Score 0. 912y AUC 0. 974. Al agregar la capa LSTM al modelo permito tener en cuenta el dinamismo de las imágenes y adaptarse de forma única a la entrada.

- La mentira se relaciona con la detección de la emoción falsa. La mejor máscara es una emoción falsa, que desconcierta y actúa como camuflaje. La cara es la sede primordial del despliegue de las emociones. Ponerse una máscara es la mejor manera de ocultar una emoción, si uno se cubre el rostro o parte de él con la mano o lo aparta de la persona que habla dándose media vuelta, habitualmente eso dejará traslucir que podría estar mintiendo.

6.2. Trabajos futuros

- Explorar modelos Bayesianos que permiten tener una predicción del modelo más robusto, manejar un modelo probabilístico que pueda mejorar el clasificador para el reconocimiento de microexpresiones faciales.
- Seguir explorando el reconocimiento de microexpresiones faciales y el reconocimiento de la mentira; para esto se puede trabajar con imágenes infrarrojas que aporten información de la presión sanguínea importante para el reconocimiento de la mentira.
- Trabajar sobre la aplicación de reconocimiento de microexpresiones faciales creada en este trabajo de grado, para que compile en GPUS, sea más optima y pueda probar varios modelos al tiempo; retornando la mejor predicción.
- Seguir explorando los modelos temporales ya que se obtuvieron mejores resultados con este modelo para las dos bases de datos, seguir explorando mejoras para correr sobre la base de datos de SMIC.
- Mejorar las etiquetas de las bases de datos de microexpresiones faciales, ya que la base de datos SMIC no se generó el etiquetado con FACS y eso agrega un margen de error en la predicción del modelo, como se puedo apreciar en los tres modelos montados en este trabajo de grado. El primer modelo MicroExpSTCNN el cual se adaptó y mejoro para este trabajo de grado no logro tener un accuracy mayor a 90 % haciendo difícil su entrenamiento.

6.3. Difusión publicaciones

A partir del desarrollo del trabajo de grado propuesto, se tiene planeado las siguientes publicaciones:

- Artículo: Deep models for facial micro-expressions recognition (a ser sometido). Ver Anexo b.
- Registro de software de la herramienta para el reconocimiento de microexpresiones faciales utilizando herramientas Open Source.

Capítulo VII

7. Referencias

OpenCV. (2020, 4 noviembre). *About*. Recuperado de https://opencv.org/about/

Choudhary, A. (2021, abril 5). *Optimizers in Deep Learning - Ayushi choudhary*. Medium. Recuperado de https://2809ayushic.medium.com/optimizers-in-deep-learning-31db684c73cf

Scikit-Learn, herramienta básica para el Data Science en Python. (2019, 8 abril). Máster en Data Science. https://www.master-data-scientist.com/scikit-learn-data-science/

Verma, S. (2019, 20 septiembre). *Understanding 1D and 3D Convolution Neural Network | Keras*. Medium. https://towardsdatascience.com/understanding-1d-and-3d-convolution-neural-network-keras-9d8f76e29610

M. Shreve, S. Godavarthy, D. Goldgof and S. Sarkar, "Macro- and micro-expression spotting in long videos using spatio-temporal strain", *Proc. IEEE Int. Conf. Automat. Face Gesture Recog. Workshops*, pp. 51-56, 2011.

Merghani, W. (2018, 7 mayo). *A Review on Facial Micro-Expressions Analysis: Datasets, Features. . .* ArXiv.Org. Recuperado de https://arxiv.org/abs/1805.02397

M. Owayjan, A. Kashour, N. Al Haddad, M. Fadel & G. Al Souki, "The design and development of a Lie Detection System using facial micro-expressions," *2012 2nd International Conference on Advances in Computational Tools for Engineering Applications (ACTEA)*, 2012, pp. 33-38, doi: 10.1109/ICTEA.2012.6462897.

S. Li & W. Deng, "Deep Facial Expression Recognition: A Survey," in *IEEE Transactions on Affective Computing,* doi: 10.1109/TAFFC.2020.2981446.

Fita, J. (2016, 2 marzo). *Cómo detectar la mentira a través del lenguaje corporal*. La Vanguardia. Recuperado de https://www.lavanguardia.com/vida/20160301/40123303858/detectar-mentira-lenguaje-corporal.html

Agencyworld. (2019, 9 diciembre). *Técnicas para detectar mentiras*. Detectives Privados en Madrid. Recuperado de https://www.agencyworld.org/blog/tecnicas-para-detectar-mentiras

Andrea, P., Cardona, N., & Quintero, M. V. (2016). *Detección de mentiras mediante reconocimiento de patrones faciales utilizando procesamiento digital de imágenes Event Representation in Pre-conceptual Schemas by using Semantic Roles and Mathematical Equations View project Detección de mentiras mediante el reconocimiento de patrones faciales y del discurso usando linguística computacional y procesamiento digital de imágenes View project*. Recuperado de https://www.researchgate.net/publication/303895612

Van Edwards, V. (2014). The Definitive Guide to Reading Microexpressions (Facial Expressions). *Scienceofpeople*. Published. Recuperado de https://www.scienceofpeople.com/microexpressions/

Babich, N. (2016, 27 enero). *How to Detect Lies: Micro Expressions - Nick Babich*. Medium. Recuperado de https://medium.com/@101/how-to-detect-lies-microexpressions-b17ae1b1181e

Ortega González, M. (2010). *COMPORTAMIENTO MENTIROSO: UN ANÁLISIS CONCEPTUAL DESDE UNA PERSPECTIVA INTERCONDUCTUAL.*

ichi.pro. (2021, 16 febrero). *Cómo detectar mentiras con una máquina y microexpresiones*. Recuperado de https://ichi.pro/es/como-detectar-mentiras-con-una-maquina-y-microexpresiones-276674219479585

Sharma, G. (2020, 14 abril). *CK+48 5 emotions*. Kaggle. Recuperado de https://www.kaggle.com/gauravsharma99/ck48-5-emotions#__sid=js0

X. Li, T. Pfister, X. Huang, G. Zhao & M. Pietikäinen, "A Spontaneous Micro-expression Database: Inducement, collection and baseline," *2013 10th IEEE International Conference and Workshops on Automatic Face and Gesture Recognition (FG)*, 2013, pp. 1-6, doi: 10.1109/FG.2013.6553717.

Yan WJ, Li X, Wang SJ, Zhao G, Liu YJ, Chen YH, Fu X. CASME II: an improved spontaneous micro-expression database and the baseline evaluation. PLoS One. 2014 Jan 27;9(1):e86041. doi: 10.1371/journal.pone.0086041. PMID: 24475068; PMCID: PMC3903513.

A. K. Davison, C. Lansley, N. Costen, K. Tan and M. H. Yap, "SAMM: A Spontaneous Micro-Facial Movement Dataset," in IEEE Transactions on Affective Computing, vol. 9, no. 1, pp. 116-129, 1 Jan.-March 2018, doi: 10.1109/TAFFC.2016.2573832.

P. Ekman & W. V. Friesen, *Facial Action Coding System: A Technique for the Measurement of Facial Movement*, Palo Alto, CA, USA:Consulting Psychologists Press, 1978

Wen-Jing Yan, Q. Wu, Yong-Jin Liu, Su-Jing Wang & X. Fu, "CASME database: A dataset of spontaneous micro-expressions collected from neutralized faces," *2013 10th IEEE International Conference and Workshops on Automatic Face and Gesture Recognition (FG),* 2013, pp. 1-7, doi: 10.1109/FG.2013.6553799.

Wen-Jing Yan, Q. Wu, J. Liang, Y.-H. Chen & X. Fu, "How fast are the leaked facial expressions: The duration of micro-expressions", *J. Nonverbal Behavior*, vol. 37, no. 4, pp. 217-230, 2013.

Chavali, G. K. (2014). *Micro-Expression Extraction For Lie Detection Using Eulerian Video (Motion and Color) Magnication. DIVA*. Recuperado de https://www.diva-portal.org/smash/record.jsf?pid=diva2%3A830774&dswid=-3836

P. Ekman (2021, 16 febrero). *Universal Emotions* | *What are Emotions?* Paul Ekman Group. Recuperado de https://www.paulekman.com/universal-emotions/

Montejano, S. (2020, 18 febrero). *¿Porqué la gente miente?* PsicoGlobal. Recuperado de https://www.psicoglobal.com/blog/porque-mienten-las-personas

P. Ekman. Why lies fail and what behaviors betray a lie. In JohnC. Yuille, editor, *Credibility Assessment*, volume 47 of Nato Science, pages 71–81. Springer Netherlands, 1989.

P. Ekman. Deception, lying, and demeanor. *States of Mind: American and Post-Soviet Perspectives on Contemporary Issues in Psychology*, page 93, 1997.

P. Ekman. Should we call it expression or communication? Innovation, 10(4):333, 1997.

P. Ekman. Darwin's contributions to our understanding of emotional expressions. *Philosophical Transactions: Biological Sciences*, 364(1535):3449–3451, 2009.

P. Ekman, "Darwin deception and facial expression", *Ann. New York Academy Sci.*, vol. 1000, no. 1, pp. 205-221, 2003.

P. Ekman. Facial expression and emotion. *The American Psychologist*, 48(4):384– 392, 1993.

P. Ekman. An argument for basic emotions. *Cognition & Emotion*, 6(3-4):169– 200, 1992.

P. Ekman. Are there basic emotions? Psychological review, 99(3):550–553, 1992.

D. Cordaro & P. Ekman. What is meant by calling emotions basic. *Emotion Review*, 3(4):364–370, 2011.

D. Matsumoto & P. Ekman. The relationship among expressions, la- bels, and descriptions of contempt. *Journal of personality and social psychology*, 87(4):529–540, 2004.

Hao-Yu Wu, M. Rubinstein, E. Shih, J. Guttag, F. Durand, & W. Freeman. *Eulerian video magnification for revealing subtle changes in the world*. ACMTrans. Graph., 31(4):65:1–65:8, July 2012.

Kamath, D. (2017, 18 diciembre). *GitHub - dhanushkamath/PyEmotionRecognition: A Python project to recognize the emotion expressed on an individual's face, given an image.* GitHub. Recuperado de https://github.com/dhanushkamath/PyEmotionRecognition

Zartha Suarez, N. (2021, March 23). *natzasu13/landmark_detector*. GitHub. Recuperado de https://github.com/natzasu13/landmark_detector

Kekre, S. (2020). *Tutorial: Facial Expression Recognition with Keras*. Coursera. Recuperado de https://www.coursera.org/projects/facial-expression-recognition-keras

Consortium (2020). Recuperado el 23 de junio de 2020 de http://www.consortium.ri.cmu.edu/ckagree/

Paulvangent (2021). Recuperado el 23 de junio de 2021 de http://www.paulvangent.com/2016/08/05/emotion-recognition-using-facial-landmarks/

Li, Y., Huang, X., & Zhao, G. (2021). Micro-expression action unit detection with spatial and channel attention. *Neurocomputing*, 436, 221–231. Recuperado de https://doi.org/10.1016/j.neucom.2021.01.032

X. Li *et al*., "Towards Reading Hidden Emotions: A Comparative Study of Spontaneous Micro-Expression Spotting and Recognition Methods," in *IEEE Transactions on Affective Computing*, vol. 9, no. 4, pp. 563-577, 1 Oct.-Dec. 2018, doi: 10.1109/TAFFC.2017.2667642.

F. Qu, S.-J. Wang, W.-J. Yan, H. Li, S. Wu, & X. Fu, "Cas (me)ˆ 2: A database for spontaneous macro-expression and micro- expression spotting and recognition," *IEEE Transactions on Affective Computing*, 2017.

W. Li, F. Abtahi, Z. Zhu, Action unit detection with region adaptation, multi- labeling learning and optimal temporal fusing, in, in: Proceedings of *the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 1841– 1850.

D. Acharya, Z. Huang, D. Pani Paudel, L. Van Gool, Covariance pooling for facial expression recognition, in, in: Proceedings of *the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2018, pp. 367–374.

D. Y. Choi & B. C. Song, "Facial Micro-Expression Recognition Using Two-Dimensional Landmark Feature Maps," in *IEEE Access*, vol. 8, pp. 121549-121563, 2020, doi: 10.1109/ACCESS.2020.3006958.

J. See, M. H. Yap, J. Li, X. Hong & S.-J. Wang, "MEGC 2019—The second facial micro-expressions grand challenge", *Proc. 14th IEEE Int. Conf. Autom. Face Gesture Recognit. (FG)*, pp. 1-5, May 2019.

P. Ekman & W. V. Friesen, *Manual for the Facial Action Coding System, Consulting Psychologists Press*, 1977.

P. Ekman & W. V. Friesen, "*Nonverbal leakage and clues to deception*", *Psychiatry*, vol. 32, no. 1, pp. 88-106, 1969

S. Porter, "Reading between the lies identifying concealed and falsified emotions in universal facial expressions", *Psychological Sci.*, vol. 19, no. 5, pp. 508-514, 2008.

Nebauer, C. (1998) "*Evaluation of convolutional neural networks for visual recognition*." IEEE Transactions on Neural Networks 9 (4): 685-696.

Zhang, Z. (2016) "*Derivation of Backpropagation in Convolutional Neural Network (CNN)".*

Fieres, J., Schemmel, J., & Meier, K. (2006) "*Training convolutional networks of threshold neurons suited for low-power hardware implementation*." In Neural Networks, 2006. IJCNN'06. International Joint Conference on. IEEE. (pp. 21-28).

Timoshenko, D., & Grishkin, V. (2013) "*Composite face detection method for automatic moderation of user avatars*." Computer Science and Information Technologies (CSIT'13)

Arel, I., Rose, D. C., & Karnowski, T. P. (2010) "*Deep machine learning-a new frontier in artificial intelligence research [research frontier*]." IEEE computational intelligence magazine 5 (4): 13-18.

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998) "*Gradient-based learning applied to document recognition*." Proceedings of the IEEE 86 (11): 2278-2324.

Tivive, F. H. C., & Bouzerdoum, A. (2005) "*Efficient training algorithms for a class of shunting inhibitory convolutional neural networks*." IEEE Transactions on Neural Networks 16 (3): 541-556

Wang, J., Lin, J., & Wang, Z. (2016) "*Efficient convolution architectures for convolutional neural network*." In Wireless Communications and Signal Processing (WCSP), 2016 8th International Conference on (pp. 1-5).

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). *"Imagenet classification with deep convolutional neural networks*." In Advances in neural information processing systems (pp. 1097-1105)

Zeiler, M. D., & Fergus, R. (2013) "*Stochastic pooling for regularization of deep convolutional neural networks*." arXiv preprint arXiv:1301.3557.

Donahue, J., Jia, Y., Vinyals, O., Hoffman, J., Zhang, N., Tzeng, E., & Darrell, T. (2014) "*Decaf: A deep convolutional activation feature for generic visual recognition.*" In International conference on machine learning (pp. 647-655)

Sainath, T. N., Kingsbury, B., Mohamed, A. R., Dahl, G. E., Saon, G., Soltau, H., Ramabhadran, B. (2013) "*Improvements to deep convolutional neural networks for LVCSR*." In Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on (pp. 315-320).

Szegedy, C., Toshev, A., & Erhan, D. (2013) "*Deep neural networks for object detection.*" In Advances in neural information processing systems (pp. 2553-2561).

Luo, X., Shen, R., Hu, J., Deng, J., Hu, L., & Guan, Q. (2017) "*A Deep Convolution Neural Network Model for Vehicle Recognition and Face Recognition*." Procedia Computer Science 107: 715-720.

Pratt, H., Coenen, F., Broadbent, D. M., Harding, S. P., & Zheng, Y. (2016) "*Convolutional neural networks for diabetic retinopathy*." Procedia Computer Science 90: 200-205

Uçar, A. (2017, July) "*Deep Convolutional Neural Networks for facial expression recognition*." In Innovations in Intelligent Systems and Applications (INISTA), 2017 IEEE International Conference on (pp. 371-375)

Indolia, S., Goswami, A. K., Mishra, S. P., & Asopa, P. (2018). *Conceptual Understanding of Convolutional Neural Network- A Deep Learning Approach.*

Procedia Computer Science, 132, 679–688. Recuperado de https://doi.org/10.1016/j.procs.2018.05.069

B. Fasel, "Robust face analysis using convolutional neural networks", *Pattern Recognition 2002. Proceedings. 16th International Conference on*, vol. 2, pp. 40-43, 2002.

F. Beat, "Head-pose invariant facial expression recognition using convolutional neural networks", *Fourth IEEE International Conference on Multimodal Interfaces 2002*, pp. 529-534, 2002.

M. Matsugu, K. Mori, Y. Mitari & Y. Kaneda, "Subject independent facial expression recognition with robust face detection using a convolutional neural network", *Neural Networks*, vol. 16, no. 5–6, pp. 555-559, 2003.

S. Singh & F. Nasoz, "*Facial Expression Recognition with Convolutional Neural Networks,*" 2020 10th Annual Computing and Communication Workshop and Conference (CCWC), 2020, pp. 0324-0328, doi: 10.1109/CCWC47524.2020.9031283.

TensorFlow. (2021). Recuperado el 23 de junio de 2021 de https://www.tensorflow.org/

Team, K. (2021). Keras documentation: About Keras. Recuperado el 23 de junio de 2021 de https://keras.io/about/

cs231n. (s. f.). *CS231n Convolutional Neural Networks for Visual Recognition.* Recuperado 23 de junio de 2021, de https://cs231n.github.io/convolutional-networks/

Matlab. (s. f.). *Redes neuronales convolucionales*. MATLAB & Simulink. Recuperado 28 de septiembre de 2021, de https://es.mathworks.com/discovery/convolutional-neural-network-matlab.html

Kingma, D. P., & Ba, J. L. (2015). *Adam: a Method for Stochastic Optimization.* International Conference on Learning Representations, 1–13

Ruder, S. (2016, 19 junio). *An overview of gradient descent optimization algorithms.* Sebastian Ruder. Recuperado el 4 de julio 2021 de https://ruder.io/optimizing-gradient-descent/index.html#adam

DeepAI. (2020, 25 junio). *Max Pooling.* Recuperado de https://deepai.org/machine-learning-glossary-and-terms/max-pooling

Baeldung. (2020, 13 agosto). *How ReLU and Dropout Layers Work in CNNs*. Baeldung on Computer Science. Recuperado el 4 de julio de 2021 de https://www.baeldung.com/cs/ml-relu-dropout-layers

Wood, T. (2020, 25 junio). *Softmax Function*. DeepAI. Recuperado de https://deepai.org/machine-learning-glossary-and-terms/softmax-layer

Dahal, P. (2021). *Classification and Loss Evaluation - Softmax and Cross Entropy Loss*. Recuperado el 5 de julio de 2021 de https://deepnotes.io/softmax-crossentropy

Wikipedia contributors. (2021, 13 octubre). *Receiver operating characteristic*. Wikipedia. Recuperado el 5 de julio de 2021 de https://en.wikipedia.org/wiki/Receiver_operating_characteristic

Nicholson, C. (s. f.). *Evaluation Metrics for Machine Learning - Accuracy, Precision, Recall, and F1 Defined*. Pathmind. Recuperado el 6 de julio de 2021 de https://wiki.pathmind.com/accuracy-precision-recall-f1

Arce, J. (2020). *La matriz de confusión y sus métricas – Inteligencia Artificial –*. Recuperado el 6 de julio de 2021 de https://www.juanbarrios.com/la-matriz-de-confusion-y-sus-metricas/

Prof. Xiaolan Fu's Group. (2008). *Welcome to Professor Fu's Lab. CASME II Database*. Recuperado de http://fu.psych.ac.cn/CASME/casme2-en.php

University of oulu. (2018). SMIC - Spontaneous Micro-expression Database*. SMIC - Spontaneous Micro-Expression Database.* Recuperado de https://www.oulu.fi/cmvs/node/41319

S. P. Teja Reddy, S. Teja Karri, S. R. Dubey & S. Mukherjee, "*Spontaneous Facial Micro-Expression Recognition using 3D Spatiotemporal Convolutional Neural Networks*," 2019 International Joint Conference on Neural Networks (IJCNN), 2019, pp. 1-8, doi: 10.1109/IJCNN.2019.8852419.

