PONTIFICIA UNIVERSIDAD CATOLICA DEL ECUADORFACULTAD

DE INGENIERÍA

ESCUELA DE SISTEMAS Y COMPUTACIÓN

DISERTACIÓN PREVIA A LA OBTENCIÓN DEL TÍTULO DE

INGENIERO DE SISTEMAS Y COMPUTACIÓN

“Mejora del rendimiento y la interpretabilidad en el reconocimiento deunidades de acción facial con redes neuronales profundas”

AUTOR

Felipe Marcelo Serrano Flores

Quito, 2021

Agradecimientos

Todos estos años, tuve el inmenso placer de formar parte del grupo de la muerte. Me gustaría dar las gracias a todos mis colegas, por su espíritu, presencia, humor y largos días (y a veces noches) de trabajo duro. Eran verdaderos compañeros. Gracias, David Almachi, Luis Benavides, Edwin Ortega, David Maldonado, Bryan Morales. Gracias Nelson por su compromiso excepcional, su perspicacia y gran liderazgo. Cada vez que tendré que construir algo donde nada estaba. Creíste mucho. A mi madre es una verdadera inspiración para mí. Porúltimo, pero no menos importante, me gustaría dar las gracias a mis queridos amigos y familiares.

“La cara es el espejo del alma, y los ojos, sin hablar, revelan los secretos del corazón.”

San Jerónimo

Contenido

RESUMEN

Esta disertación tiene las siguientes contribuciones principales: a) Definición de taxonomíacompleta de los enfoques automáticos de visión artificial para el reconocimiento automático de expresiones faciales seguido de usar una red neuronal recurrente, una revisión extendida delas tendencias históricas y actuales en reconocimiento automático de las expresiones faciales.

b) Propuesta de un modelo que aprende la representación, parche y estructura de salida de la cara de extremo a extremo c) Introducción de una topología de inferencia de estructura que replica el algoritmo en modelos gráficos probabilísticos d) Estudio de ablación ampliado y análisis experimental de la arquitectura recién propuesta e) Análisis y mejora del rendimiento de la arquitectura previamente propuesta para la arquitectura de expresión facial utilizando el nuevo marco teórico. f) Formulación de un nuevo marco general para el análisis de redes neuronales profundas basadas en topología algebraica g) Análisis de diferencias topológicas fundamentales entre redes neuronales profundas que aprenden y redes neuronales profundas que memorizan.

Justificación

Las expresiones faciales son formas vitales de comunicación entre los seres humanos en contextos sociales. Se utilizan como marcadores conversacionales y transmiten información sobre el estado afectivo y cognitivo. Muchas aplicaciones se beneficiarían del avance del Reconocimiento Automático de la Experiencia Facial (RAEF). Un Robusto RAEF mejoraría la interacción humano-ordenador, aumentaría la seguridad de conducción. En los últimos años sehan realizado avances significativos en RAEF con el uso de Redes Neuronales Profundas (RNP). Desafortunadamente, este aumento en el rendimiento vino con el aumento de la opacidad. El estado actual de las RNP como modelo de "caja negra" dificulta el avance del campo. En esta disertación, se propone un nuevo marco general para analizar las redes neuronales profundasbasadas en el estudio sistemático de su topología mientras son patrones de aprendizaje en los datos. Se utiliza este marco para estudiar una RNP recién propuesta, especialmente diseñadapara el reconocimiento de la Unidad de Acción, lo que se traduce en una mejor comprensión, control y un mayor rendimiento.

El estudio acerca de redes neuronales profundas puede ver la mejora que tiene con cada algoritmo, y como este actúa para diferentes casos. Esto permitirá ver con que algoritmo se tiene la seguridad de tener mayor efectividad frente a otros algoritmos.

Planteamiento del problema

En los últimos años se ha realizado un avance significativo en RAEF con el uso de RNP, Lamentablemente, este aumento en el rendimiento vino con unamayor opacidad. El estado actual de los RNP como modelo de "caja negra" dificulta el avancedel campo. En esta disertación, se propone un análisis general para analizar redes neuronalesprofundas basado en el estudio sistemático de su topología mientras aprenden patrones en los datos.

La cara es un componente fundamental de la identidad humana. Al mirar la cara de alguien, loshumanos pueden saber casi instantáneamente si conocen a la persona, su género y edad, dóndeestá mirando y si sienten dolor o se divierten.

Un algoritmo robusto cualquiera, en general, tendría que aprender qué información relevante considerar y qué ignorar para tomar una decisión. Estos problemas son comunes en cualquier tipo de tareas de reconocimiento de patrones, ya sea de imágenes, audio o cualquier otro tipo de señal.

Sin embargo, se necesita más trabajo a este respecto. Hasta hace poco, la anotación manual deexpresiones fáciles era el único medio para medir la expresión facial de la emoción. Debido a que la anotación manual es laboriosa, la replicación de los estudios es limitada.

De esto se pueden identificar el siguiente problema principal:

- El acceso a la información sobre las redes neuronales profundas y su aplicación que eneste caso es para reconocimiento de acciones faciales es limitado.

Y los siguientes problemas secundarios:

- La documentación sobre el manejo de algoritmos es un problema en la actualidad.
- Se ha buscado documentación a nivel local, es un tema nuevo y esta es la motivación para realizar el tema de investigación.

En función de lo expuesto como problemática se plantean la siguiente pregunta de investigación principal:

- ¿Cómo aprenden las redes neuronales profundas para identificar las acciones facialesde una persona?

Y las siguientes preguntas de investigación secundarias:

- ¿Qué es la taxonomía en redes neuronales profundas?
- ¿Cuáles son los trabajos relacionados a redes neuronales profundas?
- ¿Cuáles son las tendencias de redes neuronales profundas?
- ¿Cómo se optimizaría el aprendizaje por parte de las redes profundas para el reconocimiento de acciones faciales?

Objetivos

General

Obtener la comprensión de lo que significa aprender en redes neuronales profundas en unidadesde acción facial, ya que las redes neuronales profundas son cada vez más opacas a medida quecrecen en rendimiento, complejidad y proponer un marco general, aplicable a una miríada de problemas.

Específicos

- Analizar la precisión de ejecución del programa con mejora con el que no lo tiene.
- Desarrollar el código de entrenamiento de la red neuronal con una mejora juntamente con el código para la ejecución del programa.
- Examinar el aprendizaje de Unidades de Acción Facial con la Red de Inferencia de Estructura Profunda.
- Interpretar los algoritmos de aprendizaje profundo que reconocen expresiones facialesa partir de imágenes.

CAPITULO 1: Marco Teórico

1.1 **Antecedentes**

Las redes neuronales, aunque tiene ya tres décadas de ser descubiertas, en los últimos 5 años es donde la investigación ha avanzado, revolucionando varios sectores de la sociedad.

El Sistema de Unidad de Acción Facial (FACS) (Paul Ekman, 2002). es un esquema de codificación descriptivo de EF que se centra en lo que puede hacer la cara sin asumir ningún valor cognitivo o emocional. Sus componentes básicos se denominan Unidades de Acción (UA) y se combinan para formar una representación completa de las EF.

Las redes neuronales profundas utilizadas en el reconocimiento de objetos generalmente aprenden a extraer entidades de bajo nivel similares a Gabor en las primeras capas, combinan estas entidades de bajo nivel para representar partes de un objeto en capas de nivel medio y, finalmente, las combinan para generar una representación muy compleja de objetos que son invariables a los cambios de imagen, por ejemplo, pose, iluminación y transformaciones afines.(Zhou, Bau, Oliva y Torralba, 2018) propuso un marco de aprendizaje disperso multitarea para el aprendizaje de parches discriminatorios comunes y específicos para diferentes expresiones. La ubicación del parche estaba predefinida y no tenía en cuenta la geometría facial.

Con el paso de los años las mejoras de la tecnología han logrado avances en redes neuronales a inicios de 1943 McCulloch & Pitts crean un modelo computacional de la neurona umbral. Pero no es hasta el año de 1989 los científicos logran crear un algoritmo capaz de usar redes neuronales profundas, sin embargo, para tener resultado en lo que se quería analizar el algoritmo se demoraba días, siendo inutilizable para la vida real. En la actualidad mediante el estudio de los algoritmos para redes neuronales y con recursos más avanzados en tecnología se logra un desempeño mayor y de esta manera resolver problemas con mayor

rapidez. (Busy, 2020)

El aprendizaje profundo no quiere decir que las computadoras piensen como los humanos, sino que, mediante grandes cantidades de datos, procesadores, y algoritmos, ahora las computadoraspueden desarrollar lo que antes solo era imprescindible por los humanos como son: reconocimiento de imágenes, voz, tomar decisiones, entre otras aplicaciones. (Busy, 2020)

Las expresiones faciales (EF) son sistemas de señalización vitales de un efecto, que transmitenseñales sobre el estado emocional de las personas. Junto con la voz, el lenguaje, las manos y lapostura del cuerpo, forman un sistema fundamental de comunicación entre humanos en contextos sociales. El reconocimiento automático de EF es un dominio interdisciplinario que se encuentra en el cruce de la ciencia del comportamiento, la neurología y la inteligencia artificial.

Los estudios de la cara fueron muy influidos en los tiempos premodernos por las teorías populares de la fisonomía y el creacionismo. La fisonomía supone que el carácter o la personalidad de una persona se puede juzgar por su apariencia externa, especialmente la cara.(Richard Wiseman, 2020)

1.2 Alcance

El tema propuesto está enfocado en mejorar el rendimiento aplicando los algoritmos mejorados de expresión facial a partir de imágenes utilizando una red descrita en el algoritmo que se utiliza como base para mejorar en el rendimiento del análisis del rostro facial y al mismo tiempo obtener una comprensión sobre cómo estos modelos aprenden.

De acuerdo con los objetivos específicos se logró cumplir cada punto.

El trabajo de titulación termina con el desarrollo del programa que utiliza el algoritmo mejorado para validar expresiones faciales y la documentación técnica.

1.1. Introducción

La forma facial tiene una gran diversidad y singularidad entre los seres humanos. La morfología específica del cráneo y de los tejidos blandos, músculos, grasa y piel que lo cubren,es única para cada persona y cambia sólo muy lentamente con la edad. Pero además de su formarígida que transmite información esencial sobre la identidad humana, los rostros también son altamente morfodinámico siendo un transportador vital de señales sociales. Los rostros son muy importantes en la forma en que se comunica e imagina las acciones faciales y se piensa en la emoción. Por ejemplo, una de las formas más utilizadas de representarla emoción humana en el arte es a través de expresiones faciales (ver Figura 1.)

Figura 1 Experimentos sobre cómo se producen las expresiones faciales

Fuente: (Boulogne, 2016)

Al mirar la cara de alguien, la mayoría de las veces se dice si esa persona está triste o feliz, si está en duda, si lo que dicen es irónico o grave o si puede estar sufriendo. En general, la cara es una fuente muy rica de información afectiva y cognitiva y un importante canal de comunicación. Las expresiones faciales son una de las primeras habilidades de coordinación motora que los humanos poseen. Reconocer las expresiones faciales es esencial para poder decir lo que los demás sienten y piensan e influir en sus acciones, que tiene gran importancia para la supervivencia como se muestra en la figura 2.

Figura 2 Esculturas de Franz Messerschmidt, artista del siglo XVIII

Fuente: (Fahrenheit, 2020)

Esta increíble capacidad humana se da en gran medida por sentada. Rara vez se piensa cómo es posible ser capaces de reconocer tantas expresiones faciales diferentes con tanta precisión y robustez. Pero en cuanto a muchas otras capacidades cognitivas, su complejidad sólo se hace evidente cuando se trata de replicarlos en un dispositivo de ingeniería humana como una computadora. *Cómo un ordenador podría replicar estas capacidades increíbles con mayor* rendimiento *y mayor transparencia.*

Se imagina por un segundo un almuerzo con amigos. En un momento dado, uno de ustedes toma su teléfono inteligente, toma una foto y la comparte en alguna red social. Tal vezla foto se parece a la figura 3 lo que todos los demás verán es una buena imagen con un montónde queridos amigos alrededor de una mesa llena de buena comida. Un observador más interesado inferirá una increíble cantidad de cosas con sólo echar un vistazo a esta foto. Ellos reconocerán que usted está compartiendo una ensalada y la mayoría de ustedes están tomando

una copa de vino. Se darán cuenta de qué tipo de ropa usas y combinada con la fuerte luz que entra por la ventana probablemente pensarán que el clima es cálido. Por el fondo, también pensarán que estás en la sala de estar de alguien, no en un restaurante y el contexto es informal. Luego, mirando a las personas, fácilmente podrán decir que hay tres mujeres y

tres hombres alrededor de la mesa y que tienen alrededor de 30 años. Más atención que prestarían a los rostros de todas las señales sociales se volverán evidentes. Uno puede decir dónde está mirando la gente, si está a punto de decir algo y con quién están interactuando.

