Universidad de Navarra | DATAI
INSTITUTO DE CIENCIA DE LOS DATOS E INTELIGENCIA ARTIFICIAL

Optimización de la precisión y eficiencia del reconocimiento de emociones en tiempo real mediante redes neuronales convolucionales en sistemas de videovigilancia.

Trabajo Fin de Máster

Máster Universitario en Big Data Science

Curso académico 2022-2023

AUTOR: Raúl Artigues Femenia

TUTOR ACADÉMICO: Borja Balparda de Marco

Madrid a 14 de junio del 2023

RESUMEN

En el panorama tecnológico actual, la inteligencia artificial desempeña un papel crucial, especialmente en el campo de la visión por computadora. Las redes neuronales convolucionales han surgido como un enfoque poderoso para el reconocimiento y análisis de imágenes.

El objetivo principal es desarrollar una herramienta capaz de identificar cuatro emociones estándar -alegría, tristeza, sorpresa y neutralidad- basándose únicamente en las expresiones faciales. El modelo creado se presenta como el Producto Mínimo Viable para la puesta en marcha de una Start-Up.

Las contribuciones del presente proyecto son significativas. Desde un punto de vista científico, se avanza en el campo de la visión por computador y el reconocimiento de emociones mediante técnicas de Deep Learning. El modelo desarrollado destaca el potencial de las redes neuronales convolucionales en la interpretación precisa de las emociones humanas a partir de las expresiones faciales. A nivel empresarial, esta herramienta puede ser de gran utilidad para comprender y analizar las experiencias emocionales de los clientes, lo que puede tener un impacto importante en la mejora de la satisfacción del cliente, la adaptación de estrategias de marketing y la optimización de los servicios.

Palabras Clave: Redes neuronales convolucionales, reconocimiento de emociones, Web Scraping, Data Warehouse, reportes, tiempo real.

ABSTRACT

In the current technological landscape, artificial intelligence plays a crucial role, especially in the field of computer vision. Convolutional neural networks have emerged as a powerful approach for image recognition and analysis.

The main objective is to develop a tool capable of identifying four standard emotions -happiness, sadness, surprise, and neutrality- based solely on facial expressions. The created model is presented as the Minimum Viable Product for a Start-Up.

The contributions of this project are significant. From a scientific perspective, it advances the field of computer vision and emotion recognition using deep learning techniques. The developed model highlights the potential of convolutional neural networks in accurately interpreting human emotions from facial expressions. From a business standpoint, this tool can be highly useful for understanding and analyzing customers' emotional experiences, which can have a significant impact on improving customer satisfaction, adapting marketing strategies, and optimizing services.

Keywords: Convolutional Neural Networks, Emotion Recognition, Web Scraping, Data Warehouse, Reports, Real-time.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

CAPÍTULO 1: INTRODUCCIÓN

La inteligencia artificial se ha consolidado como uno de los campos más dinámicos y en constate evolución en los últimos años, permitiendo el desarrollo de herramientas y técnicas cada vez más precisas para el análisis de datos complejos. En particular, las **redes neuronales** se han convertido en una metodología eficaz para el reconocimiento de patrones en datos complejos, incluyendo las expresiones faciales.

Este trabajo final de máster tiene como objetivo principal la creación de una herramienta basada en redes neuronales para la identificación de las **cuatro emociones** estándar que las personas pueden expresar exclusivamente a través de sus rostros: *alegría, disgusto, sorpresa y neutral.* El modelo permitirá clasificar a las personas en los ámbitos emocionales pertinentes, lo que posibilitará la interpretación de las sensaciones tanto positivas como negativas experimentadas.

El modelo que se desarrollará en el presente proyecto será el producto mínimo viable de una Start-Up que busca implementar la inteligencia artificial en el mundo cotidiano. El primer proyecto de esta empresa será la implementación del modelo creado en cámaras de videovigilancia para el análisis de las emociones en los establecimientos de los clientes. Esto permitirá un análisis innovador de gran importancia para cualquier industria, ya que permitirá a los propietarios de negocios y gerentes mejorar la experiencia de los clientes y, por ende, su satisfacción.

En definitiva, este trabajo final de máster busca contribuir al desarrollo de tecnologías de inteligencia artificial aplicadas al **reconocimiento de emociones** y su potencial impacto en la mejora de la experiencia del usuario en diversos contextos empresariales. Además, este será un primer paso para la creación de una empresa con un enfoque innovador y disruptivo en el campo de la inteligencia artificial y es por eso por lo que se sigue una metodología *end-to-end*, desde la obtención de imágenes hasta la puesta en producción.

CAPÍTULO 2: DESCRIPCIÓN DEL PROBLEMA

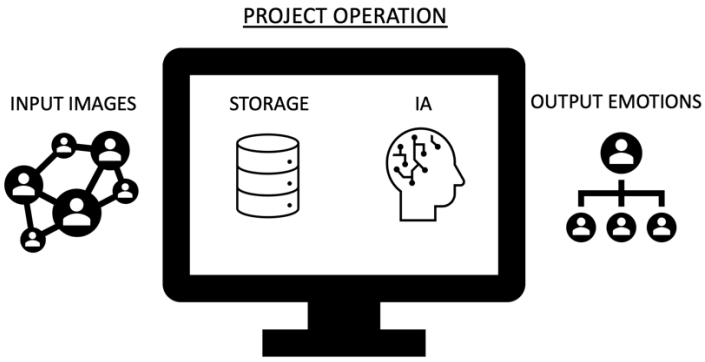
2.1 Planteamiento del proyecto

Hoy en día, en numerosos establecimientos, es común observar cámaras de vigilancia cuya principal función es garantizar la seguridad de los clientes y el personal del local. Aunque este es el enfoque primordial de dichos dispositivos, es posible ampliar su aplicación, almacenando y analizando cada fotograma de los videos capturados durante los 365 días del año para emplearlos en contextos completamente distintos.

El sector por excelencia que utiliza las cámaras de seguridad implementadas por todo el país es el de la seguridad vial. La Dirección General de tráfico emplea estos dispositivos para rastrear aquel vehículo que conduce de una manera inadecuada o infringiendo las leyes mediante el uso de sistemas de inteligencia artificial que son capaces que capturar el número de matrícula y conectarse a la base de datos estatal e identificar el titular de ese transporte privado.

Dado que existen empresas o entes públicos que emplean el reconocimiento de imágenes para generar beneficios y mejorar el bienestar social: ¿Por qué no implementar una **red neuronal capaz de detectar las emociones de las personas**?

El **reconocimiento de las emociones humanas** es esencial para facilitar la interacción social y mejorar la experiencia del usuario en diversos entornos, como los establecimientos recreativos. A pesar de los avances en la tecnología de reconocimiento facial, la identificación precisa y en tiempo real de las emociones humanas sigue siendo un desafío debido a la complejidad de las expresiones faciales y las diferencias individuales. Este proyecto de investigación aborda este problema al desarrollo un modelo basado en **redes neuronales convolucionales para analizar las expresiones faciales** de las personas y clasificarlas en las cuatro emociones estándar: *alegría, disgusto, sorpresa y neutral*.



Figuras 1. Funcionamiento del proyecto.

Para alcanzar el objetivo se requiere inicialmente obtener un conjunto de datos suficientemente amplio y diverso de imágenes faciales para entrenar y validar el modelo de redes neuronales. Esto implicará la implementación de técnicas de **Web Scraping** en *Google Images* para recolectar imágenes de rostros humanos con diversas emociones además de la utilización de conjunto de datos abiertos prestados por la *Universidad de Denver* (Mollahosseini et al., 2017).

El segundo bloque y uno de los más importantes, es la limpieza de los datos, ya que a la hora de aplicar la anterior técnica se descargan URLs que no resultan interesantes para la creación de la red neuronal (imágenes que no tienen la emoción que se quiere, dibujos animados, etc.).

Por otro lado, se necesite un entorno en la nube para almacenar y procesar las imágenes ya procesadas de una manera eficiente, lo que requiere la creación de un **Data Warehouse** en *Oracle*. Una vez que se haya establecido la infraestructura de los datos, será necesario adquirir un conocimiento profundo del modelo matemático subyacente a **las redes neuronales convolucionales** y desarrollar un algoritmo específico para la detección de emociones faciales. Este algoritmo deberá ser implementado en tiempo real para permitir la clasificación las sensaciones de las personas que salen de diferentes áreas de un establecimiento cualquiera (esa es la filosofía de la *Start-Up*).

Por último, es fundamental generar informes que evalúen y presenten los resultados obtenidos a partir del modelo propuesto. Estos informes, creados en *Power BI* y publicados en *Wix* (simulando un SharePoint de la Start-Up), permitirán la validación de la eficacia del modelo en la identificación de emociones para mejorar la experiencia del usuario en los establecimientos, por ejemplo, en un parque de atracciones.

2.2 Objetivos y materiales y métodos

2.2.1 Objetivos

El presente proyecto tiene como principal objetivo analizar, mediante el uso de redes neuronales, las expresiones faciales de las personas. Se desarrollará una herramienta capaz de identificar las **cuatro emociones** estándar que los seres humanos pueden expresar exclusivamente a través de sus rostros: *alegría, disgusto, sorpresa y neutral*. Con este modelo, se pretende clasificar a las personas en los ámbitos emocionales pertinentes, permitiendo así interpretar las sensaciones tanto positivas como negativas experimentadas en cualquier negocio. El modelo será el producto mínimo viable para la *Start-Up* que se creará.

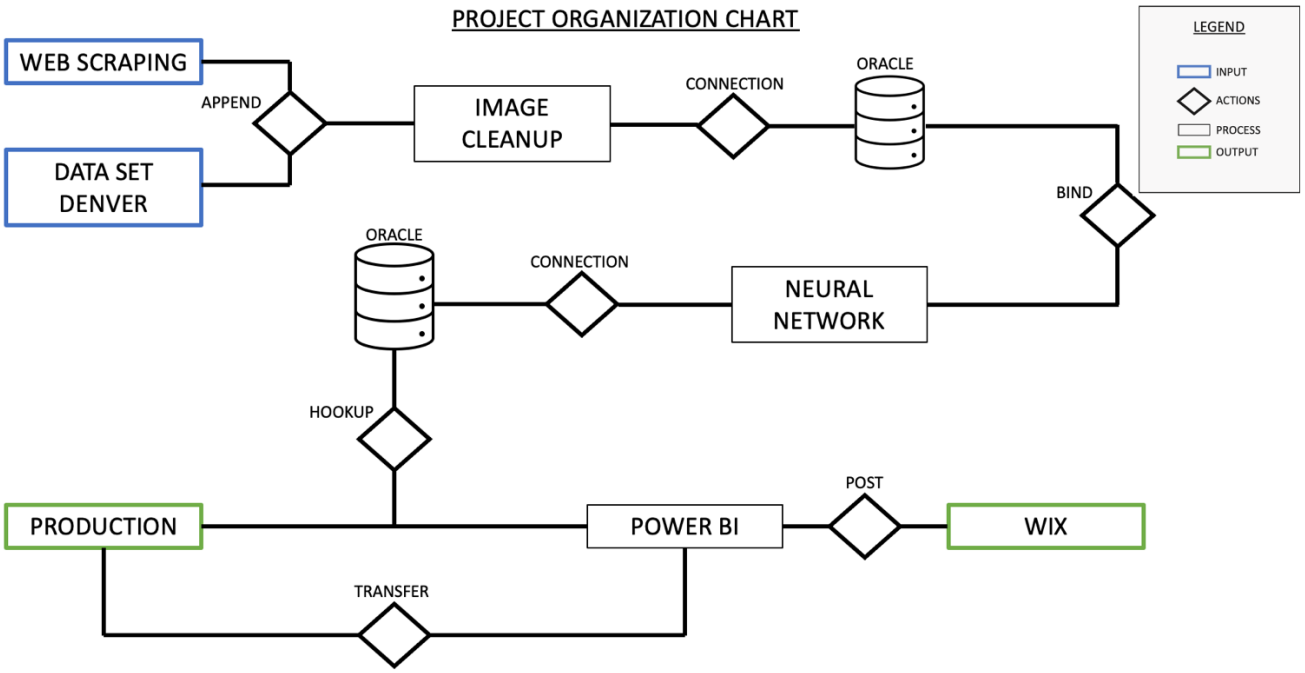
Para alcanzar este objetivo, se requiere abordar las siguientes tareas:

1. Obtención de los datos a través de la aplicación de técnicas de **Web Scraping.**
2. Creación de un entorno en la nube (**Data Warehouse**) para el almacenamiento y procesamiento de las imágenes en *Oracle*.
3. Adquisición de conocimientos sobre el modelo matemático subyacente a las redes neuronales convolucionales.
4. Desarrollo del algoritmo de **redes neuronales convolucionales** aplicado a la detección de emociones faciales.
5. **Implementación** del modelo **en tiempo real** para la clasificación de las emociones.
6. **Generación de informes** que permiten evaluar y presentar los resultados obtenidos.

2.2.2 Materiales y métodos

Se ha emulado un proyecto empresarial de inteligencia artificial mediante el uso de herramientas de programación, como *Python*, de sistemas de gestión de datos en la nube, como *Oracle*, y un instrumento de reportes, *Power BI*. El modelo creado tiene la característica de ofrecer los **resultados en tiempo real** mediante la utilización de la webcam del ordenador, simulando una cámara de videovigilancia. Toda la información recolectada se guardará de forma automática en una base de datos autónoma en *Oracle*, que a su vez está conecta con una página web en *Wix* que contiene un reporte en tiempo real mediante el uso de *Power BI*.

Para la creación de *Optimización de la precisión y eficiencia del reconocimiento de emociones en tiempo real mediante redes neuronales convolucionales en sistemas de videovigilancia* es necesario seguir los pasos que se indican en la Figura1 que se proceden a explicar brevemente.



Figuras 2. Organigrama del proyecto.

En primer lugar, se realizará la obtención de imágenes a través de técnicas de **Web Scraping** y la utilización del data set que ofrece de forma gratuita la *Universidad de Denver*. A continuación, se aplicarán métodos de limpieza de aquellas imágenes que no sean válidas a través de una serie de criterios.

Nada más obtener los datos limpios, se realizará una conexión de forma automática con el servicio que ofrece *Oracle*, **Oracle Autonomous Database**, para optimizar el almacenamiento y procesamiento de los datos.

Una vez adquiridos los conocimientos básicos y necesarios del *Deep learning*, se procederá con la continuación del flujo del proyecto, creación de la red neuronal. Para el correcto funcionamiento de la **red convolucional** para la detección de **emociones faciales**, habrá que testear con distintas arquitecturas hasta encontrar la que maximice los parámetros establecidos para la correcta clasificación: precisión, sensibilidad, especificidad y acierto.

Posteriormente, **se implementará el modelo en tiempo real para la clasificación de las emociones**, integrándolo en la cámara del ordenador en el que aparecen los valores predichos, los cuales indicarán si una persona está sonriendo, enfadada, etc.

Finalmente, se **generará un informe interactivo** que permite evaluar y presentar los resultados obtenidos. Dicho reporte incluirá tanto la descripción detalla de las arquitecturas testeadas como sus resultados en términos de precisión y eficacia.

2.3 Motivación

El **análisis de emociones** ha sido y continúa siendo un tema de interés tanto para individuos como para empresas, debido a que la investigación en este campo puede generar numerosas oportunidades de negocio y tener un gran impacto en la sociedad a nivel mundial.

Al prestar atención a las situaciones cotidianas, es posible darse cuenta de que durante gran parte del día, las personas están siendo observadas por cámaras de seguridad en espacios públicos, establecimientos comerciales o lugares de trabajo. Aún más notorio, aunque a menudo no se tenga en cuenta, el uso de dispositivos móviles personales y corporativos requieren desbloqueo mediante reconocimiento facial (Face ID, Face Unlock, etc.), siendo más del 67% de la población mundial poseedora de un teléfono móvil.

Más allá de esta realidad, a partir del año 2022, tanto *Apple* como *Google*, las dos potencias tecnológicas más importantes del mundo, desarrollaron sistemas de inteligencia artificial capaces de identificar, a través de la galería de fotos del móvil, las personas con las que el usuario se relaciona y crear carpetas con las fotografías que el cliente tiene con sus amistades. Además del ámbito de los dispositivos móviles, esta nueva tecnología de análisis y clasificación de individuos según su rostro facial se ha implementado, por ejemplo, en la búsqueda de personas desaparecidas desarrollada por la asociación *SOSDesaparecidos* en España.

Debido a estas herramientas mencionadas, se ha decidido llevar a cabo un proyecto cuyo objetivo principal **es analizar el rostro facial de los seres humanos**, aportando un enfoque diferencial en comparación al resto, al centrase en la indagación de las cuatro emociones estándar de las personas.

Este proyecto pretende emular una problemática a la que una *Start-Up* tecnológica podría enfrentarse si un establecimiento recreativo deseara estudiar el nivel de satisfacción de sus clientes después de experimentar sus atracciones. Para esta simulación, será necesario crear un entorno empresarial que incluya desde la obtención de las imágenes, pasando por un repositorio en la nube, hasta el informe o dashboard presentado a la alta dirección.

El principal beneficio que ofrece el presente proyecto radica en su aplicabilidad a diversos sectores empresariales. El algoritmo creado y su implementación en producción pueden extrapolarse a múltiples industrias y proporciona una visión clara de las inquietudes y satisfacciones de los clientes.

2.4 Estado del arte

En la revisión del estado del arte, se abordarán los avances más significativos en la identificación **de emociones humanas a través de expresiones faciales**, así como las herramientas y técnicas utilizadas para el correcto análisis de estas.

Para la recopilación de imágenes de personas se utiliza la técnica denominada en inglés como **Web Scraping**. Es una metodología automatizada que consiste en

extraer imágenes de *Google Images* de manera sistemática y estructurada. Es utilizada con frecuencia en el campo de minería de datos, inteligencia artificial y aprendizaje automático para recopilar grandes cantidades de información y utilizarlas para entrenar los modelos de reconocimiento facial.

Según indican (Fernando & Sivarajah, 2015) esta técnica de obtención de información tiene como objetivo convertir los datos no estructurados de páginas web en estructurados para poder almacenarlos y procesarlos ya sea en local o en Cloud. Además, se matiza que existen distintos métodos que incluye en copiar y pegar tradicional, captura de texto, coincidencia de expresiones regulares, análisis HTML (el que se aplica en el presente proyecto), análisis DOM, entre otros muchos.

El proceso que ha de seguirse a la hora de aplicar esta técnica es comprender la estructura de la página web a analizar, diseñar un patrón de expresión regular y finalmente usar ese patrón para obtener ciertos datos. Toda esta información se puede contrastar gracias a (Teguh & Purnama, 2023).

El hecho diferencial del **Web Scraping** aplicado en el presente proyecto en la identificación de patrones en el código HTML de *Google Images* y la descarga de las URLs de las imágenes en un archivo CSV es la obtención de únicamente las direcciones de internet y la no descarga de las imágenes en el propio local en formato jpg.