Prasanna Teja Reddy, B. S. (2019, 2 octubre). *micro-expression-recognition*. GitHub. Recuperado de https://github.com/bogireddytejareddy/micro-expression-recognition

Mañas, A. M. (s. f.). *Capítulo 8 Métodos basados en Deep Learning | Notas sobre pronóstico del flujo de tráfico en la ciudad de Madrid*. Bookdown. Recuperado 26 de septiembre de 2021, de https://bookdown.org/amanas/traficomadrid/m%C3%A9todos-basados-en-deep-learning.html#lstm-univariado

K. He, X. Zhang, S. Ren & J. Sun, "Deep Residual Learning for Image Recognition," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770-778, doi: 10.1109/CVPR.2016.90.

A. Budhiman, S. Suyanto & A. Arifianto, "Melanoma Cancer Classification Using ResNet with Data Augmentation," *2019 International Seminar on Research of Information Technology and Intelligent Systems (ISRITI)*, 2019, pp. 17-20, doi: 10.1109/ISRITI48646.2019.9034624.

K. O'Shea & R. Nash, *"An Introduction to Convolutional Neural Networks*", November 2015.

H. Wu, X. Ma & Y. Li, "Spatiotemporal Multimodal Learning with 3D CNNs for Video Action Recognition," in *IEEE Transactions on Circuits and Systems for Video Technology*, doi: 10.1109/TCSVT.2021.3077512.

S. Ji, W. Xu, M. Yang & K. Yu, "3D Convolutional Neural Networks for Human Action Recognition," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 1, pp. 221-231, Jan. 2013, doi: 10.1109/TPAMI.2012.59.

Google. (s. f.). *Google Colab*. research.google. Recuperado 27 de septiembre de 2021, de https://research.google.com/colaboratory/faq.html?authuser=0&hl=es

Elhawary, M. M. (2020, 26 febrero). *An easy way to create cloud-synced Data Science project using Google Ecosystem*. Medium. Recuperado de https://medium.com/analytics-vidhya/an-easy-way-to-create-your-portable-data-science-project-environment-46a2c2ec889a

Scikit-Learn, *herramienta básica para el Data Science en Python*. (2019, 8 abril). Máster en Data Science. Recuperado de https://www.master-data-scientist.com/scikit-learn-data-science/

Muñoz, J. D. (2020, 22 diciembre)*. Qué es Flask*. OpenWebinars.net. Recuperado de https://openwebinars.net/blog/que-es-flask/

Google. (s. f.-a). *Classification: ROC Curve and AUC | Machine Learning Crash Course*. Google Developers. Recuperado 28 de septiembre de 2021, de https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc

Zunair, H. (2020, 23 septiembre). *3D_image_classification_CT*. Google Colab. Recuperado de https://colab.research.google.com/drive/14vczr_vjxT9608bgHscaGZhV2e5-xLNQ?usp=sharing

R. (2017, 14 noviembre). *Convolutional Neural Network (CNN) - Raycad*. Medium. Recuperado de https://medium.com/@raycad.seedotech/convolutional-neural-network-cnn-8d1908c010ab

Carchenilla. (2016, 27 junio). *Una Introducción Práctica al Deep Learning con Caffe y Python.* Los mundos de Carchenilla. Recuperado de https://carchenilla.wordpress.com/2016/06/27/80/

Cagatay C. (2012). *Performance Evaluation Metrics for Software Fault Prediction Studies*. Acta Polytechnica Hungarica. Recuperado de http://acta.uni-obuda.hu//Catal_36.pdf

Anaconda. (s. f.). *Anaconda | The World's Most Popular Data Science Platform*. Recuperado 29 de septiembre de 2021, de https://www.anaconda.com/

Cagatay C. *Performance Evaluation Metrics for Software Fault Prediction Studies*. Acta Polytechnica Hungarica, 9
(4):193–206, 2012. ISSN 17858860.

Hochreiter, Sepp, and Jürgen Schmidhuber. 1997. "Long Short-Term Memory." *Neural Computation* 9 (8). MIT Press: 1735–80.

Anexo a. Modelos Implementados en Google Colab

A continuación, se presenta el codigo que se creó en Google Colab para los tres modelos, cada uno para las dos bases de datos de microexpresiones faciales CASME II y SMIC. Tambien se adjunta la url de GitHub donde se encuentra el repositorio de los modelos.

Modelo 1. Modelo convolucional MicroExpSTCNN 3D para el reconocimiento de microexpresiones faciales

Base de datos CASME II

El siguiente codigo se creó en Google Colab. Tambien se puede ver en la url https://colab.research.google.com/drive/1Jf6gW9g4IAPjtzfQLKNHmBPy4clJGwrZ y en la url de Github https://github.com/natzasu13/microexpresion-recognition/blob/casmeii-microexpresion/Microexpression_recognition_MicroExpSTCNN_CASMEII_Maestria_v1_2.ipynb

```
import os
import cv2
import numpy
import imageio
from sklearn.metrics import confusion_matrix, accuracy_score, classification_report, precision_recall_curve, roc_curve, auc
from keras.models import Sequential
```

```
from keras.layers.core import Dense, Dropout, Activation, Flatten
from keras.layers.convolutional import Convolution3D, MaxPooling3D
from tensorflow.keras.optimizers import SGD
from keras.callbacks import ModelCheckpoint
from keras.utils import np_utils, generic_utils
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from sklearn import preprocessing
from keras import backend as K
import sys
from matplotlib import pyplot as plt
from tensorflow import keras

K.set_image_data_format('channels_first')
```

Import files from google drive

```
from google.colab import drive
drive.mount('/content/drive')
```

Load images

```
image_rows, image_columns, image_depth = 64, 64, 18
training_list = []

CODE_PATH = 'drive/MyDrive/MAESTRIA ING DE SISTEMAS/1. MICROEXPRESIONES FACIALES TRA
BAJO DE GRADO/3. Codigo/microexpresion-recognition-colab/CASMEII'
CASMEII_path = CODE_PATH + '/datasets/CASMEII'

HSpath =CASMEII_path +'/CASME2_preprocessed_small_Li Xiaobai/Cropped/'
```

```
    rootDirectory = os.listdir(HSpath)
    for s in rootDirectory:
        sPath = HSpath + s + "/"
        microPath = os.listdir(sPath)
        for video in microPath:
            videopath = sPath + video

            frames = []
            framelisting = os.listdir(videopath)
            framesCount = image_depth if len(framelisting) > image_depth else 0
            framerange = [x for x in range(18)]

            if(framesCount != 0):
                for frame in framerange:
                      imagepath = videopath + "/" + framelisting[frame]

                      image = cv2.imread(imagepath)
                      imageresize = cv2.resize(image, (image_rows, image_columns), inter
polation = cv2.INTER_AREA)
                      grayimage = cv2.cvtColor(imageresize, cv2.COLOR_BGR2GRAY)
                      frames.append(grayimage)

                frames = numpy.asarray(frames)
                videoarray = numpy.rollaxis(numpy.rollaxis(frames, 2, 0), 2, 0)

                training_list.append(videoarray)
                print(len(training_list))
```

Training

```
#257 imagenes que se conviernes en escala de gris
#64x64 tamaño
#18 frames
training_list = numpy.asarray(training_list)
trainingsamples = len(training_list)

print("trainingsamples")
print(trainingsamples)

traininglabels = numpy.zeros((trainingsamples, ), dtype = int)

traininglabels[0:76] = 0
traininglabels[76:170] = 1
traininglabels[170:206] = 2

traininglabels = np_utils.to_categorical(traininglabels, 3)

training_data = [training_list, traininglabels]
(trainingframes, traininglabels) = (training_data[0], training_data[1])

training_set = numpy.zeros((trainingsamples, 1, image_rows, image_columns, image_dep
th))

print("range(trainingsamples) is the value of h")
print(range(trainingsamples))

for h in range(trainingsamples):
    training_set[h][0][:][:][:] = trainingframes[h, :, :, :]
```

```
training_set = training_set.astype('float32')



new_path = '/content/drive/MyDrive/TesisMScNataliaZartha/'
#Save in the new path to data augmentation
numpy.save(new_path + '/microexpstcnn_images_tr_casmeII.npy', training_set)
numpy.save(new_path + '/microexpstcnn_labels_tr_casmeII.npy', traininglabels)


training_set -= numpy.mean(training_set) #SE COMENTA PARA GUARDAR LOS NPARRAY
training_set /= numpy.max(training_set) #SE COMENTA PARA GUARDAR LOS NPARRAY

# Save training images and labels in a numpy array
numpy.save(CODE_PATH+'/numpy_training_datasets/microexpstcnn_images.npy', training_s
et)
numpy.save(CODE_PATH+'/numpy_training_datasets/microexpstcnn_labels.npy', trainingla
bels)

# Load training images and labels that are stored in numpy array
"""
training_set = numpy.load(CODE_PATH+'/numpy_training_datasets/microexpstcnn_images.n
py')
traininglabels =numpy.load(CODE_PATH+'/numpy_training_datasets/microexpstcnn_labels.
npy')
"""
```

Save train and validation images and labels in a numpy array

```
"""
new_path = '/content/drive/MyDrive/TesisMScNataliaZartha/'
```

```
   #Save in the new path to data augmentation
   numpy.save(new_path + '/microexpstcnn_images_tr_casmeII.npy', training_set)
   numpy.save(new_path + '/microexpstcnn_labels_tr_casmeII.npy', traininglabels)
   """
```

Model

```
   # MicroExpSTCNN Model
   model = Sequential()
   model.add(Convolution3D(32, (3, 3, 15), input_shape=(1, image_rows, image_columns, i
mage_depth), activation='relu'))
   model.add(MaxPooling3D(pool_size=(3, 3, 3)))
   model.add(Dropout(0.5))
   model.add(Flatten())
   model.add(Dense(128, kernel_initializer='normal', activation='relu'))
   model.add(Dropout(0.5))
   model.add(Dense(3, kernel_initializer='normal'))
   model.add(Activation('softmax'))
   model.compile(loss = 'categorical_crossentropy', optimizer = 'SGD', metrics = ['accu
racy'])

   model.summary()

   filepath=CODE_PATH+"/weights_microexpstcnn/weights-improvement-{epoch:02d}-
{val_accuracy:.2f}.hdf5"
   checkpoint = ModelCheckpoint(filepath, monitor='val_accuracy', verbose=1, save_best_
only=True, mode='max')
   callbacks_list = [checkpoint]

   # Load pre-trained weights
```

```
   #model.load_weights(CODE_PATH+'/weights_microexpstcnn/weights-improvement-40-
0.69.hdf5')
   #model.load_weights(CODE_PATH+'/weights_microexpstcnn/model_weights.h5')

   # Spliting the dataset into training and validation setsl
   train_images, validation_images, train_labels, validation_labels = train_test_split
(training_set, traininglabels, test_size=0.35, random_state=4)

   # Save validation set in a numpy array
   numpy.save(CODE_PATH+'/numpy_validation_dataset/microexpstcnn_val_images.npy', valid
ation_images)
   numpy.save(CODE_PATH+'/numpy_validation_dataset/microexpstcnn_val_labels.npy', valid
ation_labels)

   # Load validation set from numpy array
   validation_images = numpy.load(CODE_PATH+'/numpy_validation_dataset/microexpstcnn_va
l_images.npy')
   validation_labels = numpy.load(CODE_PATH+'/numpy_validation_dataset/microexpstcnn_va
l_labels.npy')
   # Training the model
   hist = model.fit(train_images, train_labels, validation_data = (validation_images, v
alidation_labels), callbacks=callbacks_list, batch_size = 16, epochs = 150, shuffle=Tr
ue)

   plt.plot(hist.history['accuracy'], label='accuracy')
   plt.plot(hist.history['val_accuracy'], label = 'val_accuracy')
   plt.xlabel('Epoch')
   plt.ylabel('Accuracy')
   plt.ylim([0.5, 1])
   plt.legend(loc='lower right')
```

```
   test_loss, test_acc = model.evaluate(validation_images,  validation_labels, verbose=
2)

   import matplotlib.pyplot as plt
   plt.plot(hist.history['accuracy'])
   plt.plot(hist.history['val_accuracy'])
   plt.title('model accuracy')
   plt.ylabel('accuracy')
   plt.xlabel('epoch')
   plt.legend(['train', 'test'], loc='upper left')
   plt.show()

plt.plot(hist.history['loss'],)

   plt.plot(hist.history['val_loss'])
   plt.title('model loss')
   plt.ylabel('loss')
   plt.xlabel('epoch')
   plt.legend(['train', 'test'], loc='upper right')
   plt.show()
```

Model architecture graph

```
   !pip install keras-resnet
   from keras_resnet.models import ResNet50, ResNet101, ResNet152
   import numpy as np
   from tensorflow import keras

   #https://keras.io/api/utils/model_plotting_utils/
   keras.utils.plot_model(
```

```
        model,
        to_file="model.png",
        show_shapes=False,
        show_dtype=False,
        show_layer_names=True,
        rankdir="TB",
        expand_nested=False,
        dpi=96,
        layer_range=None,
    )
```

Predictions

```
    predictions = model.predict(validation_images)
    predictions_labels = numpy.argmax(predictions, axis=1)

    validation_labels_y = numpy.argmax(validation_labels, axis=1)

cfm = confusion_matrix(validation_labels_y, predictions_labels)

    print('Accuracy score :', accuracy_score(validation_labels_y,predictions_labels) )
    print('Classification report :', classification_report(validation_labels_y,predictio
ns_labels) )

    from sklearn.metrics import roc_auc_score, f1_score, precision_score, recall_score,
hamming_loss, jaccard_similarity_score, log_loss, zero_one_loss, matthews_corrcoef

    def evaluation_analysis(true_label,predicted):
        '''
```

```
        return all metrics results
        '''
        print("accuracy (EXACTITUD)",accuracy_score(true_label, predicted))
        print("f1 score macro",f1_score(true_label, predicted, average='macro'))
        print("f1 score micro",f1_score(true_label, predicted, average='micro'))
        print("precision score (PRESICION)",precision_score(true_label, predicted, avera
ge='macro'))
        print("recall score (SENSIBILIDAD)",recall_score(true_label, predicted, average=
'macro'))
        print("hamming_loss",hamming_loss(true_label, predicted))
        print("classification_report", classification_report(true_label, predicted))
        print("jaccard_similarity_score", jaccard_similarity_score(true_label, predicted
))
        print("zero_one_loss", zero_one_loss(true_label, predicted))
        print("matthews_corrcoef", matthews_corrcoef(true_label, predicted))

evaluation_analysis(validation_labels_y,predictions_labels)
```

ROC and Precision-Recall curves

```
X_train = train_images
y_train = train_labels
X_test = validation_images
Y_test = validation_labels
```

```
import numpy as np
from sklearn import metrics
```

```
prediction = model.predict(validation_images)
```

```
from sklearn.metrics import roc_curve
```

```
y_pred_keras = model.predict(X_test).ravel()
fpr_keras, tpr_keras, thresholds_keras = roc_curve(Y_test.ravel(), y_pred_keras)
```

```
from sklearn.metrics import auc
auc_keras = auc(fpr_keras, tpr_keras)
```

```
import matplotlib.pyplot as plt
plt.figure(1)
plt.plot([0, 1], [0, 1], 'k--')
plt.plot(fpr_keras, tpr_keras, label='Keras (area = {:.3f})'.format(auc_keras))
plt.xlabel('False positive rate')
plt.ylabel('True positive rate')
plt.title('ROC curve')
plt.legend(loc='best')
plt.show()
```

Save model as json string

```
model_json = model.to_json()
with open("model.json", "w") as json_file:
    json_file.write(model_json)
```

Base de datos SMIC

El siguiente codigo se creó en Google Colab. Tambien se puede ver en la url https://colab.research.google.com/drive/1Hrq9bOrzS3IxOSUivqvz2hLPfNoPXpEk y en la url de Github https://github.com/natzasu13/microexpresion-recognition/blob/smic-microexpresion/Microexpression_recognition_MicroExpSTCNN_SMIC_Maestria_v1_2.ipynb

```
import os
import cv2
import numpy
import imageio
from sklearn.metrics import confusion_matrix, accuracy_score, classification_report,
precision_recall_curve, roc_curve, auc
from keras.models import Sequential, model_from_json
from keras.layers.core import Dense, Dropout, Activation, Flatten
from keras.layers.convolutional import Convolution3D, MaxPooling3D
from tensorflow.keras.optimizers import SGD
from keras.callbacks import ModelCheckpoint
from keras.utils import np_utils, generic_utils
from sklearn.model_selection import train_test_split
from keras import backend as K
import sys
from matplotlib import pyplot as plt
import tensorflow as tf
from tensorflow.keras import layers
```

```
from keras.layers import GaussianNoise
import matplotlib.pyplot as plt
from keras.preprocessing.image import ImageDataGenerator
from sklearn.metrics import roc_auc_score, f1_score, precision_score, recall_score,
hamming_loss, jaccard_similarity_score, log_loss, zero_one_loss, matthews_corrcoef
from sklearn import metrics
from sklearn.metrics import roc_curve
import datetime
from tensorflow import keras

K.set_image_data_format('channels_first')
```

Initialize variables

```
image_rows, image_columns, image_depth = 64, 64, 18
images_list = []
negative_images_count, positive_images_count, surprise_images_count = 0,0,0
current_date = datetime.datetime.today().strftime("%Y-%m-%d_%H:%M:%S")


#From drive
CODE_PATH = 'drive/MyDrive/MAESTRIA ING DE SISTEMAS/1. MICROEXPRESIONES FACIALES TRA
BAJO DE GRADO/3. Codigo/microexpresion-recognition-colab/SMIC'
SMIC_path = CODE_PATH + '/datasets/SMIC'
ZIP_PATH = CODE_PATH + '/datasets/SMIC/SMIC_all_cropped.zip'
HSpath =SMIC_path +'/SMIC_all_cropped/HS/'


#From .zip
"""
you can find the .zip file at this shared url https://drive.google.com/drive/u/0/fol
ders/1g2hPdDahKFaxBhA5DkswE6Jz3M31F4kp
```

```
   copy the .zip file into the CODE_PATH folder
   """
   CODE_PATH = 'drive/MyDrive/MAESTRIA ING DE SISTEMAS/1. MICROEXPRESIONES FACIALES TRA
BAJO DE GRADO/3. Codigo/microexpresion-recognition-colab/SMIC'
   ZIP_PATH = CODE_PATH + '/datasets/SMIC/SMIC_all_cropped.zip'
```