Y por las expresiones faciales, también se podrían inferir estados cognitivos y afectivos y marcadores deconversación. Por ejemplo, que algunos son un poco divertidos, otros un poco impresionados y al menos un tipo trata de actuar falsamente disgustado (Khomami Abadi, y otros, 2015).

Estrictamente hablando, la imagen en sí misma de una expresión facial es sólo una colección de números con una estructura específica similar a una cuadrícula (véase la Figura 4). El enfoque principal de la visión por computadora es tratar de alcanzar un nivel más alto sin soportar las imágenes. Esto se puede utilizar para el reconocimiento de objetos, el análisis de escenas, la segmentación semántica o el reconocimiento de expresiones faciales a partir deesta sencilla colección de números representados por una imagen. El enfoque está en el Reconocimiento Automático de Expresión Facial (RAEF) a partir de imágenes.

En los desafíos que puede tener la IA al reconocer los diferentes rasgos faciales, probablemente la gran mayoría de los humanos, si se les pregunta, estarían de acuerdo en que el hombre de la izquierda y la dama rubia de la derecha son un poco divertidos (ver figura 3). Pero estos son los resultados de un complejo algoritmo de reconocimiento de patrones construido en el cerebro. Pero las colecciones de números contenidas en las fotos serían muy diferentes de cara a cara debidoa diferentes identidades, diferentes poses y diferentes oclusiones o estilos de cabello. Un algoritmo robusto tendría que aprender qué información relevante considerar y qué ignorar para tomar una decisión. Estos problemas son comunes en cualquier tipo de tareas de reconocimiento de patrones, ya sea a partir de

imágenes, audio o cualquier otro tipo de señal.

Figura 3 Diferentes rasgos faciales en comida de amigos

Fuente: (Shutterstock, 2020)

A finales de los años 40, un modelo muy poderoso, llamado Deep Neural Networks (RNP), logró un rendimiento sin precedentes en un sin fin de tareas, generalmente cuando hay una gran cantidad de datos disponibles. Desafortunadamente, al mismo tiempo estos modelos de aprendizaje, a medida que se volvieron más poderosos también se volvieron más complejos y opacos. Las redes neuronales no son una idea nueva. La intuición a los diseñadores de un aprendizaje de un algoritmo que debe buscar inspiración para el cerebro de los mamíferos era una forma central de pensar desde el comienzo de la Inteligencia Artificial.

Científicos e innovadores se centraron en modelos matemáticos muy simplificados quedescriben un comportamiento de neurona biológica. Se demostró teóricamente que, dada la escala correcta, las neuronas simplificadas, cuando se juntan son capaces de aproximar cualquier tipo de función, no importa cuán compleja. Desafortunadamente, esto plantea un problema importante. Aunque el código para especificar la arquitectura y el entrenamiento deun modelo puede ser simple, los resultados pueden ser muy complejos, efectivamente dando lugar a "cajas negras". Cuando una RNP mira una imagen y decide que contiene la cara de unapersona que se ve "feliz" o "triste" (ver Figura 4.), los procesos funcionales exactos que generanestas salidas son difíciles de interpretar incluso para los científicos que generaron

los algoritmos en primer lugar, aunque se está haciendo algún progreso en la interpretación. Finalmente se termina con una situación paradójica, en la que se tiene modelos que funcionan bien pero realmente no se sabe cómo y por qué. Por lo tanto, se interesa en mejorar el rendimiento en RAEF a partir de imágenes que utilizan RNP y al mismo tiempoobtener comprensión sobre cómo estos modelos aprenden.

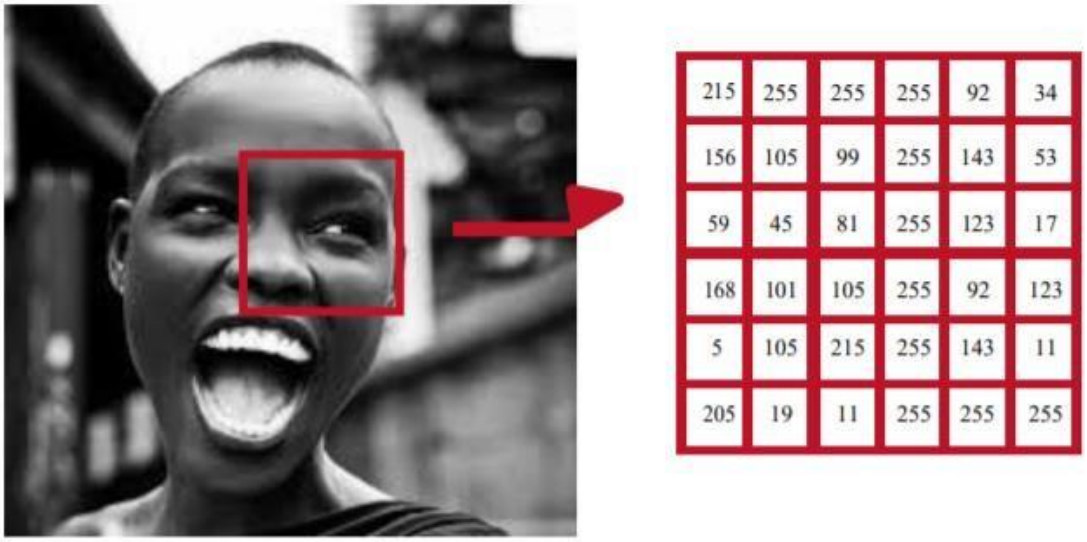


Figura 4 Diferentes rasgos faciales en comida de amigos

Fuente: (Pinterest, 2020)

1.2. Motivación y objetivos

La motivación es la de tener un enfoque hacia las redes neuronales profundas y tener un mejoramiento en los algoritmos como se muestra en la figura 5. Se reafirma aquí junto con los principales objetivos que, en consecuencia, se derivan. Todo el manuscrito está estructurado en estas líneas:

Las expresiones faciales (FE) son sistemas vitales de señalización de afecto, transmitiendo indicaciones sobre el estado emocional y cognitivo de los seres humanos.

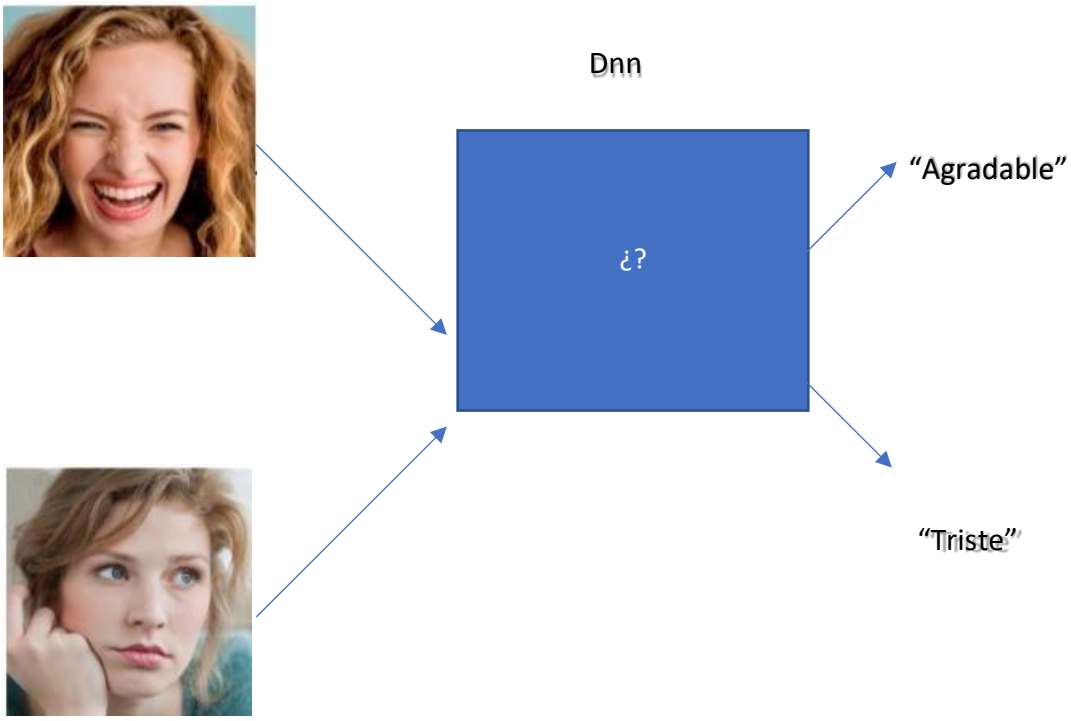


Figura 5 Muestra la interrogativa de como aprende las redes neuronales profundas para que dé un resultado

Fuente: (MasterFile,2020)

Las redes neuronales profundas (RNP) son cada vez más opacas a medida que crecen en rendimiento y complejidad. Se consideran modelos de "caja negra" (Castelvecchi, 2016). Para un mejor rendimiento, usabilidad y aceptación del público en general es importante obtener comprensión de lo que significa aprender en redes neuronales profundas. Si bien este es un objetivo general y se propone un marco general, aplicable a una miríada de problemas, se centra en la atención en la motivación inicial y objetivo, a saber,en lo que significa aprender expresiones faciales en redes neuronales profundas y cómopodría utilizarse para un mayor rendimiento.

1.3. **Contribuciones y Esquema**

Se presenta varias contribuciones al problema del reconocimiento de la expresión facial. Se agrupan en tres categorías principales: contribuciones relacionadas con la taxonomía del reconocimiento facial de expresión, contribuciones a la mejora del rendimiento del reconocimiento automático de la expresión facial y contribuciones relacionadas con la interpretabilidad de las redes neuronales profundas. Hay una correspondencia directa entre esta categorización y el esquema de este manuscrito, por ello las contribuciones son las siguientes:

- Expresión Facial Automática. Marco General, Perspectiva y Tendencias Evolucionarias. Se define una taxonomía integral de enfoques automáticos de visión computarizada para el reconocimiento automático de la expresión facial (RAEF). La definición y las opciones de los componentes diferentes se analizan y discuten. Esto secomplementa con una sección dedicada a la evolución histórica de los enfoques y un análisis en profundidad de las últimas tendencias. Además, se proporciona una introducción a la inferencia de efectos desde el rostro de la cara a través de una perspectiva evolutiva. Se hace hincapié en la investigación producida desde la última revisión importante de RAEF en 2009. El enfoque en inferir el efecto, definir una taxonomía integral y tratar diferentes modalidades tiene como objetivo proponer una perspectiva más general sobre RAEF y sus tendencias actuales.

Las principales contribuciones son:

  a. Una perspectiva evolutiva de la inferencia de afectación desde el rostro
  b. Definición de taxonomía integral de enfoques automáticos de visión computarizada para el reconocimiento automático de la expresión facial
  c. Encuesta ampliada de las tendencias históricas y actuales en RAEF

- Rendimiento en Reconocimiento de Expresión Facial. Se propone una arquitectura

profunda neural que aborda el aprendizaje de la representación local y el aprendizaje de la estructura de clases combinando características locales y globales aprendidas en sus etapas iniciales y replicando un algoritmo de paso de mensajes entre clases similar a un enfoque de inferencia de modelo gráfico en etapas posteriores. Se demuestra queal entrenar el modelo de extremo a extremo con una mayor supervisión se mejora el reconocimiento de la unidad de acción facial de última generación.

- Las principales contribuciones son:
    a. Propuesta de un modelo que aprende la representación, el parche y la estructurade salida de la cara de extremo a extremo
    b. Introducción de una topología de inferencia de estructura que replica el algoritmo de inferencia en modelos gráficos probabilísticos mediante una red neuronal recurrente
    c. Estudio de ablación ampliado y análisis experimental de la arquitectura recién propuesta
- Interpretabilidad en reconocimiento de expresión facial. Se deriva un enfoque novedoso para definir lo que significa aprender en redes profundas. Se muestra cómo esto define la capacidad de una red para generalizar a muestras de prueba no vistos y, lo que es más importante, por qué este es el caso. Más concretamente, hay una contribución principal:
    a. Formulación de un nuevo marco general para el análisis de redes neuronales profundas basadas en la topología algebraic

CAPÍTULO 2: METODOLOGÍA DE INVESTIGACIÓN Y DESARROLLO

2.1. Metodología de investigación

La propuesta considera los siguientes pasos:

a) Inicio

b) Identificar el problema a resolver

¿Cuál es el problema que pretende resolver?