En cuanto a otra consideración, se hace mención a las imágenes que han sido proporcionadas por la *Universidad de Denver*, específicamente (Mollahosseini et al., 2017). Esta base de datos contiene información relativa a las expresiones faciales en su ambiente natural. Cabe destacar que la particularidad que distingue el uso de este conjunto de datos es que únicamente la mitad de las imágenes han sido categorizadas manualmente, mientras que el resto han sido categorizadas mediante un algoritmo diseñado por el autor, clasificando un total de **4 emociones/expresiones faciales humanas.**

Toda la información recopilada con las técnicas mencionadas anteriormente no es útil si no se realiza un preprocesamiento y limpieza de los datos. Existe multitud de “papers” y tesis enfocadas únicamente en esta fase.

El propósito del preprocesamiento de las imágenes es corregir las inconsistencias de los datos que serán la base del análisis en proceses de redes neuronales, tal y como dice (Olaya-Rodríguez & Suárez-Ruiz, 2014).

Para almacenar la cantidad enorme de imágenes procesadas es necesario crear un repositorio **Data Warehouse** en *Oracle*. Este es la centralización de fuentes de contenido dispares que pueden ser combinadas y procesadas utilizando técnicas en Big Data.

Como bien indican (Zhang & Jagadish, 2016) el gran desafío de hoy en día es estructurar los datos de la misma forma ya que la gran diversidad de fuentes de información provoca a menudo unos esquemas heterogéneos, datos no integrados y miles de silos diferentes.

Existe la aparición de una base de datos que promete garantizar la mejor disponibilidad y tiempos de respuesta posibles sin necesidad de intervención humana, **Oracle Autonomous Database**. Funciona mediante el uso de aprendizaje automático para auto ajustarse, aliviando a los administradores de base de datos (Oracle, 2018).

El campo del *Deep Learning* ha experimentado un rápido crecimiento en los últimos años, con un gran número de investigadores y empresas trabajando en aplicaciones de inteligencia artificial en diferentes ámbitos. Un tema importante en este campo es el procesamiento de imágenes y señales, para lo cual las **redes neuronales convolucionales** han demostrado ser muy efectivas.
Un artículo seminal es este tema (LeCun et al, 1998) en el cual introdujeron las redes neuronales convolucionales y las aplicaron en el reconocimiento de dígitos escritos a mano.

Un libro de referencia en *Deep Learning* (LeCun et al., 2015) el cual proporciona una introducción completa a la teoría y la práctica de dicha metodología. En este

libro, se explican las **redes neuronales convolucionales** y su uso en tareas como el reconocimiento de imágenes y el procesamiento del lenguaje natural.

Sin embargo, hay que tener en cuenta que existe un sesgo en la inteligencia artificial. Los algoritmos de aprendizaje automático pueden ser entrenados con conjuntos de datos que no son representativos de la diversidad de la población, lo que puede llevar a resultados injustos o discriminatorios.

Un ejemplo de este fue el caso de *Google* en 2015, donde su herramienta de reconocimiento de imágenes etiquetó incorrectamente a las personas negras como “gorilas” o “monos” (Salas, 2018). Este incidente fue ampliamente criticado cono un ejemplo de sesgo racial en la inteligencia artificial y llevó a la compañía a retirar la función y emitir disculpas públicas.

Con el fin de evitar la problemática anterior, se ha implementado un sistema totalmente automatizado que se encarga de recolectar, limpiar y procesar las imágenes. Esto se logra mediante la creación de Scripts y del desarrollo de un modelo **Data Warehouse** en Oracle Autonomous Database.

La particularidad del trabajo radica en su enfoque integral, el cual abarca todas las etapas del proceso de gestión de proyectos, desde la recopilación hasta la puesta en producción y la generación de informes: *end-to-end*. La simulación exhaustiva de este proceso permite asegurar la eficiencia y eficacia del **reconocimiento de las cuatro emociones estándar**.

CAPÍTULO 3: METODOLOGÍA

En la presente sección, se presentará en detalle el enfoque utilizado para optimizar el **reconocimiento de emociones en tiempo real**. El proceso comienza con la recopilación de imágenes mediante técnicas de **Web Scraping** y preparación de un conjunto de datos adecuado, que abarque una amplia variedad de expresiones fáciles y emociones. Se considerará la inclusión de diferentes grupos demográficos para asegurar la representatividad de las muestras.

Seguido a la obtención de datos, el paso crucial para la detección de emociones es la limpieza del conjunto de imágenes que se van a utilizar. Esta es esencial para garantizar la homogeneidad, coherencia y fiabilidad de las muestras de entrenamiento y validación del algoritmo creado, ya que existirán imágenes ruidosas e irrelevantes.

El tercer apartado de la metodología, se describe el proceso de creación de la conexión con **Oracle Autonomous Database** para el almacenamiento y gestión de los datos utilizado. La conexión con un **Data Warehouse** es esencial para asegurar la disponibilidad, persistencia y seguridad de los datos recopilados y procesados.

Posteriormente, se describirá la arquitectura de la **red neuronal convolucional** diseñada para el este proyecto. Se explicarán las capas, filtros y estructuras utilizadas, así como los hiper parámetros seleccionados para el entrenamiento del modelo. La elección de la función de pérdida y del algoritmo de optimización será fundamentada, buscando maximizar tanto la precisión como la eficiencia en el reconocimiento de emociones en tiempo real.

La evaluación del rendimiento de la **red neuronal convolucional optimizada** será realizada mediante técnicas de validación cruzada y se utilizarán métricas estándar, como la sensibilidad, especificidad, pérdida, precisión, F1-Score y el tiempo de procesamiento. Además, se explorarán técnicas de mejora de la eficiencia computacional, como la reducción de la dimensionalidad y la

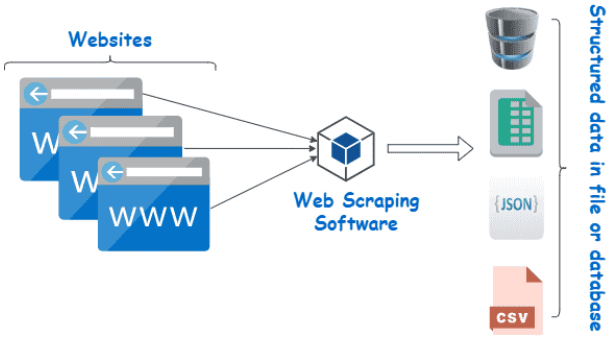
optimización de los hiper parámetros, para asegurar una implementación óptima del sistema de videovigilancia.

En el último apartado, se aborda la explicación del código de puesta en **producción del modelo óptimo** emulando un sistema de videovigilancia utilizando la webcam del ordenador. Este paso es de vital importancia para demostrar la viabilidad y aplicabilidad de la red desarrollada en un entorno práctico y en tiempo real.

3.1 Web Scraping

A continuación, se describirá en detalle el procedimiento emprado para la recopilación de las cuatro emociones específicas mediante el uso del lenguaje de programación Python y técnicas de **Web Scraping** en *Google Images*. Esta metodología se ha empleado con el objetivo de obtener un conjunto de imágenes representativas de cada **emoción facial**. Aunque aparentemente puede resultar buena idea utilizar dicha técnica, hay que considerar ciertos aspectos como la ética y la legalidad, realizando un uso responsable y lícito.

Cabe matizar que el procedimiento que se ha seguido es el mismo para **las cuatro emociones**, únicamente hay que cambiar el texto a consultar es por ese motivo que dicha técnica de recopilación de imágenes se explica de manera general.



Figuras 3. Web Scraping. Fuente: Kinsta

En primer lugar, se debe realizar una preparación del entorno de desarrollo, con esto se refiere a la instalación de bibliotecas para llevar a cabo el procesamiento de imágenes en *Python*. Esto incluye la instalación de paquetes como *selenium* y

webdriver_manager, los cuales permiten la automatización de un navegador web para acceder a *Google Images* y extraer las URLs relevantes.

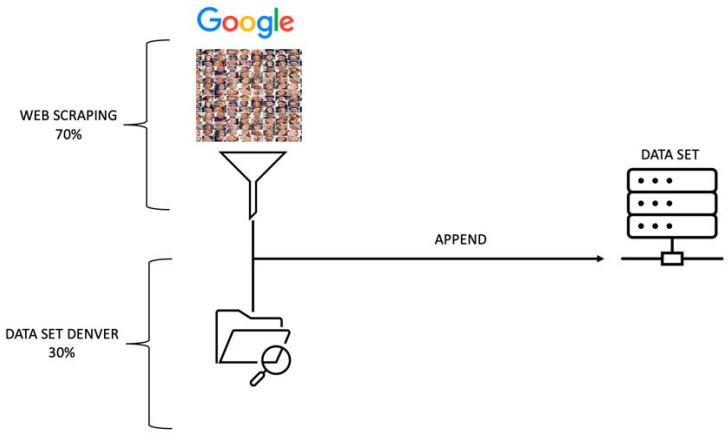
El código proporcionado en el anexo [1], consta de tres funciones que realizan tareas específicas relacionadas con el desplazamiento hacia abajo (scroll) para cargar más imágenes, la construcción de la consulta en *Google* y la actualización de la página, el consentimiento de cookies, y la extracción y guardado de URLs. El objetivo de la primera función, denominada "*scroll_to_end*", es utilizar un controlador web para desplazarse hacia debajo y cargar más fotografías. Esta funcionalidad es necesaria ya que los datos de *Google* se actualizan de manera dinámica según se vaya avanzando.

Posteriormente, la función "*fetch_image_urls*" se encarga de construir la URL de búsqueda en la propia web aplicando la consulta proporcionada como input y/o variable denominada *search_url*. Esta dirección incluye una query en la parte correspondiente y tiene parámetros adicionales, como el tamaño de las imágenes y el número máximo de iteraciones a descargar. Una vez creada, se localiza y hace clic en el botón de cookies haciendo uso de un selector CSS, el cual permite acceder a las fotografías.

Seguidamente, se utiliza la función "*search_and_download*" para iterar a través de todas las imágenes encontradas hasta alcanzar el número máximo establecido y obtener las URLs correspondientes. Una vez completada la acción, todas las direcciones web se extraen y se almacenan en un archivo CSV.

Finalmente, se definen los valores de detección, como "*Famosos sonriendo*" o "*Persona Disgustada*", el número de imágenes a descarga para cada búsqueda y la ubicación del archivo CSV. Además, se crea un DataFrame homogéneo que permite realizar la transpuesta en las URLs, logrando una estructura y etiquetado adecuados para la correcta inserción de estas en el archivo.

La implementación del código anterior previamente explicado se puede llevar a cabo tanto en la consola del ordenador con en un entorno de desarrollo como *Jupyter Notebook.* Se recomienda este último para poder seguir las iteraciones de las consultas y obtener una mejor visión de los resultados obtenidos.



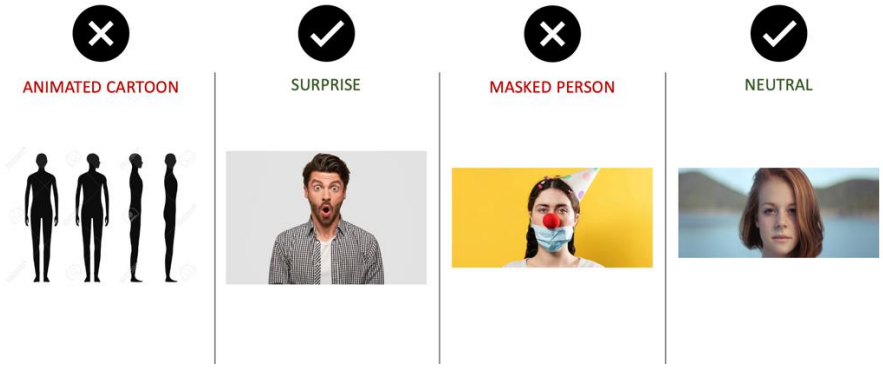
Figuras 4. Recopilación de imágenes.

Es importante destacar que las imágenes obtenidas mediante el uso de técnicas de **Web Scraping** representan aproximadamente el 70% de los datos utilizados para el entrenamiento de las redes neuronales. El 30% restante se obtiene de la base de datos Open Source proporcionada por la *Universidad de Denver* (Mollahosseini et al., 2017), que abarca una amplia variedad de emociones y expresiones faciales, todas ellas debidamente etiquetadas. Esta combinación de datos enriquece el conjunto de datos de entrenamiento, permitiendo a **las redes neuronales** aprender y reconocer una mayor diversidad de **emociones faciales**.

3.2 Image Cleanup

En el ámbito de la investigación y el análisis de datos, es común encontrarse con conjuntos de imágenes que requieren una limpieza y preparación adecuada para su correcta utilización. La calidad y precisión de los datos de fotografías son aspectos críticos para garantizar la validez y confiabilidad de los resultados obtenidos en el presente proyecto.

El script *Image Cleanup*, ubicado en el anexo [2], tiene como objetivo abordar este desafío, presentado un código diseñado para eliminar imágenes incorrectas o irrelevantes de un conjunto de datos. Este proceso permite filtrar observaciones no deseadas, como dibujos animados, fotografías ruidosas o cualquier otro contenido que no se ajuste a los criterios de análisis establecidos.



Figuras 5. Filtración de fotografías.

El código proporcionado en esta sección utiliza técnicas de procesamientos de datos y expresiones regulares para identificar y eliminar las URLs de imágenes inadecuadas. Se aplica sobre un archivo CSV que contiene la información de **las cuatro emociones**, permitiendo generar un nuevo fichero con las imágenes filtradas y preparadas para su posterior uso.

A continuación, se explicará el funcionamiento del código que se encarga de realizar el procesamiento de las imágenes correspondientes a **las cuatro emociones principales** que experimenta un ser humano.

En primer lugar, se importan las bibliotecas necesarias para el procesamiento de datos en *Python,* incluyendo las populares librerías *pandas, numpy y re*. Estas son fundamentales para realizar operaciones de lecturas y manipulación de archivos .csv en el lenguaje de programación. Una vez importadas las librerías, se procede a realizar cambios en la nomenclatura de las variables y a eliminar aquellas columnas del DataFrame que no aportan ningún valor añadido en el proceso de preparación de datos.

Posteriormente, se crea una función encargada de llevar a cabo la limpieza de las URLs de las imágenes. Dicha sintaxis recibe como entrada una lista de valores que representan las direcciones incorrectas o no relevantes. La determinación de estas se realiza ya sea mediante una revisión manual o a través de características específicas extraídas de las propias URLs.

Para garantizar que las URLs exactas sean tratadas como literales en las expresiones regulares, se utiliza la función de escape provista por la biblioteca *re*. Esta técnica permite optimizar la búsqueda de coincidencias estrictas y asegura

que las direcciones web sean correctamente reconocidas, incluyendo elementos como protocolos (http o https) o extensiones de archivo (.png p .jpg).

Una vez que las URLs han sido homogeneizadas y escapadas, se procede a eliminar aquellos registros del DataFrame que contiene valores inusuales, como caracteres especiales (@ o #) o valores vacíos. Como resultado, se obtiene un conjunto de datos limpio que contiene únicamente imágenes que representa **las cuatro emociones** a analizar: *alegría, disgusto, sorpresa y neutral*.

Por último, se muestran en pantalla las métricas relevantes que indican el tamaño del nuevo conjunto de datos, el número de URLs correspondientes a cada emoción y la cantidad de estas eliminadas durante el proceso de limpieza. Además, se crea un nuevo archivo .csv que contendrá los datos procesados, el cual será importado posteriormente en el **Data Warehouse** de la **Autonomous Database de Oracle**.

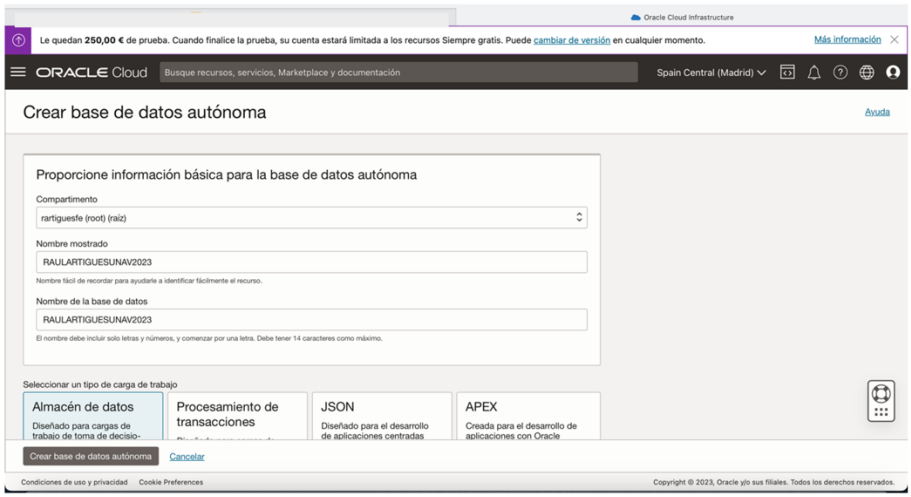
3.3 Connection to Autonomous Oracle Database

La creación de un **Data Warehouse**, anexo [3], en este caso en *Oracle*, juega un papel esencial en los proyectos de Data Science por su capacidad para albergar, organizar y gestionar grandes volúmenes de datos procedentes de diversas fuentes y en diferentes formatos. Este repositorio centralizado de información es crítico para la toma de decisiones basadas en datos, ya que proporciona una visión integral y coherente de las operaciones de la *Start-Up.* Al facilitar el acceso a los datos históricos y actuales, un **Data Warehouse** permite a los científicos de datos realizar análisis exhausticos, descubrir patrones u tendencias, y obtener valiosos insights para informar las estrategias de negocio. Además, la estructura de los datos facilita su procesamiento y análisis mediante algoritmos de aprendizaje automático, abriendo así las puertas de la implementación de soluciones de inteligencias artificial.

El propósito de esta sección es detallar los métodos y procedimientos que se han seguido para el desarrollo de la base de datos y el procesamiento de las imágenes. La metodología empleada comprende varias etapas, que incluyen la configuración de la base de datos en *Oracle*, su conexión remota, la creación de

tablas, la inserción de URLs de imágenes procesadas y limpias en las tablas, la realización de consultas SQL y la correcta extrapolación de las direcciones descargadas a archivos con la extensión apropiada: jpeg y jpg. Además, se verifica la integridad y la funcionalidad de la base de datos autónoma de *Oracle*.

En primer lugar, para la creación de una **base de datos autónoma** es necesario crearse un usuario en *Oracle Cloud* () para posteriormente iniciar una nueva instancia de la base de datos en la nube. Durante este proceso, se pedirá que se especifiquen los detalles como el nombre del **Data Warehouse**, el tipo de carga de trabajo (almacenamiento de datos), la cantidad de núcleos de CPU y la cantidad de memoria necesaria.



Figuras 6. Creación base de datos autónoma de Oracle.