Mount google drive

```
   from google.colab import drive
   drive.mount('/content/drive')
```

Extract Images from .zip

```
   """
   Extract the .zip files into the 'train data' folder.
   """
   from zipfile import ZipFile

   #you can find the .zip file at this shared url https://drive.google.com/drive/u/0/fo
lders/1g2hPdDahKFaxBhA5DkswE6Jz3M31F4kp
   zip_name = ZIP_PATH
   with ZipFile(zip_name, 'r') as zip:
     zip.extractall('smic_data')
     print("Extracted all images files into the folder 'smic_data'")


   HSpath ='smic_data/SMIC_all_cropped/HS/'
```

Load images

```
   rootDirectory = os.listdir(HSpath)
   for s in rootDirectory:
       sPath = HSpath + s + "/micro/negative/"
       microPath = os.listdir(sPath)
```

```
        for video in microPath:
            videopath = sPath + video
            frames = []
            framelisting = os.listdir(videopath)
            framesCount = image_depth if len(framelisting) > image_depth else 0
            framerange = [x for x in range(image_depth)]
            if(framesCount != 0):
                for frame in framerange:
                      imagepath = videopath + "/" + framelisting[frame]
                      image = cv2.imread(imagepath)
                      imageresize = cv2.resize(image, (image_rows, image_columns), inter
polation = cv2.INTER_AREA)
                      grayimage = cv2.cvtColor(imageresize, cv2.COLOR_BGR2GRAY)
                      frames.append(grayimage)

                frames = numpy.asarray(frames)
                videoarray = numpy.rollaxis(numpy.rollaxis(frames, 2, 0), 2, 0)

                images_list.append(videoarray)
                negative_images_count  += 1



        sPath = HSpath + s + "/micro/positive/"
        microPath = os.listdir(sPath)
        for video in microPath:
            videopath = sPath + video
            frames = []
            framelisting = os.listdir(videopath)
            framesCount = image_depth if len(framelisting) > image_depth else 0
            framerange = [x for x in range(image_depth)]
            if(framesCount != 0):
```

```
                for frame in framerange:
                      imagepath = videopath + "/" + framelisting[frame]
                      image = cv2.imread(imagepath)
                      imageresize = cv2.resize(image, (image_rows, image_columns), inter
polation = cv2.INTER_AREA)
                      grayimage = cv2.cvtColor(imageresize, cv2.COLOR_BGR2GRAY)
                      frames.append(grayimage)

                frames = numpy.asarray(frames)
                videoarray = numpy.rollaxis(numpy.rollaxis(frames, 2, 0), 2, 0)

                images_list.append(videoarray)
                positive_images_count +=1


        sPath = HSpath + s + "/micro/surprise/"
        microPath = os.listdir(sPath)
        for video in microPath:
            videopath = sPath + video
            frames = []
            framelisting = os.listdir(videopath)
            framesCount = image_depth if len(framelisting) > image_depth else 0
            framerange = [x for x in range(image_depth)]
            if(framesCount != 0):
                for frame in framerange:
                      imagepath = videopath + "/" + framelisting[frame]
                      image = cv2.imread(imagepath)
                      imageresize = cv2.resize(image, (image_rows, image_columns), inter
polation = cv2.INTER_AREA)
                      grayimage = cv2.cvtColor(imageresize, cv2.COLOR_BGR2GRAY)
                      frames.append(grayimage)
```

```
            frames = numpy.asarray(frames)
            videoarray = numpy.rollaxis(numpy.rollaxis(frames, 2, 0), 2, 0)

            images_list.append(videoarray)
            surprise_images_count += 1


    print("negative_images_count", negative_images_count)
    print("positive_images_count", positive_images_count)
    print("surprise_images_count", surprise_images_count)
```

Preparate data

```
#159 imagenes que se conviernes en escala de gris
#64x64 tamaño
#18 frames

#negative_images_count 66
#positive_images_count 50
#surprise_images_count 43


training_list = numpy.asarray(images_list)
trainingsamples = len(training_list)

traininglabels = numpy.zeros((trainingsamples, ), dtype = int)

traininglabels[0:66] = 0 #negativepath
traininglabels[66:113] = 1 #positivepath
traininglabels[113:156] = 2  #surprise
```

```
    traininglabels = np_utils.to_categorical(traininglabels, 3)

    training_data = [training_list, traininglabels]
    (trainingframes, traininglabels) = (training_data[0], training_data[1])

    training_set = numpy.zeros((trainingsamples, 1, image_rows, image_columns, image_dep
th))

    for h in range(trainingsamples):
        training_set[h][0][:][:][:] = trainingframes[h, :, :, :]

    training_set = training_set.astype('float32')

    new_path = '/content/drive/MyDrive/TesisMScNataliaZartha/'
    #Save in the new path to data augmentation
    numpy.save(new_path + '/traning_set.npy', train_images)
    numpy.save(new_path + '/traininglabels.npy', train_labels)

    training_set -= numpy.mean(training_set)
    training_set /= numpy.max(training_set)

    # Spliting the dataset into training and validation sets
    train_images, validation_images, train_labels, validation_labels = train_test_split
(training_set, traininglabels, test_size=0.3, random_state=4)
```

Save train and validation images and labels in a numpy array

```
    # Save training images and labels in a numpy array
    ruta = '/content/drive/MyDrive/TesisMScNataliaZartha/'
```

```
   numpy.save(CODE_PATH+'/numpy_training_datasets/microexpstcnn_images.npy', training_s
et)
   numpy.save(CODE_PATH+'/numpy_training_datasets/microexpstcnn_labels.npy', trainingla
bels)

   # Load training images and labels that are stored in numpy array
   """
   training_set = numpy.load(CODE_PATH+'/numpy_training_datasets/microexpstcnn_images.n
py')
   traininglabels =numpy.load(CODE_PATH+'/numpy_training_datasets/microexpstcnn_labels.
npy')
   """


   # 111 images
   # Save train images and labels in a numpy array
   numpy.save(CODE_PATH+'/numpy_training_datasets/microexpstcnn_images.npy', train_imag
es)
   numpy.save(CODE_PATH+'/numpy_training_datasets/microexpstcnn_labels.npy', train_labe
ls)



   # Load train images and labels that are stored in numpy array
   """
   train_images = numpy.load(CODE_PATH+'/numpy_training_datasets/microexpstcnn_images.n
py')
   train_labels =numpy.load(CODE_PATH+'/numpy_training_datasets/microexpstcnn_labels.np
y')
   """
```

```
   # 48 images
   # Save validation set in a numpy array
   numpy.save(CODE_PATH+'/numpy_validation_dataset/microexpstcnn_val_images.npy', valid
ation_images)
   numpy.save(CODE_PATH+'/numpy_validation_dataset/microexpstcnn_val_labels.npy', valid
ation_labels)

   # Load validation set from numpy array
   """
   validation_images = numpy.load(CODE_PATH+'/numpy_validation_dataset/microexpstcnn_va
l_images.npy')
   validation_labels = numpy.load(CODE_PATH+'/numpy_validation_dataset/microexpstcnn_va
l_labels.npy')
   """




   numpy.save(CODE_PATH+'/numpy_training_datasets/traning_set.npy', train_images)
   numpy.save(CODE_PATH+'/numpy_training_datasets/traininglabels.npy', train_labels)


   """
   #Save in the new path to data augmentation
   numpy.save(new_path + '/traning_set.npy', train_images)
   numpy.save(new_path + '/traininglabels.npy', train_labels)
   numpy.save(new_path + '/validation_images.npy', validation_images)
   numpy.save(new_path + '/validation_labels.npy', validation_labels)
   """
```

Model

```
# MicroExpSTCNN Model
model = Sequential()
model.add(Convolution3D(60, kernel_size=(3, 3, 15), input_shape=(1, image_rows, image_columns, image_depth), activation='relu'))
#model.add(Convolution3D(60, kernel_size=(3, 3, 15), input_shape=(1, image_depth, image_rows, image_columns), activation='relu'))
model.add(MaxPooling3D(pool_size=(3, 3, 3)))
model.add(Dropout(0.5))
model.add(Flatten())
model.add(Dense(128, kernel_initializer='normal', activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(3, kernel_initializer='normal'))
model.add(Activation('softmax'))
                                                # optimizer = 'adam'
model.compile(loss = 'categorical_crossentropy', optimizer = 'SGD', metrics = ['accuracy'])

model.summary()
```

Model architecture graph

```
!pip install keras-resnet
from keras_resnet.models import ResNet50, ResNet101, ResNet152
import numpy as np

#https://keras.io/api/utils/model_plotting_utils/
keras.utils.plot_model(
    model,
    to_file="model.png",
    show_shapes=False,
```

```
        show_dtype=False,
        show_layer_names=True,
        rankdir="TB",
        expand_nested=False,
        dpi=96,
        layer_range=None,
    )
```

Add Callback

```
    filepath=CODE_PATH+"/weights_microexpstcnn/weights-improvement-{epoch:02d}-
{val_accuracy:.2f}.hdf5"
    ######filepath=CODE_PATH+"/weights_microexpstcnn/model_weights.h5"
    checkpoint = ModelCheckpoint(filepath, monitor='val_accuracy', verbose=1, save_best_
only=True, mode='max')
    callbacks_list = [checkpoint]

    # Load pre-trained weights
    #model.load_weights(CODE_PATH+'/weights_microexpstcnn/weights-improvement-40-
0.69.hdf5')
    #model.load_weights(CODE_PATH+'/weights_microexpstcnn/weights-improvement-01-
1.00.hdf5')

    """
    https://www.kaggle.com/moghazy/guide-to-cnns-with-data-augmentation-keras
       # Stop training when `val_loss` is no longer improving
            monitor='val_loss',
            # "no longer improving" being defined as "no better than 1e-2 less"
            min_delta=1e-3,
            # "no longer improving" being further defined as "for at least 2 epochs"
            patience=25,
            verbose=1
```

```
"""


# Define callbacks.
checkpoint_cb = keras.callbacks.ModelCheckpoint(
    filepath, save_best_only=True
)
early_stopping_cb = keras.callbacks.EarlyStopping(monitor="val_acc", patience=15)
```

Training Model

```
hist = model.fit(
    train_images,
    train_labels,
    validation_data = (validation_images, validation_labels),
    callbacks=callbacks_list,
    batch_size = 16,
    epochs = 200,
    shuffle=True)
```

Model accuracy graph

```
plt.plot(hist.history['accuracy'], label='accuracy')
plt.plot(hist.history['val_accuracy'], label = 'val_accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.ylim([0.5, 1])
plt.legend(loc='lower right')

print("----------Model Evaluate------- ")
test_loss, test_acc = model.evaluate(validation_images,  validation_labels, verbose=
2)
```

```
print("test_acc", test_acc)

print("test_loss", test_loss)
```

```
plt.plot(hist.history['accuracy'])

plt.plot(hist.history['val_accuracy'])

plt.title('model accuracy')

plt.ylabel('accuracy')

plt.xlabel('epoch')

plt.legend(['train', 'test'], loc='upper left')

plt.show()
```

```
plt.plot(hist.history['loss'],)

plt.plot(hist.history['val_loss'])

plt.title('model loss')

plt.ylabel('loss')

plt.xlabel('epoch')

plt.legend(['train', 'test'], loc='upper right')

plt.show()
```

Predictions

```
# Finding Confusion Matrix using pretrained weights


predictions = model.predict(validation_images)

predictions_labels = numpy.argmax(predictions, axis=1)


validation_labels_y = numpy.argmax(validation_labels, axis=1)
```

```
cfm = confusion_matrix(validation_labels_y, predictions_labels)
print("confusion_matrix",cfm )

def evaluation_analysis(true_label,predicted):
    '''
    return all metrics results
    '''
    print("accuracy (EXACTITUD)",accuracy_score(true_label, predicted))
    print("f1 score macro",f1_score(true_label, predicted, average='macro'))
    print("f1 score micro",f1_score(true_label, predicted, average='micro'))
    print("precision score (PRESICION)",precision_score(true_label, predicted, avera
ge='macro'))
    print("recall score (SENSIBILIDAD)",recall_score(true_label, predicted, average=
'macro'))
    print("hamming_loss",hamming_loss(true_label, predicted))
    print("classification_report", classification_report(true_label, predicted))
    print("jaccard_similarity_score", jaccard_similarity_score(true_label, predicted
))
    #print("log_loss", log_loss(true_label, predicted))
    print("zero_one_loss", zero_one_loss(true_label, predicted))
    #print("AUC&ROC",roc_auc_score(true_label, predicted))
    print("matthews_corrcoef", matthews_corrcoef(true_label, predicted))


evaluation_analysis(validation_labels_y,predictions_labels)
```

ROC and Precision-Recall curves

```
X_train = train_images
y_train = train_labels
X_test = validation_images
```

```
Y_test = validation_labels


prediction = model.predict(validation_images)
y_pred_keras = model.predict(X_test).ravel()


fpr_keras, tpr_keras, thresholds_keras = roc_curve(Y_test.ravel(), y_pred_keras)


from sklearn.metrics import auc
auc_keras = auc(fpr_keras, tpr_keras)


import matplotlib.pyplot as plt
plt.figure(1)
plt.plot([0, 1], [0, 1], 'k--')
plt.plot(fpr_keras, tpr_keras, label='Keras (area = {:.3f})'.format(auc_keras))
plt.xlabel('False positive rate')
plt.ylabel('True positive rate')
plt.title('ROC curve')
plt.legend(loc='best')
plt.show()
```

Save model as json string

```
path = CODE_PATH + '/models'
model.save(path +  '/model_'+current_date)


model_json = model.to_json()
with open("model.json", "w") as json_file:
    json_file.write(model_json)
```

Modelo 2. Modelo convolucional 3D para el reconocimiento de microexpresiones faciales con data augmentation

Base de datos CASME II

El siguiente codigo se creó en Google Colab. Tambien se puede ver en la url https://colab.research.google.com/drive/1wvFxCNX37V49so_3fsd3aVmj6flUfHOz#scrollTo=Exmb8EytZ_eM y en la url de Github https://github.com/natzasu13/microexpresion-recognition/blob/casmeii_data_augmentation/ExampleVolumeDataAugmentation_casmeII.ipynb

```
import os
import zipfile
import numpy as np

from tensorflow import keras
from tensorflow.keras import layers
import tensorflow as tf
device_name = tf.test.gpu_device_name()
if device_name != '/device:GPU:0':
  raise SystemError('GPU device not found')
print('Found GPU at: {}'.format(device_name))

from google.colab import drive
drive.mount('/content/drive')
```

Cargar numpy arrays

```
ruta = '/content/drive/MyDrive/TesisMScNataliaZartha/'

import numpy as np


X = np.load(ruta+'microexpstcnn_images_tr_casmeII.npy')

t = np.load(ruta+'microexpstcnn_labels_tr_casmeII.npy')


from sklearn.model_selection import train_test_split


img = np.squeeze(X)/255.

lbl = t


x_train, x_val, y_train, y_val = train_test_split(img,lbl,test_size = 0.4,random_sta
te = 123)
```

Data augmentation

La secuencia de emociones también aumentó al girar en ángulos aleatorios durante el entrenamiento. Dado que los datos se almacenan en tensores de forma de rango 3 (muestras, altura, ancho, profundidad (fotogramas)), agregamos una dimensión de tamaño 1 en el eje 4 para poder realizar convoluciones 3D en los datos. La nueva forma es así (muestras, altura, ancho, profundidad, 1).

```
import random

from scipy import ndimage



@tf.function

def rotate(volume):

    """Rotate the volume by a few degrees"""


    def scipy_rotate(volume):

        # define some rotation angles

        angles = [-20, -10, -5, 5, 10, 20]

        # pick angles at random

        angle = random.choice(angles)
```

```
            # rotate volume
            volume = ndimage.rotate(volume, angle, reshape=False)
            volume[volume < 0] = 0
            volume[volume > 1] = 1
            return volume

        augmented_volume = tf.numpy_function(scipy_rotate, [volume], tf.float32)
        return augmented_volume


    def shift(volume):
        """Shift the volume"""

        def scipy_shift(volume):
            # shift volume
            volume = ndimage.shift(volume, [2, 2, 2], order=3, mode='constant', cval=0.0
, prefilter=True)
            volume[volume < 0] = 0
            volume[volume > 1] = 1
            return volume

        augmented_volume = tf.numpy_function(scipy_shift, [volume], tf.float32)
        return augmented_volume


    def train_preprocessing(volume, label):
        """Process training data by rotating and adding a channel."""
        # Rotate volume
        volume = rotate(volume)
        # Shift volume
        #volume = shift(volume)
        volume = tf.expand_dims(volume, axis=3)
```

```
    return volume, label


def validation_preprocessing(volume, label):
    """Process validation data by only adding a channel."""
    volume = tf.expand_dims(volume, axis=3)
    return volume, label
```

Mientras se definen los datos de entrenamiento y validación, los datos de entrenamiento se pasan a través de una función de aumento que rota aleatoriamente el volumen en diferentes ángulos. Tenga en cuenta que tanto los datos de entrenamiento como los de validación ya se han reescalado para tener valores entre 0 y 1.

```
# Define data loaders.
train_loader = tf.data.Dataset.from_tensor_slices((x_train, y_train))
validation_loader = tf.data.Dataset.from_tensor_slices((x_val, y_val))

batch_size = 2
# Augment the on the fly during training.
train_dataset = (
    train_loader.shuffle(len(x_train))
    .map(train_preprocessing)
    .batch(batch_size)
    .prefetch(2)
)
# Only rescale.
validation_dataset = (
    validation_loader.shuffle(len(x_val))
    .map(validation_preprocessing)
    .batch(batch_size)
    .prefetch(2)
)
```

Visualize an augmented CT scan.

```
import matplotlib.pyplot as plt


data = train_dataset.take(1)

images, labels = list(data)[0]

images = images.numpy()

image = images[0]

print("Dimension of the Emotional sequence is:", image.shape)

plt.imshow(np.squeeze(image[:, :, 15]), cmap="gray")


def plot_slices(num_rows, num_columns, width, height, data):

    """Plot a montage of 20 CT slices"""

    data = np.rot90(np.array(data))

    data = np.transpose(data)

    print(data.shape)

    data = np.reshape(data, (num_rows, num_columns, width, height))

    rows_data, columns_data = data.shape[0], data.shape[1]

    heights = [slc[0].shape[0] for slc in data]

    widths = [slc.shape[1] for slc in data[0]]

    fig_width = 12.0

    fig_height = fig_width * sum(heights) / sum(widths)

    f, axarr = plt.subplots(

        rows_data,

        columns_data,

        figsize=(fig_width, fig_height),

        gridspec_kw={"height_ratios": heights},

    )

    for i in range(rows_data):

        for j in range(columns_data):

            axarr[i, j].imshow(data[i][j], cmap="gray")
```

```
            axarr[i, j].axis("off")
    plt.subplots_adjust(wspace=0, hspace=0, left=0, right=1, bottom=0, top=1)
    plt.show()


# Visualize montage of slices.
plot_slices(2, 9, 64, 64, image[:, :, :18])
```