El mejoramiento de algoritmo mediante redes neuronales mediante el aprendizaje de históricos y dando giros a la imagen.

c) Configuración del equipo

Se debe identificar si se tiene tarjeta de video, y el número de procesadores, se actualiza la tarjeta de video y los drivers para no tener problema alguno, así como descargar Python y en la distribución Anaconda.

d) Prueba del equipo

Si el equipo fue configurado exitosamente, y se tiene correctamente instalado Python y las librerías requeridas como son Numpy, Pyplot.

e) Implementar el algoritmo

La implementación del algoritmo en particular puede ser una tarea consumidora de tiempo, pero gracias a la tarjeta de video y los procesadores esta tarea no puede ser tan larga, se debe revisar los manuales de PyOpenCL.

f) Evaluar los resultados

Se verifica que los resultados sean correctos esto quiere decir que exista una mejora del algoritmo y si no se pasa al punto anterior que es el paso e)

g) Fin

En la figura 6 se muestra el diagrama de flujo de cómo funciona la metodología de investigación.

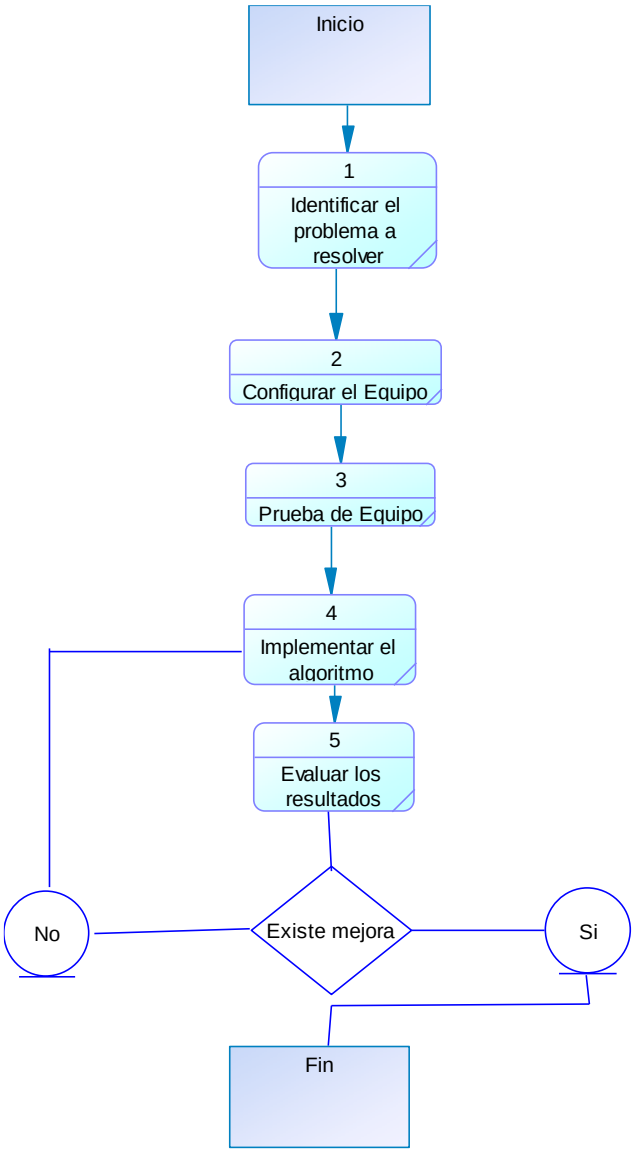


Figura 6 Diagrama de flujo sobre la metodología de investigación

2.2. Kanban

Con Kanban se diferencia de otras metodologías ágiles en que se tiene calidad garantizada, esto quiere decir que se debe realizar todo con calidad a la primera, para ahorrar el tiempo de volver, con esto no se prioriza en tiempo si no la calidad del documento y aplicación.

Reducción de desperdicio esto quiere decir que se va a utilizar lo justo y necesario sin desperdiciar nada.

Flexibilidad según las tareas que son prioritarias se las puede añadir en el backlog por orden de prioridad que sea conveniente.

Existen diferentes tipos de sistemas Kanban, el que se va a usar se llama Tablero Kanban como se muestra en la figura 7.

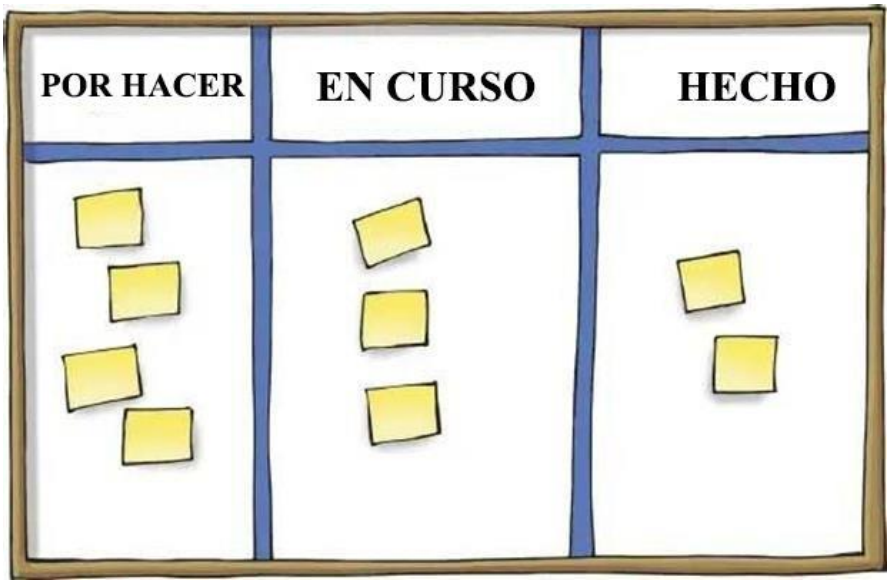


Figura 7 Fases del tablero Kanban.

El tablero Kanban es usado por muchas empresas, el cual contiene tres columnas, dependiendode la empresa, pueden ser más, por ejemplo: por hacer, en curso y hecho.

Las cartas deben estar puestas en el lugar donde se va a realizar, se usa por prioridad, las que están arriba del todo, se las debe realizar primero, aunque dependiendo de la empresa, solo lo usan sin prioridad.

2.3. Cinco prácticas básicas de Kanban

2.3.1. Visualiza el trabajo

Usa el tablero para representar el flujo de trabajo, las columnas para representar los ajustesen este proceso y enumera las tareas relevantes de los elementos de trabajo. (Jaramillo, 2017)

2.3.2. Limitar el trabajo en proceso

Ser más eficaz y conseguir más haciendo menos. Delegue solo el número ideal de tareas asu equipo. No haga más trabajo hasta que su equipo complete algo. (Jaramillo, 2017)

2.3.3. Administrar el flujo

Comience donde está, con el proceso existente, revise dónde se encuentran las ineficienciasy mejore continuamente. Aprenda de sus errores y evite que vuelvan a producirse cuellos de botella. (Jaramillo, 2017)

2.3.4. Hacer explicitas las políticas de gestión

Defina todo el proceso y asegúrese de que todos comprendan cómo funcionan las cosas y cuál es su objetivo. Por ejemplo, defina los criterios que debe cumplir cada paso del procesopara denominar "terminado". (Jaramillo, 2017)

2.3.5. Mejorar colaborativamente

Cuando su equipo comprende claramente las teorías sobre el trabajo, los procesos y el flujode trabajo, es más probable que intercambien sus ideas y sugieran acciones de mejora. (Jaramillo, 2017)

En resumen, Kanban sirve para lo siguiente:

- Poder empezar cualquier operación estándar en cualquier momento.
- Dar instrucciones basados en las condiciones actuales del área de trabajo.
- Prevenir que se agregue trabajo innecesario a aquellas ordenes ya empezadas y prevenirel exceso de documentación innecesario.

CAPÍTULO 3: DESARROLLO DE LA APLICACIÓN PARA EL RECONOCIMIENTO FACIAL

3.1. Materiales

Se usa las últimas versiones del lenguaje de programación Python 3.8.5 y la librería TensorFlow. A continuación, se explicará las librerías a usar, las cuales es de ayuda para poder efectuar adecuadamente el reconocimiento facial.

3.2. Librerías para usar

3.2.1. NumPy

Es el paquete fundamental para el cálculo numérico y define la matriz numérica y los tipos de matriz y las operaciones básicas en ellos.

NumPy array es un objeto de matriz N-dimensional, en forma de filas y columnas con respectiva ubicación en memoria. cada elemento ocupa el mismo tamaño de bloque de memoria, y todos los bloques se interpretan inexacto de la misma manera. La forma en que se va a interpretar cada elemento de la matriz se especifica mediante un elemento independiente objeto de tipo de datos, uno de los cuales está asociado a cada matriz. Además de los tipos básicos (enteros, flotantes, etc.), los objetos de tipo de datos también pueden representar estructuras de datos.

3.2.2. Matplotlib

Matplotlib es un paquete de gráficos para visualización de datos en Python. Matplotlib ha surgido como un componente clave en la pila de ciencia de datos de Python y está bien integrado con NumPy y pandas. El módulo pyplot refleja de cerca los comandos de trazado de MATLAB. Por lo tanto, los usuarios de MATLAB pueden pasar fácilmente al trazado con Python. Seaborn, por otro lado, amplía la biblioteca Matplotlib para crear gráficos con Python utilizando un conjunto de métodos más sencillo. Seaborn es más integrado para trabajar con

Pandas DataFrames. Se pasa por la creación simple parcelas imprescindibles con Matplotlib y SeaBorn.

3.2.3. TensorFlow

En esencia, TensorFlow es una biblioteca para operaciones de arreglos multidimensionales eficientes con un enfoque en el aprendizaje profundo. Desarrollado por Google Brain Team, TensorFlow fue de código abierto el 9 de noviembre de 2015. Y aumentado por su conveniente capa API de Python, TensorFlow ha ganado mucha popularidad y una amplia adopción en la industria, así como en la academia.

TensorFlow comparte algunas similitudes con NumPy, como proporcionar estructuras de datos y computaciones basadas en matrices multidimensionales. Lo que hace que TensorFlow particularmente sea adecuado para el aprendizaje profundo, sin embargo, son sus primitivos para definir funciones en tensores, la capacidad de paralelizar las operaciones tensores y herramientas de conveniencia como la diferenciación automática.

Mientras que TensorFlow se puede ejecutar completamente en una CPU o varias CPU, una delas principales fortalezas de esta biblioteca es su soporte de GPU que son muy eficientes en el rendimiento de cálculos numéricos altamente paralelos. Además, TensorFlow también es compatible con sistemas distribuidos, así como plataformas de computación móvil, incluyendo Android y iOS de Apple.

En resumen, es la API para definir modelos de aprendizaje automático, entrenarlos con datos y exportarlos para su uso posterior. Se accede a la API principal a través de Python, mientras que el cálculo real se escribe en C++. Esto permite a los científicos e ingenieros de datos utilizar un entorno más fácil de usar en Python, mientras que el cálculo real se realiza con código C++ rápido y compilado. Hay una API de C++ para ejecutar modelos de TensorFlow, pero está limitada en este momento y no se recomienda para la mayoría de los usuarios. (Scarpino, 2018)

Pero ¿qué es un tensor? En términos simplificados, se piensa en tensores como matrices multidimensionales de números, como una generalización de escalares, vectores y matrices. Cuando se describe tensores, se refiere a sus "dimensiones" como el rango (u Orden) de un tensor, que se debe confundir con las dimensiones de una matriz. por ejemplo, se tiene m, n matriz dónde m es el número de filas y n es el número de columnas figura 8.

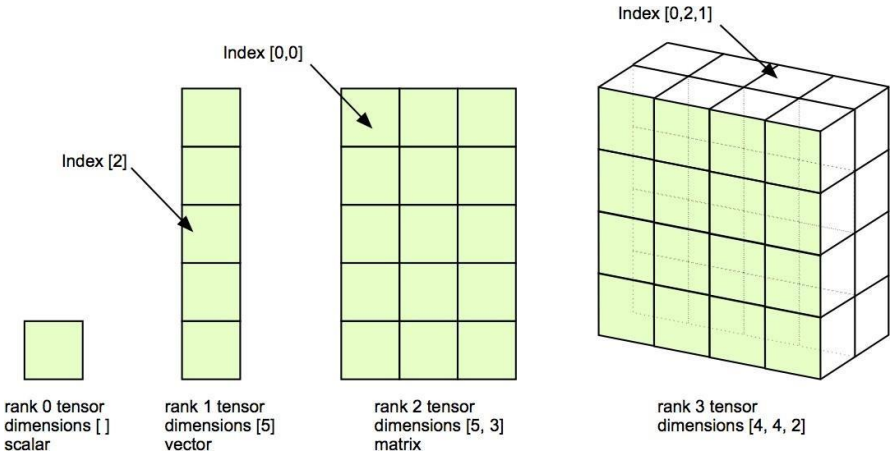


Figura 8 Rangos en tensores

Fuente: (Cheatsheet,2020)

CAPÍTULO 4: RESULTADOS RECONOCIMIENTO FACIAL

4.1 Descripción de los datos

A continuación, se muestra la cantidad de imágenes que la red neuronal uso para el entrenamiento y su clasificación: Enojado, Disgustado, Temeroso, Feliz, Neutral, Triste, Sorprendido.