Una vez creada la instancia, el siguiente paso es configurar las medidas de seguridad para la base de datos. Esto implica la generación de credenciales, que incluyen un nombre de usuario y una contraseña, y la descarga de las credenciales de red: archivo Wallet con extensión ora. Este archivo es crucial para la conexión segura a la base de datos desde aplicaciones cliente, y se debe crear una propia carpeta en el ordenador del cliente, donde se ubiquen la carpeta de *instantclient_19_8*, el *host*, el *port* y el *dns.*

Con las credenciales de red y de usuario, ya se está en posición de conectarse a la base de **datos autónoma de Oracle**. En el caso del código *Python* creado, los detalles de la conexión se den proporcionar de la siguiente manera:

1- Especificar el directorio de la carpeta *instantclient_19_8*
2- Nombre de usuario: *User*
3- La contraseña: *Password*
4- El DNS: Que puede tener propiedades: *High*, *Medium* o *Low*

Una vez descritos los settings a establecer en *Oracle Cloud* se procede a la explicación del proceso de implementación de la nube en el presente proyecto. Para comenzar, es necesario instalar la librería *cx_Oracle* en *Python* utilizando el comanda *pip install* que se debe utilizar en la propia consola/terminal del ordenador. Esta biblioteca es fundamental para establecer la conexión con *Oracle* y el propio ordenador. Otras librerías importantes son *io*, que sirve para flujos de bytes, *hashlib* que se utiliza para transformar datos a una cadena de longitud fija de caracteres y *request* para realizar solicitudes HTTP.

En primer lugar, se conecta a la base de datos de *Oracle* a través de las propiedades del DNS, que pueden ser Bajas, Medias o Altas dependiendo de las necesitades del usuario. Este procedimiento se realiza utilizando funciones como *os.chdir* para obtener la ruta del directorio donde se encuentra la carpeta de *instantclient_19_8* y *cx_Oracle.connect.* A continuación, se crea la tabla de “SMILE” utilizando la función *cursor.execute.* Es importante mencionar que este proceso se repite tantas veces como emociones se quieran detectar, en el presente caso, 4: *alegría, tristeza, sorpresa y neutral*. Estas tablas están compuestas por dos atributos: ID (identificador de cada registro) y URL (la variable objetivo).

Una vez creadas las tablas, se insertar las URLs de las imágenes procesadas y limpiadas en la tabla correspondiente. Para ello, se utiliza un archivo csv titulado “Emotions.csv”, que se abre y lee mediante la biblioteca csv que está disponible en *Python*. Las direcciones de las fotografías se insertar en la tabla mediante la función *cursor.executemany*. Este proceso de inserción se repite tantas veces como etiquetas de emociones haya.

Para verificar el correcto funcionamiento de las tablas y la correcta inserción de las URLs, se llevan a cabo consultas SQL de prueba. En este caso, se seleccionan direcciones de la tabla “SMILE” mediante la función de *cursor.execute*. Los resultados se almacenan en una lista y se transformar en un DataFrame para su posterior análisis y verificación.

Finalmente, se realiza una transformación de las URLs en imágenes en formato jpeg/jpg y se descargan al equipo. Esta etapa es vital para la visualización y análisis de las imágenes en etapas posteriores del proyecto. El código para este proceso incluye la definición de una función “*download_and_save_image*”, que descarga y guarda las imágenes proporcionadas en un directorio específico. Esta función utiliza la biblioteca *requests* para obtener la URL de la imagen, la librería *PIL* para abrir el archivo y *hashlib* para generar una clave hash única para cada fotografía. Esta clase se utiliza como nombre de archivo para la imagen descargada.

Figuras 7. Descarga de imágenes de Oracle.

Cada tipología de las fotografías se guarda en carpetas diferentes, así ya se tienen las imágenes etiquetas para su posterior implementación en **las redes neuronales** y puesta en **producción**.

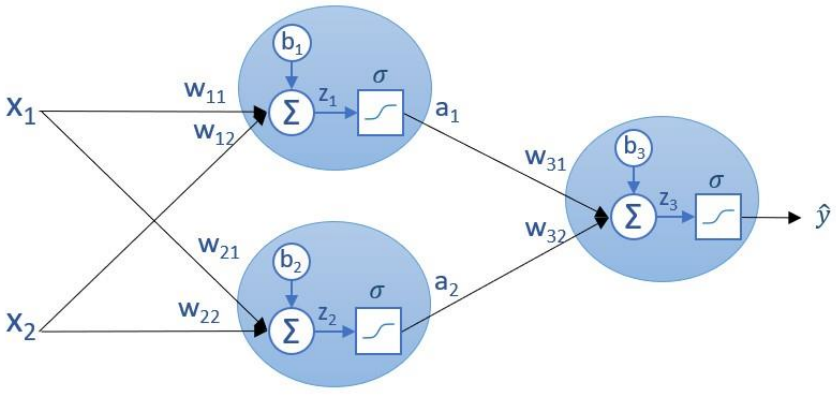
3.4 Convolutional Neural Networks

En el corazón del presente proyecto innovador, se encuentra una tecnología que ha sido la pieza clave en el avance de la inteligencia artificial durante la última década; las redes neuronales artificiales. En este caso, se centra en un tipo específico, **Redes Neuronales Convolucionales**, que será fundamental para detectar y clasificar **emociones humanas**.

En la actualidad está de modo decir inteligencia artificial o redes neuronales, pero realmente, ¿**Qué es una Red neuronal**?

Las redes neuronales artificiales son un paradigma de aprendizaje automático inspirado en el sistema nervioso de los seres humanos, en particular en el cerebro, el cual está compuesto de billones de células interconectados llamadas neuronas. Las **ANN** intentan simular este comportamiento y estructura de las neuronas bilógicas para realizar tareas de aprendizaje.

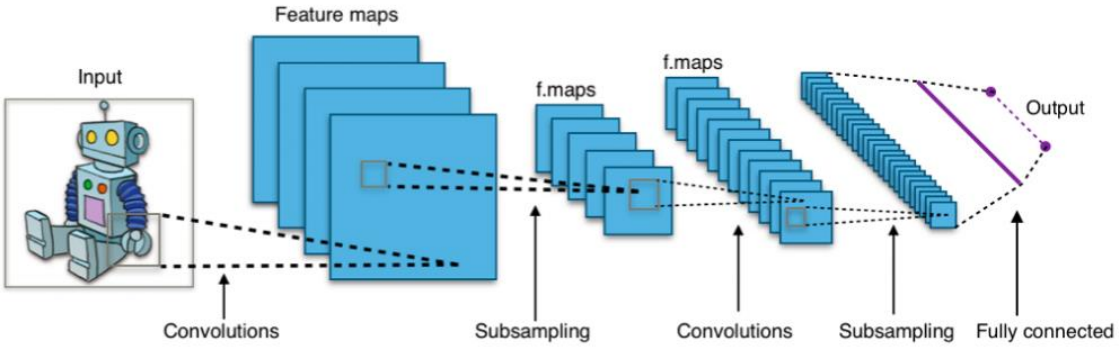
Las **ANN** están formadas por neuronas interconectas y organizadas en diferentes capas: capa de entrada, una o más capas ocultas y capa de salida. Cada neurona realiza cálculos simples sobre los datos de entrada, y el poder de estas redes proviene de la conexión de muchas neuronas y su capacidad para aprender patrones a partir de los datos de entrada.



Figuras 8. Representación de una red neuronal y sus pesos. Fuente: InteractiveChaos.

En el contexto del proyecto, se utiliza las **Redes Neuronales Convolucionales**, un tipo de especializado de **ANN** que ha demostrado ser extremadamente eficaz para el procesamiento de imágenes y la visión por computadora, debido a su capacidad para extraer automáticamente y aprender características de las fotografías, que son luego utilizadas para la detección y **clasificación de las 4 emociones principales en los rostros humanos**: *alegría, tristeza, sorpresa y neutralidad.*

Las **CNN** se estructuran en varias capas, cada una con una función específica. Normalmente, este tipo de redes tienen tres tipos de capas. En primer lugar, se encuentran las *capas convolucionales* que aplican una serie de filtros a la entrada. Cada filtro se activa en presencia de una característica visual particular, como un borde, una textura o un color. Seguidamente a estas vienen las *capas pooling* que se utilizan para reducir la dimensionalidad de los datos, manteniendo las características más importantes. Esto hace que el modelo sea más eficiente y menos propenso al sobreajuste. Las últimas capas que se encuentran al final de la red son las denominadas capas *fully connected.* Estas se utilizan para clasificar los datos en función de las características extraídas por las *capas convolucionales* y *pooling*. La salida de estas capas indica la probabilidad de que la imagen de entrenada pertenezca a cada una de las clases.



Figuras 9. Representación de una red neuronal convolucional. Fuente: Pocho Costa

En la búsqueda de la creación de un sistema eficiente y preciso para la detección de emociones, se opta por dos enfoques distintas pero complementarios para el diseño y entrenamientos de las **redes neuronales convolucionales**. Estas estrategias implicando tanto el desarrollo de una **CNN desde 0** como la utilización de la técnica de **Transfer Learning** con la arquitectura pre-entrenada de *ResNet*.

La primera estrategia que se aborda en el presente proyecto es diseñar y entrenar una propia **CNN desde 0**. Este enfoque brinda una gran flexibilidad para adaptar la arquitectura de la red a las necesidades específicas y garantizar una correcta optimización de la clasificación.

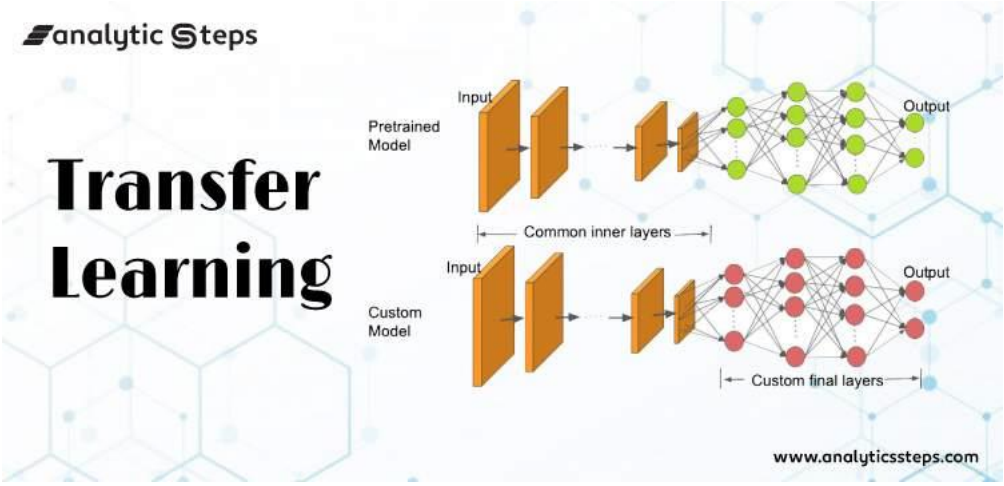
Para el entrenamiento, se utiliza un conjunto de datos compuesto por una combinación de imágenes extraídas mediante una aplicación de **Web Scraping** y un conjunto de datos proporcionado por la *Universidad de Denver*. Las imágenes recolectadas de *Google Images* constituyen el 70% de los datos, y la universidad el 30%.

El proceso de entrenamiento consiste en alimentar la **red neuronal convolucional** con estas imágenes y ajustar los pesos de la red a través de múltiples iteraciones para minimizar la diferencia entre las predicciones y las emociones reales. A través de este proceso, la red aprende a reconocer patrones y características que son indicativos de las diferentes emociones que se tratan de clasificar.

Además de la arquitectura personalizada de la **CNN**, también se experimenta con una estrategia de **Transfer Learning** utilizando *ResNet*, una arquitectura de convolución pre-entrenada muy conocida y probada. Esta técnica es un enfoque en el que se utiliza un modelo entrenado como punto de partida para una nueva tarea relacionada. En el presente caso, *ResNet* ya ha sido entrenada con millones de imágenes y ha aprendido a extraer una gran cantidad de características visuales útiles.

Al aplicar **Transfer Learning** con *ResNet*, se puede extraer características que ha aprendido, y ajustarlas a la tarea de clasificar emociones. Esta metodología puede ser particularmente útil cuando se dispone de un número relacionalmente pequeño de datos de entrenamiento (menos de 20.000 registros), ya que aprovecha el conocimiento previamente aprendido por el modelo en un conjunto de datos mucho más grande (millones).

Para implementar esta técnica se debe modificar la última capa de la red pre-entrenada y añadir más capas ocultas (convolucionales y densas) para aprender sobre los datos que se poseen.



Figuras 10. Transfer Learning. Fuente: Analytics Vidhya

Para el desarrollo de este proyecto, se ha seguido una metodología estructurada y progresiva, la cual permite explorar y entender de manera gradual la eficacia de las arquitecturas de **redes neuronales convolucionales** tanto personalizada como la que utiliza el **Transfer Learning** con *ResNet*. A continuación, se detallan los pasos que se han seguido:

Clasificación binaria (alegría y tristeza): Se comienza el trabajo creando dos arquitecturas de red, para abordar el problema de clasificación binaria, es decir, clasificar **las imágenes en dos categorías**: *alegría y tristeza*. Esta etapa proporcionada una excelente oportunidad para afinar las dos arquitecturas y entender cómo cada una de ellas aprende y clasifica emociones.

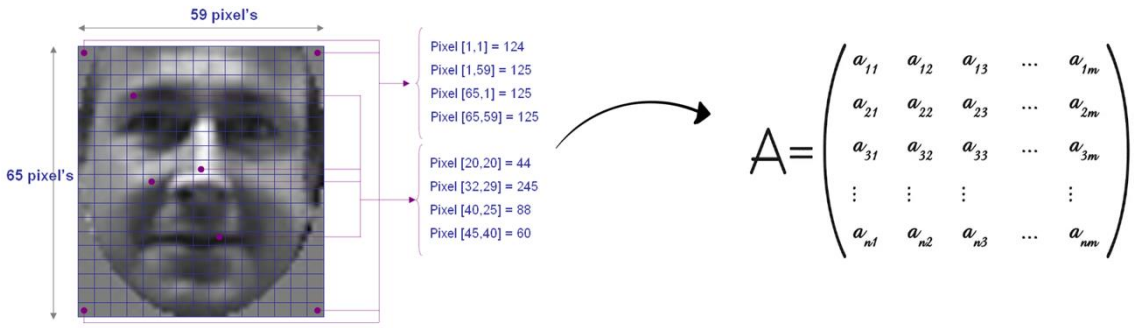
Clasificación de 3 emociones (alegría, tristeza y sorpresa): Una vez completado la clasificación binaria, se crean las primeras **redes neuronales multiclase**. En este caso, el objetivo es clasificar las imágenes en tres emociones: *alegría, tristeza y sorpresa.* Este paso permite comprender como entrenar **redes neuronales convolucionales multiclase** y que desafíos supone.

Clasificación de 4 emociones (alegría, tristeza, sorpresa y neutralidad): Finalmente, se amplían las arquitecturas de redes para clasificar las imágenes en 4 emociones. Estas arquitecturas, concretamente, la **red neuronal convolucional** utilizando *ResNet* es la utilizada para el modo de **producción**.

La construcción y entrenamiento de una red neuronal es un proceso detallado y multifacético. A continuación, se embarca en la tarea de desarrollar una red **neuronal convolucional** multicapa para reconocer y **clasificar cuatro emociones distintas**: *alegría, tristeza, sorpresa y neutralidad*. Este proceso se realiza mediante la implementación del modelo de **red neuronal convolucional ResNet50**, ubicada en el anexo [4].

En un primer lugar se realiza una importación de módulos. Estos incluyen *os* y *cv2* para el manejo de archivos y la manipulación de imágenes, *numpy* y *pandas* para el manejo de datos, *matplotlib* para la visualización, *keras* y sus módulos para la construcción y el entrenamiento de la red neuronal, y módulos de *sklearn* para dividir los datos y realizar validación cruzada.

Seguidamente y, mediante la función de "*load_images*" se cargan las imágenes de cada categoría de emociones desde una carpeta específica. Estas se convierten al espacio de color RFG, se redimensionar a 100x100 pixeles y ser realiza un preprocesamiento de los datos, es decir, se normalizan mediante la división de 255 (rango de valores de los píxeles).



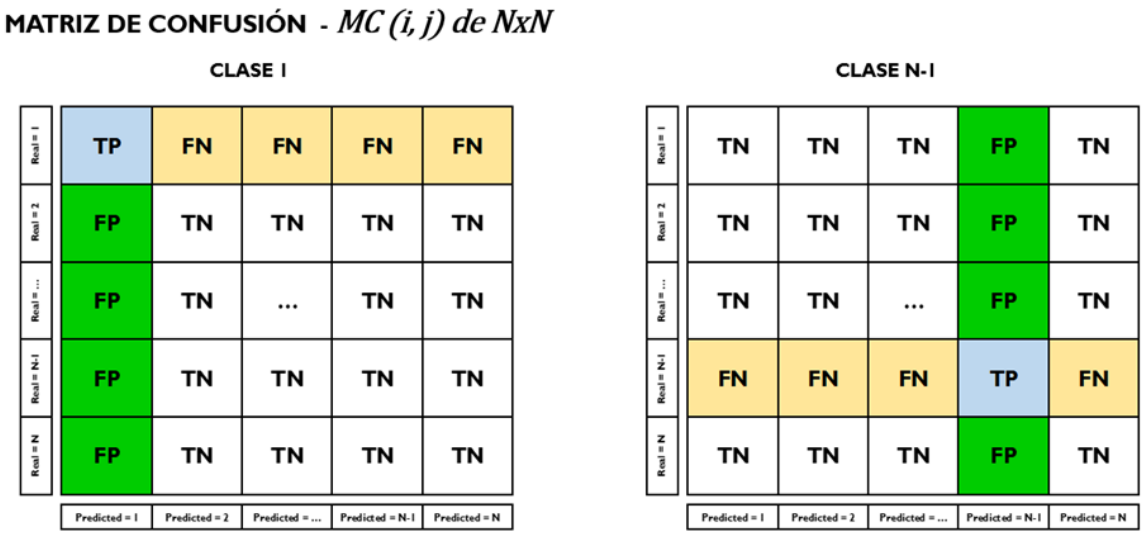
Figuras 11. Normalización de imágenes.

A continuación, se carga el modelo pre-entrenado de *ResNet50*, luego se añade las capas adicionales para adaptar el modelo a la tarea específica. Esto incluye capas de relleno (*padding*), convoluciones, de agrupación (*pooling*) y

complementado conectadas (*dense*). La última capa es una densa con 4 neuronas (una para cada categoría de emociones) y la función de activación *softmax* para la clasificación.

Seguidamente, el modelo se compila con el optimizador *Adam*, la función de pérdida de entropía cruzada categórica y el seguimiento de la métrica de precisión. El modelo se entrena en el conjunto de entrenamiento durante un máximo de 100 *épocas*, utilizando una parada temprana para finalizar el proceso si la pérdida de validación no mejora durante 5 *épocas* consecutivas. Los resultados se almacenan en la variable *history* para su análisis posterior.

Por último, se evalúan las métricas de entrenamiento de la **red neuronal** para ver si los valores predichos se ajustan con la realidad y observar si el modelo está aprendiendo de la forma adecuada o existe un sobreajuste. Además, con la utilización de la función *predict* del modelo se crean predicciones de las etiquetas de las imágenes para calcular, finalmente, la matriz de confusión. Esta compara las etiquetas reales con las etiquetas predichas y produce un resumen de los resultados de la clasificación.