Define a 3D convolutional neural network

```
def get_model(width=64, height=64, depth=18):
    """Build a 3D convolutional neural network model."""

    inputs = keras.Input((width, height, depth, 1))

    x = layers.Conv3D(filters=64, kernel_size=3, activation="relu")(inputs)
    x = layers.MaxPool3D(pool_size=2)(x)
    x = layers.BatchNormalization()(x)

    x = layers.Conv3D(filters=32, kernel_size=3, activation="relu")(x)
    x = layers.MaxPool3D(pool_size=2)(x)
    x = layers.BatchNormalization()(x)

    x = layers.GlobalAveragePooling3D()(x)
    x = layers.Dense(units=512, activation="relu")(x)
    x = layers.Dropout(0.3)(x)

    outputs = layers.Dense(units=3, activation="softmax")(x)

    # Define the model.
    model = keras.Model(inputs, outputs, name="3dcnn")
    return model
```

```
# Build model.
model = get_model(width=64, height=64, depth=18)
model.summary()
```

Train model

```
# Compile model.
initial_learning_rate = 0.0001
lr_schedule = keras.optimizers.schedules.ExponentialDecay(
    initial_learning_rate, decay_steps=100000, decay_rate=0.96, staircase=True
)
model.compile(
    loss="categorical_crossentropy",
    optimizer=keras.optimizers.Adam(learning_rate=lr_schedule),
    metrics=["acc"],
)

# Define callbacks.
checkpoint_cb = keras.callbacks.ModelCheckpoint(
    ruta+"3d_image_classification_casmeII.h5", save_best_only=True
)
early_stopping_cb = keras.callbacks.EarlyStopping(monitor="loss", patience=15)

# Train the model, doing validation at the end of each epoch
epochs = 250
model.fit(
    train_dataset,
    validation_data=validation_dataset,
    epochs=epochs,
    shuffle=True,
```

```
    verbose=2,
    callbacks=[checkpoint_cb],
    #callbacks=[checkpoint_cb, early_stopping_cb],
)
```

Visualizing model performance

```
fig, ax = plt.subplots(1, 2, figsize=(20, 3))
ax = ax.ravel()

for i, metric in enumerate(["acc", "loss"]):
    ax[i].plot(model.history.history[metric])
    ax[i].plot(model.history.history["val_" + metric])
    ax[i].set_title("Model {}".format(metric))
    ax[i].set_xlabel("epochs")
    ax[i].set_ylabel(metric)
    ax[i].legend(["train", "val"])
```

Make predictions on a single Emotional image scan

```
# Load best weights.
model.load_weights(ruta+"3d_image_classification_casmeII.h5")
prediction = model.predict(np.expand_dims(x_val[0], axis=0))[0]
scores = prediction
class_names = ["Angry", "Happy","Disgust"]
for score, name in zip(scores, class_names):
    print(
        "This model is %.2f percent confident that Emotional sequence is %s"
        % ((100 * score), name)
    )

print('Real Label: ',y_val[0])
```

```
   from sklearn.metrics import accuracy_score, confusion_matrix, ConfusionMatrixDisplay
   dataTest = np.expand_dims(x_val[0], axis=0)
   y_est = model.predict(x_val)

   acc = accuracy_score(np.argmax(y_val,axis=1),np.argmax(y_est,axis=1))
   print('Acc: ',acc)
   cMat = confusion_matrix(np.argmax(y_val,axis=1),np.argmax(y_est,axis=1))
   ax = ConfusionMatrixDisplay(cMat,class_names)
   ax.plot()
   plt.show()

   from sklearn.metrics import roc_auc_score, f1_score, precision_score, recall_score,
hamming_loss, jaccard_similarity_score, log_loss, zero_one_loss, matthews_corrcoef, co
nfusion_matrix, accuracy_score, classification_report, precision_recall_curve, roc_cur
ve, auc

   def evaluation_analysis(true_label,predicted):
       '''
       return all metrics results
       '''
       print("accuracy",accuracy_score(true_label, predicted))
       print("f1 score macro",f1_score(true_label, predicted, average='macro'))
       print("f1 score micro",f1_score(true_label, predicted, average='micro'))
       print("precision score",precision_score(true_label, predicted, average='macro'))
       print("recall score (Sensibilidad)",recall_score(true_label, predicted, average=
'macro'))
       print("hamming_loss",hamming_loss(true_label, predicted))
       print("classification_report", classification_report(true_label, predicted))
       print("jaccard_similarity_score", jaccard_similarity_score(true_label, predicted
))
```

```
    #print("log_loss", log_loss(true_label, predicted))
    print("zero_one_loss", zero_one_loss(true_label, predicted))
    #print("AUC&ROC",roc_auc_score(true_label, predicted))
    print("matthews_corrcoef", matthews_corrcoef(true_label, predicted))


evaluation_analysis(np.argmax(y_val,axis=1),np.argmax(y_est,axis=1))
```

ROC and Precision-Recall curves

```
fpr_keras, tpr_keras, thresholds_keras = roc_curve(y_val.ravel(), y_est.ravel())

from sklearn.metrics import auc
auc_keras = auc(fpr_keras, tpr_keras)

import matplotlib.pyplot as plt
plt.figure(1)
plt.plot([0, 1], [0, 1], 'k--')
plt.plot(fpr_keras, tpr_keras, label='Keras (area = {:.3f})'.format(auc_keras))
plt.xlabel('False positive rate')
plt.ylabel('True positive rate')
plt.title('ROC curve')
plt.legend(loc='best')
plt.show()
```

Save model as json string

```
import datetime
current_date = datetime.datetime.today().strftime("%Y-%m-%d_%H:%M:%S")

path = ruta + 'model/'
model.save(path +  'model_'+current_date)
```

```
model_json = model.to_json()
with open("model.json", "w") as json_file:
    json_file.write(model_json)
```

Base de datos SMIC

El siguiente codigo se creó en Google Colab. Tambien se puede ver en la url ExampleVolumeDataAugmentation_smic.ipynb - Colaboratory (google.com) y en la url de Github https://github.com/natzasu13/microexpresion-recognition/blob/smic_data_augmentation/ExampleVolumeDataAugmentation_smic.ipynb

```
import os
import zipfile
import numpy as np
# import tensorflow as tf

from tensorflow import keras
from tensorflow.keras import layers

import tensorflow as tf
device_name = tf.test.gpu_device_name()
if device_name != '/device:GPU:0':
  raise SystemError('GPU device not found')
print('Found GPU at: {}'.format(device_name))
```

```
from google.colab import drive

drive.mount('/content/drive')
```

Cargar los numpy arrays de las imagenes

```
ruta = '/content/drive/MyDrive/TesisMScNataliaZartha/'

import numpy as np


X = np.load(ruta+'microexpstcnn_images_tr.npy')

t = np.load(ruta+'microexpstcnn_labels_tr.npy')


img = np.squeeze(X)/255.

lbl = t


x_train, x_val, y_train, y_val = train_test_split(img,lbl,test_size = 0.4,random_sta
te = 123)
```

Data augmentation

La secuencia de emociones también aumentó al girar en ángulos aleatorios durante el entrenamiento. Dado que los datos se almacenan en tensores de forma de rango 3 (muestras, altura, ancho, profundidad (fotogramas)), agregamos una dimensión de tamaño 1 en el eje 4 para poder realizar convoluciones 3D en los datos. La nueva forma es así (muestras, altura, ancho, profundidad, 1).

```
import random

from scipy import ndimage


@tf.function

def rotate(volume):

    """Rotate the volume by a few degrees"""


    def scipy_rotate(volume):

        # define some rotation angles
```

```
            angles = [-20, -10, -5, 5, 10, 20]
            # pick angles at random
            angle = random.choice(angles)
            # rotate volume
            volume = ndimage.rotate(volume, angle, reshape=False)
            volume[volume < 0] = 0
            volume[volume > 1] = 1
            return volume

        augmented_volume = tf.numpy_function(scipy_rotate, [volume], tf.float32)
        return augmented_volume


    def shift(volume):
        """Shift the volume"""

        def scipy_shift(volume):
            # shift volume
            volume = ndimage.shift(volume, [2, 2, 2], order=3, mode='constant', cval=0.0
, prefilter=True)
            volume[volume < 0] = 0
            volume[volume > 1] = 1
            return volume

        augmented_volume = tf.numpy_function(scipy_shift, [volume], tf.float32)
        return augmented_volume


    def train_preprocessing(volume, label):
        """Process training data by rotating and adding a channel."""
        # Rotate volume
        volume = rotate(volume)
```

```
    # Shift volume
    #volume = shift(volume)
    volume = tf.expand_dims(volume, axis=3)
    return volume, label


def validation_preprocessing(volume, label):
    """Process validation data by only adding a channel."""
    volume = tf.expand_dims(volume, axis=3)
    return volume, label
```

Mientras se definen los datos de entrenamiento y validación, los datos de entrenamiento se pasan a través de una función de aumento que rota aleatoriamente el volumen en diferentes ángulos. Tenga en cuenta que tanto los datos de entrenamiento como los de validación ya se han reescalado para tener valores entre 0 y 1.

```
# Define data loaders.
train_loader = tf.data.Dataset.from_tensor_slices((x_train, y_train))
validation_loader = tf.data.Dataset.from_tensor_slices((x_val, y_val))

batch_size = 2
# Augment the on the fly during training.
train_dataset = (
    train_loader.shuffle(len(x_train))
    .map(train_preprocessing)
    .batch(batch_size)
    .prefetch(2)
)
# Only rescale.
validation_dataset = (
```

```
    validation_loader.shuffle(len(x_val))
    .map(validation_preprocessing)
    .batch(batch_size)
    .prefetch(2)
)
```

Visualize an augmented.

```
import matplotlib.pyplot as plt

data = train_dataset.take(1)
images, labels = list(data)[0]
images = images.numpy()
image = images[0]
print("Dimension of the Emotional sequence is:", image.shape)
plt.imshow(np.squeeze(image[:, :, 15]), cmap="gray")



def plot_slices(num_rows, num_columns, width, height, data):
    """Plot a montage of 20 CT slices"""
    data = np.rot90(np.array(data))
    data = np.transpose(data)
    print(data.shape)
    data = np.reshape(data, (num_rows, num_columns, width, height))
    rows_data, columns_data = data.shape[0], data.shape[1]
    heights = [slc[0].shape[0] for slc in data]
    widths = [slc.shape[1] for slc in data[0]]
    fig_width = 12.0
    fig_height = fig_width * sum(heights) / sum(widths)
    f, axarr = plt.subplots(
```

```
        rows_data,
        columns_data,
        figsize=(fig_width, fig_height),
        gridspec_kw={"height_ratios": heights},
    )
    for i in range(rows_data):
        for j in range(columns_data):
            axarr[i, j].imshow(data[i][j], cmap="gray")
            axarr[i, j].axis("off")
    plt.subplots_adjust(wspace=0, hspace=0, left=0, right=1, bottom=0, top=1)
    plt.show()
```

Define a 3D convolutional neural network

```
def get_model(width=64, height=64, depth=18):
    """Build a 3D convolutional neural network model."""

    inputs = keras.Input((width, height, depth, 1))

    x = layers.Conv3D(filters=350, kernel_size=3, activation="relu")(inputs)
    x = layers.MaxPool3D(pool_size=3)(x)
    x = layers.BatchNormalization()(x)

    x = layers.Conv3D(filters=100, kernel_size=3, activation="relu")(x)
    x = layers.MaxPool3D(pool_size=2)(x)
    x = layers.BatchNormalization()(x)

    x = layers.GlobalAveragePooling3D()(x)
    x = layers.Dense(units=600, activation="relu")(x)
    x = layers.Dropout(0.3)(x)

    outputs = layers.Dense(units=3, activation="softmax")(x)
```

```
    # Define the model.
    model = keras.Model(inputs, outputs, name="3dcnn")
    return model


# Build model.
model = get_model(width=64, height=64, depth=18)
model.summary()
```

Model architecture graph

```
!pip install keras-resnet
from keras_resnet.models import ResNet50, ResNet101, ResNet152
import numpy as np
from tensorflow.keras.utils import plot_model

plot_model(model, to_file='model.png', show_shapes=True, show_layer_names=True)
```

Train model

```
# Compile model.
initial_learning_rate = 0.00001
lr_schedule = keras.optimizers.schedules.ExponentialDecay(
    initial_learning_rate, decay_steps=100000, decay_rate=0.96, staircase=True
)
model.compile(
    loss="categorical_crossentropy",
    optimizer=keras.optimizers.Adam(learning_rate=lr_schedule),
    #optimizer=keras.optimizers.Adam(learning_rate=0.001),
    metrics=["acc"],
)
```

```
# Define callbacks.
checkpoint_cb = keras.callbacks.ModelCheckpoint(
    ruta+"3d_image_classification_smic.h5", save_best_only=True
)
early_stopping_cb = keras.callbacks.EarlyStopping(monitor="loss", patience=15)

# Train the model, doing validation at the end of each epoch
epochs = 300
model.fit(
    train_dataset,
    validation_data=validation_dataset,
    epochs=epochs,
    shuffle=True,
    verbose=2,
    callbacks=[checkpoint_cb],
    #callbacks=[checkpoint_cb, early_stopping_cb],
)

# Load the TensorBoard notebook extension.
%load_ext tensorboard

import tensorboard
tensorboard.__version__
from datetime import datetime

# Define the Keras TensorBoard callback.
logdir="logs/fit/" + datetime.now().strftime("%Y%m%d-%H%M%S")
tensorboard_callback = keras.callbacks.TensorBoard(log_dir=logdir)

# Train the model.
```

```
epochs = 50
model.fit(
    train_dataset,
    validation_data=validation_dataset,
    epochs=epochs,
    shuffle=True,
    verbose=2,
    callbacks=[tensorboard_callback],
    #callbacks=[checkpoint_cb, early_stopping_cb],
)

# Examining the TensorFlow Graph  |  TensorBoard

%tensorboard --logdir logs

!tensorboard dev upload \
  --logdir logs \
  --name "Sample op-level graph" \
  --one_shot
```

Visualizing model performance

```
fig, ax = plt.subplots(1, 2, figsize=(20, 3))
ax = ax.ravel()

for i, metric in enumerate(["acc", "loss"]):
    ax[i].plot(model.history.history[metric])
    ax[i].plot(model.history.history["val_" + metric])
    ax[i].set_title("Model {}".format(metric))
    ax[i].set_xlabel("epochs")
    ax[i].set_ylabel(metric)
```

```
    ax[i].legend(["train", "val"])
```

Make predictions on a single Emotional image scan

```
# Load best weights.
model.load_weights(ruta+"3d_image_classification_smic.h5")


print(x_val.shape)
print(x_val[0].shape)

prediction = model.predict(np.expand_dims(x_val[0], axis=0))[0]
scores = prediction
class_names = ["Negativo", "Positivo","Sorpresa"]
for score, name in zip(scores, class_names):
    print(
        "This model is %.2f percent confident that Emotional sequence is %s"
        % ((100 * score), name)
    )

print('Real Label: ',y_val[0])

plt.imshow(np.squeeze(x_val[0,:, :, 10]), cmap="gray")

from sklearn.metrics import accuracy_score, confusion_matrix, ConfusionMatrixDisplay
dataTest = np.expand_dims(x_val[0], axis=0)
y_est = model.predict(x_val)

acc = accuracy_score(np.argmax(y_val,axis=1),np.argmax(y_est,axis=1))
print('Acc: ',acc)
cMat = confusion_matrix(np.argmax(y_val,axis=1),np.argmax(y_est,axis=1))
ax = ConfusionMatrixDisplay(cMat,class_names)
```

```
    ax.plot()
    plt.show()

    from sklearn.metrics import roc_auc_score, f1_score, precision_score, recall_score,
hamming_loss, jaccard_similarity_score, log_loss, zero_one_loss, matthews_corrcoef, co
nfusion_matrix, accuracy_score, classification_report, precision_recall_curve, roc_cur
ve, auc

    def evaluation_analysis(true_label,predicted):
        '''
        return all metrics results
        '''
        print("accuracy",accuracy_score(true_label, predicted))
        print("f1 score macro",f1_score(true_label, predicted, average='macro'))
        print("f1 score micro",f1_score(true_label, predicted, average='micro'))
        print("precision score",precision_score(true_label, predicted, average='macro'))
        print("recall score (Sensibilidad)",recall_score(true_label, predicted, average=
'macro'))
        print("hamming_loss",hamming_loss(true_label, predicted))
        print("classification_report", classification_report(true_label, predicted))
        print("jaccard_similarity_score", jaccard_similarity_score(true_label, predicted
))
        #print("log_loss", log_loss(true_label, predicted))
        print("zero_one_loss", zero_one_loss(true_label, predicted))
        #print("AUC&ROC",roc_auc_score(true_label, predicted))
        print("matthews_corrcoef", matthews_corrcoef(true_label, predicted))


    evaluation_analysis(np.argmax(y_val,axis=1),np.argmax(y_est,axis=1))
```

ROC and Precision-Recall curves

```
fpr_keras, tpr_keras, thresholds_keras = roc_curve(y_val.ravel(), y_est.ravel())


from sklearn.metrics import auc

auc_keras = auc(fpr_keras, tpr_keras)


import matplotlib.pyplot as plt

plt.figure(1)

plt.plot([0, 1], [0, 1], 'k--')

plt.plot(fpr_keras, tpr_keras, label='Keras (area = {:.3f})'.format(auc_keras))

plt.xlabel('False positive rate')

plt.ylabel('True positive rate')

plt.title('ROC curve')

plt.legend(loc='best')

plt.show()
```

Save model as json string

```
import datetime

current_date = datetime.datetime.today().strftime("%Y-%m-%d_%H:%M:%S")


path = ruta + 'model/'

model.save(path +  'model_'+current_date)



model_json = model.to_json()

with open("model.json", "w") as json_file:

    json_file.write(model_json)
```

Modelo 3. Modelo temporal profundo para el reconocimiento de expresiones faciales

Base de datos CASME II

El siguiente codigo se creó en Google Colab. Tambien se puede ver en la url y en la url https://colab.research.google.com/drive/1P9GVLtvskIm7qAAYeyOtGDhisqEawIZz?authuser=2 de Github https://github.com/natzasu13/microexpresion-recognition/blob/timeDistributed_CASMEII/timeDistributedExample_KERAS_CASMEII.ipynb

```
import keras
from keras.layers import Input ,Dense, Dropout, Activation, LSTM
from keras.layers import Convolution2D, MaxPooling2D, Flatten, Reshape
from keras.models import Sequential
from keras.layers.wrappers import TimeDistributed
from keras.layers.pooling import GlobalAveragePooling1D
from keras.utils import np_utils
from keras.models import Model

import numpy as np

import tensorflow as tf
device_name = tf.test.gpu_device_name()
if device_name != '/device:GPU:0':
```

```
  raise SystemError('GPU device not found')
print('Found GPU at: {}'.format(device_name))

from google.colab import drive
drive.mount('/content/drive')
```