4.1.1. Prueba

Los datos de prueba son:

Enojado: 958 imágenes

Disgustado: 111 imágenes

Temeroso: 1024 imágenes

Feliz: 1774 imágenes

Neutral: 1233 imágenes

Triste: 1247 imágenes

Sorprendido: 831 imágenes

4.1.2. Entrenar

Los datos para entrenar son:

Enojado: 3995 imágenes

Disgustado: 436 imágenes

Temeroso: 4097 imágenes

Feliz: 7215 imágenes

Neutral: 4965 imágenes

Triste: 4830 imágenes

Sorprendido: 3171 imágenes

4.2. **Proceso**

4.2.1. Entrenamiento del modelo sin mejora:

Con las siguientes librerías nympy, argparse, matplotlib.pyplot, tensorflow.keras.models, tensorflow.keras.layers, tensorflow.keras.optimizers, tensorflow.keras.layers, tensorflow.keras.preprocessing.image (Team, 2021) se describe el siguiente código mejorando la aplicación del algoritmo:

- import numpy as np
- import argparse
- import matplotlib.pyplot as plt
- import cv2 (Numpy, 2021)
- from tensorflow.keras.models import Sequential
- from tensorflow.keras.layers import Dense, Dropout, Flatten
- from tensorflow.keras.layers import Conv2D
- from tensorflow.keras.optimizers import Adam
- from tensorflow.keras.layers import MaxPooling2D
- from tensorflow.keras.preprocessing.image import ImageDataGenerator (Team, 2021)

Se definió una función para la traza de curvas de precisión y pérdida

- def plot_model_history(model_history):

En esta parte se guarda un histórico, con el que se llama cada vez, para poder realizar un nuevo histórico y mejorar el algoritmo.

- fig, axs = plt.subplots(1,2,figsize=(15,5)) (Subplots, 2021)

Se resume el historial para tener mayor precisión cada que se lo llama.

- “axs[0].plot(range(1,len(model_history.history['accuracy'])+1),model_history.history['accuracy'])
- axs[0].plot(range(1,len(model_history.history['val_accuracy'])+1),model_history.history['val_accuracy'])
- axs[0].set_title('Model Accuracy')
- axs[0].set_ylabel('Accuracy')
- axs[0].set_xlabel('Epoch')
- axs[0].set_xticks(np.arange(1,len(model_history.history['accuracy'])+1),len(model_history.history['accuracy'])/10)
- axs[0].legend(['train', 'val'], loc='best') (parneetk/parneetk.github.io, 2021)

En esta parte se resume el historial de perdida

- axs[1].plot(range(1,len(model_history.history['loss'])+1),model_history.history['loss'])
- axs[1].plot(range(1,len(model_history.history['val_loss'])+1),model_history.history['val_loss'])
- axs[1].set_title('Model Loss')
- axs[1].set_ylabel('Loss')
- axs[1].set_xlabel('Epoch')
- axs[1].set_xticks(np.arange(1,len(model_history.history['loss'])+1),len(model_history.history['loss'])/10)
- axs[1].legend(['train', 'val'], loc='best')
- fig.savefig('plot.png')
- plt.show() (Subplots, 2021)

Se define los datos de la carpeta donde se encuentra el entrenamiento

- train_folder = 'data/train'

Carpeta donde se encuentra las imágenes de prueba

- val_folder = 'data/test'

Variables constantes para el modelo

Tamaño del lote

- NUM_BATCHS = 64

Numero de repeticiones para cada imagen

- EPOCHS = 50

Numero de imágenes a entrenar

- TRAIN_NUM = 28709

Numero de imágenes de prueba

- VAL_NUM= 7178

Se crea un generador de imágenes para tener una escala única para los datos deentrenamiento

- train_datagen = ImageDataGenerator(rescale=1./255)

Se crea un generador de imágenes para tener una escala única para los datos de prueba

- val_datagen = ImageDataGenerator(rescale=1./255)

Se genera el directorio para el entrenamiento de las imágenes a entrenar

- train_generator = train_datagen.flow_from_directory(
    - train_folder,
    - target_size=(48,48),
    - batch_size=NUM_BATCHS,
    - color_mode="grayscale",

    - class_mode='categorical')

Se genera el valor que va a obtener las imágenes de prueba

- validation_generator = val_datagen.flow_from_directory((Team, 2021)
    - val_folder,
    - target_size=(48,48),
    - batch_size=NUM_BATCHS,
    - color_mode="grayscale",
    - class_mode='categorical')

Se crea el modelo

- modelCNN = Sequential()
- modelCNN.add(Conv2D(16 ,kernel_size = (3, 3), activation='relu',input_shape = (48 , 48 , 1)))
- modelCNN.add(Conv2D(64, kernel_size=(3, 3), activation='relu'))
- modelCNN.add(MaxPooling2D(pool_size=(2, 2)))
- modelCNN.add(Dropout(0.25))
- modelCNN.add(Conv2D(64, kernel_size=(3, 3), activation='relu'))
- modelCNN.add(MaxPooling2D(pool_size=(2, 2)))
- modelCNN.add(Dropout(0.25)) (Team, 2021)

Se añade una capa plana

- modelCNN.add(Flatten())
- modelCNN.add(Dense(512, activation='relu'))
- modelCNN.add(Dropout(0.5))
- modelCNN.add(Dense(7, activation='softmax')) (Team, 2021)

Se configura el modelo para el entrenamiento

- modelCNN.compile(loss='categorical_crossentropy',optimizer=Adam(lr=0.000 1, decay=1e-6),metrics=['accuracy']) (Team, 2021)

Se entrena al modelo

- model_info = modelCNN.fit_generator(
    - train_generator,
    - steps_per_epoch=TRAIN_NUM // NUM_BATCHS,
    - epochs=EPOCHS,
    - validation_data=validation_generator,
    - validation_steps=VAL_NUM// NUM_BATCHS)

Se guarda el modelo entrenado

- modelCNN.save_weights('model.h5')
- plot_model_history(model_info)

4.2.2. Entrenamiento del modelo con mejora:

Se describe el siguiente código mejorando la aplicación del algoritmo:

- from tensorflow.keras.callbacks import CSVLogger, ModelCheckpoint, EarlyStopping
- from tensorflow.keras.callbacks import ReduceLROnPlateau
- from tensorflow.keras.preprocessing.image import ImageDataGenerator
- import matplotlib.pyplot as plt
- from cnn import improved_cnn
- import numpy as np

Se traza curvas de precisión y pérdida

- def plot_model_history(model_history):

Se traza las curvas de precisión y pérdida dado el model_history

  - fig, axs = plt.subplots(1,2,figsize=(15,5))

Se resume el historial para mayor precisión

  - axs[0].plot(range(1,len(model_history.history['accuracy'])+1),model_history.history['accuracy'])
  - axs[0].plot(range(1,len(model_history.history['val_accuracy'])+1),model_history.history['val_accuracy'])
  - axs[0].set_title('Model Accuracy')
  - axs[0].set_ylabel('Accuracy')
  - axs[0].set_xlabel('Epoch')
  - axs[0].set_xticks(np.arange(1,len(model_history.history['accuracy'])+1),len(model_history.history['accuracy'])/10)
  - axs[0].legend(['train', 'val'], loc='best')

Se resume el historial de pérdidas

  - axs[1].plot(range(1,len(model_history.history['loss'])+1),model_history.history['loss'])
  - axs[1].plot(range(1,len(model_history.history['val_loss'])+1),model_history.history['val_loss'])
  - axs[1].set_title('Model Loss')
  - axs[1].set_ylabel('Loss')
  - axs[1].set_xlabel('Epoch')
  - axs[1].set_xticks(np.arange(1,len(model_history.history['loss'])+1),len(model_history.history['loss'])/10)
  - axs[1].legend(['train', 'val'], loc='best')

    - fig.savefig('plot.png')
    - plt.show()

Se declara los parametros

- input_shape = (48, 48, 1)
- verbose = 1
- num_classes = 7
- patience = 50
- base_path = 'models/'

Tamaño del lote

- NUM_BATCHS = 32

Número de epoch

- EPOCHS = 100

Número de imágenes a entrenar

- TRAIN_NUM = 28709

Número de imágenes de prueba

- VAL_NUM= 7178

Carpeta de imágenes de entrenamiento

- train_folder = 'data/train'

Carpeta de imágenes de prueba

- val_folder = 'data/test'
- def preprocess_input(x):
    - x = x.astype('float32')
    - x = x / 255.0
    - v2=True
    - if v2:

        - x = x - 0.5
        - x = x * 2.0
    - return x
- train_datagen = ImageDataGenerator(
        - featurewise_center=False,
        - featurewise_std_normalization=False,
        - rotation_range=10,
        - width_shift_range=0.1,
        - height_shift_range=0.1,
        - zoom_range=.1,
        - preprocessing_function=preprocess_input,
        - horizontal_flip=True)
- val_datagen = ImageDataGenerator(
        - featurewise_center=False,
        - featurewise_std_normalization=False,
        - rotation_range=10,
        - width_shift_range=0.1,
        - height_shift_range=0.1,
        - zoom_range=.1,
        - preprocessing_function=preprocess_input,
        - horizontal_flip=True)

Genera lotes de datos de tensores para imágenes de entrenamiento

- train_generator = train_datagen.flow_from_directory(
    - train_folder,
    - target_size=(48,48),

    - batch_size=NUM_BATCHS,
    - color_mode="grayscale",
    - class_mode='categorical')

Genera lotes de datos de tensores para imágenes válidas

- validation_generator = val_datagen.flow_from_directory(
    - val_folder,
    - target_size=(48,48),
    - batch_size=NUM_BATCHS,
    - color_mode="grayscale",
    - class_mode='categorical')

Parámetros de modelo / compilación

- model = improved_cnn(input_shape, num_classes)
- model.compile(optimizer='adam', loss='categorical_crossentropy',metrics=['accuracy'])
- model.summary()

Devolución de llamadas

- log_file_path = base_path + '_emotion_training.log'
- csv_logger = CSVLogger(log_file_path, append=False)
- early_stop = EarlyStopping('val_loss', patience=patience)
- reduce_lr = ReduceLROnPlateau('val_loss', factor=0.1, patience=int(patience/4), verbose=1)
- trained_models_path = base_path + '_improved_model'
- model_names = trained_models_path + '.{epoch:02d}-{val_accuracy:.2f}.hdf5'
- model_checkpoint = ModelCheckpoint(model_names, 'val_loss', verbose=1, save_best_only=True)

- callbacks = [model_checkpoint, csv_logger, early_stop, reduce_lr]

Entrenamiento

- model_info = model.fit_generator(
    - train_generator,
    - steps_per_epoch=TRAIN_NUM // NUM_BATCHS,
    - epochs=EPOCHS,
    - verbose=1, callbacks=callbacks,
    - validation_data=validation_generator,
    - validation_steps=VAL_NUM// NUM_BATCHS)
- plot_model_history(model_info)
- model.save_weights('improved_model2.h5')

4.2.3. Reconocimiento del modelo sin mejora:

Se describe el siguiente código de la aplicación del algoritmo:

- import numpy as np
- import argparse
- import matplotlib.pyplot as plt
- import cv2
- from tensorflow.keras.models import Sequential
- from tensorflow.keras.layers import Dense, Dropout, Flatten
- from tensorflow.keras.layers import Conv2D
- from tensorflow.keras.optimizers import Adam
- from tensorflow.keras.layers import MaxPooling2D

- from tensorflow.keras.preprocessing.image import ImageDataGenerator
- import tkinter as tk
- from tkinter import filedialog
- import os
- from cnn import cnn
- from tensorflow.keras.preprocessing.image import ImageDataGenerator
- root = tk.Tk() #root of tkinter
- root.withdraw() #hide root window tkinter

Se selecciona la imagen

- file_path = filedialog.askopenfilename(initialdir=os.getcwd()+'/images', title="Select file")

Se crea el modelo de CNN

- modelCNN = cnn()
- modelCNN.load_weights('model.h5')
- modelCNN.compile(loss='categorical_crossentropy',optimizer=Adam(lr=0.0001, decay=1e-6),metrics=['accuracy'])

Peso de carga del modelo entrenado

- modelCNN.load_weights('model.h5')
- modelCNN.summary()

Etiqueta emocional

- emotion_label = {0: "Enojado", 1: "Disgustado", 2: "Temeroso", 3: "Feliz", 4: "Neutral", 5: "Triste", 6: "Sorprendido"}

Lectura de la imagen

- img = cv2.imread(file_path)

Muestra imagen de entrada

- cv2.imshow('input image', cv2.resize(img,(800,600),interpolation = cv2.INTER_CUBIC))

Detector facial de OPENCV

- facecasc = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')

Si la imagen es una imagen gris, no es necesario convertir la imagen a una imagen gris

- if(len(img.shape)<3):
- gray = img

Si la imagen es una imagen en color, la imagen se convierte en una imagen gris

- elif len(img.shape)==3
- gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

Se detecta rostros en la imagen de entrada

- faces = facecasc.detectMultiScale(gray,scaleFactor=1.3, minNeighbors=5)

x, y son coordenadas de caras y w, h son peso y altura de cara

- for (x, y, w, h) in faces:

Se dibuja el rectángulo de la cara

    - cv2.rectangle(img, (x, y-50), (x+w, y+h+10), (255, 0, 0), 2)

Imagen de la cara

    - roi_gray = gray[y:y + h, x:x + w]

Se redimensiona el tamaño de la imagen de la cara a 48x48

    - cropped_img = np.expand_dims(np.expand_dims(cv2.resize(roi_gray, (48,

48)), -1), 0)

Predice la emoción usando el modelo CNN entrenado

- prediction = modelCNN.predict(cropped_img)

Indice de emoción predicha

- maxindex = int(np.argmax(prediction))

Se escribe la etiqueta de la emoción en la imagen

- cv2.putText(img, emotion_label[maxindex], (x+10, y-10), cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 255, 255), 2, cv2.LINE_AA)

Muestra la imagen prevista

- cv2.imshow('regonixzed image', cv2.resize(img,(800,600),interpolation = cv2.INTER_CUBIC))
- cv2.waitKey(0)
- cv2.destroyAllWindows()

4.2.4. Resultados sin mejora:

En la figura 9 se ve la recopilación y el reconocimiento del algoritmo base.