Figuras 12. Matriz de confusión multiclase. Fuente: Wayner Barrios Bustamante

Una vez se ha entrenado la **red neuronal convolucional** y su respectiva evaluación, se guarda el modelo mediante la función *save* que incluye *keras* en formato *h5*. Esta incluye la arquitectura del modelo, los valores de los pesos del modelo después del entrenamiento, la configuración de entrenamiento

(optimizador, función de pérdida, etc.), y el estado del optimizador, lo que permite continuar el entrenamiento exactamente donde se dejó.

El siguiente paso es guardar todas las métricas generadas en el entrenamiento, los *hiper parámetros* establecidos y los valores de los pesos en un archivo csv para posteriormente realizar un análisis exploratorio de datos en el **reporting** de *Power BI*.

El último fragmento del presente código creado para la detección de las **4 emociones** principales que experimentan los seres humanos mediante la utilización de **redes neuronales convolucionales** es la realización de prueba con imágenes del propio desarrollador del presente proyecto.

Inicialmente, se carga una imagen usando la función *imread* de la librería *OpenCV*, se especifica la ruta de la imagen y se aplica un filtro de color. Posteriormente, se redimensionar la imagen a 100x100 píxeles y se normaliza mediante la división de 255 para así conseguir que todos los valores de los píxeles se encuentren entre 0 y 1, lo cual es un requisito común para los modelos de *Deep Learning*.

A continuación, el modelo se utiliza para hacer una predicción en la imagen preprocesada. Aquí, la fotografía se pasa como una lista de un elemento para cumplir con el formato de entrada de la red entrenada. Una vez hecha la predicción, se verifica la mayor probabilidad de las clases existentes. Si la probabilidad de una clase es superior al 0.5, se selecciona esa etiqueta y se imprime, junto con la probabilidad específica.

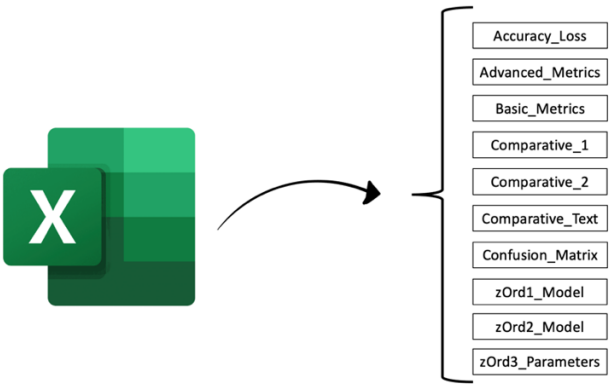
Finalmente, la imagen se muestra junto con la etiqueta predicha utilizando *Matplotlib*. La función *imshow* se usa para mostrar la imagen, y la función *title* para mostrar la etiqueta de la emoción predicha. Con este enfoque, es posible tomar cualquier imagen nueva, predecir su etiqueta utilizando el modelo óptimo entrenado y visualizar el resultado.

3.5 Metrics from CNN to Oracle

En el presente estudio, se realiza un análisis exhaustivo de los datos generales a partir de una serie de modelos de **redes neuronales convolucionales**: CNN binario y multiclase. Estos modelos, entrenados y probados con diferentes configuraciones e *hiper parámetros,* proporcionan una gran cantidad de métricas e indicadores de rendimiento que son esenciales para comprender la eficacia y eficiencia de las redes. El análisis de estos valores puede proporcionar información crucial para la mejora y optimización de futuros modelos de **CNN**.

Para manejar eficientemente este volumen de datos, se emplea una **base de datos autónoma de Oracle,** que proporciona una plataforma de gestión de datos segura, flexible y escalable. Con la ayuda de este recurso, se puede almacenar, gestionar y recuperar las métricas generadas por las distintas **redes neuronales convolucionales** creadas.

El código presentado en el anexo [5], se detalla el proceso utilizado para importar los datos obtenidos de las **CNN** a la base de datos de *Oracle*. El primer paso en este proceso es establecer una conexión con la base de datos que se creó previamente (*3.3 Connection to Autonomous Oracle Database*) mediante la utilización del módulo *cx_Oracle* de *Python*. Luego, se crean tantas tablas como emociones se quieran detectar por 2. Esto hace un total de 8 tablas más 2 que se encargan de interconectarlas todas entre sí. Por tanto, hay que crear 10 tablas diferentes en *Oracle*. Para entender mejor el procedimiento, se explica la creación de la tabla "*Accuracy and Loss*", que almacena la información sobre la precisión y la pérdida de los modelos en las etapas de entrenamiento y prueba.



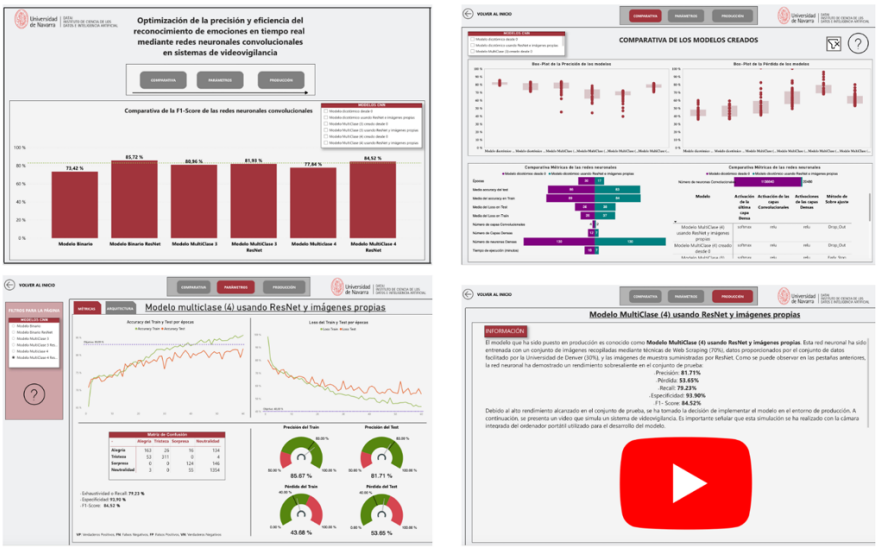
Figuras 13. Estructura del fichero de métricas de la CNN.

Posteriormente, se utiliza la biblioteca *openxl* para abrir un archivo Excel que contiene los datos a importar a la base de datos. Estos datos son insertados en la tabla correspondiente, en este caso, "*Accuracy and Loss*" por lotes, para optimizar el rendimiento y minimizar la carga en la base de datos.

Finalmente, se realizan consultas SQL a la base de datos para recuperar los datos almacenados. Estos datos son luego almacenados en un DataFrame de *pandas* para comprobar que la importación de las métricas e hiper parámetros ha sido correcta.

Estos datos recopilados en la base de datos *Oracle* son posteriormente empleados para generar un **cuadro de mando** en *Power BI* que se visualiza en *Wix*. Esta herramienta permite visualizar y analizar los datos de forma interactiva, facilitando la interpretación de los resultados y permitiendo la toma de decisiones basada en los datos. De esta forma, este enfoque combinado permite un análisis completo y eficaz de los datos generados por las **redes neuronales convolucionales**, lo que conduce a una mejor comprensión y mejora de estos complejos modelos de aprendiza automático.

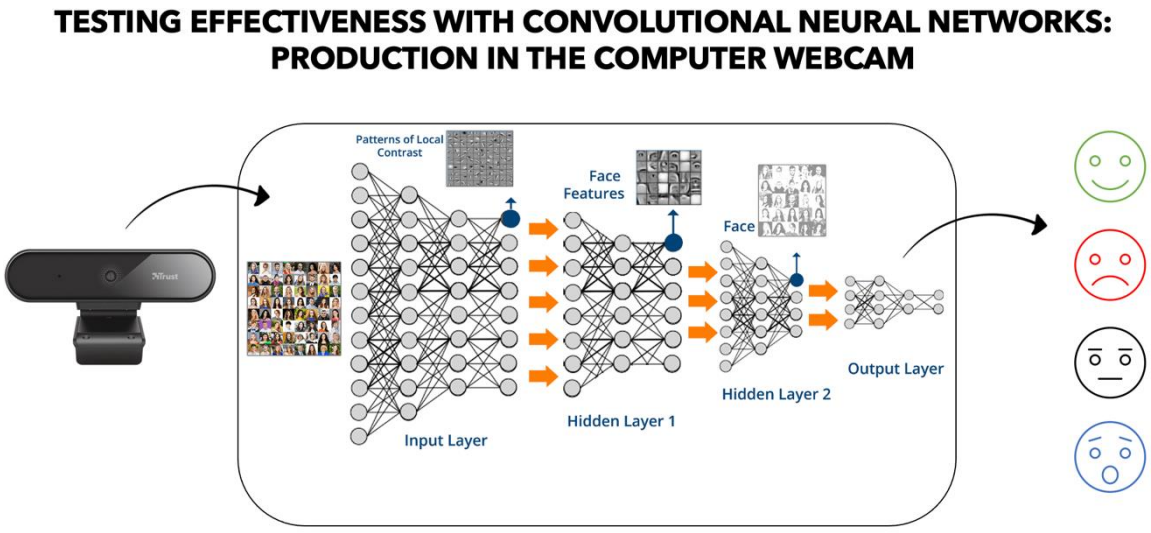
CUADRO DE MANDO DEL PROYECTO



Figuras 14. Cuadro de mando en Power BI

3.6 Production of the Convolutional Neural Network

En esta fase final del proyecto, se pone en **producción el modelo de red neuronal convolucional** entrenado previamente para **detectar** y **clasificar** las **emociones humanas** presentes en los rostros. En particular, este modelo se ha diseñado para identificar cuatro emociones distintas: *alegría, tristeza, sorpresa y neutralidad*. La implementación se realiza a través de un cámara de ordenador en tiempo real, lo que ofrece una prueba tangible de la eficacia del modelo en un entorno de uso práctico y en tiempo real simulando un servicio de video vigilancia para cualquier establecimiento.



Figuras 15. Diagrama de flujo del modelo en Producción.

El código proporcionado, ubicado en el anexo [6], detalla la implementación de esta fase de producción. En primer lugar, se importan las librerías necesarias, incluyendo *OpenCV* para el procesamiento de imágenes y video y *TensorFlow* para la carga y uso del modelo **CNN** previamente creado, entrenado y guardado. Luego se cargan dos modelos: la modelo cascada de *Haar* para la detección de rostros y el modelo de **red neuronal.**

Después de iniciar la captura de video de la cámara del ordenador con *cv2.VideoCapture(0),* se entre en un bucle que procesa continuamente cada fotograma del video. Para cada uno de estos, se aplica el modelo de detección de rostros para identificar cualquier aspecto en el cuadro. Luego se extrae la región

de interés (ROI) que contiene cada rostro detectado y se prepara esta imagen para ser procesa por el modelo **CNN**.

El modelo **CNN** entonces recibe esta imagen de entrada y emite una predicción de la emoción presente en el rostro. En este punto, el programa examina la salida del modelo para determinar qué emoción tiene la mayor probabilidad según la predicción del modelo. A continuación, se superpone en el fotograma original un texto indicativo de la emoción detecta, junto con una caja alrededor del rostro. Este fotograma anotado se muestra en la pantalla **en tiempo real.**

El bucle continúa procesando fotograma y mostrando las emociones detectas hasta que el usuario interrumpe el proceso, en cuyo momento el programa libera la cámara y cierra la venta de visualización.

En resumen, este código demuestra la capacidad del modelo de **red neural convolucional** para detectar y clasificar emociones en **tiempo real** en un entorno de uso práctico. Este es un logro significativo y representa un hito importante en la verificación de la efectividad del modelo en **la detección de emociones humanas**: *alegría, tristeza, sorpresa y neutralidad.*

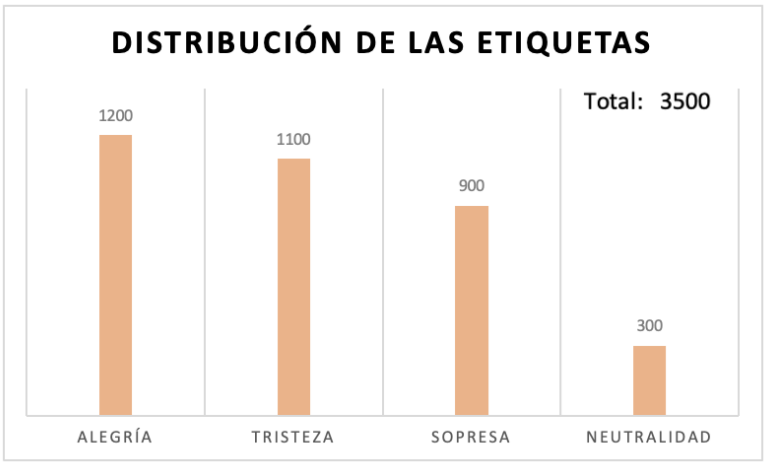
CAPÍTULO 4: RESULTADOS

El capítulo actual se dedica a presentar y analizar los resultados obtenidos a lo largo del proceso de investigación de este trabajo final de máster. Se han divido estos resultados en cinco subsecciones, cada una reflejando una etapa distinta del procedimiento empleado para llevar a cabo este estudio. Esta estructuración permite una descripción clara y detallada del proceso completo, desde la recolección inicial de los datos hasta el análisis final de los mismos mediante modelos de aprendizaje profundo con variados grados de complejidad.

4.1 Recopilación de imágenes

El primer paso en la presente investigación en la recopilación de datos a través de la técnica de **Web Scraping**, específicamente en *Google Images*. Este proceso consiste en recopilar automáticamente URLs de imágenes que correspondan a diversas **emociones humanas**: *alegría, tristeza, sorpresa y neutralidad*.

La recolección de datos arroja un total de 1200 imágenes de *alegría*, 1100 de *tristeza*, 900 de *sorpresa* y 300 de *neutralidad*. La cantidad variable de fotografías refleja la disponibilidad relativa de imágenes para cada emoción en *Google Images.* El conjunto de datos final, por tanto, estuvo compuesto por 3500 sin aplicar el procesamiento de estas.



Figuras 16. Distribución etiquetas Web Scraping.

Una característica destacada del conjunto de imágenes recolectadas a través de esta técnica es la variabilidad intra-emocional. Esto se refiere a la diversidad de expresiones faciales y situaciones que se pueden encontrar dentro de una única

categoría emocional. La recopilación en *Google Images*, ofrece una amplia gama de representaciones para cada emoción, proporcionando una variedad enriquecedora que refleja las múltiples formas en que una emoción puede ser expresada y percibida.

Por ejemplo, en la categoría de *alegría*, se pueden encontrar imágenes que van desde sonrisas sutiles hasta risas abiertas y exuberantes. De manera similar, las fotografías de *tristeza* pueden incluir expresiones de leve descontento hasta llanto y angustia. Esta variabilidad *intra-emocional* es una fortaleza del conjunto de datos, ya que permite a los modelos aprender a reconocer una gama más amplia y realista de **expresiones emocionales**.

Figuras 17. Imágenes descargadas Intra-emocionales.

En cuanto a la composición de las imágenes, se observan ciertas particularidades. Algunas de ellas presentan a las personas cubriéndose la cara, con gafas, o una combinación de estas, lo que añade un grado de dificultad en la interpretación de las expresiones faciales. Además, aunque la gran mayoría de estas (aproximadamente el 90%) son a color, también se incluyen algunas imágenes en blanco y negro.

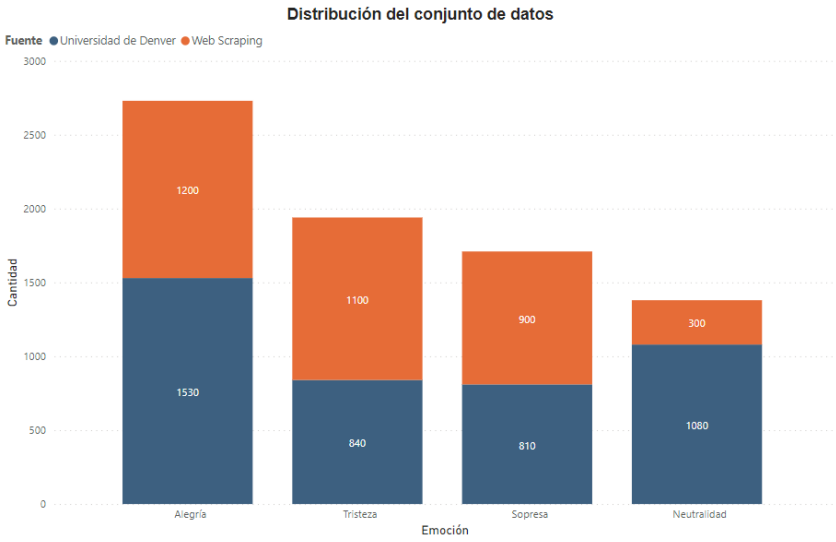
Cabe destacar que la mayoría de las imágenes recopilas a través del **Web Scraping** no solo muestran la cara, sino aproximadamente la mitad del cuerpo. Aunque esto va más allá de los necesario, la presencia de gestos corporales y posturas provoca un fuerte ruido en el reconocimiento de emociones, es por eso motivo que se aplicará posteriormente un preprocesamiento de datos.

Para contrarrestar las limitaciones asociadas a la recopilación de imágenes mediante **Web Scraping**, y particularmente para abordar el desequilibrio en el número de observaciones, se recurre a un recurso adicional de alta calidad, la base de datos de **rostros humanos** ofrecido por la *Universidad de Denver*.

Esta base de datos es reconocida por su rigor y exhaustividad, ofreciendo un conjunto significativo de imágenes de **expresiones faciales humanas** categorizadas por emoción. Este conjunto de datos representa una valiosa adición al conjunto de imágenes recolectadas a través de *Google Images*, ya que aporta una mayor uniformidad y control en términos de calidad y relevancia de las fotografías.

De esta manera, el número de datos para cada emoción aumenta de manera significativa. Se agregan 5100 imágenes para la categoría de *alegría*, 2800 para *tristeza*, 2700 para *sorpresa* y 3600 para la *neutralidad*. Sin embargo, es importante destacar que solo se ha decidido utilizar el 30% de estas imágenes en cada categoría de emoción, buscando asegurar un control más estricto sobre la calidad y la relevancia de las imágenes utilizadas en el presente estudio.

Este criterio de selección permite no solo incrementar el volumen total de los datos, sino también equilibrar de manera más efectiva la representación emocional, lo cual es fundamental para el entrenamiento y evaluación de los modelos de **redes neuronales convolucionales**. Al optar por un subconjunto del total de imágenes disponibles, se tiene la oportunidad de manejar un conjunto de datos más reducido y de calidad controlada al mismo tiempo que se preserva la diversidad y representatividad de las emociones a clasificar.



Figuras 18. Distribución Data set original.