Cargar numpy array

```
ruta = '/content/drive/MyDrive/TesisMScNataliaZartha/'
import numpy as np
from sklearn.model_selection import train_test_split

X = np.squeeze(np.load(ruta+'microexpstcnn_images_tr_casmeII.npy'))
t = np.squeeze(np.load(ruta+'microexpstcnn_labels_tr_casmeII.npy'))

X = np.moveaxis(X, -1, 1)


img = np.squeeze(X)/255.
lbl = t

X_train, X_val, y_train, y_val = train_test_split(img,lbl,test_size = 0.4,random_sta
te = 123)

X_train = np.expand_dims(X_train,axis= 0)
X_train = np.moveaxis(X_train, 0, -1)

X_val = np.expand_dims(X_val,axis= 0)
X_val = np.moveaxis(X_val, 0, -1)
```

```
print(X_val.shape)
```

Crear modelo

```
num_frames = 18
width = 64
height = 64
num_classes = 3


model = Sequential()


model.add(
    TimeDistributed(
        Convolution2D(64, (3, 3), activation='relu'),
        input_shape=(num_frames, width, height, 1)
    )
)
model.add(TimeDistributed(MaxPooling2D((2, 2), strides=(1, 1))))

model.add(TimeDistributed(Convolution2D(128, (4,4), activation='relu')))
model.add(TimeDistributed(MaxPooling2D((2, 2), strides=(2, 2))))

model.add(TimeDistributed(Convolution2D(256, (4,4), activation='relu')))
model.add(TimeDistributed(MaxPooling2D((2, 2), strides=(2, 2))))

# extract features and dropout
model.add(TimeDistributed(Flatten()))
model.add(Dropout(0.5))

# input to LSTM
```

```
model.add(LSTM(256, return_sequences=False, dropout=0.5))

# classifier with sigmoid activation for multilabel
model.add(Dense(num_classes, activation='softmax'))
model.summary()



from tensorflow import keras

# Compile model.
initial_learning_rate = 0.0001
lr_schedule = keras.optimizers.schedules.ExponentialDecay(
    initial_learning_rate, decay_steps=100000, decay_rate=0.96, staircase=True
)
model.compile(
    loss="categorical_crossentropy",
    optimizer=keras.optimizers.Adam(learning_rate=lr_schedule),
    #optimizer=keras.optimizers.Adam(learning_rate=0.001),
    metrics=["acc"],
)

# Define callbacks.
checkpoint_cb = keras.callbacks.ModelCheckpoint(
    ruta+"3d_image_classification_casmeII.h5", save_best_only=True
)
early_stopping_cb = keras.callbacks.EarlyStopping(monitor="acc", patience=25)

# Train the model, doing validation at the end of each epoch
epochs = 1000
model.fit(X_train,y_train,
          validation_data=(X_val,y_val),
```

```
            epochs=epochs,
            shuffle=True,
            verbose=2,
            callbacks=[checkpoint_cb,early_stopping_cb],
    )

    import matplotlib.pyplot as plt
    history = model.history
    # Trainig Process
    print(model.history.history.keys())
    # summarize history for accuracy
    plt.plot(history.history['acc'])
    plt.plot(history.history['val_acc'])
    plt.title('model accuracy')
    plt.ylabel('accuracy')
    plt.xlabel('epoch')
    plt.legend(['train', 'test'], loc='upper left')
    plt.show()
    # summarize history for loss
    plt.plot(history.history['loss'])
    plt.plot(history.history['val_loss'])
    plt.title('model loss')
    plt.ylabel('loss')
    plt.xlabel('epoch')
    plt.legend(['train', 'test'], loc='upper left')
    plt.show()
```

Model Evaluation

```
    from sklearn.metrics import roc_auc_score, confusion_matrix, accuracy_score, f1_scor
e, precision_score,ConfusionMatrixDisplay, recall_score, classification_report, hammin
g_loss, jaccard_similarity_score, log_loss, zero_one_loss, matthews_corrcoef
```

```
    yEst_val = model.predict(X_val)


    tEst_val = np.argmax(yEst_val,axis=1)
    t_val = np.argmax(y_val,axis=1)


    print('Acc Val:',accuracy_score(t_val,tEst_val))


    cMat_val = confusion_matrix(t_val,tEst_val)


    ax = ConfusionMatrixDisplay(cMat_val,display_labels= ["Angry", "Happy","Disgust"])
    ax.plot(xticks_rotation = 45)


    def evaluation_analysis(true_label,predicted):
        '''
        return all metrics results
        '''
        print("accuracy (EXACTITUD)",accuracy_score(true_label, predicted))
        print("f1 score macro",f1_score(true_label, predicted, average='macro'))
        print("f1 score micro",f1_score(true_label, predicted, average='micro'))
        print("precision score (PRESICION)",precision_score(true_label, predicted, avera
ge='macro'))
        print("recall score (SENSIBILIDAD)",recall_score(true_label, predicted, average=
'macro'))
        print("hamming_loss",hamming_loss(true_label, predicted))
        print("classification_report", classification_report(true_label, predicted))
        print("jaccard_similarity_score", jaccard_similarity_score(true_label, predicted
))
        print("zero_one_loss", zero_one_loss(true_label, predicted))
        print("matthews_corrcoef", matthews_corrcoef(true_label, predicted))
```

```
evaluation_analysis(t_val,tEst_val)
```

```
   scores = yEst_val[0]
   class_names = ["Negativo", "Positivo","Sorpresa"]
   for score, name in zip(scores, class_names):
       print(
           "This model is %.2f percent confident that Emotional sequence is %s"
           % ((100 * score), name)
       )

   print('Real Label: ',y_val[0])
```

ROC and Precision-Recall curves

```
   from sklearn.metrics import roc_auc_score, f1_score, precision_score, recall_score,
hamming_loss, jaccard_similarity_score, log_loss, zero_one_loss, matthews_corrcoef, co
nfusion_matrix, accuracy_score, classification_report, precision_recall_curve, roc_cur
ve, auc

   fpr_keras, tpr_keras, thresholds_keras = roc_curve(y_val.ravel(), yEst_val.ravel())

   from sklearn.metrics import auc
   auc_keras = auc(fpr_keras, tpr_keras)

   import matplotlib.pyplot as plt
   plt.figure(1)
   plt.plot([0, 1], [0, 1], 'k--')
   plt.plot(fpr_keras, tpr_keras, label='Keras (area = {:.3f})'.format(auc_keras))
```

```
plt.xlabel('False positive rate')
plt.ylabel('True positive rate')
plt.title('ROC curve')
plt.legend(loc='best')
plt.show()
```

Save model as json string

```
import datetime
current_date = datetime.datetime.today().strftime("%Y-%m-%d_%H:%M:%S")

path = ruta + 'model/'
model.save(path +  'model_'+current_date)

model_json = model.to_json()
with open("model.json", "w") as json_file:
    json_file.write(model_json)
```

Base de datos SMIC

El siguiente codigo se creó en Google Colab. Tambien se puede ver en la url y en la url https://colab.research.google.com/drive/1FpfzA_DJzDm368Yb8mZjKxK67YGBK2Fm?authuser=2#scrollTo=AD_qaz5TpbqR de Github https://github.com/natzasu13/microexpresion-recognition/blob/timeDistributed_SMIC/timeDistributedExample_KERAS_SMIC.ipynb

```
import keras
from keras.layers import Input ,Dense, Dropout, Activation, LSTM
from keras.layers import Convolution2D, MaxPooling2D, Flatten, Reshape
from keras.models import Sequential
from keras.layers.wrappers import TimeDistributed
from keras.layers.pooling import GlobalAveragePooling1D
from keras.utils import np_utils
from keras.models import Model

import numpy as np

"""
%tensorflow_version 2.x
"""
import tensorflow as tf
device_name = tf.test.gpu_device_name()
if device_name != '/device:GPU:0':
  raise SystemError('GPU device not found')
print('Found GPU at: {}'.format(device_name))

from google.colab import drive
drive.mount('/content/drive')
```

Cargar numpy array de las imágenes

```
ruta = '/content/drive/MyDrive/TesisMScNataliaZartha/'
import numpy as np
from sklearn.model_selection import train_test_split

X = np.squeeze(np.load(ruta+'microexpstcnn_images_tr.npy'))
```

```
    t = np.squeeze(np.load(ruta+'microexpstcnn_labels_tr.npy'))
    X = np.moveaxis(X, -1, 1)


    img = np.squeeze(X)/255.
    lbl = t

    X_train, X_val, y_train, y_val = train_test_split(img,lbl,test_size = 0.4,random_sta
te = 123)

    X_train = np.expand_dims(X_train,axis= 0)
    X_train = np.moveaxis(X_train, 0, -1)

    X_val = np.expand_dims(X_val,axis= 0)
    X_val = np.moveaxis(X_val, 0, -1)


    print(X_val.shape)

    num_frames = 18
    width = 64
    height = 64
    num_classes = 3


    model = Sequential()

    model.add(
        TimeDistributed(
            Convolution2D(64, (3, 3), activation='relu'),
            input_shape=(num_frames, width, height, 1)
        )
```

```
)
model.add(TimeDistributed(MaxPooling2D((2, 2), strides=(1, 1))))

model.add(TimeDistributed(Convolution2D(128, (4,4), activation='relu')))
model.add(TimeDistributed(MaxPooling2D((2, 2), strides=(2, 2))))

model.add(TimeDistributed(Convolution2D(256, (4,4), activation='relu')))
model.add(TimeDistributed(MaxPooling2D((2, 2), strides=(2, 2))))

model.add(TimeDistributed(Convolution2D(326, (4,4), activation='relu')))
model.add(TimeDistributed(MaxPooling2D((2, 2), strides=(2, 2))))

# extract features and dropout
model.add(TimeDistributed(Flatten()))
model.add(Dropout(0.5))

# input to LSTM
model.add(LSTM(256, return_sequences=False, dropout=0.5))

# classifier with sigmoid activation for multilabel
model.add(Dense(num_classes, activation='softmax'))
model.summary()

from tensorflow import keras
```

Entrenar el modelo

```
# Compile model.
initial_learning_rate = 0.0001
lr_schedule = keras.optimizers.schedules.ExponentialDecay(
    initial_learning_rate, decay_steps=100000, decay_rate=0.96, staircase=True
```

```
)
model.compile(
    loss="categorical_crossentropy",
    optimizer=keras.optimizers.Adam(learning_rate=lr_schedule),
    #optimizer=keras.optimizers.Adam(learning_rate=0.001),
    metrics=["acc"],
)

# Define callbacks.
checkpoint_cb = keras.callbacks.ModelCheckpoint(
    ruta+"3d_image_classification_smic.h5", save_best_only=True
)
early_stopping_cb = keras.callbacks.EarlyStopping(monitor="acc", patience=25)

# Train the model, doing validation at the end of each epoch
epochs = 500
model.fit(X_train,y_train,
          validation_data=(X_val,y_val),
          epochs=epochs,
          shuffle=True,
          verbose=2,
          callbacks=[checkpoint_cb,early_stopping_cb],
          #callbacks=[checkpoint_cb],
)
```

Observe TensorFlow speedup on GPU relative to CPU

```
import timeit

device_name = tf.test.gpu_device_name()
if device_name != '/device:GPU:0':
```

```
  print(
      '\n\nThis error most likely means that this notebook is not '
      'configured to use a GPU. Change this in Notebook Settings via the '
      'command palette (cmd/ctrl-shift-P) or the Edit menu.\n\n')
  raise SystemError('GPU device not found')

def cpu():
  with tf.device('/cpu:0'):
    random_image_cpu = tf.random.normal((100, 100, 100, 3))
    net_cpu = tf.keras.layers.Conv2D(32, 7)(random_image_cpu)
    return tf.math.reduce_sum(net_cpu)

def gpu():
  with tf.device('/device:GPU:0'):
    random_image_gpu = tf.random.normal((100, 100, 100, 3))
    net_gpu = tf.keras.layers.Conv2D(32, 7)(random_image_gpu)
    return tf.math.reduce_sum(net_gpu)

cpu()
gpu()

# Run the op several times.
print('Time (s) to convolve 32x7x7x3 filter over random 100x100x100x3 images '
      '(batch x height x width x channel). Sum of ten runs.')
print('CPU (s):')
cpu_time = timeit.timeit('cpu()', number=10, setup="from_main_import cpu")
print(cpu_time)
print('GPU (s):')
gpu_time = timeit.timeit('gpu()', number=10, setup="from_main_import gpu")
print(gpu_time)
print('GPU speedup over CPU: {}x'.format(int(cpu_time/gpu_time)))
```

FLOPS calculation with tf.keras

```
import tensorflow as tf
from tensorflow.python.framework.convert_to_constants import convert_variables_to_co
nstants_v2_as_graph


from tensorflow.keras import Model, Input
from tensorflow.keras.layers import Dense, Flatten


def get_flops(model, batch_size=None):
    if batch_size is None:
        batch_size = 1


    real_model = tf.function(model).get_concrete_function(tf.TensorSpec([batch_size]
 + model.inputs[0].shape[1:], model.inputs[0].dtype))
    frozen_func, graph_def = convert_variables_to_constants_v2_as_graph(real_model)


    run_meta = tf.compat.v1.RunMetadata()
    opts = tf.compat.v1.profiler.ProfileOptionBuilder.float_operation()
    flops = tf.compat.v1.profiler.profile(graph=frozen_func.graph,
                                          run_meta=run_meta, cmd='op', options=opt
s)
    return flops.total_float_ops


flops = get_flops(model)
print(f"FLOPS: {flops}")
return model
```

Graficar el modelo

```
from tensorflow.keras.utils import plot_model

plot_model(model, to_file='model.png', show_shapes=True, show_layer_names=True)
```

Gráficas de accuracy y loss

```
import matplotlib.pyplot as plt
history = model.history
# Trainig Process
print(model.history.history.keys())
# summarize history for accuracy
plt.plot(history.history['acc'])
plt.plot(history.history['val_acc'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
# summarize history for loss
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```

Model Evaluation

```
from sklearn.metrics import roc_auc_score, confusion_matrix, accuracy_score, f1_scor
e, precision_score,ConfusionMatrixDisplay, recall_score, classification_report, hammin
g_loss, jaccard_similarity_score, log_loss, zero_one_loss, matthews_corrcoef
```

```
   yEst_val = model.predict(X_val)


   tEst_val = np.argmax(yEst_val,axis=1)
   t_val = np.argmax(y_val,axis=1)


   print('Acc Val:',accuracy_score(t_val,tEst_val))


   cMat_val = confusion_matrix(t_val,tEst_val)


   ax = ConfusionMatrixDisplay(cMat_val,display_labels=['Negativo','Positivo','Neutro']
)
   ax.plot(xticks_rotation = 45)




   def evaluation_analysis(true_label,predicted):
       '''
       return all metrics results
       '''
       print("accuracy (EXACTITUD)",accuracy_score(true_label, predicted))
       print("f1 score macro",f1_score(true_label, predicted, average='macro'))
       print("f1 score micro",f1_score(true_label, predicted, average='micro'))
       print("precision score (PRESICION)",precision_score(true_label, predicted, avera
ge='macro'))
       print("recall score (SENSIBILIDAD)",recall_score(true_label, predicted, average=
'macro'))
       print("hamming_loss",hamming_loss(true_label, predicted))
       print("classification_report", classification_report(true_label, predicted))
       print("jaccard_similarity_score", jaccard_similarity_score(true_label, predicted
))
```

```
        print("zero_one_loss", zero_one_loss(true_label, predicted))
        print("matthews_corrcoef", matthews_corrcoef(true_label, predicted))

evaluation_analysis(t_val,tEst_val)
```

ROC and Precision-Recall curves

```
from sklearn.metrics import roc_auc_score, f1_score, precision_score, recall_score, hamming_loss, jaccard_similarity_score, log_loss, zero_one_loss, matthews_corrcoef, confusion_matrix, accuracy_score, classification_report, precision_recall_curve, roc_curve, auc

fpr_keras, tpr_keras, thresholds_keras = roc_curve(y_val.ravel(), yEst_val.ravel())

from sklearn.metrics import auc
auc_keras = auc(fpr_keras, tpr_keras)

import matplotlib.pyplot as plt
plt.figure(1)
plt.plot([0, 1], [0, 1], 'k--')
plt.plot(fpr_keras, tpr_keras, label='Keras (area = {:.3f})'.format(auc_keras))
plt.xlabel('False positive rate')
plt.ylabel('True positive rate')
plt.title('ROC curve')
plt.legend(loc='best')
plt.show()
```

Save model as json string