Figura 9 Recopilación de entrenamiento sin mejora en el algoritmo herramienta Spyder

Fuente: (Pexels,2020)

4.2.5. Reconocimiento con mejora:

Se describe el siguiente código con mejora de la aplicación del algoritmo:

- cv2.waitKey(0)
- cv2.destroyAllWindows()
- import numpy as np

- import argparse
- import matplotlib.pyplot as plt
- import cv2
- import tkinter as tk
- from tkinter import filedialog
- import os
- from cnn import improved_cnn
- from tensorflow.keras.preprocessing.image import ImageDataGenerator
- from tkinter import messagebox
- def preprocess_input(x):
    - x = x.astype('float32')
    - x = x / 255.0
    - v2=True
    - if v2:
        - x = x - 0.5
        - x = x * 2.0
    - return x
- root = tk.Tk() #root of tkinter
- root.withdraw() #hide root window tkinter

Se selecciona la imagen

- file_path = filedialog.askopenfilename(initialdir=os.getcwd()+'/images', title="Select file")

Se crea el modelo CNN

- input_shape = input_shape = (48, 48, 1)
- num_classes = 7
- modelCNN = improved_cnn(input_shape, num_classes)
- modelCNN.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

Se carga los pesos

- modelCNN.load_weights('improved_model.h5')
- modelCNN.summary()

Etiqueta de emoción

- emotion_label = {0: "Enojado", 1: "Disgustado", 2: "Temeroso", 3: "Feliz", 4: "Neutral", 5: "Triste", 6: "Sorprendido"}

Se lee la imagen

- img = cv2.imread(file_path)

Lectura de la imagen

- cv2.imshow('input image', cv2.resize(img,(800,600),interpolation = cv2.INTER_CUBIC))

Detector de rostros de OPENCV

- facecasc = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
- catoon_face_casc = cv2.CascadeClassifier('lbpcascade_animeface.xml')

Si la imagen es una imagen gris, no es necesario convertir la imagen a una imagen gris

- if(len(img.shape)<3):

- gray = img

Si la imagen es una imagen en color, la imagen se convierte en una imagen gris

- elif len(img.shape)==3:
    - gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

Detecta imagenes animadas

- faces_cat = catoon_face_casc.detectMultiScale(gray,scaleFactor=1.3, minNeighbors=5)
- if len(faces_cat)>0:
    - messagebox.showinfo("No face", No se encontro un rostro")
    - cv2.destroyAllWindows()
    - cv2.waitKey(0)
- else:

Se detecta rostros en la imagen de entrada

- faces = facecasc.detectMultiScale(gray,scaleFactor=1.3, minNeighbors=5)

x, y son coordenadas de caras y w, h son peso y altura de cara

- for (x, y, w, h) in faces:

Se dibuja el rectángulo de la cara

- cv2.rectangle(img, (x, y-50), (x+w, y+h+10), (255, 0, 0), 2)

Imagen de la cara

- roi_gray = gray[y:y + h, x:x + w]

Se redimensiona el tamaño de la imagen de la cara a 48x48

- cropped_img = np.expand_dims(np.expand_dims(cv2.resize(roi_gray, (48, 48)), -1), 0)

- cropped_img = preprocess_input(cropped_img)

Predice la emoción usando el modelo CNN entrenado

- prediction = modelCNN.predict(cropped_img)

Indice de emoción predicha

- maxindex = int(np.argmax(prediction))

Se escribe la etiqueta de la emoción en la imagen

- cv2.putText(img, emotion_label[maxindex], (x+10, y-10), cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 255, 255), 2, cv2.LINE_AA)

Muestra la imagen prevista

- cv2.imshow('regonixzed image', cv2.resize(img,(800,600),interpolation = cv2.INTER_CUBIC))
- if len(faces)>1:
  - messagebox.showinfo("face", Se encontro "+str(len(faces)) +" rostros
- cv2.waitKey(0)
- cv2.destroyAllWindows()
- im = cv2.imread('data/test/angry/im5.png')
- prediction = modelCNN.predict(cropped_img)
- maxindex = int(np.argmax(prediction))

4.2.6. Resultados con mejora:

En la figura 10 se ve una recopilación y el reconocimiento del algoritmo con mejora.

Figura 10 Recopilación de entrenamiento con mejora en el algoritmo herramienta Spyder.

Fuente: (Pexels,2020)

4.3. **Análisis**

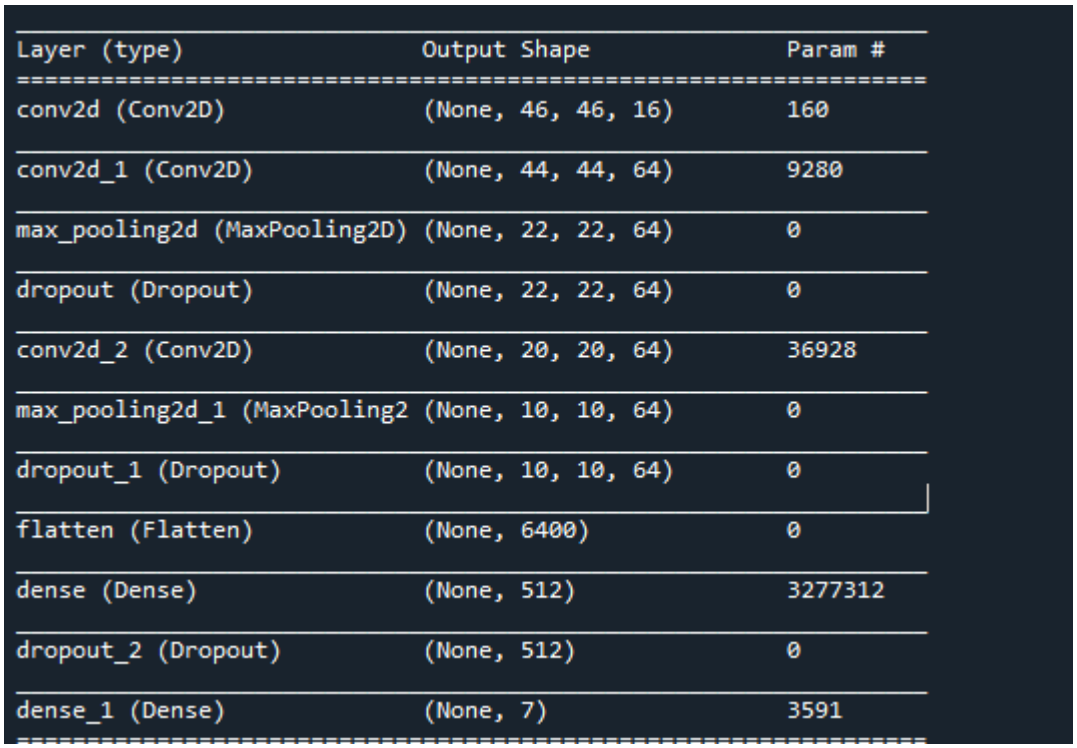
Resultados de las técnicas y herramientas mostradas en el presente proyecto de investigación que se propuso entrenar una red neuronal, y mejorarla, en el presente proyecto se llegó a una media de 60% y se logró una mejora del 3% esto quiere decir el 63.27% y en un segundo entrenamiento este cambio a 63.55% como se indica en la figura.

4.3.1. Algoritmo original:

El modelo de CNN del algoritmo original es la siguiente figura 11.

Esta imagen nos muestra los parámetros que se usa en el modelo original.

Modelo CNN:



Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 46, 46, 16)	160
conv2d_1 (Conv2D)	(None, 44, 44, 64)	9280
max_pooling2d (MaxPooling2D)	(None, 22, 22, 64)	0
dropout (Dropout)	(None, 22, 22, 64)	0
conv2d_2 (Conv2D)	(None, 20, 20, 64)	36928
max_pooling2d_1 (MaxPooling2	(None, 10, 10, 64)	0
dropout_1 (Dropout)	(None, 10, 10, 64)	0
flatten (Flatten)	(None, 6400)	0
dense (Dense)	(None, 512)	3277312
dropout_2 (Dropout)	(None, 512)	0
dense_1 (Dense)	(None, 7)	3591

Figura 11 Modelo de red neuronal sin mejora herramienta Spyder.

El modelo está entrenado en el conjunto de datos fer2013 (precisión de la prueba del 60%) el cual sirve para realizar una igualdad entre los conjuntos de datos y los de fer2013 para llegar a una medida de nuestro algoritmo figura 12 y 13.

```
Epoch 50/50
448/448 [==============================] - 123s
275ms/step - loss: 0.3689 - accuracy: 0.8699 -
val_loss: 1.3023 - val_accuracy: 0.6003
```

Figura 12 Entrenamiento sin mejora herramienta Spyder.

Se visualiza de mejor manera en el siguiente gráfico el valor que tiene en el entrenamiento, con los datos de prueba y los entrados.

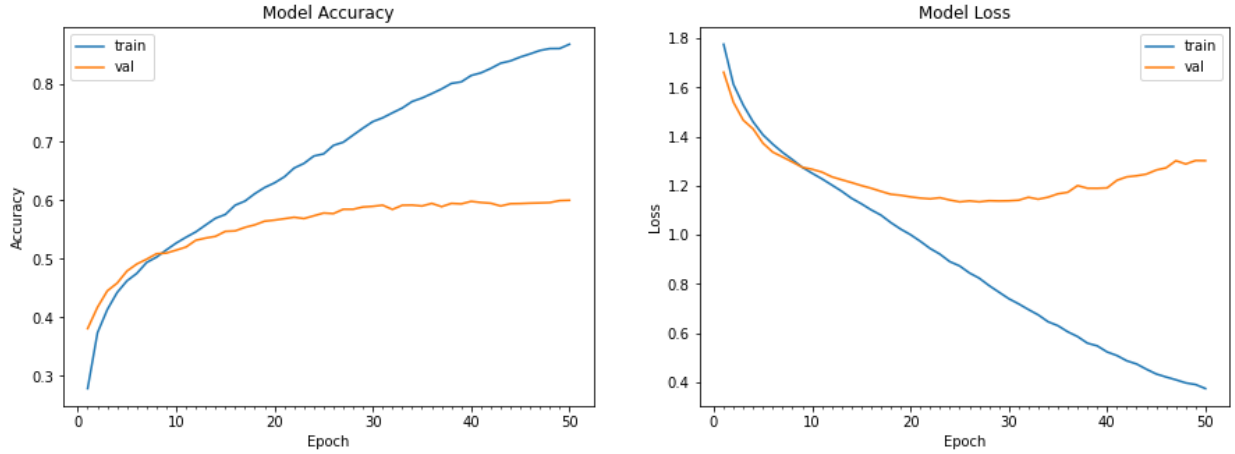


Figura 13 Gráfico de modelo herramienta Spyder.

4.3.2. Algoritmo con mejora:

El modelo de CNN del algoritmo mejorado es el siguiente en la figura 14, 15 y 16.

Aquí se visualiza los parámetros que se aumento para tener una mejora, en el que se usa funciones para llegar a un nivel de mejora superior.