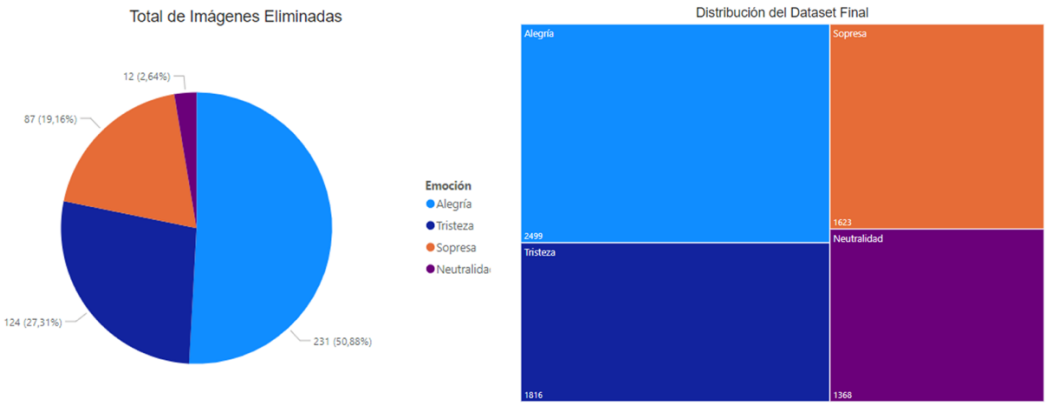
A pesar de los desafíos asociados a la variabilidad de las imágenes recopiladas mediante **Web Scraping** y a los desequilibrios iniciales en el número de observaciones por categoría, se consigue un conjunto de datos robusto y equilibrado. Esta base de datos no sólo abarca una amplia gama de manifestaciones de cada emoción, sino también incorpora variaciones en el formato y la composición de imágenes, siendo un total de 7760, divididas en 2730, 1940, 1710 y 1380, respectivamente.

4.2 Preprocesamiento de los datos

El preprocesamiento de los datos es un paso crucial en cualquier análisis de datos. En este caso, resulta especialmente importante dada la diversidad de imágenes recopiladas a través del **Web Scraping**. Para abordar este desafio, se ha diseñado e implementado un proceso de limpieza de datos que permite filtrar y eliminar aquellas imágenes que no son relevantes o adecuadas para el proyecto. Este proceso tiene como objetivo principal **la eliminación de fotografías incorrectas** o irrelevantes, como dibujos animados, figuras ruidosas o cualquier otro contenido que no se ajuste a los criterios establecidos.

Figuras 19. Fotografías no deseadas.

Este procedimiento se basa en técnicas de procesamiento de datos y en el uso de expresiones regulares para identificar y eliminar las URLs de las imágenes inadecuadas. Dicha fase se aplica sobre un archivo csv que contiene la información de las **4 emociones,** generando un nuevo archivo con las imágenes filtradas y listas para su posterior uso.



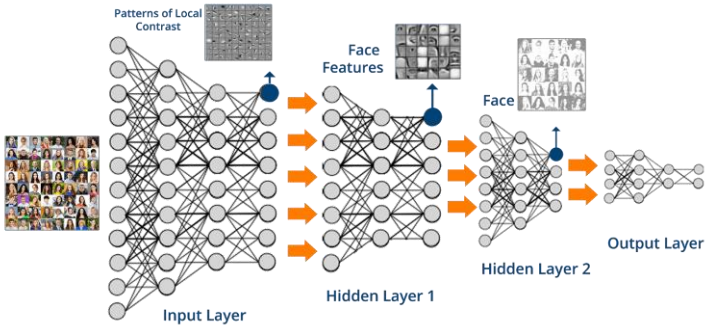
Figuras 20. Distribución del conjunto de datos definitivo.

En la aplicación de este proceso, se han obtenido resultados significativos que han contribuido a la mejora de la calidad del conjunto de datos. Específicamente, se han eliminado 231 imágenes de *alegría*, 124 de *tristeza*, 87 de *sorpresa* y 12 de *neutralidad*, quedando un conjunto de datos final de 7306 observaciones. Este proceso de limpieza ha permitió mejorar el ruido de la base de datos creada, eliminando las imágenes que pueden introducir un sesgo en los **modelos de clasificación emocional**. Así, este proceso ha contribuido a la creación de un conjunto de datos más coherente y adecuada para poder aplicar **redes neuronales convolucionales.**

4.3 Modelos de las redes neuronales convolucionales

En esta sección, se presentan los resultados obtenidos al aplicar **modelos de redes neuronales convolucionales** para el reconocimiento y clasificación de **emociones humanas**. El enfoque se centra en la utilización de 3 arquitecturas de modelos diferentes para abordar la tarea de clasificación emocional**: un modelo binario** para distinguir entre *alegría* y *tristeza*, un modelo **multiclase** para identificar *alegría*, *tristeza* y *sorpresa*, y **otro modelo multiclase** para incluir además la categoría *neutralidad*.

El uso de las redes **neuronales convolucionales** en la clasificación de emociones ha demostrado ser eficaz debido a su capacidad para aprender características relevantes directamente de las imágenes. Estas arquitecturas se aplican en diversas tareas de reconocimiento visual, y su adaptación al análisis emocional se ha convertido en un área de investigación activa.



Figuras 21. Red Neuronal Convolucional. Fuente: Developers Breach

4.3.1 Modelo Binario: Alegría y Tristeza

El modelo binario óptimo utilizando para distinguir entre las **emociones** de *alegría* y *tristeza* se basa en la implementación de **Transfer Learning** con *ResNet50*, una red neuronal convolucional pre-entrenada en el conjunto de datos de *ImageNet*. A continuación se presenta un resumen de la arquitectura.

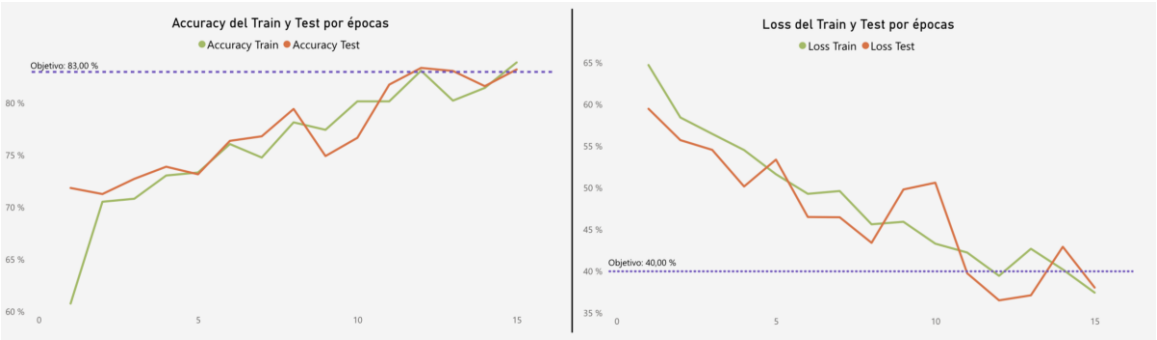
Los pesos del modelo se mantienen congelados, lo que significa que no se actualizan durante el entrenamiento. Esto aprovecha los conocimientos previos adquiridos por el modelo entrenado en *ResNet*. Se añaden capas adicionales, aplicando capas de *ZeroPadding2D*, convolucionales con 32 y 64 filtros y funciones de activación *ReLu*. Además, se incluyen dos capas densas con 64

unidades y función de activación *ReLu*. Por último, la capa de salida consta de una capa densa con 2 neuronas y una función de activación *softmax*, que produce la probabilidad de pertenencia a cada una de las **emociones**: *alegría* y *tristeza*.

Una vez claros los hiper parámetros de la **red neuronal** se procede a la explicación de los resultados. Durante el entrenamiento del modelo de aprendizaje, se logran resultados prometedores en términos de precisión y pérdida. En el conjunto de entrenamiento, se obtiene una *precisión* del 83.91%, lo que significa que la red es capaz de clasificar las imágenes en las categorías de *alegría* y *tristeza*. Por el otro lado, en el conjunto de prueba, se alcanza una *precisión* del 83.24%, lo que sugiere que el modelo también generaliza bien en datos no vistos previamente.

Al analizar la *pérdida*, un indicador de qué tan bien se ajusta la **red neuronal convolucional** a los datos, se encuentra que en el conjunto de entrenamiento se obtiene un valor de *pérdida* del 37.45%. Esto significa que este tiene un margen de error de este valor a la hora de predecir las **emociones** de las imágenes. En la prueba, la *pérdida* es ligeramente mayor, alcanzando el 38.05%.

A continuación, se presentan dos gráficas que muestran el proceso del entrenamiento de la **red neuronal convolucional binaria**. El primer gráfico muestra la evolución de la *precisión* a lo largo de las épocas, 15, mientras que el segundo ilustra la disminución de la pérdida durante el mismo período. Estas figuras proporcionan una visualización clara del rendimiento del modelo a medida que se entrena, permitiendo una evolución más detallada de su capacidad para capturar y aprender las características relevantes de las **emociones** en las fotografías.



Figuras 22. Entrenamiento red binaria.

Para analizar los resultados de la **red neuronal convolucional** mediante *ResNet50*, se utiliza la matriz de confusión como una herramienta visual para evaluar su desempeño. Esta brinda información sobre la cantidad de imágenes que son clasificadas correctamente y aquellas que son segmentados incorrectamente por el modelo.

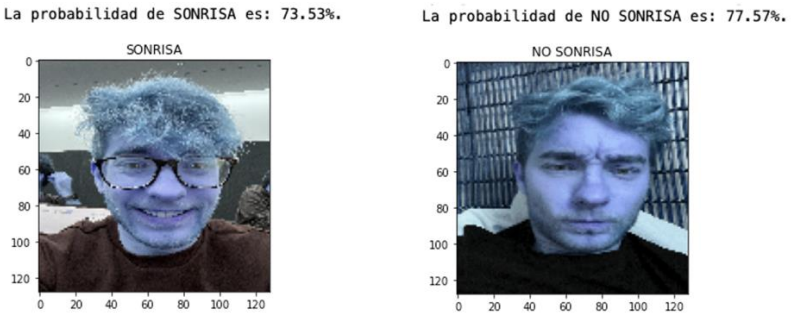
Matriz de Confusión				
-	Alegría	Tristeza	Sorpresa	Neutralidad
Alegría	243	83		
Tristeza	32	328		

Tablas 1. Matriz de confusión red binaria.

En este caso específico, la matriz de confusión revela que el modelo logra un *Recall* del 88.36%, es decir, la identificación correcta de las imágenes de la categoría *alegría*. Además, se obtiene una especificidad del 79.81% indicando que el modelo es capaz de identificar correctamente las imágenes de *tristeza*.

El *F1-Score*, una métrica que combina tanto la precisión como la exhaustividad (*Recall*), alcanza un valor de 85.72%. Esta media general del rendimiento de la red en la clasificación de las dos categorías tiene en cuenta tanto los casos positivos como negativos, proporcionando una evaluación integral del modelo.

La aplicación práctica de la **red neuronal convolucional** creada permite probar su efectividad en la clasificación de **emociones**. Para ello, se presentan dos imágenes: en la izquierda se representa la emoción de *alegría* y en la derecha se muestra *tristeza*.



Figuras 23. Prueba con imágenes reales red binaria.

En el caso de la izquierda, el modelo realiza una predicción con una probabilidad de *alegría* de 0.7353, lo cual concuerda con la emoción representada. Esta predicción confirma la capacidad de la red para identificar adecuadamente dicha emoción. Por otro lado, la red de aprendizaje predice una probabilidad de *tristeza* de 0.7757, lo cual también concuerda con la emoción ilustrada.

En conclusión, el modelo binario de **reconocimiento de emociones** basado en la **red neuronal convolucional** utilizando *ResNet50* demuestra ser efectiva en la clasificación de las emociones de *alegría* y *tristeza*. Mediante el uso de la *matriz de confusión* y métricas como el *Recall*, *Especificidad* y F1-Score, se evalúa el rendimiento del modelo con resultados satisfactorios. Además, al aplicar el modelo en imágenes de prueba, se ha confirmado su capacidad de predecir correctamente las etiquetas correspondientes.

4.3.2 Modelo Multiclase: Alegría, Tristeza y Sorpresa

En el presente apartado, se presenta los resultados obtenidos mediante el uso de una **red neuronal convolucional multiclase** óptima basa en **Transfer Learning** (*ResNet50*) para la clasificación de las emociones de *alegría*, *tristeza* y sorpresa. La arquitectura del modelo se construye agregando capas adicionales a la base del modelo pre-entrenado.

Para empezar, se especifica una forma de entrada de (100,100,3), lo que representa imágenes de entrada de 100x100 píxeles con 3 canales de color RGB. A continuación, se aplica la base del modelo *ResNet50* a las entradas, con los pesos congeladas para aprovechar el conocimiento previo adquirido en *ImageNet*.

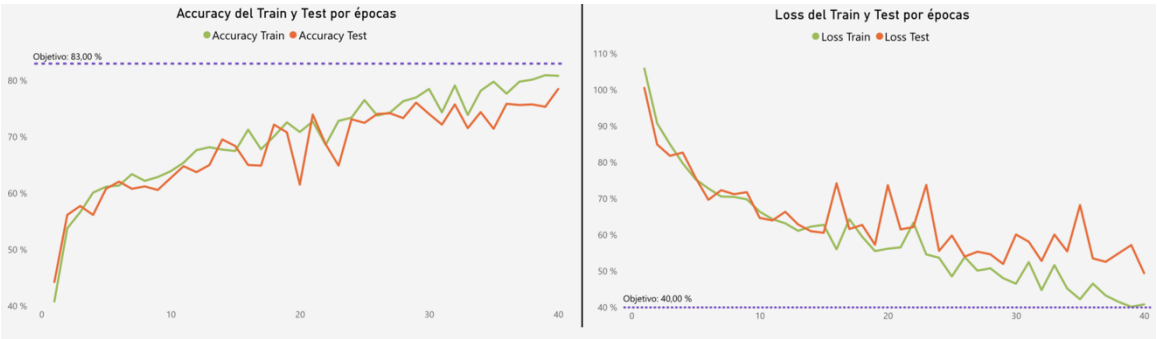
Para adaptar el modelo a la tarea específica, se agregan capas adicionales. Primero, se incluyen capas de *ZeroPladding2D* para asegurar un tamaño de salida compatible con las capas de convolución subsiguientes. Luego, se añades tres capas de convolución con activación *ReLu*, utilizando filtros 64,64 y 128, respectivamente. A continuación, se aplica una capa de *Global Average Pooling* para reducir la dimensionalidad de los mapas de características a un vector unidimensional. Seguidamente, se añaden dos capas densas con activación

ReLu. Entre estas capas, se incluye una capa *Dropout* con una tasa de 0.5 para regularizar el modelo y reducir el riesgo de sobreajuste.

La capa de salida consiste en una capa densa con 3 unidades y función de activación *softmax*, lo que produce la probabilidad de pertenencia a cada una de las 3 clases: *alegría*, *tristeza* y *sorpresa*.

Durante el entrenamiento de la **red neuronal convolucional**, se obtienen resultados alentadores en términos de *precisión* y *pérdida*. El modelo logra una *precisión* del 80.84% en el conjunto de entrenamiento y del 78.50% en la prueba, lo que indica que aprende a clasificar con rigurosidad las **emociones** de *alegría*, *tristeza* y *sorpresa* de las imágenes. En cuanto a la *pérdida*, se registra un valor del 40.83% en el conjunto de entrenamiento y un 49.45% en prueba. Estos datos reflejan la cantidad de información que el modelo no captura durante el entrenamiento. A medida que disminuye la *pérdida*, el modelo se acerca cada vez más a una clasificación precisa de las emociones.

Para evaluar y visualizar el progreso del entrenamiento, se genera una serie de gráficos que muestran la *precisión* y la *pérdida* a lo largo de las 40 *épocas*.



Figuras 24. Entrenamiento red multiclase (3).

En los gráficos anteriores, se observa un entrenamiento muy adecuado, donde la *precisión* aumenta progresivamente a lo largo de las épocas, mientras que la *pérdida* disminuye de manera constante. Además, el modelo logra generalizar de forma correcta y no memoriza los datos de entrenamiento ni tiene dificultades para adaptarse a nuevos datos.

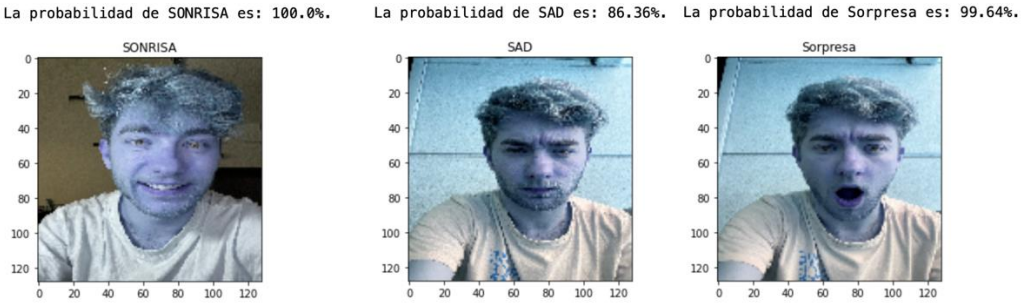
Por otro lado, el análisis de la *matriz de confusión* y las métricas asociadas proporciona una visión más detallada del desempeño del modelo de clasificación. Dicha matriz ofrece una representación visual de la segmentación de las **emociones** indicando los valores en la diagonal el correcto etiquetado y el resto equivocaciones, por ejemplo, el modelo predice que son *alegría* 49 pero en realidad son *tristeza*. Cabe destacar que está matriz tiene un dimensionamiento 3x3 debido a que se está estudiando 3 emociones de los rostros humanos.

Matriz de Confusión				
-	Alegría	Tristeza	Sorpresa	Neutralidad
Alegría	249	49	42	
Tristeza	41	323	2	
Sorpresa	69	1	173	

Tablas 2. Matriz de confusión red multiclase (3).

En este caso, la matriz anterior revela resultados alentadores. El *Recall* obtuvo un valor del 78.50%, indicando los casos que han sido clasificados correctamente. Por el otro lado, la *especificidad* alcanza un valor del 89.25%, el cual proporciona la información de aquellas imágenes que no pertenecen a las emociones a clasificar, es decir, que identifica de forma óptima el tipo de emoción. Finalmente, el *F1-Score* es del 81.93%. Cabe destacar que los valores presentados anteriormente son el promedio de las 3 clases ya que es la forma óptima de presentar los resultados generales del presente modelo.

Habiendo evaluado el rendimiento del modelo en la clasificación de las **emociones** de *alegría*, *tristeza* y *sorpresa* mediante el análisis de las métricas anteriores, se procede a poner en práctica con tres imágenes reales. Estas se someten a la **red neuronal convolucional multiclase** y se obtiene los siguientes resultados.



Figuras 25. Prueba con imágenes reales red multiclase (3).

En la imagen de *alegría*, el modelo predijo con certeza una probabilidad de 1. Para la emoción de *tristeza*, la probabilidad es de 0.8636. Por último, la probabilidad de *sorpresa* es de 0.9964. Estos resultados coherentes entre las predicciones del modelo y las **emociones** reales representadas en las imágenes fortalecen aún más la evaluación positiva del modelo.