```
import datetime
```

```
current_date = datetime.datetime.today().strftime("%Y-%m-%d_%H:%M:%S")


path = ruta + 'model/'

model.save(path +  'model_'+current_date)



model_json = model.to_json()

with open("model.json", "w") as json_file:

    json_file.write(model_json)
```

Anexo b. Artículo científico

Faculty of Engineering Magazine

Deep models for facial micro-expressions recognition

Modelos profundos para el reconocimiento de microexpresiones faciales

Natalia Zartha Suarez **rnan Felipe García Arias**

1 Facultad de Ingeniería, rsidad Tecnológica de Pereira. Carrera 27 # 10-02. C. P.660003. Barrio Álamos, Pereira, Risaralda - Colombia.

KEYWORDS

Deep learning, Facial micro-expression recognition, Convolutional Neural Networks, LSTM short/long-term memory networks.

Aprendizaje profundo, Reconocimiento de microexpresiones faciales, Redes neuronales convolucionales, Redes de memoria a corto/largo plazo LSTM.

ABSTRACT: The result of recognizing facial micro-expressions using a deep machine learning model is presented. For this purpose, 3 models are developed each for two databases of facial micro-expressions SMIC [1] and CASME II [2]. The first model implemented was MicroExpSTCNN, which was proposed by [129] using the same databases of facial micro-expressions, this grade work achieved a higher accuracy for both databases (90% for CASME II and 91.6% for SMIC); than that reported by the reference, which was 87.80% for the CASME II database. The second model implemented was a CNN 3D with data augmentation rotating the images with a certain number of degrees chosen randomly, for this model it was possible to improve the accuracy for the CASME II database (94.2%). The third model was built with a temporary 2D CNN and an LSTM layer, which significantly improved the prediction for both databases of facial micro-expressions. An application was also developed where the neural network model was created, and the previously trained weights were loaded for both SMIC [1] and CASME II [2] databases. The Flask framework was used to view the video and show the facial micro-expression predicted by the model.

RESUMEN: Se presenta el resultado de reconocer las microexpresiones faciales mediante un modelo profundo de aprendizaje automático. Para este fin, se desarrollan 3 modelos cada uno para dos bases de datos de microexpresiones faciales SMIC [1] y CASME II [2]. El primer modelo implementado fue MicroExpSTCNN el cual fue propuesto por [129] utilizando sobre las mismas bases de datos de microexpresiones faciales, este trabajo de grado logró obtener un accuracy mayor para ambas bases de datos (90 % para CASME II y 91.6 % para SMIC); que el reportado por la referencia, el cual fue de 87.80 % para la base de datos CASME II. El segundo modelo implementado fue un CNN 3D con data augmentation rotando las imágenes con cierto número de grados escogidos aleatoriamente, para este modelo se logró mejorar el accuracy para la base de datos CASME II (94.2 %). El tercer modelo se construyó con una CNN 2D temporal y una capa de LSTM, lo cual logró mejorar notablemente la predicción para ambas bases de datos de microexpresiones faciales. También se desarrolló una aplicación donde se creó el modelo de la red neuronal y se le cargaron los pesos entrenados previamente para ambas bases de datos de SMIC [1] y CASME II [2]. Se usó el framework Flask para visualizar el video y mostrar la microexpresión facial que predice el modelo.

1. *Introduction*

Revealing a criminal and proving that he is lying becomes a challenge at the time of being tried. Today criminal behavior cannot be biased only to the physical aspects of a person, this depends on multiple factors such as: political, cultural, economic, psychological, academic, behavioral, among others. But it can detect when a person is lying to be the first approach to discovering a criminal; lying can be detected by facial expression recognition FER (Facial Expression Recognition); they can recognize emotions through facial micro-expressions. Facial microexpressions are very short facial movements, generated between 1/15 and 1/25 seconds; Because they are so brief, they can reveal truths that the human eye cannot perceive. When a person is consciously aware that a facial expression is occurring, the person may try to suppress the expression because displaying it may not be appropriate or it could be due to a cultural display rule. [33]

The recognition of facial expressions, despite being a very powerful system, still presents some obstacles when analyzing the data, such as: changes in lighting, non-frontal postures of the head and occlusions. Also in systems Deep FER (Deep Facial Expression Recognition) presents the challenge of the lack of data in terms of quantity and quality. It is scientifically proven that when faced with a threatening situation, the sympathetic system is triggered and the body becomes alert. "All lies generate a psychophysiological reaction, and if the observer is well trained, he sees it." Not even the most experienced liar can indefinitely control what science has come to call facial micro-expressions. The detection of facial micro-expressions and points of conflict (scratching the nose, the head, touching the chin, among others) indicate that there is discomfort in the individual, that is, an internal dispute. [36]

Researchers are developing various techniques to detect lying individuals. British airport authorities are testing a system based on the Facial Action Coding System (FACS). The human face is a signaling vehicle that sends messages using not only its basic structure and muscle tone, but also changes in the face that transmit expressions, such as smiles, frowning, among others. The mood and intentions of the person can be read from facial expressions.

“Emotions are a process, a particular kind of automatic appraisal influenced by our evolutionary and personal past, in which we sense that something important to our welfare is occurring, and a set of psychological changes and emotional behaviors begins to deal with the situation." - Paul Ekman, PhD

In other words, emotions prepare us to face important events without having to think about them. These emotional responses just happen spontaneously, which means we don't choose to feel them, they happen automatically to us. Of all the human emotions that we experience, there are universal emotions that we all feel (Happiness, Sadness, Anger, Disgust, Surprise, Fear, Contempt), which transcend linguistic, regional, cultural and ethnic differences [37] These facial expressions are indifferent to the race or culture of the world regions. These expressions can be faked and it is the small movements that can tell us if an expression is being real or a lie. These small movements are called facial micro-expressions, which occur between 1/15 and 1/25 seconds and are imperceptible to the human eye. Facial expressions are one of the most powerful, natural, and universal signals through which human beings convey their emotions, states, and intentions. Several Facial Expression Recognition Systems (FER) have been explored to encode expression information from facial representations. [3]

1.1. The 7 Universal Emotions

Each emotion or family of emotions has a variety of expressions associated with it, but not visually identical. For example, anger has 60 visually non-identical expressions with equal core properties [25]. This core property differentiates the anger family from the fear family. A family of emotions is distinguished from another family of emotions based on 8 different characteristics [26]. Universal emotions are also called basic emotions. Every researcher of emotions agreed on the universality of six basic emotions: anger, disgust, sadness, joy, fear and surprise. In recent years, there is one more universal emotion addition called contempt [27]. [28]

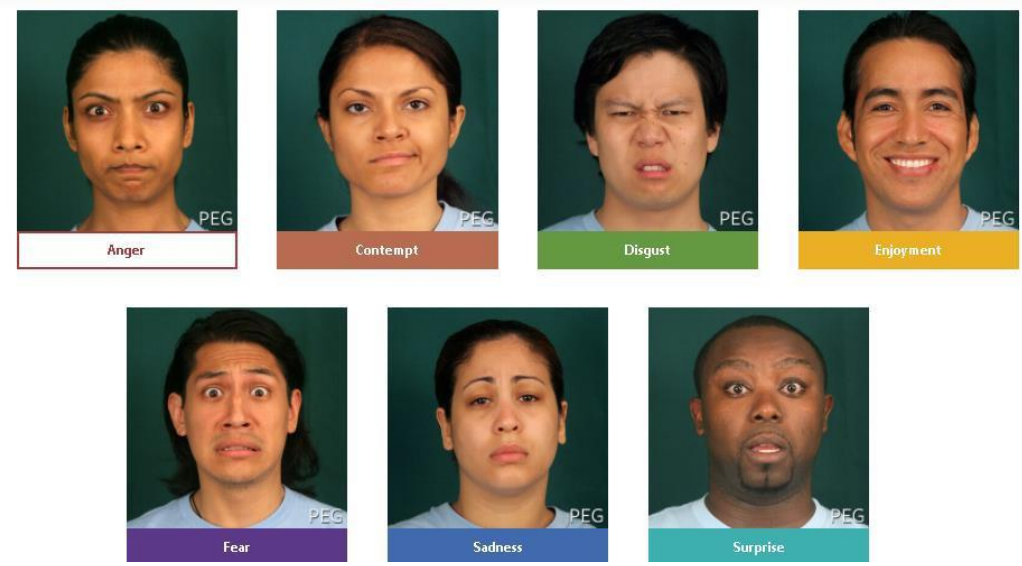


Figure 1 The 7 Universal Emotions [31].

1.1.1. Anger

It happens when a person is upset or the response when a person is attacked or hurt by someone. Anger can also be a response developed from extreme hatred [29].

1.1.2. Dislike

It happens when a person feels repulsively provoked by something offensive or disgusting [29].

1.1.3. Afraid

It happens when a person feels a threat, harm or pain [29].

1.1.4. Happiness

It happens when a person feels content, happy or pleasant [29]. Side eye muscles can detect false happiness. The following image shows real happiness and another false one. [29]

Figure 2 Real versus fake happiness. [30]

1.1.5. Sadness

It happens when a person feels unhappy or loses someone or something [29].

1.1.6. Surprise

It happens when a person feels something sudden and unexpected [29].

1.1.7. Contempt

It happens when one person feels superior to another [29].

1.2. Micro facial expressions

Microexpressions were first discovered by researchers Haggard and Isaacs. Dr. Paul Ekman popularized the term "microexpression" and greatly expanded the research. Ekman found that in all of these countries people expressed and identified the 7 universal emotions in the same way. He even ventured to a remote and primitive tribe called Fore in Papau New Guinea and found that they expressed the same emotions as us. The face is the best indicator of a person's emotions. [32]

Facial microexpressions are very short facial movements, generated between 1/15 and 1/25 seconds; Because they are so brief, they can reveal truths that the human eye cannot perceive.

1.3. Neural Networks CNN Convolutions

CNNs have been used extensively in a variety of computer vision applications, including RES. A convolutional neural network can have tens or hundreds of layers that learn to detect different characteristics of an image. Filters are applied to each training image at different resolutions, and the output of each convoluted image is used as input for the next layer. Filters can range from very simple features, such as brightness and edges, to more complex features, such as features that uniquely define the object. [4] Convolutional neural networks take advantage of the fact that the input consists of images and constrain the architecture more sensitively. In particular, unlike a normal neural network, the layers of a ConvNet have neurons arranged in 3 dimensions: width, height, depth. (Note that the word depth here refers to the third dimension of an activation volume, not the depth of an entire neural network, which can refer to the total number of layers in a network). [5]

1.4. LSTM Short / Long Term Memory Networks

LSTM neural networks are an extension of RNN Recurrent Neural Networks. RNNs are a type of network capable of recognizing and predicting data sequences over time, such as texts, genomes, spoken speech or numerical series. These types of networks are based on loops that allow the exit of the network or a part of it at a given moment to serve as the entrance of the network itself at the next moment. [6]

2. Materials and methods

There are several databases of facial microexpressions, among the most recognized are SMIC [1], USF-HD, CASME, CASME II [2] and SAMM [7].

	Polikovsky et al. [13]	SMIC [14]	USF-HD [15]	CASME [16]	CASME II [17]	SAMM
Micro-Movements	42	164	100	195	247	159
Participants	10	16	N/A	35	35	32
Resolution	640×480	640×480	720×1280	640×480/720×1280	640×480	2040×1088
Facial Resolution	N/A	190×230	N/A	150×190	280×340	400×400
FPS	200	100	29.7	60	200	200
Spontaneous/Posed	Posed	Spontaneous	Posed	Spontaneous	Spontaneous	Spontaneous
FACS Coded	No	No	No	Yes	Yes	Yes
Emotion Classes	6	3	6	7	5	7
Mean Age (SD)	N/A	26.7 (N/A)	N/A	22.03 (SD = 1.60)	22.03 (SD = 1.60)	33.24 (SD = 11.32)
Ethnicities	3	3	N/A	1	1	13

Figure 3 Summary of Publicly Available Datasets Containing Facial Microexpressions [7]

2.1. Facial microexpression databases

For the development of this article we worked with two databases of facial microexpressions SMIC [1] and CASME II [2].

2.1.1 CASME II

CASME II [2] is a 200 fps frame rate facial microexpression database with 247 FACS-encoded microexpressions from 26 participants. The facial area used for analysis is larger than CASME and SMIC at 280 × 340 pixels. This dataset includes only Chinese participants and categorizes in the same way. CASME II uses participants mostly students with a mean age of 22.03 (SD = 1.60). In addition to using only one ethnicity, this dataset uses only young participants, restricting the dataset to the analysis of similar-looking participants (based on age characteristics). [7]

The CASME II database is obtained through an FTP, which has a folder called "CASME2_preprocessed_small_Li Xiaobai" containing the pre-processed images and faces in a resolution of 64 x 64 pixels. This database is obtained from [8]. The CASME II database has the facial expressions of anger, happiness, and disgust.

2.1.2 SMIC

The SMIC dataset [1] uses an interrogation setup to cause high stress on subjects. Since micro-expressions are often displayed in high-stress situations where emotions are repressed, the interrogation environment allowed natural and spontaneous micro-expressions to occur [9]. This database consists of 164 spontaneous micro-expressions filmed at 100 fps and was one of the first to include spontaneous micro-expressions obtained through emotional induction experiments. However, this data set was not encoded by the Facial Actions Coding System (FACS) [10] and does not provide information on neutral sequences (the participant's face does not move before onset). The protocols for the induction experiment consisted of showing the participants videos to react and asking them to repress their emotions, however, without the FACS coding, the categorization of the emotion labels was left to the participant's own self-information. Leaving the categorization to the participants allows the introduction of subjectivity on emotional stimuli. The recording quality was also reduced due to flickering light and the facial area was 190 × 230 pixels. The SMIC

included a broader demographic of participants, 6 women and 14 men. Ethnicity was more diverse than previous data sets with ten Asians, nine Caucasians, and one African participant, however, this still only includes three ethnicities and does not provide a good overview of a population. [7]

The SMIC database is accessed through a Google Drive folder and has a folder called "SMIC_all_cropped" containing the pre-processed images and faces in a resolution of 64 x 64 pixels. This database is obtained from [11]. The SMIC database has the facial expressions of Negative, Positive and Surprise. In Figure 1 the most common facial microexpression databases are shown. For the development of the deep model, the CASME II [2] and SMIC [1] databases were used.

2.2. Used tools

2.2.1 Google Colaboratory

Collaboratory, also called Colab, is a product of Google Research. Colab allows everyone to write and execute arbitrary Python code in the browser. It is ideal for application in machine learning, data analytics and education projects. More technically, Colab is a Jupyter hosted notebook service that requires no configuration to use and provides free access to computational resources, including GPUs. Colab resources are not guaranteed or unlimited, and usage limits sometimes fluctuate. This is necessary so that Colab can offer the resources for free. Colab notebooks are stored in Google Drive, but can also be loaded fromGitHub. Colab notebooks can be shared in the same way as Google Sheets or Docs. [15]

2.2.2 TensorFlow

TensorFlow is an end-to-end open source platform for machine learning. It has a comprehensive and flexible ecosystem of tools, libraries, and community resources that enables researchers to advance the state of the art in ML and developers to easily create and deploy ML-powered applications. [16]

2.2.3 Keras

Keras is a deep learning API written in Python, which runs on the TensorFlow machine learning platform. It was developed with a focus on allowing quick experimentation. Being able to get from idea to result as quickly as possible is key to doing good research. Keras reduces the cognitive load on the developer so you can focus on the parts of the problem that matter. Keras embraces the principle of progressive disclosure of complexity: simple workflows must be quick and easy, while arbitrarily advanced workflows must be possible through a clear path that builds on what you have already learned. Keras provides industry-strength performance and scalability - it is used by organizations and companies such as NASA,

23. Evaluation of the Models

2.3.1 F1 Score

$$F_1 = \frac{2 * Precisión * Sensibilidad\ (recall)}{Precisión + Sensibilidad\ (recall)} \quad (1)$$

F1 is a general measure of the precision of a model that combines precision and recall, in that strange way where addition and multiplication just mix two ingredients to make a completely separate dish. That is, a good F1 score means you have few false positives and few false negatives, so you are correctly identifying real threats and are not bothered by false alarms. An F1 score is considered perfect when it is 1, while the model is a total failure when it is 0.

All models will generate some false negatives, some false positives, and possibly both. While you can adjust a model to minimize one or the other, you often face a trade-off, in which a decrease in false negatives leads to an increase in false positives, or vice versa. You will need to optimize the performance metrics that are most useful for your specific problem. [18]

2.3.2 Accuracy

The precision metric is used to be able to know what percentage of values that have been classified as positive are really positive, that is, it evaluates the data by the performance of positive predictions. [19]

$$\frac{TP}{(TP + FP)} \quad (2)$$

Equation 2 shows us how to calculate precision, where:

$TP = True\ Positive$, are the values that the algorithm classifies as positive and that really are positive.

$FP = False\ Positive$**,** are the values that the algorithm classifies as positive when they really are negative.

2.3.3 Sensitivity

Sensitivity and specificity are two values that indicate the ability of our estimator to discriminate positive cases from negative ones. Sensitivity is represented as the fraction of true positives, while specificity is the fraction of true negatives.