Modelo de CNN:

separable_conv2d_2 (SeparableCo	(None, 22, 22, 32)	656	add[0][0]
batch_normalization_6 (BatchNor	(None, 22, 22, 32)	128	separable_conv2d_2[0][0]
activation_3 (Activation)	(None, 22, 22, 32)	0	batch_normalization_6[0][0]
separable_conv2d_3 (SeparableCo	(None, 22, 22, 32)	1312	activation_3[0][0]
batch_normalization_7 (BatchNor	(None, 22, 22, 32)	128	separable_conv2d_3[0][0]
conv2d_6 (Conv2D)	(None, 11, 11, 32)	512	add[0][0]
max_pooling2d_3 (MaxPooling2D)	(None, 11, 11, 32)	0	batch_normalization_7[0][0]
batch_normalization_5 (BatchNor	(None, 11, 11, 32)	128	conv2d_6[0][0]
add_1 (Add)	(None, 11, 11, 32)	0	max_pooling2d_3[0][0] batch_normalization_5[0][0]
separable_conv2d_4 (SeparableCo	(None, 11, 11, 64)	2336	add_1[0][0]
batch_normalization_9 (BatchNor	(None, 11, 11, 64)	256	separable_conv2d_4[0][0]
activation_4 (Activation)	(None, 11, 11, 64)	0	batch_normalization_9[0][0]
separable_conv2d_5 (SeparableCo	(None, 11, 11, 64)	4672	activation_4[0][0]
batch_normalization_10 (BatchNo	(None, 11, 11, 64)	256	separable_conv2d_5[0][0]
conv2d_7 (Conv2D)	(None, 6, 6, 64)	2048	add_1[0][0]
max_pooling2d_4 (MaxPooling2D)	(None, 6, 6, 64)	0	batch_normalization_10[0][0]
batch_normalization_8 (BatchNor	(None, 6, 6, 64)	256	conv2d_7[0][0]
add_2 (Add)	(None, 6, 6, 64)	0	max_pooling2d_4[0][0] batch_normalization_8[0][0]

Figura 14 Parámetros del modelo de red neuronal con mejora imagen 1 herramienta Spyder.

separable_conv2d_2 (SeparableCo	(None, 22, 22, 32)	656	add[0][0]
batch_normalization_6 (BatchNor	(None, 22, 22, 32)	128	separable_conv2d_2[0][0]
activation_3 (Activation)	(None, 22, 22, 32)	0	batch_normalization_6[0][0]
separable_conv2d_3 (SeparableCo	(None, 22, 22, 32)	1312	activation_3[0][0]
batch_normalization_7 (BatchNor	(None, 22, 22, 32)	128	separable_conv2d_3[0][0]
conv2d_6 (Conv2D)	(None, 11, 11, 32)	512	add[0][0]
max_pooling2d_3 (MaxPooling2D)	(None, 11, 11, 32)	0	batch_normalization_7[0][0]
batch_normalization_5 (BatchNor	(None, 11, 11, 32)	128	conv2d_6[0][0]
add_1 (Add)	(None, 11, 11, 32)	0	max_pooling2d_3[0][0] batch_normalization_5[0][0]
separable_conv2d_4 (SeparableCo	(None, 11, 11, 64)	2336	add_1[0][0]
batch_normalization_9 (BatchNor	(None, 11, 11, 64)	256	separable_conv2d_4[0][0]
activation_4 (Activation)	(None, 11, 11, 64)	0	batch_normalization_9[0][0]
separable_conv2d_5 (SeparableCo	(None, 11, 11, 64)	4672	activation_4[0][0]
batch_normalization_10 (BatchNo	(None, 11, 11, 64)	256	separable_conv2d_5[0][0]
conv2d_7 (Conv2D)	(None, 6, 6, 64)	2048	add_1[0][0]
max_pooling2d_4 (MaxPooling2D)	(None, 6, 6, 64)	0	batch_normalization_10[0][0]
batch_normalization_8 (BatchNor	(None, 6, 6, 64)	256	conv2d_7[0][0]
add_2 (Add)	(None, 6, 6, 64)	0	max_pooling2d_4[0][0] batch_normalization_8[0][0]

Figura 15 Parámetros del modelo de red neuronal con mejora imagen 2 herramienta Spyder.

separable_conv2d_6 (SeparableCo	(None, 6, 6, 128)	8768	add_2[0][0]
batch_normalization_12 (BatchNo	(None, 6, 6, 128)	512	separable_conv2d_6[0][0]
activation_5 (Activation)	(None, 6, 6, 128)	0	batch_normalization_12[0][0]
separable_conv2d_7 (SeparableCo	(None, 6, 6, 128)	17536	activation_5[0][0]
batch_normalization_13 (BatchNo	(None, 6, 6, 128)	512	separable_conv2d_7[0][0]
conv2d_8 (Conv2D)	(None, 3, 3, 128)	8192	add_2[0][0]
max_pooling2d_5 (MaxPooling2D)	(None, 3, 3, 128)	0	batch_normalization_13[0][0]
batch_normalization_11 (BatchNo	(None, 3, 3, 128)	512	conv2d_8[0][0]
add_3 (Add)	(None, 3, 3, 128)	0	max_pooling2d_5[0][0] batch_normalization_11[0][0]
conv2d_9 (Conv2D)	(None, 3, 3, 7)	8071	add_3[0][0]
global_average_pooling2d (Globa	(None, 7)	0	conv2d_9[0][0]
predictions (Activation)	(None, 7)	0	global_average_pooling2d[0][0]

Figura 16 Parámetros del modelo de red neuronal con mejora imagen 3 herramienta Spyder.

El modelo mejorado se entrena en el conjunto de datos fer2013 (precisión de la prueba 63,3%) figura 17 y 18.

En este caso ya se muestra una mejora del 3,3% a diferencia del algoritmo original.

```
Epoch 100/100
897/897 [==============================] - 187s
208ms/step - loss: 0.8029 - accuracy: 0.7008 -
val_loss: 1.0049 - val_accuracy: 0.6327
```

Figura 17 Entrenamiento con mejora herramienta Spyder.

Se visualiza de mejor manera en el siguiente gráfico el valor que tiene en el entrenamiento, con los datos de prueba y los entrados.

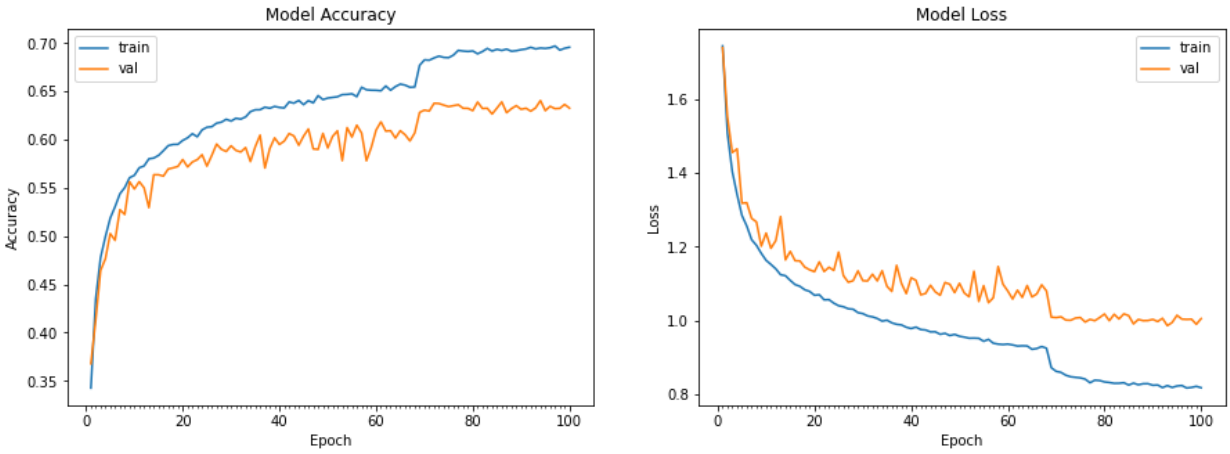


Figura 18 Gráfico de modelo herramienta Spyder

El modelo mejorado se entrena por segunda vez para aumentar el porcentaje de mejora en el conjunto de datos fer2013 (precisión de la prueba 63,6%) figura 19.

```
897/897 [==============================] - 200s 223ms/step - loss: 0.8054 - accuracy: 0.7046 -
val_loss: 1.0040 - val_accuracy: 0.6355
```

Figura 19 Segundo entrenamiento con mejora herramienta Spyder

CAPÍTULO 5: CONCLUSIONES Y RECOMENDACIONES

5.1. Conclusiones:

- La metodología de desarrollo Kanban fue muy acertada ya que se tuvo un orden de las tareas, se empezó desde las tareas pendientes que se tenía, las cuales las primordiales se utilizó el color rojo, mientras que las tareas con menos importancia se usó el color verde esto nos ayudó a una correcta organización de la información y programación, lo que se está haciendo y lo terminado. Esto fue eficaz en cada capítulo desde su inicio tanto para las correcciones.
- La precisión de ejecución tuvo un efecto positivo en esta investigación y se comprobó con 13 imágenes, las cuales el programa sin mejora mostraba resultados erróneos, mientras que el algoritmo con mejora muestra resultados positivo, con esto se llega a la conclusión de que el algoritmo con mejora tuvo un mayor desempeño con un porcentaje de mejora del 3% frente al algoritmo normal, todo esto gracias a dos funciones que se añadieron y una función en el cual partimos desde la base y se añadió 4 módulos donde en cada Epoch debía pasar y este comprobaba cada pixel de entrenamiento, entre más porcentaje tenía daba el resultado en pantalla.
- Se logro una mejora del 3% en la media que el otro, el algoritmo sin mejor tiene 59% mientras que el algoritmo con mejora alcanza el 62% esto se debe gracias al entrenamiento de la red neuronal teniendo parámetros de entrenamiento input_shape = (48, 48, 1), verbose = 1, num_classes = 7, patience = 50 con esto la red tiene claro cómo se debe desarrollar el algoritmo ya que en el entrenamiento sin mejor se lo deja con los datos predeterminados, a su vez se aumentó el número de epotch a EPOCHS = 100 para tener un menor margen de error.

5.2. Recomendaciones

- Se recomienda usar parámetros basados en el modelo y datos que se van a realizar, como se obtuvo los resultados positivos se los dejo con estos parámetros mientras que en otro entrenamiento es diferente, así que se debe usar los parámetros que mejor convenga en el entrenamiento que se va a utilizar y realizar pruebas constantes hasta que sea el resultado esperado.
- Es importar elegir la metodología a utilizar, ya que esta brinda facilidad en desarrollar un proyecto y las fases que el proyecto contenga, para proyectos mucho más grandes, se puede usar metodologías diferente, la recomendada por la industria es SCRUM, ya que este tiene un control en cada una de sus fases.
- Utilizar las funcionalidades y librerías de Python en conjunto con TensorFlow a fin de sacarle el mayor provecho posible por ejemplo con el tiempo de ejecución y el porcentaje de mejora.

6. Bibliografía

Busy. (2020). Retrieved from Busy: https://busy.org/@iars.geo/breve-historias-de-las-redes-neuronales-artificiales-articulo-1

Dentist, O. (2020, 8 30). *O2Dental*. Retrieved from O2Dental: https://o2dental.ca/top-reasons-regularly-visit-dentist-west-end/

Ekman, P., W., F., & J., H. (2002). Facs manual.

FaceFirst. (2018, 08 29). *Breve Historia de Reconocimiento Facial*. Retrieved from Breve Historia de Reconocimiento Facial: https://www.facefirst.com/blog/breve-historia-del-reconocimiento- facial/

Fahrenheit Magazine. (2020, 8 28). Retrieved from Fahrenheit Magazine: https://fahrenheitmagazine.com/arte/la-locura-genial-del-escultor-franz-xaver-messerschmidt

hamgardi. (2020, 08 30). Retrieved from hamgardi: https://hamgardi.com/fa/Post/1517-رنتن-اداب-مهمازی

Kapoor, A., Burleson, W., & Picard, R. W. (2007). Automatic prediction. *SciencieDirect*, 724-736.

Kulkarni, K., Corneanu, C., Ikechukwu, O., Escalera, E., Baro, X., Hyniewska, S., . . . Anbarjafari, G. (2017). Automatic recognition of deceptive facial expressions of emotion. *arXiv preprint*.

Márquez, M. (2012). *Reconocimiento de expresiones faciales emepleando memorias asociativas Alfa- Beta.* Retrieved from Reconocimiento de expresiones faciales emepleando memorias asociativas Alfa-Beta:

https://www.saber.cic.ipn.mx/SABERv3/Repositorios/webVerArchivo /26047/1 Paul Ekman, W. F. (2002). *Facs manual. a human face.*

Richard Wiseman, R. (2020). *newscientist*. Retrieved from newscientist: https://www.newscientist.com/article/mg20126957-300-how-your-looks-betray-your- personality/

Taigman, Y., Yang, M., Ranzato, M., & Wolf, L. (204). Deepface: Closing the gap to human-level performance in face verification. *In Proceedings of the IEEE conference on computer vision and pattern recognition*, 1701-1708.