4.3.3 Modelo Multiclase: Alegría, Tristeza, Sorpresa y Neutralidad

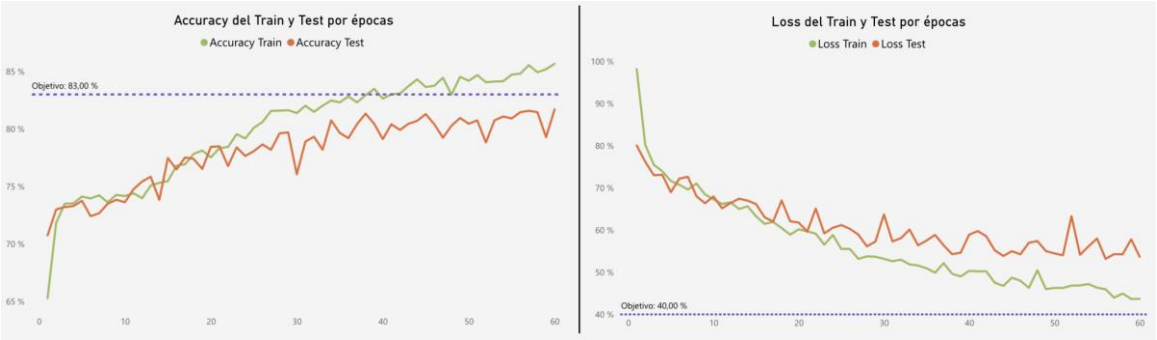
El modelo que se analiza a continuación es el **modelo óptimo de la red neuronal convolucional multiclase** con 4 categorías (*alegría*, *tristeza*, *sorpresa* y *neutralidad*) utilizando **Transfer Learning** mediante la arquitectura *ResNet*. Este modelo es el objetivo principal del proyecto, ya que tiene como propósito identificar y clasificar las **cuatro emociones principales** en rostros humanos, y es el que se encuentra en producción.

La arquitectura del modelo consta de una combinación de capas pre-entrenadas de la red *ResNet* junto con capas adicionales agregadas. A continuación, se detalla de forma breve dicha arquitectura.

Se especifica una forma de entrada de (100,100,3), representado imágenes de entrada de 100x100 píxeles con 3 canales de color: RGB. Se utiliza **Transfer Learning** para aprovechar los conocimientos adquiridos en tareas anteriores de clasificación de imágenes. Se añaden capas de *ZeroPadding2D* para asegurar un tamaño de salida compatible con las capas de convolución subsiguientes. Seguidamente, se introducen 3 capas convolucionales con activación *ReLu*, utilizando tamaños de filtro (3,3) y un número creciente de filtros (64,128,128). Se aplica una capa de *Global Average Pooling* para reducir la dimensionalidad de los mapas de características a un vector unidimensional. Se incluyen dos capas densas con la misma activación, junto con una capa de *Dropout* con una tasa del 50% para regularizar el modelo y mitigar el riesgo de sobreajuste. Finalmente, la capa de salida consta de 4 neuronas con una activación *softmax*, que proporciona las probabilidades de pertenencia a cada una de las **4 emociones**.

El modelo se compila utilizado el optimizar *Adam* y la función de pérdida *categorical_crossentropy*. Además, se establece la métrica de precisión para evaluar el rendimiento del modelo durante el entrenamiento.

Durante el entrenamiento de la **red neuronal convolucional**, se realizan registros de la *precisión* y la *pérdida* en cada una de las 60 *épocas*. Estas métricas brindan una comprensión detallada del progreso del modelo a lo largo del entrenamiento.



Figuras 26. Entrenamiento red multiclase (4).

La *precisión* en el conjunto de entrenamiento alcanza un valor del 85.67%, lo que indica que el modelo es capaz de clasificar correctamente las imágenes utilizadas en el entrenamiento. Esta alta precisión demuestra la capacidad del modelo para aprender patrones.

En el conjunto de prueba, se logra una *precisión* del 81.71%. Esta métrica muestra la habilidad del modelo para generalizar y clasificar correctamente muevas imágenes que no son utilizadas en el entrenamiento. La alta precisión indica que aprende a reconocer y clasificar las **emociones** de manera efectiva.

En cuento a la *pérdida*, se obtiene un valor de 43.68% en el entrenamiento, por tanto, el modelo logra reducir significantemente la discrepancia entre las etiquetas reales y las predicciones durante el entramiento. Por otro lado, la *pérdida* en el conjunto de prueba es del 53.65%, lo que sugiere que el modelo generaliza de manera efectiva al mantener un bajo nivel de pérdida en datos no vistos previamente.

A continuación, se analizan los resultados obtenidos en la *matriz de confusión* de la **red neuronal convolucional multiclase.** Esta matriz es una representación visual de las clasificaciones realizadas por el modelo, donde cada fila representa la clase real y cada coluna representa la clase predicha. En este caso, al estudiar y clasificar **4 emociones**, la matriz de confusión tiene una estructura 4x4.

Matriz de Confusión				
-	Alegría	Tristeza	Sorpresa	Neutralidad
Alegría	163	26	16	134
Tristeza	53	311	0	4
Sorpresa	0	0	124	146
Neutralidad	3	0	55	1354

Tablas 3. Matriz de confusión red multiclase (4).

Además de la *matriz de confusión*, se calculan las tres métricas importantes para cada una de las categorías: *Recall*, *Especificidad* y *F1-Score*. Estas métricas proporcionan información más detallada sobre el rendimiento del modelo en la clasificación de cada emoción específica. Es importante analizar las métricas por separada para cada etiqueta, ya que permite evaluar el desempeño del modelo en la identificación de **emociones** individuales. Cada emoción puede tener diferentes características y desafíos asociados, por lo que es crucial comprender cómo el modelo desempeña en cada una de ellas

Métricas Red Neuronal Convolucional Multiclase (4)				
Métricas	Alegría	Tristeza	Sorpresa	Neutralidad
Recall	48,08 %	85,51 %	45,93 %	95,89 %
Especificidad	94,68 %	97,46 %	93,52 %	54,96 %
F1-Score	58,43 %	88,23 %	53,33 %	88,79 %

Tablas 4. Métricas red multiclase (4).

Para la emoción de *alegría*, el modelo ha identificado correctamente el 48.08% de las imágenes que representan esta emoción. La *especificidad* del modelo indica que el 94.68% de las veces reconoce acertamiento que las imágenes no corresponden a *alegría*. El *F1-Score*, que combina la *precisión* y el *Recall*, alcanza un valor del 58.43%, esto indica que la red logra un equilibrio entre la capacidad de identificar los casos positivos y negativos.

En cuanto a la emoción de *tristeza*, se observa un *Recall* del 85.51%. La *especificidad* obtenida es del 97.46%, lo que significa que el modelo ha clasificado correctamente aquellas imágenes que no pertenecen a dicha categoría. El *F1-*

Score para esta emoción alcanza un valor del 88.23%, es decir, existe un buen equilibrio entre los valores True Positive y True Negative.

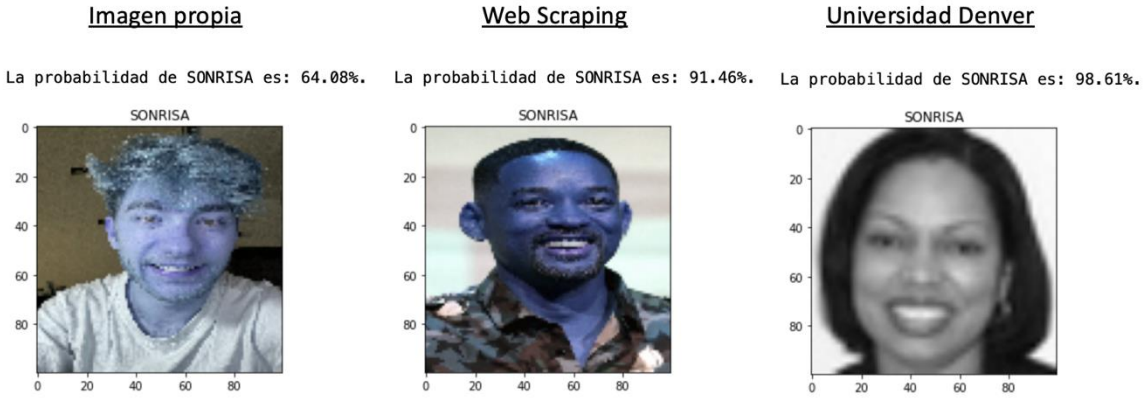
En cuanto a la emoción de *sorpresa*, se obtiene un *Recall* del 45.93%. La *especificidad* alcanza un valor del 93.52%, es decir, que identifica bien aquellas imágenes que no corresponden a esta emoción. El *F1-Score* se sitúa en 53.33%, lo que indica un mejorable equilibrio entre la *precisión* y el *Recall* en la clasificación de esta etiqueta.

Finalmente, para la emoción de neutralidad, el modelo indica correctamente el 95.89% de las imágenes que representan dicha emoción. La red clasifica correctamente el 54.96% de aquellas fotografías que no corresponden con el rostro a analizar. El *F1-Score* para la *neutralidad* alcanza un valor del 88.79%, indicando un buen equilibrio entre la *precisión* y el *Recall* en la clasificación de esta emoción.

Además de las métricas individuales, también se calculan los valores generales del modelo, que brindan una evaluación global de su rendimiento en la clasificación de las **4 emociones**. Estos resultados se consiguen mediante la consideración tanto de los casos positivos como negativos, obteniendo un *Recall* del 79.23%, una *Especificidad* del 93.90% y un *F1-Score* del 84.52%.

Para evaluar la efectividad de la **red neuronal** en la clasificación de las **4 emociones** (*alegría*, *tristeza*, *sorpresa* y *neutralidad*), se realizan predicciones utilizando diferentes imágenes representativas. Estas imágenes abarcan distintos escenarios y su utilizan para evaluar el desempeño general del modelo en situaciones diversas. En cada **categoría emocional** se van a realizar **3 escenario predictivos**. El primer, se utiliza una imagen capturada directamente por el creador del presente proyecto utilizando su **smartphone**. Esta figura es la más importante, ya que será la clave para realizar una previsualización de cómo funcionará el modelo en producción. En el segundo escenario, se utiliza una imagen totalmente desconocida obtenida a través de técnicas de **Web Scraping**. Por último, se selecciona una imagen del conjunto de datos proporcionado por la **Universidad de Denver**.

A través de estos tres escenarios para cada categoría emocional, se puede evaluar la capacidad del modelo para clasificar correctamente las emociones en diferentes situaciones y contextos.



Figuras 27. Predicciones alegría.

En el escenario de la **imagen propia** del creado, el modelo predijo una probabilidad de *alegría* de 0.6408. En la casuística de la fotografía de **Web Scraping,** la probabilidad predicha es de 0.9156. Por último, la figura que proviene del conjunto de datos de la **Universidad de Denver**, la probabilidad que indica la red neuronal es de 0.9861.

Estos resultados en la categoría de *alegría* muestran que la **red neuronal convolucional** es capaz de reconocer y clasificar de manera efectiva la categoría en cuestión, incluyendo 3 escenarios distintos.

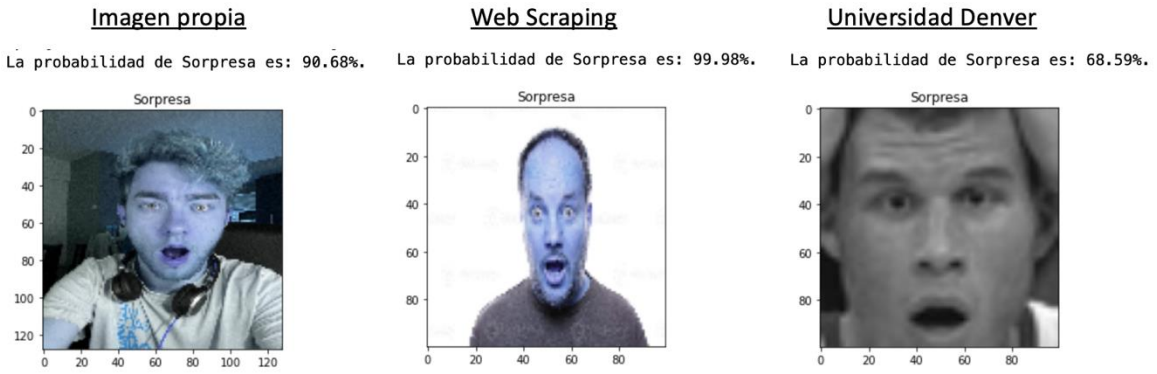
En cuanto a la categoría de *tristeza*, se realizan pruebas similares utilizando 3 escenarios. Estas predicciones permiten evaluar cómo el modelo clasifica esta emoción en diferentes contextos.

Figuras 28. Predicciones tristeza.

El escenario de la **imagen propia** del creador, el modelo predijo una probabilidad de 0.6426 para la emoción de tristeza. La imagen de **Web Scraping,** la red neuronal muestra una mayor confianza en la categoría, con una probabilidad predicha de 0.7654. La última casuística proveniente del conjunto de datos de la **Universidad de Denver**, el modelo muestra una probabilidad más baja, con una predicción de 0.5308.

Estos resultados en la categoría de *tristeza* revelan que la **red neuronal convolucional** puede reconocer y clasificar dicha emoción, con una cierta variabilidad según el escenario.

En el caso de la categoría de *sorpresa*, se realiza pruebas utilizando el mismo criterio que en los dos casos anteriores. Estas permiten evaluar el rendimiento del modelo en la clasificación de la emoción de *sorpresa* en diferentes situaciones.

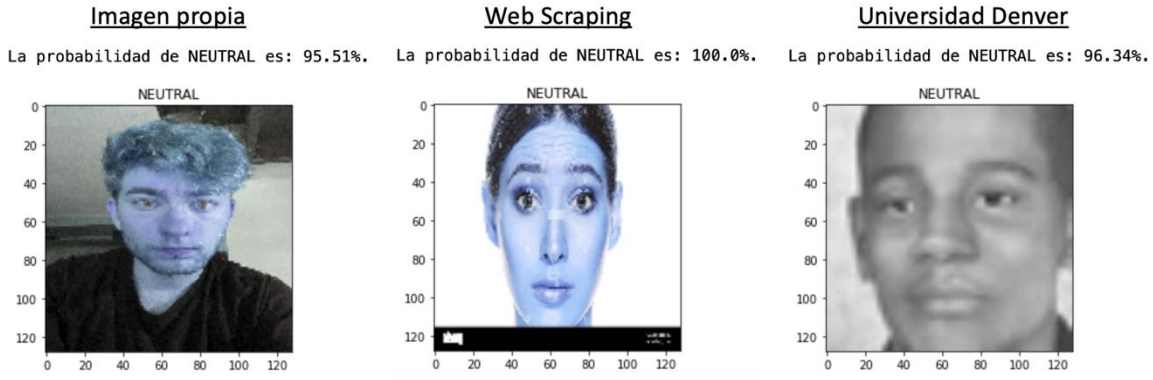


Figuras 29. Predicciones sorpresa.

En el escenario de la imagen propia desde el **smartphone**, el modelo predice una probabilidad de 0.9068 para esta emoción. Seguidamente, la fotografía de **Web Scraping**, la probabilidad que indica la red es de 0.9998. Por último, con la figura del conjunto de datos de la **universidad** la probabilidad es la más bajo, siendo 0.6859.

Estos resultados en la categoría de *sorpresa* destacan la capacidad del modelo para reconocer y clasificar dicha emoción en diferentes contextos, siendo los resultados muy favorables.

Por último, se lleva a cabo las pruebas de la emoción de *neutralidad* utilizando, una vez más, los 3 escenarios anteriores. Estas pruebas permiten evaluar el rendimiento y efectividad de la **red neuronal convolucional** para detectar dicha emoción.



Figuras 30. Predicciones neutralidad.

En el escenario de la imagen propia, el modelo indica una probabilidad de *neutralidad* de 0.9551. Seguidamente, la probabilidad que indica la red para la imagen es de 1, por lo que se clasifica de manera muy precisa en esta imagen desconocida. Por último, con la figura de la **universidad** la red predice con una probabilidad de 0.9634, es decir, realiza una segmentación a la perfección

Estos resultados en la categoría de *neutralidad* demuestran que el modelo es capaz de reconocer y clasificar de manera muy efectiva dicha emoción. Los valores respaldan la capacidad del modelo para diferenciar entre las **4 emociones** específicas de *alegría*, *tristeza*, *sorpresa* y *neutralidad*.

4.3.4 Red Neuronal Convolucional Multiclase puesta en Producción

Tras la implementación de la **red neuronal convolucional** en un entorno de **producción**, los resultados han demostrado un desempeño general sobresaliente. El modelo ha sido capaz de reconocer y clasificar con precisión las **emociones** de *alegría*, *tristeza*, *sorpresa* y *neutralidad* en una variedad de escenarios y condiciones.

Figuras 31. Predicciones modelo en Producción.

Las probabilidades que se pueden observar en la figura anterior en las **4 emociones** indican, una vez más, que el modelo realiza de forma correcta la segmentación. Sin embargo, se ha identificado una dificultad particular en la emoción de *tristeza*. Aunque el modelo logra clasificar correctamente algunas imágenes de estas, se ha observado una menor confianza y precisión en comparación al resto.

Es importante abordar esta complicación y seguir mejorando el modelo en la identificación de la emoción de *tristeza*. Se puede explorar diversas estrategias para optimizar el rendimiento, como ajustar los hiper parámetros o aumentar el tamaño y diversidad del conjunto de datos de entrenamiento.

A pesar de esta limitación, el rendimiento general de la **red neuronal convolucional** en la clasificación de las **4 emociones** es altamente **satisfactoria**. El enlace al vídeo de YouTube titulado "TESTING EFFECTIVENESS WITH CONVOLUTIONAL NEURAL NETWORKS: PRODUCTION IN THE COMPUTER WEBCAM", proporciona la evidencia visual de esa efectividad.

El general, la implementación de la **red neuronal** en producción es exitosa y, los resultados obtenidos respaldan su utilidad en la *Start-UP*.

CAPÍTULO 5: CONCLUSIONES

Este trabajo final de máster ha abordado de manera integral el reconocimiento de **emociones humanas** a través de la extracción de imágenes de *Google Images* mediante técnicas de **Web Scraping** y la combinación del conjunto de datos de la *Universidad de Denver*, el procesamiento y limpieza de datos, la creación de un **DataWarehouse** en *Oracle*, el diseño y entrenamiento de una **red neuronal multiclase,** la creación de un **reporte** en *Power BI* y la puesta en **producción** del modelo para clasificar las **cuatro emociones principales**: *alegría*, *tristeza*, *sorpresa* y *neutralidad*.

Además, es importante destacara que este proyecto ha sido concebido y desarrollado como un enfoque *end-to-end*, abarcando desde la recopilación de datos hasta la implementación en **producción**, simulando un producto mínimo viable de una *Start-Up*.

Durante el proceso de extracción de imágenes a través de **Web Scraping** y el uso del conjunto de datos de la *Universidad de Denver*, se recopilan un total de 7760, las cuales son sometidas a un riguroso procesamiento y limpieza para garantizar la calidad y consistencia de los datos. Como resultado, se redujo el conjunto de fotografías a 7306, lo que evidencia una adecuada depuración y preparación para su posterior análisis.