The sensitivity is also known as the True Positive Rate or TP. It is the proportion of positive cases that were correctly identified by the algorithm. It is calculated like this:[20]

$$\frac{TP}{TP + FN} \quad (3)$$

Equation 3 shows how sensitivity is calculated where:

$TP = True\ Positive$, are the values that the algorithm classifies as positive and that really are positive.

$FN = False\ \boldsymbol{Negative}$,are the values that the algorithm classifies as negative when they really are positive.

2.3.4 Specificity

Also known as the True Negative Rate, or TN. These are the negative cases that the algorithm has classified correctly. It expresses how well the model can detect that class. It is calculated: [20]

$$\frac{TN}{TN + FP} \qquad (4)$$

Specificity is calculated with equation 4 where:

$TN = True\ negatives$, They are values that the algorithm classifies as negative and that really are negative.

$FP = False\ positives$, are the values that the algorithm classifies as positive when they really are negative.

2.3.5 ROC curve

A receiver performance characteristic curve, or ROC curve, is a graphical diagram that illustrates the diagnostic ability of a binary classifier system as its discrimination threshold varies. The method was originally developed for military radar receiver operators beginning in 1941, which led to its name.

The ROC curve is created by plotting the true positive rate (TPR) versus the false positive rate (FPR) at various threshold values. The true positive rate is also known as sensitivity, recall, or probability of detection [21] in machine learning. The false positive rate is also known as the false alarm probability [21] and can be calculated as (1 - specificity). It can also be considered as a plot of the power versus the type I error of the decision rule (when the yieldis calculated from a sample of the population, it can be considered as an estimator of these quantities). The ROC curve is therefore the sensitivity or recovery as a function of the dip. [22]

True Positive Rate (TPR) is synonymous with recovery and is therefore defined as follows:

$$TPR = \frac{TP}{TP + FN} \qquad (5)$$

$$FPR = \frac{FP}{FP + TN} \quad (6)$$

An ROC curve plots TPR versus FPR at different classification thresholds. If the classification threshold is lowered, more items are classified as positive, increasing both false positives and true positives. Figure 2 shows a typical ROC curve. [23]

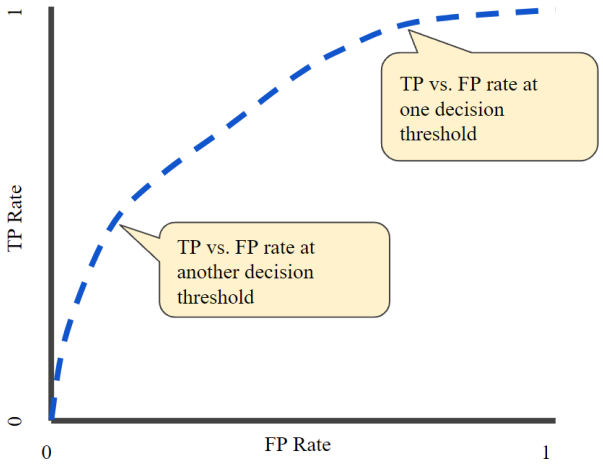


Figure 4 ROC curve

2.3.6 Confusion Matrix

The confusion matrix is a matrix that describes the performance of an algorithm, containing information about the real class and the model's predictions. [24] This matrix is essential when evaluating the performance of the model as it counts the successes and errors of each of the classes in the classification. In theTable 1 An example is given of which values a confusion matrix shows for each class being evaluated.

Table 1 Confusion Matrix

	NO (Prediction)	YES (Prediction)

NO (current)	True Negative (TN) A	False Positive (FP) B
YES (Current)	False Negative (FN) C	True Positive (TP) D

2.4. Deep models for facial micro-expression recognition

2.4.1 CNN 3D with data augmentation

For this model, the images in numpy array format generated with the previous model were loaded, for the two databases of facial microexpressions SMIC [1] and CASME II [2], a rotation function was added to increase the number of images, based on the model created by Hasib Zunair called "3D image classification from CT scans" [14] and adapted to our needs.

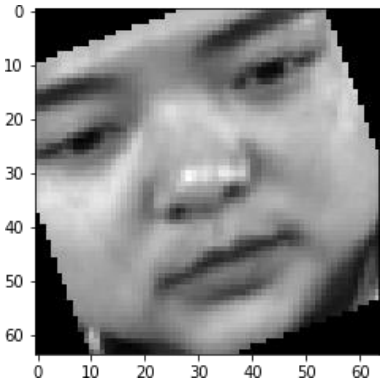


Figure 5 Randomly rotated face for CASME II database

Figure 6. 18 randomly rotated frames for CASME II database

For this model the images were rotated choosing a random angle within this range [-20, -10, -5, 5, 10, 20]. In Figure 3 the image of a randomly rotated face is displayed and in Figure 4 the 18 rotated frames are shown. For this model, 18 frames were used for both databases of facial microexpressions. For this model, 250 epochs were used with an EarlyStopping, 64 filters for the Convolutional 3D layer sight and 32 for the second, with a kernel size of 3 and a Softmax activation function, in Figure 5 The architecture created for the CASME II database is shown and in Figure 6 the architecture for the SMIC database.

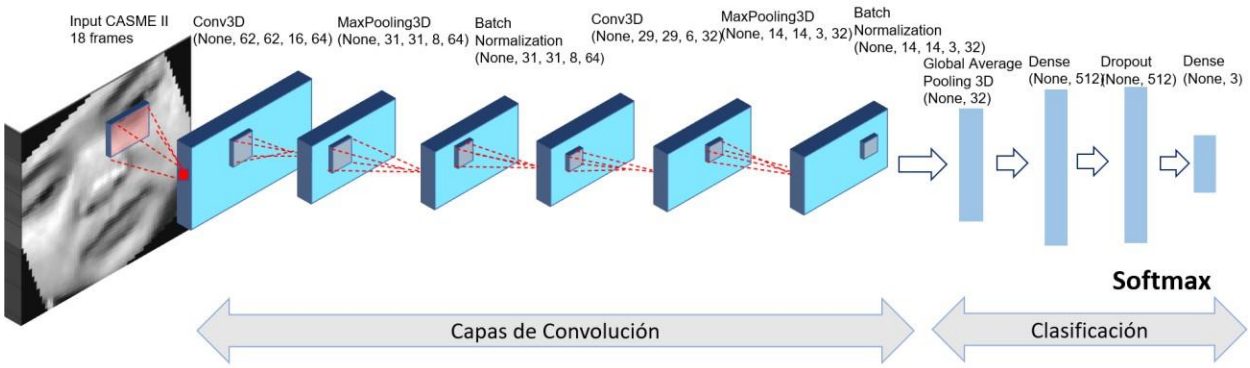


Figure 7 3D Convolutional Neural Network Architecture with Data Augmentation for the CASME II database

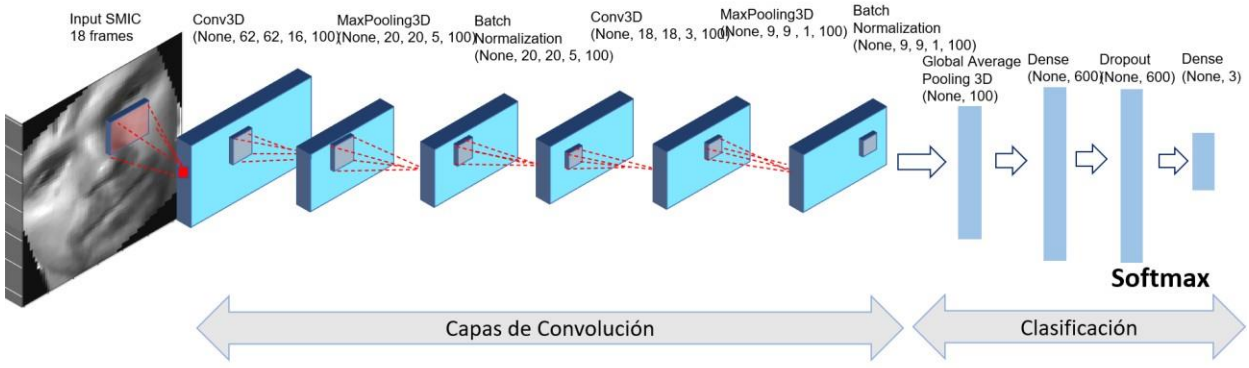


Figure 8 3D Convolutional Neural Network Architecture with Data Augmentation for the SMIC database

2.4.2 temporary CNN with an LSTM layer

This model was created by implementing a 2D Convolutional Neural Network with a time distribution and a Long short term memory LSTM layer, delivering better results than the two previously mentioned models for both databases of facial microexpressions.

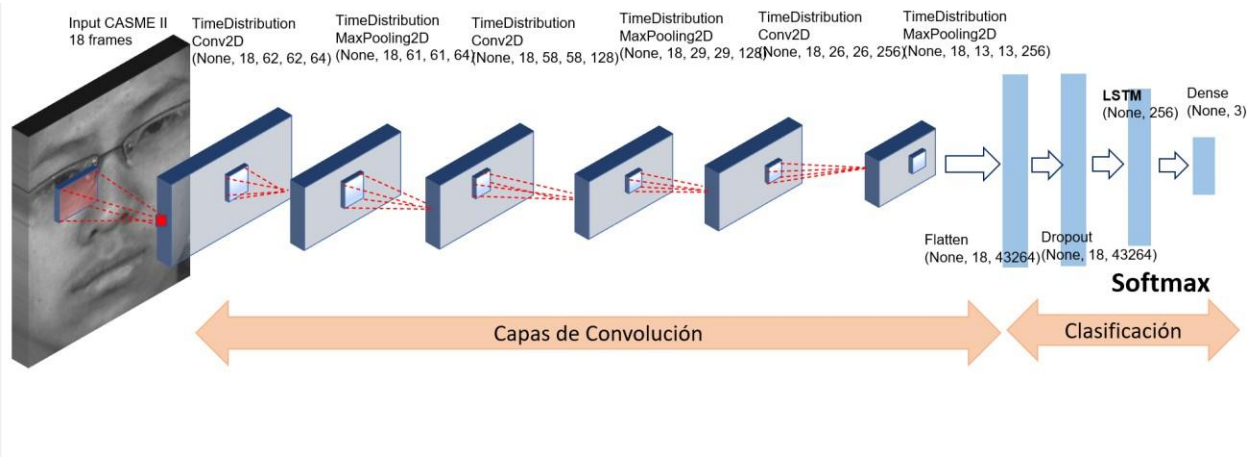


Figure 9 2D Convolutional Neural Network architecture with time distribution and LSTM for the CASME II database.

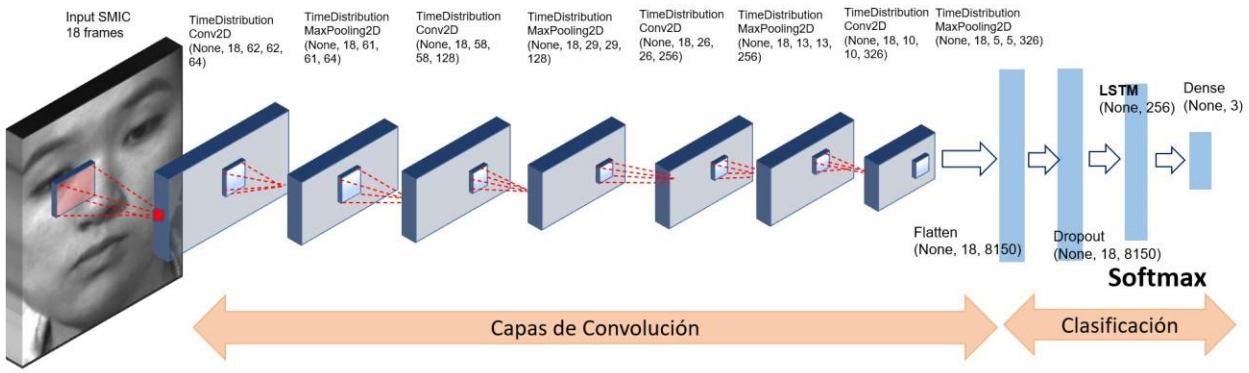


Figure 10 2D Convolutional Neural Network architecture with time distribution and LSTM for the SMIC database.

A 2D Convolutional Neural Network was built with a Sequential model, a time distribution, a filter of 128 for the first Convolution layer, 256 for the second and 326 for the third, unlike the CASME II database, which added one more

Convolution layer. An LSTM (Long Short Term Memory) layer was also added with a 256 filter. This model generated 10601545 parameters. In Figure 7 The network architecture that was created for the CASME II database is shown and in Figure 8 architecture for the SMIC database.

3. Results

In theTable 2 you can see the results for the two models implemented for the two databases of facial microexpressions CASME II [2] and SMIC [1]. For the first model implemented with Data Augmentation, the metrics for the CASME II database could be improved, obtaining an accuracy of 94.2%, on the other hand, for the SMIC database, accuracy decreased with a value of 84.4%. Finally, a temporary model was created with LSTM where it was possible to improve the metrics for both SMIC and CASME II databases; for CASME II an accuracy of 98% was obtained and for SMIC of 92.2%.

Table 2 Result of the metrics for the three models implemented.

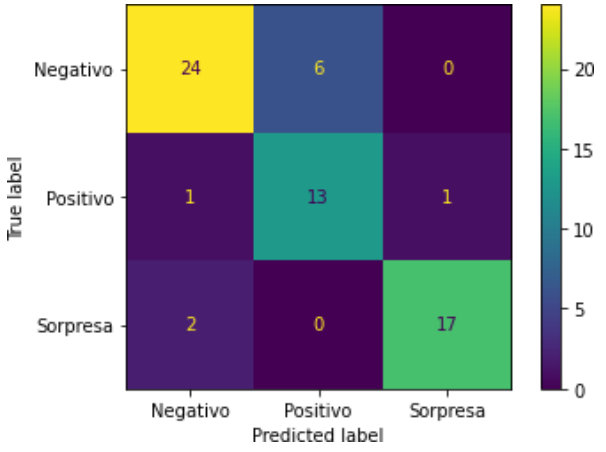
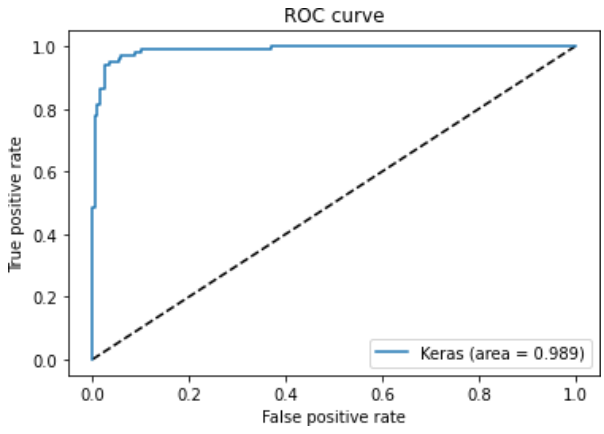
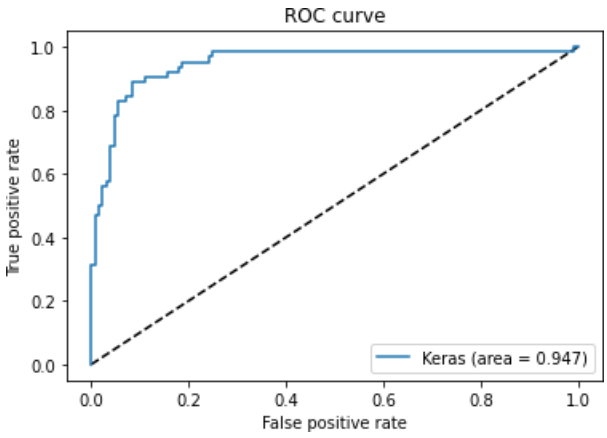
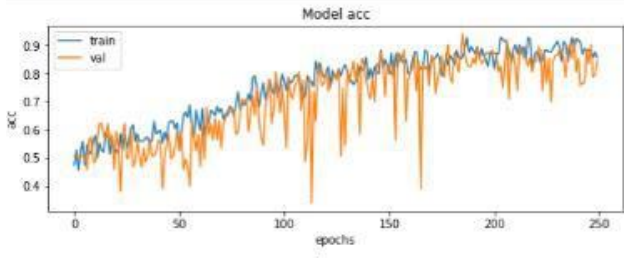
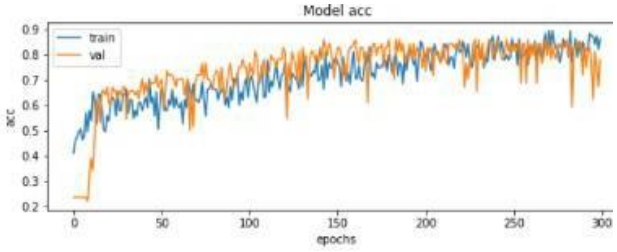
Model	Database	Precision	Sensitivity	Accuracy	F1 Score	AUC
CNN 3D with Data Augmentation	CASME II [2]	0.936	0.937	0.942	0.935	0.989
CNN 3D with Data Augmentation	SMIC [1]	0.839	0.854	0.844	0.842	0.947
CNN Temporal with LSTM	CASME II [2]	0.984	0.962	0.980	0.972	0.999
CNN Temporal with LSTM	SMIC [1]	0.906	0.922	0.922	0.912	0.974

3.1 CNN 3D with data augmentation

With the model implemented with Data Augmentation, the metrics for the CASME II database were improved, obtaining an accuracy of 94.2%, on the other hand, for the SMIC database, accuracy decreased with a value of 84.4%.

In theTable 3 You can see the results of the confusion matrix, accuracy and ROC curve for the two databases of facial microexpressions.

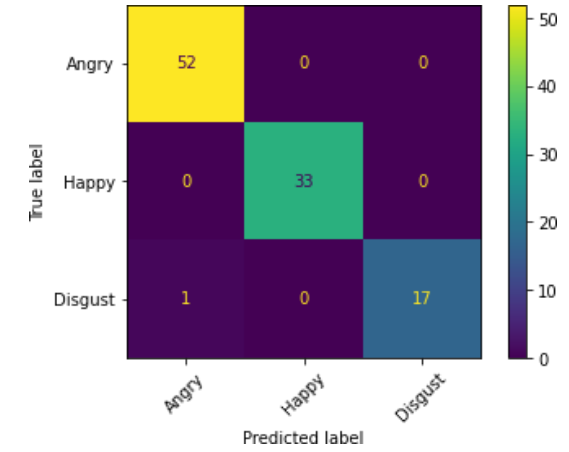
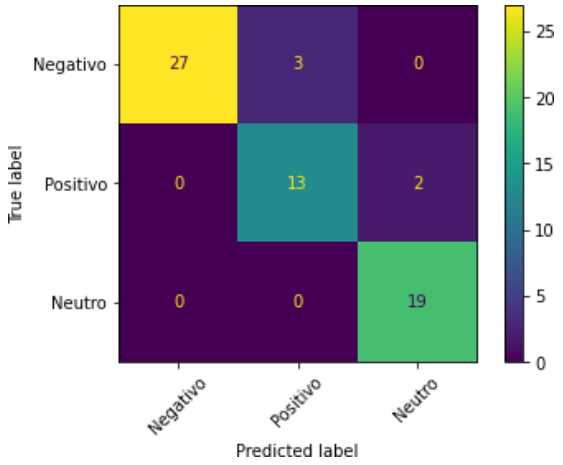
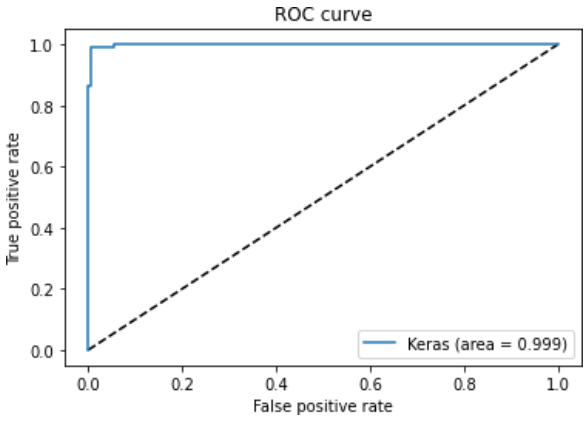
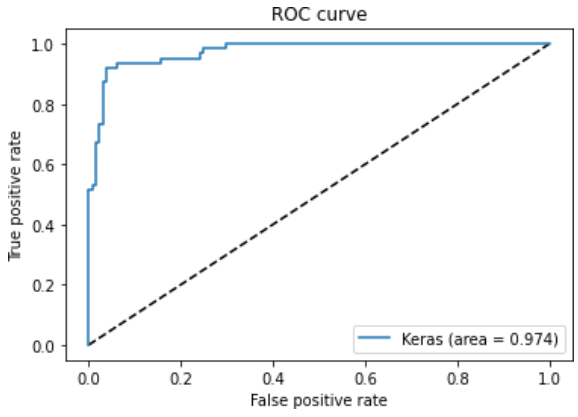
Table 3 Result of the CNN 3D model with data augmentation for CASME II and SMIC.

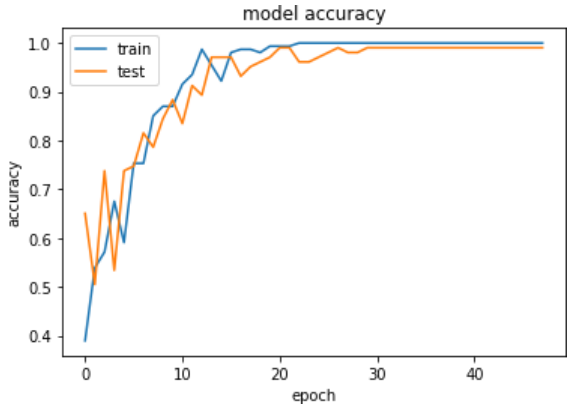
CASME II [2]	**SMIC [2]**
	
	
	

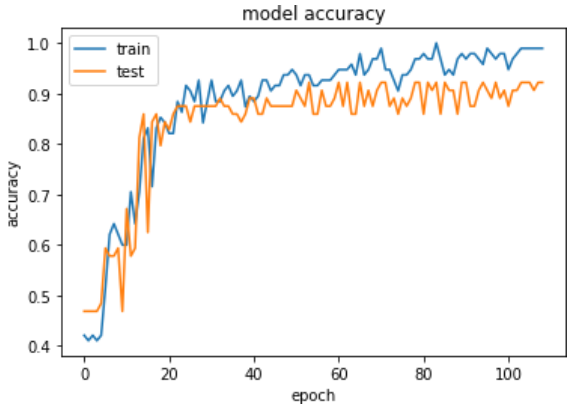
3.2 temporary CNN with an LSTM layer

A temporary model was created with LSTM where it was possible to improve the metrics for both SMIC and CASME II databases; for CASME II an accuracy of 98% was obtained and for SMIC of 92.2%. In theTable 4 You can see the results of the confusion matrix, accuracy and ROC curve for the two databases of facial microexpressions.

Table 4 Result of the temporary CNN model with an LSTM layer.

CASME II [2]	**SMIC [2]**
	
	





4. Conclusions

For the two proposed models, they obtained better prediction results for the CASME II database than for the SMIC database. CASME II is encoded with FACS of 26 participants, while the SMIC database is not. In addition, since the SMIC database did not have FACS coding, the categorization of the emotion labels was left to the participant's own self-information. This may have generated a mislabeling of emotion on the SMIC database.

By adding data augmentation to the set of images on a 3D Convolutional Neural Network, it was possible to improve the metrics for the CASME II database, while for the SMIC database, no improvement was observed.

For the temporal Convolutional Neural Network model with an LSTM layer, which notably improved the prediction of facial microexpressions for both CASME and SMIC databases.

This degree work achieved an important approach to the recognition of facial microexpressions, for the CNN Temporal model with LSTM, as a result of this degree work and the CASME database, the following metrics were obtained: Precision 0.984, Sensitivity 0.962, Accuracy 0.980, F1 Score 0.972 and AUC 0.999. Now the same model for the SMIC database Precision 0. 906, Sensitivity 0. 922, Accuracy 0. 922, F1 Score 0. 912 and AUC 0. 974.

5. Declaration of competing interest

We declare that we have no significant competing interests including financial or non-financial, professional, or personal interests interfering with the full and objective presentation of the work described in this manuscript

6. Author contributions

Garcia Arias, HF Contributed data or analysis tools. Zartha Suarez, N. Wrote the paper

References

[1] X. Li, T. Pfister, X. Huang, G. Zhao and M. Pietikäinen, "A Spontaneous Micro-expression Database: Inducement, collection and baseline," 2013 10th IEEE International Conference and Workshops on Automatic Face and Gesture Recognition (FG), Shanghai, China, 2013, pp. 1-6, doi: 10.1109/FG.2013.6553717.

[2] W.-J. Yan et al., "Casme II: An improved spontaneous micro-expression database and the baseline evaluation", PLoS ONE, vol. 9, no. 1, Jan. 2014.

[3] S. Li and W. Deng, "Deep Facial Expression Recognition: A Survey," in IEEE Transactions on Affective Computing, doi: 10.1109/TAFFC.2020.2981446.

[4] Matlab. (s. f.). Redes neuronales convolucionales. MATLAB & Simulink. Recuperado 28 de septiembre de 2021, de https://es.mathworks.com/discovery/convolutional-neural-network-matlab.html.

[5] cs231n. (s. f.). CS231n Convolutional Neural Networks for Visual Recognition. Recuperado 23 de junio de 2021, de https://cs231n.github.io/convolutional-networks/.

[6] Mañas, A. M. (s. f.). Capítulo 8 Métodos basados en Deep Learning | Notas sobre pronóstico del flujo de tráfico en la ciudad de Madrid. Bookdown. Recuperado 26 de septiembre de 2021, de https://bookdown.org/amanas/traficomadrid/m%C3%A9todos-basados-en-deep-learning.html#lstm-univariado.

[7] A. K. Davison, C. Lansley, N. Costen, K. Tan and M. H. Yap, "SAMM: A Spontaneous Micro-Facial Movement Dataset," in IEEE Transactions on Affective Computing, vol. 9, no. 1, pp. 116-129, 1 Jan.-March 2018, doi: 10.1109/TAFFC.2016.2573832.

[8] Prof. Xiaolan Fu's Group. (2008). Welcome to Professor Fu's Lab. CASME II Database. http://fu.psych.ac.cn/CASME/casme2-en.php.

[9] Cómo detectar mentiras con una máquina y microexpresiones. (2021, February 16). ICHI.PRO. https://ichi.pro/es/como-detectar-mentiras-con-una-maquina-y-microexpresiones-276674219479585

[10] P. Ekman and W. V. Friesen, Facial Action Coding System: A Technique for the Measurement of Facial Movement, Palo Alto, CA, USA:Consulting Psychologists Press, 1978.

[11] University of oulu. (2018). SMIC - Spontaneous Micro-expression Database. SMIC - Spontaneous Micro-Expression Database. https://www.oulu.fi/cmvs/node/41319.

[12] S. P. Teja Reddy, S. Teja Karri, S. R. Dubey and S. Mukherjee, "Spontaneous Facial Micro-Expression Recognition using 3D Spatiotemporal Convolutional Neural Networks," 2019 International Joint Conference on Neural Networks (IJCNN), 2019, pp. 1-8, doi: 10.1109/IJCNN.2019.8852419.

[13] B. (2019). GitHub - bogireddytejareddy/micro-expression-recognition: Spontaneous Facial Micro Expression Recognition using 3D Spatio-Temporal Convolutional Neural Networks. GitHub. https://github.com/bogireddytejareddy/micro-expression-recognition.

[14] Zunair, H. (2020, 23 septiembre). 3D_image_classification_CT. Google Colab. https://colab.research.google.com/drive/14vczr_vjxT9608bgHscaGZhV2e5-xLNQ?usp=sharing.

[15] Google. (s. f.). *Google Colab*. research.google. Recuperado 27 de septiembre de 2021, de https://research.google.com/colaboratory/faq.html?authuser=0&hl=es

[16] TensorFlow. (2021). Retrieved 23 June 2021, from https://www.tensorflow.org/

[17] Team, K. (2021). Keras documentation: About Keras. Retrieved 23 June 2021, from https://keras.io/about/

[18] Evaluation Metrics for Machine Learning - Accuracy, Precision, Recall, and F1 Defined. (2021). Retrieved 6 July 2021, from https://wiki.pathmind.com/accuracy-precision-recall-f1

[19] Cagatay C. *Performance Evaluation Metrics for Software Fault Prediction Studies*. Acta Polytechnica Hungarica. http://acta.uni-obuda.hu//Catal_36.pdf

[20] Arce, J. (2020). La matriz de confusión y sus métricas – Inteligencia Artificial –. Retrieved 6 July 2021, from https://www.juanbarrios.com/la-matriz-de-confusion-y-sus-metricas/

[21] M. Owayjan, A. Kashour, N. Al Haddad, M. Fadel and G. Al Souki, "The design and development of a Lie Detection System using facial micro-expressions," *2012 2nd International Conference on Advances in Computational Tools for Engineering Applications (ACTEA)*, 2012, pp. 33-38, doi: 10.1109/ICTEA.2012.6462897.

[22] Receiver operating characteristic - Wikipedia. (2021). Retrieved 5 July 2021, from https://en.wikipedia.org/wiki/Receiver_operating_characteristic

[23] Google. (s. f.-a). *Classification: ROC Curve and AUC | Machine Learning Crash Course*. Google Developers. Recuperado 28 de septiembre de 2021, de https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc

[24] A. Budhiman, S. Suyanto and A. Arifianto, "Melanoma Cancer Classification Using ResNet with Data Augmentation," *2019 International Seminar on Research of Information Technology and Intelligent Systems (ISRITI)*, 2019, pp. 17-20, doi: 10.1109/ISRITI48646.2019.9034624.

[25] P. Ekman. Facial expression and emotion. *The American Psychologist*, 48(4):384– 392, 1993.

[26] P. Ekman. An argument for basic emotions. *Cognition & Emotion*, 6(3-4):169– 200, 1992.

[27] P. Ekman. Are there basic emotions? Psychological review, 99(3):550–553, 1992.

[28] Chavali, G. K. (2014). *Micro-Expression Extraction For Lie Detection Using Eulerian Video (Motion and Color) Magnication*. DIVA. https://www.diva-portal.org/smash/record.jsf?pid=diva2%3A830774&dswid=-3836
[29] D. Cordaro and P. Ekman. What is meant by calling emotions basic*. Emotion Review*, 3(4):364–370, 2011.
[30] How to Detect Lies: Micro Expressions, https://medium.com/@101/how-to-detect-lies-microexpressions-b17ae1b1181e
[31] Universal Emotions | *What are Emotions?* (2021, February 16). Paul Ekman Group. https://www.paulekman.com/universal-emotions/
[32] Van Edwards, V. (2014). The Definitive Guide to Reading Microexpressions (Facial Expressions). *Scienceofpeople*. Published. https://www.scienceofpeople.com/microexpressions/
[33] Merghani, W. (2018, 7 mayo). *A Review on Facial Micro-Expressions Analysis: Datasets, Features. . .* ArXiv.Org. Recuperado de https://arxiv.org/abs/1805.02397
[34] Agencyworld. (2019, 9 diciembre). *Técnicas para detectar mentiras*. Detectives Privados en Madrid. Recuperado de https://www.agencyworld.org/blog/tecnicas-para-detectar-mentiras
[35] Andrea, P., Cardona, N., & Quintero, M. V. (2016). *Detección de mentiras mediante reconocimiento de patrones faciales utilizando procesamiento digital de imágenes Event Representation in Pre-conceptual Schemas by using Semantic Roles and Mathematical Equations View project Detección de mentiras mediante el reconocimiento de patrones faciales y del discurso usando linguística computacional y procesamiento digital de imágenes View project*. Recuperado de https://www.researchgate.net/publication/303895612
[36] Fita, J. (2016, 2 marzo). *Cómo detectar la mentira a través del lenguaje corporal*. La Vanguardia. Recuperado de https://www.lavanguardia.com/vida/20160301/40123303858/detectar-mentira-lenguaje-corporal.html
[37] P. Ekman (2021, 16 febrero). *Universal Emotions | What are Emotions?* Paul Ekman Group. Recuperado de https://www.paulekman.com/universal-emotions/

Anexo c. Manual técnico Aplicación

Aplicación para el reconocimiento de microexpresión faciales

Manual Técnico

Trabajo de Grado: Desarrollo de un sistema automático de análisis de expresiones faciales para la detección de la mentira en adultos utilizando técnicas de aprendizaje automático.

Universidad Tecnológica de Pereira

Facultad de Ingenieras

Maestría en Ingeniera de Sistemas y Computación

Pereira Risaralda, Colombia

15 de octubre de 2021

Elaborado por: Natalia Zartha Suarez

1. Introduccion

La aplicación se desarrolló sobre el lenguaje de programación Python, las url para acceder a codigo son: en github https://github.com/natzasu13/Facial_Microexpression_Recognition_Keras_frames, y en Google drive para acceder a las carpetas del proyecto completo es https://drive.google.com/drive/folders/13H9WjW3p2jDYNoW-_1HX8bR_lawbkDMl?usp=sharing. El proyecto se llama Facial_Microexpression_Recognition_Keras_frames.

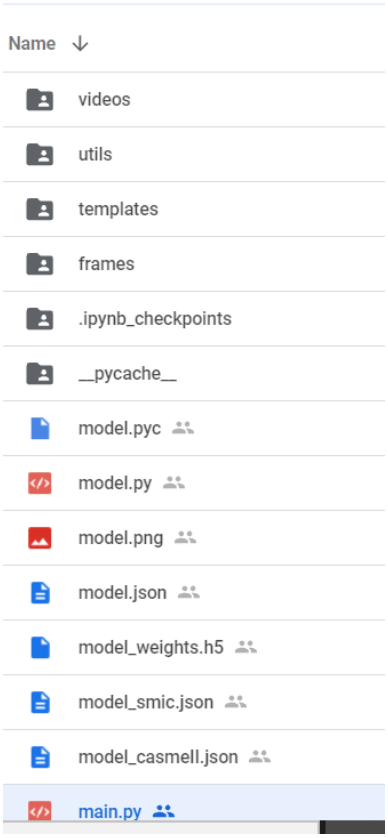
En la siguiente tabla se puede ver las herramientas usadas, seguido de sus versiones.

Herramienta/Lenguaje/Framework	Version
Python	3.6.12
Numpy	1.19.2
OpenCV (cv2)	4.5.1.48
Flask	1.1.2
TensorFlow	2.1.0
Keras	2.3.1
Anaconda3	2020

El proyecto se monto sobre Windows 10, con la ayuda de Anaconda para correr las librerías de TensorFlow.

2. Estructura de la Aplicación

Cuando se abra el link de Google Drive, usted podrá ver la siguiente estructura de archivos:



Donde en la carpeta de Videos, se guarda el video que se va a procesar, en la carpeta Templates es visualizar Flask, la carpeta frames es donde se guardan los 18 frames para milisegundo. Los demás archivos que se encuentran por fuera son principales para que la aplicación corra. A continuación, se muestran los archivos de .py que ejecuta la aplicación

2.1. Main.py