TheCut. (2020, 08 30). Retrieved from TheCut: https://www.thecut.com/2011/06/meet_the_new_girl_grace_bol.html

Vinciarelli, A., ntic, M. P., & Bourlard, H. (2009). Social signal processing. *Image and Vision Computing*, 1743-1759.

Zhiwei Deng, A. V. (2016). *Structure inference machines: Recurrent neural networks for analyzing relations in group activity recognition.* IEEE CVPR.

Kanade, T., Cohn, J. F., & Tian, Y. (2000, March). Comprehensive database for facial expression analysis. In *Proceedings Fourth IEEE International Conference on Automatic Faceand Gesture Recognition (Cat. No. PR00580)* (pp. 46-53). IEEE.

Lucey, P., Cohn, J. F., Kanade, T., Saragih, J., Ambadar, Z., & Matthews, I. (2010, June). Theextended cohn-kanade dataset (ck+): A complete dataset for action unit and emotion-specifiedexpression. In *2010 ieee computer society conference on computer vision and pattern recognition-workshops* (pp. 94-101). IEEE.

Pantic, M., & Patras, I. (2006). Dynamics of facial expression: recognition of facial actions andtheir temporal segments from face profile image sequences. *IEEE Transactions on*

Systems, Man, and Cybernetics, Part B (Cybernetics), *36*(2), 433-449.

Gross, R., Matthews, I., Cohn, J., Kanade, T., & Baker, S. (2010). Multi-pie. *Image and VisionComputing*, *28*(5), 807-813.

Yan, W. J., Wu, Q., Liu, Y. J., Wang, S. J., & Fu, X. (2013, April). CASME database: a dataset of spontaneous micro-expressions collected from neutralized faces. In *2013 10th IEEE international conference and workshops on automatic face and gesture recognition (FG)* (pp.1-7). IEEE.

Mavadati, S. M., Mahoor, M. H., Bartlett, K., Trinh, P., & Cohn, J. F. (2013). Disfa: A spontaneous facial action intensity database. *IEEE Transactions on Affective Computing*, *4*(2),151-160.

Dhall, A., Goecke, R., Lucey, S., & Gedeon, T. (2011). Acted facial expressions in the wild database. *Australian National University, Canberra, Australia, Technical Report TR-CS-11*, *2*,1.

McDuff, D., Kaliouby, R., Senechal, T., Amr, M., Cohn, J., & Picard, R. (2013). Affectiva-mit facial expression dataset (am-fed): Naturalistic and spontaneous facial expressions collected. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops* (pp. 881-888).

McKeown, G., Valstar, M., Cowie, R., Pantic, M., & Schroder, M. (2011). The semaine database: Annotated multimodal records of emotionally colored conversations between a person and a limited agent. *IEEE transactions on affective computing*, *3*(1), 5-17.

Yin, L., Wei, X., Sun, Y., Wang, J., & Rosato, M. J. (2006, April). A 3D facial expression database for facial behavior research. In *7th international conference on automatic face and gesture recognition (FGR06)* (pp. 211-216). IEEE.

Savran, A., Alyüz, N., Dibeklioğlu, H., Çeliktutan, O., Gökberk, B., Sankur, B., & Akarun, L. (2008, May). Bosphorus database for 3D face analysis. In *European workshop on biometrics and identity management* (pp. 47-56). Springer, Berlin, Heidelberg.

Zhang, X., Yin, L., Cohn, J. F., Canavan, S., Reale, M., Horowitz, A., & Liu, P. (2013, April). A high-resolution spontaneous 3d dynamic facial expression database. In *2013 10th IEEE International Conference and Workshops on Automatic Face and Gesture Recognition (FG)* (pp. 1-6). IEEE.

Zhang, X., Yin, L., Cohn, J. F., Canavan, S., Reale, M., Horowitz, A., & Liu, P. (2013, April). A high-resolution spontaneous 3d dynamic facial expression database. In *2013 10th IEEE International Conference and Workshops on Automatic Face and Gesture Recognition (FG)* (pp. 1-6). IEEE.

Zhang, X., Yin, L., Cohn, J. F., Canavan, S., Reale, M., Horowitz, A., & Liu, P. (2013, April). A high-resolution spontaneous 3d dynamic facial expression database. In *2013 10th IEEE International Conference and Workshops on Automatic Face and Gesture Recognition (FG)* (pp. 1-6). IEEE.

Zhang, X., Yin, L., Cohn, J. F., Canavan, S., Reale, M., Horowitz, A., & Liu, P. (2013, April). A high-resolution spontaneous 3d dynamic facial expression database. In *2013 10th IEEE International Conference and Workshops on Automatic Face and Gesture Recognition (FG)* (pp. 1-6). IEEE.

McKeown, G., Valstar, M., Cowie, R., Pantic, M., & Schroder, M. (2011). The semaine database: Annotated multimodal records of emotionally colored conversations between a person and a limited agent. *IEEE transactions on affective computing*, *3*(1), 5-17.

Wang, S., Liu, Z., Lv, S., Lv, Y., Wu, G., Peng, P., ... & Wang, X. (2010). A natural visible and infrared facial expression database for expression recognition and emotion inference.

IEEETransactions on Multimedia, *12*(7), 682-691.

Nguyen, H., Kotani, K., Chen, F., & Le, B. (2013, October). A thermal facial emotion databaseand its analysis. In *Pacific-Rim Symposium on Image and Video Technology* (pp. 397-408). Springer, Berlin, Heidelberg.

El Kaliouby, R., & Robinson, P. (2005, October). Generalization of a vision-based computational model of mind-reading. In *International Conference on Affective Computing and Intelligent Interaction* (pp. 582-589). Springer, Berlin, Heidelberg.

Ji, Q., Lan, P., & Looney, C. (2006). A probabilistic framework for modeling and real-time monitoring human fatigue. *IEEE Transactions on systems, man, and cybernetics-Part A: Systems and humans*, *36*(5), 862-875.

Littlewort, G. C., Bartlett, M. S., & Lee, K. (2007, November). Faces of pain: automated measurement of spontaneousallfacial expressions of genuine and posed pain. In *Proceedings of the 9th international conference on Multimodal interfaces* (pp. 15-21).

Biel, J. I., Teijeiro-Mosquera, L., & Gatica-Perez, D. (2012, October). Facetube: predicting personality from facial expressions of emotion in online conversational video. In *Proceedingsof the 14th ACM international conference on Multimodal interaction* (pp. 53-56).

Cohn, J. F., Kruez, T. S., Matthews, I., Yang, Y., Nguyen, M. H., Padilla, M. T., ... & De la Torre, F. (2009, September). Detecting depression from facial actions and vocal prosody. In *2009 3rd International Conference on Affective Computing and Intelligent Interaction and Workshops* (pp. 1-7). IEEE.

Williamson, J. R., Quatieri, T. F., Helfer, B. S., Ciccarelli, G., & Mehta, D. D. (2014, November). Vocal and facial biomarkers of depression based on motor incoordination and

timing. In *Proceedings of the 4th International Workshop on Audio/Visual Emotion Challenge* (pp. 65-72).

Busy. (2020). Obtenido de Busy: https://busy.org/@iars.geo/breve-historias-de-las-redes-neuronales-artificiales-articulo-1

Dentist, O. (30 de 8 de 2020). *O2Dental*. Obtenido de O2Dental: https://o2dental.ca/top-reasons-regularly-visit-dentist-west-end/

Ekman, P., W., F., & J., H. (2002). Facs manual.

FaceFirst. (29 de 08 de 2018). *Breve Historia de Reconocimiento Facial*. Obtenido de Breve Historia de Reconocimiento Facial: https://www.facefirst.com/blog/breve-historia-del-reconocimiento-facial/

Fahrenheit Magazine. (28 de 8 de 2020). Obtenido de Fahrenheit Magazine: https://fahrenheitmagazine.com/arte/la-locura-genial-del-escultor-franz-xaver-messerschmidt

hamgardi. (30 de 08 de 2020). Obtenido de hamgardi: https://hamgardi.com/fa/Post/1517--اداب-هممازی

Kapoor, A., Burleson, W., & Picard, R. W. (2007). Automatic prediction. *SciencieDirect*, 724-736.

Kulkarni, K., Corneanu, C., Ikechukwu, O., Escalera, E., Baro, X., Hyniewska, S., . . . Anbarjafari, G. (2017). Automatic recognition of deceptive facial expressions of emotion. *arXiv preprint*.

Márquez, M. (2012). *Reconocimiento de expresiones faciales emepleando memorias asociativas Alfa- Beta.* Obtenido de Reconocimiento de expresiones faciales emepleando memorias asociativas Alfa-Beta:

https://www.saber.cic.ipn.mx/SABERv3/Repositorios/webVerArchivo/26047/1

Paul Ekman, W. F. (2002). *Facs manual. a human face.*

Richard Wiseman, R. (2020). *newscientist*. Obtenido de newscientist: https://www.newscientist.com/article/mg20126957-300-how-your-looks-betray-your- personality/

Taigman, Y., Yang, M., Ranzato, M., & Wolf, L. (204). Deepface: Closing the gap to human-level performance in face verification. *In Proceedings of the IEEE conference on computer vision and pattern recognition*, 1701-1708.

TheCut. (30 de 08 de 2020). Obtenido de TheCut:https://www.thecut.com/2011/06/meet_the_new_girl_grace_bol.html

Vinciarelli, A., ntic, M. P., & Bourlard, H. (2009). Social signal processing. *Image and Vision Computing*, 1743-1759.

Zhiwei Deng, A. V. (2016). *Structure inference machines: Recurrent neural networks for analyzingrelations in group activity recognition.* IEEE CVPR.

Dhall, A., Goecke, R., Lucey, S., & Gedeon, T. (2011). Acted facial expressions in the wild database. *AustralianNational University, Canberra, Australia, Technical Report TR-CS-11*,

bogotobogo. (27 de Abril de 2021). Obtenido de bogotobogo: https://www.bogotobogo.com/python/OpenCV_Python/python_opencv3_Image_Object_Detection_Face_Detection_Haar_Cascade_Classifiers.php

Boulogne, G. D. (2016). *Artsandculture [Fotógrafia]*. Obtenido de Artsandculture: https://artsandculture.google.com

Castelvecchi, D. (2016). Can we open the black box of ai? Nature News.

Cheatsheet [Imagen]. (2020). Obtenido de Cheatsheet: https://ml-cheatsheet.readthedocs.io

Fahrenheit, M. (2020). *Getty [Fotográfia]*. Obtenido de Getty: http://www.getty.edu

Geeks For Geeks. (27 de Abril de 2020). Obtenido de Geeks For Geeks: https://www.geeksforgeeks.org/python-opencv-cv2-puttext-method/

Keras. (27 de Abril de 2021). *StackOverFlow*. Obtenido de StackOverFlow: https://stackoverflow.com/questions/47555829/preprocess-input-method-in-keras

Khomami Abadi, M., Miranda Correa, J. A., Wache, J., Yang, H., Patras, I., & Sebe, N. (2015). Inference of personality traits and affect schedule by analysis of spontaneous reactions to affective videos. FG.

MasterFile[Fotografía]. (2020). Obtenido de MasterFile: https://www.masterfile.com

Numpy. (27 de Abril de 2021). Obtenido de https://numpy.org

parneetk/parneetk.github.io. (27 de Abril de 2021). Obtenido de https://github.com/parneetk/parneetk.github.io

Pexels. (20 de 6 de 2021). Obtenido de Pexels: https://www.pexels.com/es-es/

Pinterest[Fotografía]. (2020). Obtenido de Pinterest: https://www.pinterest.com

Prabhu, A. K. (27 de Abril de 2021). *stackoverflow*. Obtenido de stackoverflow: https://stackoverflow.com/questions/21596281/how-does-one-convert-a-grayscale-image-to-rgb-in-opencv-python

programcreek. (27 de Abril de 2021). Obtenido de programcreek: https://www.programcreek.com/python/example/83399/cv2.putText

realpython. (27 de Abril de 2021). Obtenido de realpython: https://realpython.com/face-recognition-with-python/

Scarpino, M. (2018). *TensorFlow For Dummies.* New Jersey: Wiley.

Shutterstock. (2020). *shutterstock[Fotográfia]*. Obtenido de shutterstock:

https://www.shutterstock.com.