La creación y entrenamiento de la **red neuronal convolucional multiclase** es un aspecto fundamental de esta investigación. El modelo ha demostrado un rendimiento sólido en la clasificación de las emociones objetivo, alcanzando métricas de *precisión*, *Recall*, *Especificidad* y *F1-Score* en promedio de 82.24%, 78.44%, 90.17% y 84.12%, respectivamente.

Todos los resultados y métricas obtenidas se encuentran documentadas en el reporting publicado en Wix, brindando una visión detallada de los hallazgos y análisis realizados en este estudio. Asimismo, se ha proporcionado un enlace a un video de YouTube, titulado "TESTING EFFECTIVENESS WITH CONVOLUTIONAL NEURAL NETWORKS: PRODUCTION IN THE COMPUTER WEBCAM", que muestra

la visualización del modelo en producción, permitiendo una apreciación práctica y concreta de su funcionamiento.

En conjunto, este trabajo final de máster representa un hito significativo en el campo del reconocimiento de **emociones de seres humanos** y destaca la importancia de la integración de técnicas de extracción de datos, procesamiento de información y aprendizaje automático en la solución de problemas complejos. Su enfoque integral, los resultados alcanzados y su potencial comercial refuerzan la relevancia y el impacto de esta investigación en el ámbito académico y empresarial.

CAPÍTULO 6: PRÓXIMOS PASOS

En cuanto a los próximos pasos y futuras mejoras, existen varios aspectos que se pueden abordar para seguirá avanzado en la precisión y aplicabilidad de la solución propuesta.

En primer lugar, es importante mejorar y ampliar el conjunto de datos de **imágenes** utilizado para el entrenamiento de la **red neuronal multiclase**. Esto incluye la inclusión de imágenes con diferentes variaciones y complementos, como gafas de vista o de sol, entre otros.

Otro aspecto para considerar es la mejora del rendimiento del modelo en condiciones de baja luminosidad. El entrenamiento con imágenes que simula este tipo de escenarios o la implementación de técnicas de procesamiento de fotografías que mejoren la visibilidad en condiciones de poca luz pueden se enfoques prometedores.

También existe la posibilidad de aplicar otras redes pre-entrenadas como *MobileNet*, que puede reducir el tiempo y los recursos necesarios para el entrenamiento. Además, se puede explorar la posibilidad de agregar nuevas **emociones al modelo**, como ira o temor, ampliando así su capacidad para reconocer una gama más amplia de estados emocionales humanos.

En resumen, el futuro trabajo debe enfocarse en la mejora del conjunto de imágenes, la optimización del rendimiento en condiciones de baja luminosidad, la incorporación de nuevas emociones y la exploración de técnicas avanzadas de aprendizaje automático. Estas mejoras permiten continuar desarrollando un modelo más robusto y versátil para el reconocimiento de emociones en imágenes, ampliando así su potencial en diversas circunstancias.

CAPÍTULO 7: BIBLIOGRAFÍA

Fernando, T., & Sivarajah, S. (2015). A Comparative Study on Web Scraping. Recuperado de http://ir.kdu.ac.lk/handle/345/1051

Teguh, A. K., & Purnama, P. (2023). Web Scraping Tool For Newspapers And Images Data Using Jsonfy. Journal of Advanced Science and Engineering, 26(4), 11-20.

Mollahosseini, A., Hasani, B., & Mahoor, M.H. (2017). AffectNet: A Database for Facial Expression, Valence, and Arousal Computing in the Wild. IEEE Transactions on Affective Computing, 10(1), 18-31.

Olaya-Rodríguez, H., & Suárez-Ruiz, J.F. (2014). Preprocesamiento de datos estructurados. 93-101.

Zhang, H., & Jagadish, H. V. (2016). Data Management Challenges in Production Machine Leaning. In Proceedings of the 2016 International Conference on Management of Data (pp. 1-16).

Oracle Corporation. (2018). Ensuring a Fast, Reliable, and Secure Database Through Automation: Oracle Autonomous Database. Recuperado de https://informationsecurity.report/Resources/Whitepapers/b2db3b03-d47f-4203-850e-5cdf99909190_Ensuring_wp.pdf

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-Bases Learning Applied to Document Recognition. Proceedings of the IEEE, 86(11), 2278-2324.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep Learning. Nature, 521 (7553), 436-444.

Salas, J. (2018). Google arregla su algoritmo 'racista' borrando a los gorilas. El País. Recuperado de https://elpais.com/tecnologia/2018/01/14/actualidad/1515955554_803955.html

ANEXOS

Optimización de la precisión y eficiencia del reconocimiento de emociones en tiempo real mediante redes neuronales convolucionales en sistemas de videovigilancia.

Trabajo Fin de Máster

Máster Universitario en Big Data Science

Curso académico 2022-2023

AUTOR: Raúl Artigues Femenia

TUTOR ACADÉMICO: Borja Balparda de Marco

Madrid a 14 de junio del 2023

ÍNDICE

INTRODUCCIÓN

El trabajo final de grado titulado "*Optimización de la precisión y eficiencia del reconocimiento de emociones en tiempo real mediante redes neuronales convolucionales en sistemas de videovigilancia*" se ha convertido en un recurso significativo para avanzar en la inteligencia artificial, con especial interés en la aplicación del aprendizaje profundo para el reconocimiento de emociones en tiempo real: alegría, tristeza, sorpresa y neutralidad.

Como complemento a este proyecto, este anexo contiene 6 scripts de *Python* que han sido desarrollados por el autor, como parte integrante de la investigación. Cada uno de estos cumple una función crucial dentro de la totalidad del proyecto, desempeñando tareas desde la recolección de datos hasta la implementación de la red neuronal convolucional y su correspondiente evaluación. Cabe destacar que los títulos y explicación del código está en inglés para su mayor entendimiento y alcance.

El primer código "***Downloading Facial Faces in Google Images***" se encarga de la recopilación de datos necesaria para alimentar la red neuronal. Se utilizan técnicas de web scraping para la descarga de imágenes de rostros humanos de Google Images para su posterior uso como conjunto de datos para entrenar y probar la CNN.

Seguidamente el script "***Image Cleanup***" realiza una limpieza y preprocesamiento de las imágenes recolectadas, eliminado ruido y distorsiones y, adaptándolas para que sean adecuadas.

El código "***Connection to Autonomous Oracle Database***" establece una conexión entre el ordenador, concretamente Python, y una base de datos autónoma de Oracle. Se utiliza para almacenar, recuperar y gestionar las imágenes y los resultados (Data Warehouse).

A continuación, se implementa la creación y entrenamiento de las distintas redes neuronales convolucionales probadas "***Convolutional Neural Networks***". Estas

redes son eficientes en el procesamiento de los datos y se utilizan para reconocer patrones en las imágenes en los rostros humanos.

Tras entrenar y probar las redes neuronales óptimas, es crucial evaluar sus rendimientos. Es por ese que se crea el script "**Metrics from CNN to Oracle**" que recopila las métricas de los modelos y las almacena en la base de datos de Oracle para realizar un análisis posterior en Power BI, implementado en Wix.

Por último, se crea "***Testing Effectiveness with Convolutional Neural Networks: Production in the Computer Webcam***" que pone a prueba la red óptima entrenada en un escenario de producción. Utilizando la cámara web del ordenador, el script reconoce las 4 emociones básicas en tiempo real, demostrando la aplicación práctica de la presente investigación.

En conjunto, estos scripts representan la culminación de la investigación, permitiendo un reconocimiento de emociones preciso y eficiente en sistema de videovigilancia en tiempo real. La combinación de estos componentes ofrece un amplio espectro de aplicaciones, desde mejoras en la interacción hombre-máquina hasta avances significativos en áreas de seguridad y salud mental.

DOWNLOADING FACIAL FACES IN GOOGLE IMAGES

Installation of the necessary libraries

```
!pip install selenium
!pip install webdriver_manager
```

The necessary libraries are loaded to be able to do *web scraping*

```
import os
import csv
import pandas as pd
import time
import requests
from selenium import webdriver
from selenium.webdriver.common.by import By
from webdriver_manager.chrome import ChromeDriverManager
```

3 functions are created that allow connecting to google images to download the URLs of the searches entered, related to SMILES, SAD, SURPRISE & NEUTRAL

```
def fetch_image_urls(query: str, max_links_to_fetch: int, wd: webdriver
, sleep_between_interactions: int = 1):
    # Scrolls down to load more images
    def scroll_to_end(wd):
        wd.execute_script("window.scrollTo(0, document.body.scrollHeigh
t);")
        time.sleep(sleep_between_interactions)

    # build the google query
    search_url = "https://www.google.com/search?safe=off&site=&tbm=isch
&source=hp&q={q}&oq={q}&gs_l=img"

    # load the page
    wd.get(search_url.format(q=query))

    consent = wd.find_element(By.CSS_SELECTOR, 'button.VfPpkd-LgbsSe.Vf
Ppkd-LgbsSe-OWXEXe-k8QpJ.VfPpkd-LgbsSe-OWXEXe-dgl2Hf.nCP5yc.AjY5Oe.DuMI
Qc.LQeN7.Nc7WLe')
    consent.click()

    image_urls = set()
    image_count = 0
    results_start = 0
    while image_count < max_links_to_fetch:
```

```
        scroll_to_end(wd)

        # get all image thumbnail results
        thumbnail_results = wd.find_elements(By.CSS_SELECTOR, 'img.Q4Lu
Wd')
        number_results = len(thumbnail_results)

        print(f"Found: {number_results} search results. Extracting link
s from {results_start}:{number_results}")

        for img in thumbnail_results[results_start:number_results]:
            # try to click every thumbnail such that we can get the rea
l image behind it
            try:

                img.click()
                time.sleep(sleep_between_interactions)
            except Exception:
                continue

            # extract image urls
            actual_images = thumbnail_results = wd.find_elements(By.CSS
_SELECTOR, 'img.n3VNCb')
            for actual_image in actual_images:
                if actual_image.get_attribute('src') and 'http' in actu
al_image.get_attribute('src'):
                    image_urls.add(actual_image.get_attribute('src'))

            image_count = len(image_urls)

            if len(image_urls) >= max_links_to_fetch:
                print(f"Found: {len(image_urls)} image links, done!")
                break
        else:
            print("Found:", len(image_urls), "image links, looking for
more ...")
            time.sleep(30)
            return
            load_more_button = wd.find_elements(By.CSS_SELECTOR, '.mye4
qd')
            if load_more_button:
                wd.execute_script("document.querySelector('.mye4qd').cl
ick();")

        # move the result startpoint further down
        results_start = len(thumbnail_results)

    with open("/enter the directory path/File.csv", "a", newline="\n")
as Test:
```

```
        Test.write(str(image_urls))
        Test.write(str("\n"))
    return image_urls

def search_and_download(search_term: str, driver_path: str, target_path
='/enter the directory path/', number_images=100):
    target_folder = os.path.join(target_path, '_'.join(search_term.lowe
r().split(' ')))
    # make the folder name inside images with the search string

    if not os.path.exists(target_folder):
        os.makedirs(target_folder) # make directory using the target pa
th if it doesn't exist already

    #with webdriver.Chrome(executable_path=driver_path) as wd:
    with webdriver.Chrome(ChromeDriverManager().install()) as wd:
        res = fetch_image_urls(search_term, number_images, wd=wd, sleep
_between_interactions=0.5)

    counter = 0
    for elem in res:
        persist_image(target_folder, elem, counter)
        counter += 1
```

Next, the values to search for in google are defined and the URLs are downloaded in a csv file: SMILES, SAD, SURPRISE & NEUTRAL

An example for each emotion appears in the following code. In practice, it is better to carry out a multitude of searches with different words, that is, if they are related to the search for the corresponding emotion.

```
DRIVER_PATH = 'C:/Program Files(x86)/Google/Chrome/Application/chromedr
iver'
search_term = 'Person smiling'
# 1- People who have a SMILE, SAD, SURPRISE, NEUTRAL-inducing facial fe
atures:
     # Person smiling.
     # Person Sad.
     # Person Surprise.
     # Person Neutral.
search_and_download(search_term=search_term
                    , number_images = 100
                    , driver_path=DRIVER_PATH) # method to download ima
ges
```

As different searches are carried out, a homogeneous csv file will be created to contain the X URLs in the case of SMILES, SAD, SURPRISE & NEUTRAL

```
df= pd.read_csv("/enter the directory path/File.csv", sep=",")
dataframe1 = df.transpose()
df2=dataframe1.reset_index(level=None, drop=False, inplace=False, col_l
evel=0, col_fill='')
df2 = df2.rename(columns={'index':'URL'})
df3 = df2.to_numpy().flatten()
df3 = pd.DataFrame(df3[df3 != None])
df3.to_csv("/enter the directory path/Final_File.csv")
```

IMAGE CLEANUP

Installation of the necessary libraries

```
import pandas as pd
import re
import numpy as np
```

SMILES, SAD, SURPRISE & NEUTRAL IMAGES

```
df= pd.read_csv("/enter the directory path/Final_File.csv",
                encoding='utf-8',escapechar="{",quotechar="'")
df.head(5)

df.drop(["Unnamed: 0"], axis=1, inplace=True)

df.rename(columns={'0':'URL'}, inplace=True)
df.head(5
```

The following code shows examples of URLs to remove from the 4 emotions.

```
# Enter the URLs to be removed in a list
values_to_remove = ['https://encrypted-tbn0.gstatic.com/images?q=tbn:AN
d9GcTY1wWbASzCKl9tvCTwlubnozfAfNsjXLx0HQ&usqp=CAU',
                    'https://encrypted-tbn0.gstatic.com/images?q=tbn:
ANd9GcT3dvclP4JvJ8VT7svAc1HuFKUg6uPK8rrr7A&usqp=CAU',
                    'https://s1.abcstatics.com/media/ciencia/2016/02/
14/sonreimos--620x300.jpg',
                    'https://encrypted-tbn0.gstatic.com/images?q=tbn:
ANd9GcSjvdLjvFgHR0rTARv3jgSUWO7N9yHuvzZxgQ&usqp=CAU',
                    'https://images.emojiterra.com/twitter/v14.0/512p
x/1f64d-2642.png',
                    'https://encrypted-tbn0.gstatic.com/images?q=tbn:
ANd9GcRJzWQHtnioel7jpbeORZHLT-OisRSh4X3WtA&usqp=CAU',
                    'http://3.bp.blogspot.com/-5o5L8R1GKOQ/VQhNe1sQyA
I/AAAAAAAAA_M/zg91MdgIxcM/s1600/ceño%2B1.png',
                    'https://encrypted-tbn0.gstatic.com/images?q=tbn:
ANd9GcTdLHAkN-y1EswI8lFcNoKMULlDQdtS_9YFMw&usqp=CAU',
                    'https://emojitool.com/img/apple/ios-13.3/man-fro
wning-type-1-2-1780.png',
                    'https://encrypted-tbn0.gstatic.com/images?q=tbn:
ANd9GcTSv_KlsZdAIpcnXeqOWBmk6Dj8BrI3F75TiA&usqp=CAU',
                    'https://www.supercoloring.com/sites/default/file
s/styles/coloring_medium/public/cif/2022/01/251-u1f64d-frown-emoji-colo
ring-page.png',
                    'https://encrypted-tbn0.gstatic.com/images?q=tbn:
ANd9GcSE3BJWf6HqzCW7JOis0gV1Ib4AwQBoOTBPiA&usqp=CAU',
                    'https://encrypted-tbn0.gstatic.com/images?q=tbn:
```

```
ANd9GcSujl28qQsgKi_Psu2gmjLGqphGU3XHh3bDag&usqp=CAU',
                      'https://www.supercoloring.com/sites/default/files/styles/coloring_medium/public/cif/2022/02/252-man-frowning-emoji-coloring-page_1.png',
                      'https://www.supercoloring.com/sites/default/files/styles/coloring_full/public/cif/2022/01/214-man-frowning-1-coloring-page.png',
                      'https://emojitool.com/img/apple/ios-12.1/woman-frowning-509.png',
                      'https://i.ebayimg.com/images/g/eXEAAOSwwlFhWuSV/s-l500.jpg',
                      'https://www.wikihow.com/images_en/thumb/b/b0/Frown-Step-1-Version-3.jpg/v4-460px-Frown-Step-1-Version-3.jpg.webp',
                      'https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcS8LAfPn-M3ogE7ujC36qdlldRJBFJJtsTi7w&usqp=CAU',
                      'https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcRxqr450y1Wk4MoBY5QUirB8VOPIincsw8ufg&usqp=CAU',
                      'https://www.wikihow.com/images_en/thumb/7/72/Frown-Step-8-Version-3.jpg/v4-460px-Frown-Step-8-Version-3.jpg',
                      'https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcSgEVo-rRab90ml3S6H9kooHGL-hD26bCuMyA&usqp=CAU',
                      'https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcQoYn2CaB8jHCaZYTNkfnZdVwtJ746itK94oQ&usqp=CAU',
                      'https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcQQC2e6siActGXXaauGQv5QuSAQ_UzBHIAx8A&usqp=CAU',
                      'https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcQrUZ-edUiNNBylPB_MINzTanQ75QcsD378Yw&usqp=CAU',
                      'https://www.supercoloring.com/sites/default/files/styles/coloring_thumbnail/public/cif/2022/02/251-person-frowning-coloring-page.png']

urls_exact_getaways = [re.escape(url) for url in values_to_remove]

# Create the regular expression
regex_url_exact = re.compile('|'.join(urls_exact_getaways))

# Apply the regular expression to the 'Text' column of the dataframe
df2= pd.DataFrame()
df2['URL'] = df['URL'].replace(regex_url_exact, np.nan)

# Show the resulting dataframe
df2.head(5)
```

```
df3= df2.dropna()
print(len(df))
print(len(df2))
print(len(df3))
```

```
df3.head(4)
```

```
data=df3.values.tolist()
datas=[]

contadorWr=0
contGood=0
cont=0
url= "'https:/"
url1= " 'https:/"

for row in range(len(data)):
    cont+=1
    d= data[row][0]
    dt= d[0:8]

    if cont<12:
        if (dt==url or dt==url1):
           contGood+=1
           datas.append(data[row])
    else:
        d= data[row][0]
        dt= d[1:9]

        if (dt==url or dt==url1):
            contGood+=1
            datas.append(data[row])

        else:
            contadorWr+=1

print(f"Number of URLs that are CORRECT: {contGood}")
print(f"Number of URLs that are INCORRECT: {contadorWr}")

dataframe= pd.DataFrame(datas,columns=["URL"])
dataframe['URL'] = dataframe['URL'].str.replace("'", "")
```

Save the DataFrame of the processed URLs in a csv file

```
dataframe.to_csv('/enter the directory path/Emotions.csv')
```

CONNECTION TO AUTONOMOUS ORACLE DATABASE

Installing and loading Oracle libraries in Python

```
!pip install cx_Oracle

import cx_Oracle
import os
import pandas as pd
import csv

import hashlib
import requests
from io import BytesIO
from PIL import Image
import re
import time
```