```
from flask import Flask, render_template, Response
from camera import VideoCamera


app = Flask(_name_)

@app.route('/')
```

```
def index():
    return render_template('index.html')

def gen(camera):
    count = 0
    sec = 0
    frameRate = 0.02 #//it will capture image in each 0.02 second
    imageDepth = 18
    frames = []
    images_list = []

    while True:

        count  += 1
        sec = sec + frameRate
        sec = round(sec, 2)

        frame = camera.get_frame(count, sec, frames, images_list)

        if count >= imageDepth :
            count =0
            frames = []
            images_list = []

        yield (b'--frame\r\n'
               b'Content-Type: image/jpeg\r\n\r\n' + frame + b'\r\n\r\n')

@app.route('/video_feed')
def video_feed():
    return Response(gen(VideoCamera()),
                    mimetype='multipart/x-mixed-replace; boundary=frame')

if_name_== '_main_':
    #app.run(host='0.0.0.0', debug=True)
    app.run('127.0.0.1', 4996)
```

2.2. Camera.py

```
3.  import cv2
4.  from model import FacialExpressionModel
5.  import numpy as np
6.
7.
8.  facec = cv2.CascadeClassifier(cv2.data.haarcascades
    +'haarcascade_frontalface_default.xml')
9.  #model = FacialExpressionModel("model.json", "model_weights.h5")
10. #model = FacialExpressionModel("model_smic.json",
    "3d_image_classification_smic.h5")
11. model = FacialExpressionModel("model_casmeII.json",
    "3d_image_classification_casmeII.h5")
12. font = cv2.FONT_HERSHEY_SIMPLEX
13.
14. class VideoCamera(object):
15.     def__init__(self):
16.         self.video = cv2.VideoCapture("videos/facial_exp.mkv")
17.         #self.video = cv2.VideoCapture("videos/presidential_debate.mp4")
18.
19.
```

```
20.     def__del__(self):
21.         self.video.release()
22.
23.     # returns camera frames along with bounding boxes and predictions
24.     def get_frame(self, count, sec, frames, images_list):
25.         image_rows, image_columns, image_depth = 64, 64, 18
26.
27.
28.
29.         self.video.set(cv2.CAP_PROP_POS_MSEC,sec*1000)
30.         hasFrames,fr = self.video.read()
31.         gray_fr = cv2.cvtColor(fr, cv2.COLOR_BGR2GRAY)
32.         faces = facec.detectMultiScale(gray_fr, 1.3, 5)
33.         if hasFrames:
34.             for (x, y, w, h) in faces:
35.                 fc = gray_fr[y:y+h, x:x+w]
36.                 roi = cv2.resize(fc, (48, 48))
37.                 #pred = model.predict_emotion(roi[np.newaxis, :, :, np.newaxis])
38.
39.                 cv2.imwrite('frames/' +"image"+str(count)+".jpg", roi)
40.
41. imageresize = cv2.resize(fc, (image_rows, image_columns),interpolation =
    cv2.INTER_AREA)
42.                 #grayimage = cv2.cvtColor(imageresize, cv2.COLOR_BGR2GRAY)
43.                 frames.append(imageresize)
44.
45.                 if count == image_depth and len(frames) ==  image_depth:
46.                     frames = np.asarray(frames)
47.                     videoarray = np.rollaxis(np.rollaxis(frames, 2, 0), 2, 0)
48.                     images_list.append(videoarray)
49.                     images_list = np.asarray(images_list)
50.                     trainingsamples = len(images_list)
51.
52.                     """
53.                     #version vieja de dimension (1,1,64,64,18)
54. training_set = np.zeros((trainingsamples, 1, image_rows,image_columns,
    image_depth))
55.
56.                     for h in range(trainingsamples):
57.                         training_set[h][0][:][:][:] = images_list[h, :, :, :]
58.
59.                     print(training_set.shape)
60.                     print(training_set[0].shape)
61.
62.                     pred = model.predict_emotion(np.expand_dims(training_set[0],
    axis=0))
63.                     """
64.                     """
65.                     #training_set del data augmentation model
66. #training_set = np.zeros((trainingsamples, image_rows,image_columns,
    image_depth))
67.                     """
68.
69. training_set = np.zeros((trainingsamples, image_depth,image_rows,
    image_columns))
70.                     training_set = np.expand_dims(training_set,axis=0)
71.                     training_set = np.moveaxis(training_set, 0, -1)
72.
73.                     print(training_set.shape)
74.                     #(1, 18, 64, 64, 1)
75.                     pred = model.predict_emotion(training_set)
```

```
76.
77.                     print(pred)
78.
79.                     cv2.putText(fr, pred, (x, y), font, 1, (255, 255, 0),  2)
80.                     cv2.rectangle(fr,(x,y),(x+w,y+h),(255,0,0),2)
81.
82.
83.         _, jpeg = cv2.imencode('.jpg', fr)
84.         return jpeg.tobytes()
85.
```

2.3. Mode.py

```
86. from tensorflow.keras.models import model_from_json
87. import numpy as np
88. import tensorflow as tf
89. from tensorflow.keras import layers
90. from tensorflow import keras
91.
92. from tensorflow.keras.models import Sequential
93. from tensorflow.keras.layers import TimeDistributed
94. from tensorflow.keras.layers import Convolution2D, MaxPooling2D, Flatten, Reshape
95. from tensorflow.keras.layers import Input ,Dense, Dropout, Activation, LSTM
96.
97. # Just disables the warning, doesn't take advantage of AVX/FMA to run faster
98. import os
99. os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2'
100.
101.     config = tf.compat.v1.ConfigProto()
102.     config.gpu_options.per_process_gpu_memory_fraction = 0.15
103.     session = tf.compat.v1.Session(config=config)
104.
105.     class FacialExpressionModel(object):
106.
107.         #EMOTIONS_LIST=["Angry", "Disgust", "Fear", "Happy", "Neutral", "Sad",
    "Surprise"]
108.
109.         #SMIC
110.         #EMOTIONS_LIST= ["Negativo", "Positivo","Sorpresa"]
111.
112.         #CASME II
113.         EMOTIONS_LIST= ["Angry", "Happy","Disgust"]
114.
115.         def __init__(self, model_json_file, model_weights_file):
116.
117.             """
118.             with open(model_json_file, "r") as json_file:
119.                 loaded_model_json = json_file.read()
120.                 self.loaded_model =
    tf.keras.models.model_from_json(loaded_model_json)
121.             """
122.             self.loaded_model = self.get_model(width=64, height=64, depth=18)
123.
124.
```

```
125.             self.loaded_model.load_weights(model_weights_file)
126.             self.loaded_model._make_predict_function()
127.
128.
129.         def predict_emotion(self, img):
130.             self.preds = self.loaded_model.predict(img)
131.             return FacialExpressionModel.EMOTIONS_LIST[np.argmax(self.preds)]
132.
133.         def get_model(self, width=64, height=64, depth=18):
134.             num_frames = 18
135.             num_classes = 3
136.
137.             model = Sequential()
138.
139.             model.add(
140.                 TimeDistributed(
141.                     Convolution2D(64, (3, 3), activation='relu'),
142.                     input_shape=(num_frames, width, height, 1)
143.                 )
144.             )
145.             model.add(TimeDistributed(MaxPooling2D((2, 2), strides=(1,  1))))
146.
147.             model.add(TimeDistributed(Convolution2D(128, (4,4),
    activation='relu')))
148.             model.add(TimeDistributed(MaxPooling2D((2, 2), strides=(2, 2))))
149.
150.             model.add(TimeDistributed(Convolution2D(256, (4,4),
    activation='relu')))
151.             model.add(TimeDistributed(MaxPooling2D((2, 2), strides=(2, 2))))
152.
153.             # extract features and dropout
154.             model.add(TimeDistributed(Flatten()))
155.             model.add(Dropout(0.5))
156.
157.             # input to LSTM
158.             model.add(LSTM(256, return_sequences=False, dropout=0.5))
159.
160.             # classifier with sigmoid activation for multilabel
161.             model.add(Dense(num_classes, activation='softmax'))
162.
163.             return model
164.
165.
166.
```