Subplots, M. (27 de Abril de 2021). Obtenido de https://www.w3schools.com/python/matplotlib_subplots.asp

Team, K. (27 de Abril de 2021). *Keras: the Python deep learning API*. Obtenido de Keras: the Python deep learning API: https://keras.io

Manual de usuario

Instalación de herramientas Anaconda y Spyder

1. Ingresar al siguiente enlace:

https://www.anaconda.com/products/individual#Downloads

2. En la página web saldrá los siguientes enlaces para el sistema operativo.

Figura 20 Instaladores para diferentes Sistemas operativos

3. Se descarga e instala.

4. Se ejecuta la consola de Anaconda

Figura 21 Carpeta que contiene los programas que nos brinda Anaconda

5. Se instala las librerías a usar.

conda install -c conda-forge keras

pip install numpy

conda install matplotlib

Se acepta la instalación.

Colocación de imágenes extra para entrenar:

Si se desea ingresar más imágenes al programa lo podemos hacer desde data en el cual se tiene dos carpetas test y train:

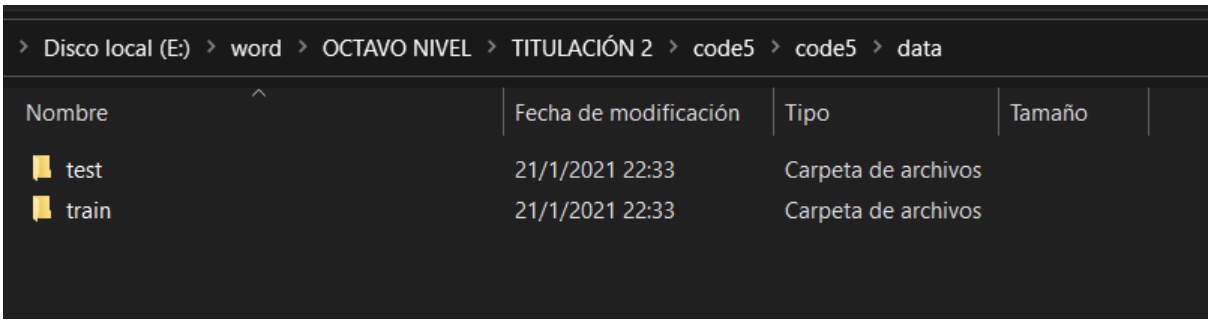


Figura 22 Carpetas donde se encuentra los datos de prueba y de entrenamiento

En la carpeta test se encuentra 7 carpetas las cuales se encuentran las imágenes a entrenar.

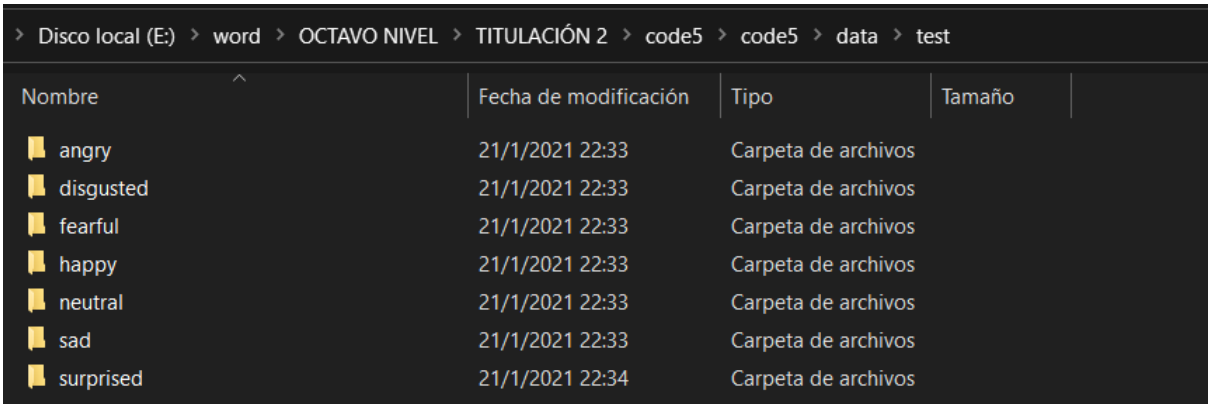


Figura 23 Carpetas donde se encuentra los datos distribuidos por estado de animo

Por ejemplo, si se quiere más imágenes de personas enojadas y se quiere añadir, se copia y guarda en la carpeta de enojados, de igual manera se puede copiar y pegar en las otras carpetas:

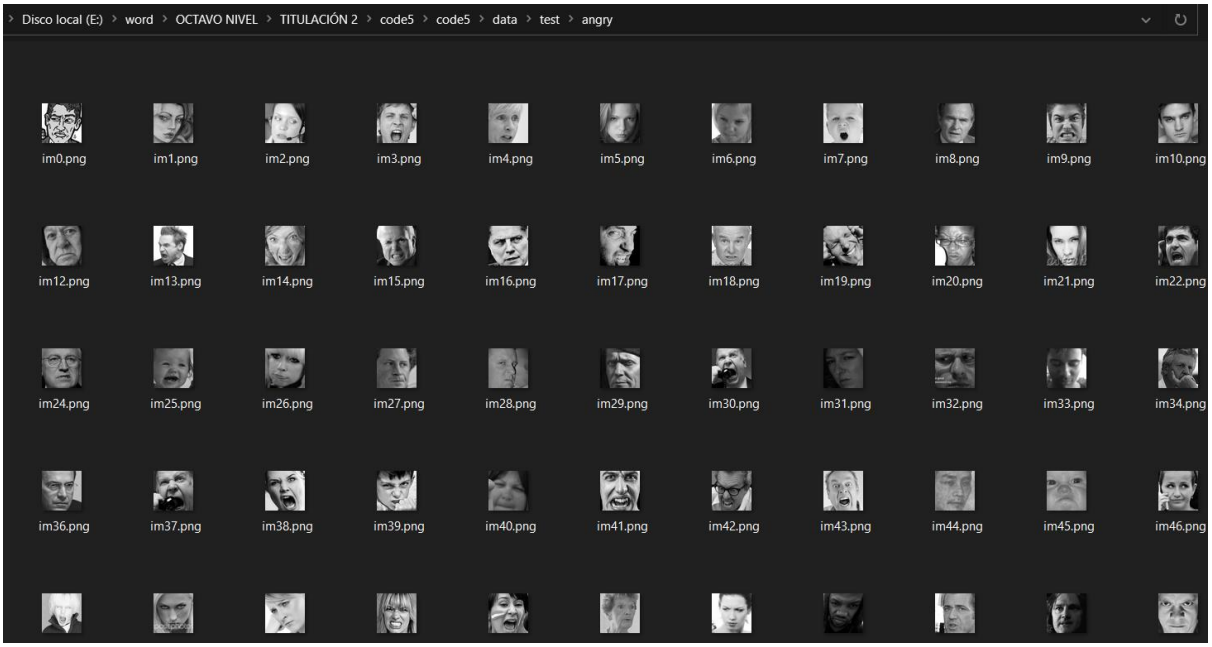


Figura 24 imágenes de personas enojadas a ser entrenadas

Guardar imágenes a reconocer:

Las imágenes que se quiere reconocer, se copian y pega en la siguiente ubicación como se muestra a continuación:

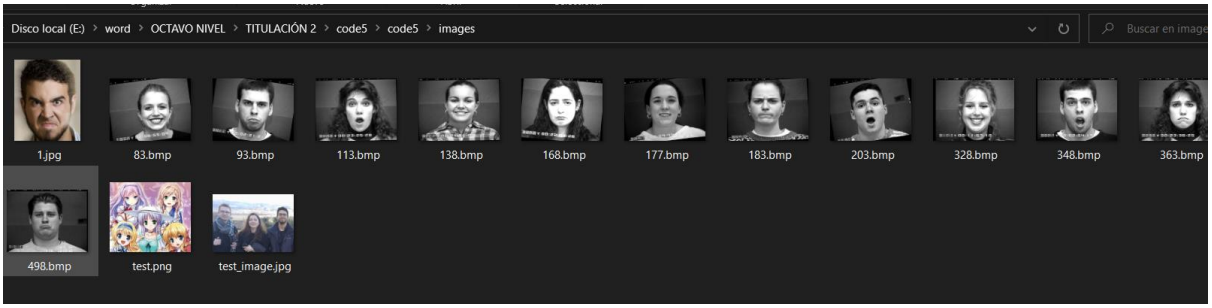


Figura 25 imágenes para ser entrenadas

Nota: No es necesario ya que con el cuadro de diálogo de selección de archivos se puede buscar en otra ubicación, pero para tener un orden de las imágenes que se va a entrenar se recomienda guardar en la ubicación mencionada.

Explicación carpetas:

A continuación, se explica el uso de cada archivo y carpeta:

Models: En esta carpeta se guarda todos los modelos guardados.

Cnn.py: este archivo se encuentra el método que se definió para realizar la mejora.

haarcascade_frontalface_default, lbpcascade_animeface: son librerías que se usa para el reconocimiento facial.

Model.h5: Es el modelo para el algoritmo original.

Improved_model2: Es el modelo para el algoritmo con mejora.

Train.py: El código para realizar el modelo original.

Train_improved: El código para realizar el modelo con mejora.

Recognition: El código para realizar el reconocimiento de las imágenes del algoritmo original.

Recognition_improved: El código para realizar el reconocimiento de las imágenes con el algoritmo con mejora.

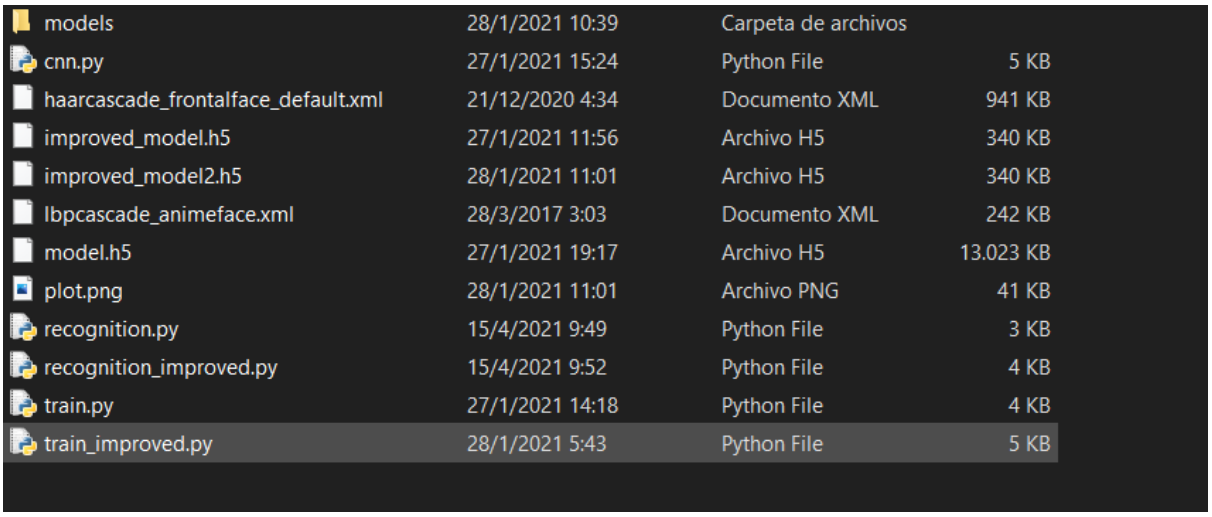


Figura 26 Carpetas y archivos del programa en general

Ejecución del programa

Reconocimiento facial de emociones en la imagen

Ejecute recognition_improved.py para el reconocimiento de emociones.

A continuación, se mostrará el cuadro de diálogo de selección de archivos.

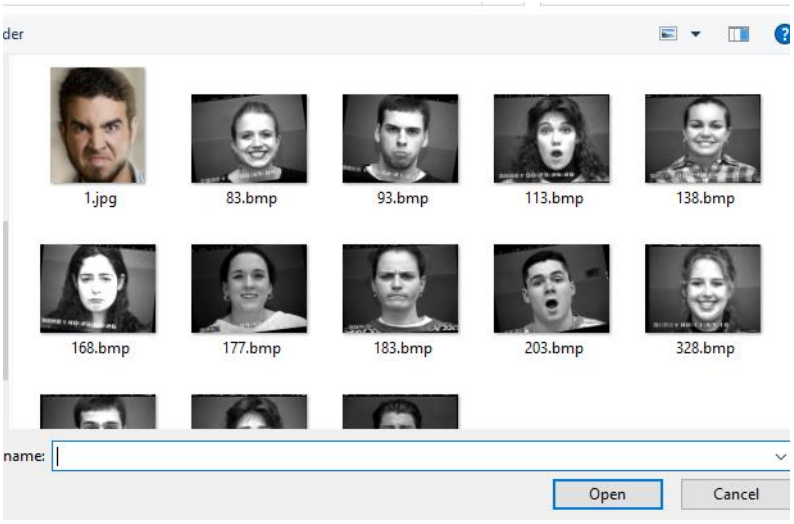


Figura 27 Selector de imágenes

Si se selecciona algún archivo, se mostrará el resultado del reconocimiento.

Figura 28 Resultado del reconocimiento