Connection with the database and creation of the 4 tables: SMILE, SAD, SURPRISE & NEUTRAL

```
os.chdir("/enter the directory path/Oracle/instantclient_19_8")

conn = cx_Oracle.connect(user='User', password='Password', dsn="enter t
he dns properties: Low, Medium or High")

cursor = conn.cursor()

#Creating table as per requirement    #SMILES, SAD, SURPRISE , NEUTRAL
sql ='''CREATE TABLE SMILES(
   ID VARCHAR2(1000),
   URL VARCHAR2(1000))'''

cursor.execute(sql)
print("Table created successfully........")

# Commit your changes in the database
conn.commit()

#Closing the connection
cursor.close()
conn.close()
```

Insertion of the URLs of the processed and cleaned images

```
os.chdir("/enter the directory path/Oracle/instantclient_19_8")
```

```
conn = cx_Oracle.connect(user='User', password='Password', dsn="enter t
he dns properties: Low, Medium or High")

cursor = conn.cursor()

# Predefine the memory areas to match the table definition
cursor.setinputsizes(None, 25)

# Adjust the batch size to meet your memory and performance requirement
s
batch_size = 10000

with open('/enter the directory path/Emotions.csv', 'r') as csv_file:
    csv_reader = csv.reader(csv_file, delimiter=',')

    encabezados = next(csv_reader)

    sql = "insert into SMILES (ID,URL) values (:1, :2)" #SMILE, SAD, SU
RPRISE, NEUTRAL
    data = []
    for line in csv_reader:
        data.append((line[0], line[1]))
        if len(data) % batch_size == 0:
            cursor.executemany(sql, data)
            data = []
    if data:
        cursor.executemany(sql, data)
    conn.commit()
```

Code to make test SQL queries

```
os.chdir("/enter the directory path/Oracle/instantclient_19_8")

conn = cx_Oracle.connect(user='User', password='Password', dsn="enter t
he dns properties: Low, Medium or High")

cursor = conn.cursor()
cursor.execute('SELECT URL FROM SMILE')

results= []
for row in cursor:
    results.append(row)

cursor.close()
conn.close()

df= pd.DataFrame(results, columns=["URL"])
df.head(4)
```

Important: Code to consult the tables of the database in Oracle to transform the URLs into jpeg/jpg and download them to the computer itself

```
# Function to download and save images
def download_and_save_image(url, folder_path, retries=3):
  for i in range(retries):
      try:
          response = requests.get(url, timeout=10)
          if 'image' in response.headers.get('content-type',''):
              img = Image.open(BytesIO(response.content))
              hash_key = hashlib.sha1(url.encode()).hexdigest()
              file_name = hash_key + '.' + img.format.lower()
              file_extension = file_name.split('.')[-1]
              file_path = os.path.join(folder_path, file_name)
              img.save(file_path, format=file_extension)
              break
      except (ConnectionError, OSError):
          if i < retries - 1:
                time.sleep(2 ** i)  # exponential wait
          else:
                print(f"Error downloading image {url}")

# Folder where the downloaded images will be saved
folder_path = '/enter the directory path/Images/' #SMILES, SAD, SURPRIS
E, NEUTRAL

# Create the folder if it doesn't exist
if not os.path.exists(folder_path):
    os.makedirs(folder_path)

# Download and save the images in the folder
for url in df['URL']:
    download_and_save_image(url, folder_path)
```

CONVOLUTIONAL NEURAL NETWORKS

Installation of the necessary libraries

```
import os
import cv2
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from tensorflow import keras
from tensorflow.keras import layers
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score, KFold
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.applications.resnet50 import ResNet50
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.wrappers.scikit_learn import KerasClassifier
from sklearn.metrics import confusion_matrix
from tensorflow.keras.models import load_model
```

RESNET MODEL PRODUCTION AND REPORTING OF 4 EMOTIONS: SMILE, SAD, SURPRISE & NEUTRAL

```
# Upload the images from the SMILE, SAD, SURPRISE and NEUTRAL folders
def load_images(folder, label):
    images, labels = [], []
    for filename in os.listdir(folder):
        img = cv2.imread(os.path.join(folder, filename))
        if img is not None:
            img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
            img = cv2.resize(img, (100, 100))
            images.append(img)
            labels.append(label)
    return images, labels

smile_images, smile_labels = load_images('/enter the directory path/Ima
ges/SMILE', 1) # SMILE
sad_images, sad_labels = load_images('/enter the directory path/Images/
SAD', 0) # SAD
surprise_images, surprise_labels = load_images('/enter the directory pa
th/Images/SURPRISE', 2) # SURPRISE
neutral_images, neutral_labels = load_images('/enter the directory path
/Images/NEUTRLA', 3) # NEUTRAL

# Match the images and tags
images = np.concatenate((smile_images, sad_images, surprise_images, neu
```

```
tral_images))
labels = np.concatenate((smile_labels, sad_labels, surprise_labels, neu
tral_labels))

# Convert to numpy arrays and normalize images
images = np.array(images) / 255.0
labels = np.array(labels)

# Split dataset into training and test
X_train, X_test, y_train, y_test = train_test_split(
    images, labels, test_size=0.33, random_state=42,shuffle=True
)
y_train = to_categorical(y_train, num_classes=4)
y_test = to_categorical(y_test, num_classes=4)

# Load pretrained model
base_model = ResNet50(
    weights="imagenet",  # Load pretrained weights into ImageNet
    input_shape=(100, 100, 3),  # Neural network input size
    include_top=False,  # Do not include 1000 class classification laye
r
)

# Freeze Pretrained Model Weights
base_model.trainable = False

# Add additional layers to the model
inputs = keras.Input(shape=(100, 100, 3))
x = base_model(inputs, training=False)
x = keras.layers.ZeroPadding2D(padding=(1, 1))(x)
x = keras.layers.Conv2D(128, (3, 3), activation='relu', strides=(1, 1))
(x)
x = keras.layers.ZeroPadding2D(padding=(1, 1))(x)
x = keras.layers.Conv2D(64, (3, 3), activation='relu', strides=(1, 1))(
x)
x = keras.layers.GlobalAveragePooling2D()(x)
x = keras.layers.Dense(64, activation="relu")(x)
x = keras.layers.Dense(64, activation="relu")(x)
outputs = keras.layers.Dense(4, activation="softmax")(x)
model = keras.Model(inputs, outputs)

# compile the model
model.compile(optimizer="adam", loss="categorical_crossentropy", metric
s=["accuracy"])

# Train the model with EarlyStopping
early_stop = EarlyStopping(monitor='val_loss', mode='min', verbose=1, p
atience=5) # Try at least 5
history = model.fit(X_train, y_train, epochs=100, batch_size=128, valid
```

```
ation_data=(X_test, y_test))

# Calculate model precision
score = model.evaluate(X_test, y_test, verbose=0)
print("Model Accuracy: %.2f%%" % (score[1] * 100))

# Plot model accuracy in training and testing
plt.plot(history.history['Accuracy'])
plt.plot(history.history['Val_Accuracy'])
plt.title('Model Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Test'], loc='upper left')
plt.show()

# Plot model loss in training and testing
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()
```

THE MODEL IS SAVED

```
model.save('/enter the directory path/Model_4_Classes.h5') # East for p
roduction
```

CONFUSION MATRIX

```
# Make predictions on the test data set
y_pred = model.predict(X_test)

# Convert predictions to labels
y_pred_labels = np.argmax(y_pred, axis=1)
y_test_labels = np.argmax(y_test, axis=1)

# Calculate the confusion matrix
conf_matrix = confusion_matrix(y_test_labels, y_pred_labels)
print("Confusión Matrix:")
print(conf_matrix)
print(y_pred_labels)
print(y_test_labels)
```

COLLECT AND SAVE INFORMATION IN A CSV FILE

```
Epochs2=[]
for i in range(len(history.history['Accuracy'])):
    Epochs2.append(i+1)

Model2=[]
for i in range(len(history.history['Accuracy'])):
    Model2.append("Multiclass model (4) using ResNet and own images")

dataAL2= {"Model":Model2,"Epochs":Epochs2, "Accuracy Train":history.his
tory['Accuracy'],"Accuracy Test":history.history['Val_Accuracy'],"Loss
Train":history.history['Loss'],"Loss Test":history.history['Val_Loss']}

dfAL2= pd.DataFrame(dataAL2)
dfAL2

dfAL2.to_csv("/enter the directory path/Reporting_Train.csv", index=Fal
se)

num_params_dense = sum(layer.count_params() for layer in model.layers i
f isinstance(layer, layers.Dense))

# Get the total number of neurons from the convolutional layers
total_conv_neurons = 0
for layer in model.layers:
    if isinstance(layer, layers.Conv2D):
        total_conv_neurons += layer.output_shape[-1] * layer.output_sha
pe[-2] * layer.filters

total_dense_neurons = 0
for layer in model.layers:
    if isinstance(layer, layers.Dense):
        total_dense_neurons += layer.units

layer1= model.layers[-2]  # Activation of the dense layers except the l
ast one
if isinstance(layer1, layers.Dense):
    Layer1_2=layer1.activation.__name__

layer2= model.layers[-1]  # Activation of the last dense layer
if isinstance(layer2, layers.Dense):
    Layer2_2=layer2.activation.__name__

TimeExecution= round(elapsed_time2/60,2)
Mean_Accuracy_Train= round(history.history['Accuracy'][-1]*100,2)
Mean_Accuracy_Test= round(history.history['Val_Accuracy'][-1]*100,2)
Mean_Loss_Train= round(np.mean(history.history['Loss']),2)
Mean_Loss_Test= round(np.mean(history.history['Val_Loss']),2)
Num_Dense_Layers= len(model.layers) - 3
Num_Dense_Neu= total_dense_neurons
Num_Conv_Layers= num_conv_layers = sum(isinstance(layer, layers.Conv2D)
```

```
 for layer in model.layers)
Num_Conv_Neu= total_conv_neurons
Activation_Dense_Layers=Layer1_2
Activation_Last_Dense_Layers=Layer2_2
Activation_Conv_Layers= "Relu"
overfitting= "Drop Out"
Model= "Multiclass model (4) using ResNet and own images"

metrics2= {"Model":Model,"Time Execution (minuts)": TimeExecution,"Mean
 Accuracy for Train":Mean_Accuracy_Train,
          "Mean Accuracy for test":Mean_Accuracy_Test,"Mean Loss for Tr
ain":Mean_Loss_Train,"Mean Loss for Test":Mean_Loss_Test,
          "Number of Dense Layers":Num_Dense_Layers,"Number of Dense ne
urons":Num_Dense_Neu,"Number of Convolutional Layers":Num_Conv_Layers,
          "Número of Convolutional neurons":Num_Conv_Neu,"Activations o
f the Dense layers":Activation_Dense_Layers,
          "Activation of the last Dense layer":Activation_Last_Dense_La
yers,"Activation of the Convolutional layers":Activation_Conv_Layers,
          "Overfit Method":overfitting}

index2=[0]
dfMetrics2= pd.DataFrame(metrics2, index=index2)
dfMetrics2

dfMetrics2.to_csv("/enter the directory path/Reporting_Train2.csv", ind
ex=False)
```

IMAGE PREDICTION TEST

In the following code a new image can be entered to predict the emotion: SMILE, SAD, SURPRISE & NEUTRAL. This code graphs the entered image and will print the predicted label and its probability.

```
# Upload image
img = cv2.imread('/enter the directory path/IMG_TEST.jpg', cv2.COLOR_GR
AY2RGB)

# Preprocess the image
img = cv2.resize(img, (128, 128))
img = img / 255.0

# Predict Image Label
prediction = model.predict(np.array([img]))

if prediction[0][1] > 0.5:
    label = 'SMILE'
    print(f"The probability of SMILE is: {round(prediction[0][1]*100,2)
}%.")
elif prediction[0][2] > 0.5:
    label = 'SURPRISE'
```

```
    print(f"The probability of SURPRISE is: {round(prediction[0][2]*100
,2)}%.")
elif prediction[0][0] > 0.5:
    label = 'SAD'
    print(f"The probability of SAD is: {round(prediction[0][0]*100,2)}%
.")
elif prediction[0][3] > 0.5:
    label = 'NEUTRAL'
    print(f"The probability of NEUTRAL is: {round(prediction[0][3]*100,
2)}%.")

# Show the image and the predicted label
plt.imshow(img.squeeze(), cmap='gray')
plt.title(label)
plt.show()
```

CHARGE THE NEURAL NETWORK TO DETECT 4 EMOTIONS: SMILE, SAD, SURPRISE & NEUTRAL

```
model = load_model('/enter the directory path/Model_4_Classes.h5')
```

METRICS FROM CNN TO ORACLE

This Script is to collect the metrics of all the convolutional neural network models created to later upload to Oracle and later to Power BI to do the reporting in Wix
Installing and loading Oracle libraries in Python

```
!pip install cx_Oracle

import cx_Oracle
import os
import openpyxl
import pandas as pd
import csv

import hashlib
import requests
from io import BytesIO
from PIL import Image
import re
import time
```

Connection with the database and creation of the 10 tables, Insertation of Data and SQL queries

In the code presented below: An example table is created, data is inserted into the created table, a query is made with SQL language and a DataFrame is created.

```
os.chdir("/enter the directory path/Oracle/instantclient_19_8")

conn = cx_Oracle.connect(user='User', password='Password', dsn="enter t
he dns properties: Low, Medium or High")

cursor = conn.cursor()

#Creating table as per requirement    ACCURACY_AND_LOSS, ADVANCED_METRI
CS, BASIC_METRICS , COMPARATIVE_1 ,
#                                    COMPARATIVE_2, COMPARATIVE_TEXT,
CONFUSION_MATRIX, ZORDEN1_MODEL,
#                                    ZORDEN2_MODEL, ZORDEN3_PARAMETERS
sql ='''CREATE TABLE ACCURACY_AND_LOSS(
   MODEL VARCHAR2(1000),
   EPOCHS INT,
   ACCTRAIN DECIMAL(10,10),
   ACCTEST DECIMAL(10,10),
   LOSSTRAIN DECIMAL(10,10),
   LOSSTEST DECIMAL(10,10))'''

cursor.execute(sql)
print("Table created successfully........")
```

```
# Commit your changes in the database
conn.commit()

#Closing the connection
cursor.close()
conn.close()
```

```
os.chdir("/enter the directory path/Oracle/instantclient_19_8")

conn = cx_Oracle.connect(user='User', password='Password', dsn="enter t
he dns properties: Low, Medium or High")

cursor = conn.cursor()

# Predefine memory areas to match table definition
cursor.setinputsizes(None, 25)

# Adjust the batch size based on your memory and performance requiremen
ts
batch_size = 10000

# Open the excel file
workbook = openpyxl.load_workbook('/enter the directory path/DATASET_RE
PORTING.xlsx')

# Gets the sheet "Accuarcy_Loss"
sheet = workbook['Accuracy_Loss']

# Prepare the SQL query
sql = "insert into ACCURACY_AND_LOSS (MODEL, EPOCHS, ACCTRAIN, ACCTEST,
 LOSSTRAIN, LOSSTEST) values (:1, :2, :3, :4, :5, :6)"

data = []

# Iterate over the rows of the sheet
for row in sheet.iter_rows(values_only=True):
    data.append((row[0], row[1], row[2], row[3], row[4], row[5]))

    if len(data) % batch_size == 0:
        cursor.executemany(sql, data)
        data = []

if data:
    cursor.executemany(sql, data)

conn.commit()
```

```
os.chdir("/enter the directory path/Oracle/instantclient_19_8")

conn = cx_Oracle.connect(user='User', password='Password', dsn="enter t
```

```
he dns properties: Low, Medium or High")

cursor = conn.cursor()
cursor.execute('SELECT * FROM ACCURACY_AND_LOSS')

results= []
for row in cursor:
    results.append(row)

cursor.close()
conn.close()

df= pd.DataFrame(results)
df.head(4)
```

TESTING EFFECTIVENESS WITH CONVOLUTIONAL NEURAL NETWORKS: PRODUCTION IN THE COMPUTER WEBCAM

Installation of the necessary libraries

```
import cv2
import tensorflow as tf
```

Load models for face detection and neural network

```
# Load the pre-trained model of Haar Cascades for face detection
face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + "haarcasca
de_frontalface_default.xml")
smile_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + "haarcasc
ade_smile.xml")

# Load the trained model for detecting the 4 emotions
# In this section you will find the links of all the models tested in p
roduction
model = tf.keras.models.load_model('/enter the directory path/Model_4_C
lasses.h5')

# Set the camera to capture video
cap = cv2.VideoCapture(0)
```

The model is called and the computer camera is opened, that is, the trained convolutional neural network is put into production

```
while True:
    # Read a video frame
    ret, frame = cap.read()

    # Do not convert image to grayscale
    gray = frame.copy()

    # Detect faces in the image
    faces = face_cascade.detectMultiScale(gray, 1.3, 5)

    # Iterate through each detected face
    for (x, y, w, h) in faces:
        # Extract the region of the image that contains the face
        face_roi = gray[y:y+h, x:x+w]

        # Preprocess the image to be classified by the model
        face_roi = cv2.resize(face_roi, (100, 100), interpolation = cv2
.INTER_AREA)
```

```
        face_roi = face_roi.astype("float") / 255.0
        face_roi = tf.keras.preprocessing.image.img_to_array(face_roi)
        face_roi = tf.expand_dims(face_roi, axis=0)

        # Predict the emotion of the person
        prediction = model.predict(face_roi)

        # If the probability that the person is smiling is greater than
 50%, show "SMILE" on the screen
        if prediction[0][1] > 0.5:
            cv2.putText(frame, "SMILE ({:.2f})".format(prediction[0][1]
), (x, y-10), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2)
            cv2.rectangle(frame, (x, y), (x+w, y+h), (0, 255, 0), 2)
        # If the probability that the person is sad is greater than 50%
, show "SAD" on the screen
        elif prediction[0][0] > 0.5:
            cv2.putText(frame, "SAD ({:.2f})".format(prediction[0][0]),
 (x, y-10), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 255), 2)
            cv2.rectangle(frame, (x, y), (x+w, y+h), (0, 0, 255), 2)
        # If the probability that the person is surprised is greater th
an 50%, show "SURPRISE" on the screen
        elif prediction[0][2] > 0.5:
            cv2.putText(frame, "SURPRISE ({:.2f})".format(prediction[0]
[2]), (x, y-10), cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 0, 0), 2)
            cv2.rectangle(frame, (x, y), (x+w, y+h), (255, 0, 0), 2)

        # If the probability that the person is neutral is greater than
 50%, show "Neutral" on the screen
        elif prediction[0][3] > 0.5:
            cv2.putText(frame, "NEUTRAL ({:.2f})".format(prediction[0][
3]), (x, y-10), cv2.FONT_HERSHEY_SIMPLEX, 1, (25, 25, 25), 2)
            cv2.rectangle(frame, (x, y), (x+w, y+h), (25, 25, 25), 2)

    # Show the result on the screen
    cv2.imshow('frame', frame)

    # Wait 1 millisecond before reading the next video frame
    if cv2.waitKey(1) == ord('q'):
        break

# Release the camera and close the viewing window
cap.release()
cv2.destroyAllWindows()
```