Artículo de Investigación

Construcción de una app nativa Android para la detección facial de emociones usando técnicas de inteligencia artificial

Construction of a native Android app for facial emotion detection using artificial intelligence techniques

Avilés Valencia, Jorge Humberto; Centeno Alomoto, Mayra Gabriela; Encarnación Umatambo, María Lucila; Trujillo Quinto, Wilfrido Amilcar

Jorge Humberto Avilés Valencia
jorge.aviles@institutotraversari.edu.ec
Instituto Superior Tecnológico Mayor Pedro Traversari - ISTPET, Ecuador

Mayra Gabriela Centeno Alomoto
mayra.centeno@institutotraversari.edu.ec
Instituto Superior Tecnológico Mayor Pedro Traversari - ISTPET, Ecuador

María Lucila Encarnación Umatambo
maria.encarnacion@institutotraversari.edu.ec
Instituto Superior Tecnológico Mayor Pedro Traversari - ISTPET, Ecuador

Wilfrido Amilcar Trujillo Quinto
wilfrido.trujillo@institutotraversari.edu.ec
Instituto Superior Tecnológico Mayor Pedro Traversari - ISTPET, Ecuador

Pro Sciences: Revista de Producción, Ciencias e Investigación
CIDEPRO, Ecuador
e-ISSN: 2588-1000
Periodicidad: Trimestral
Vol. 6, No. 45, 2022
editor@journalprosciences.com

Recepción: 9 Junio 2022
Aprobación: 7 Agosto 2022

DOI: https://doi.org/10.29018/issn.2588-1000vol6iss45.2022pp52-61

Resumen: La capacidad de reconocer emociones faciales no solamente es una habilidad propia de los seres humanos, con el avance tecnológico en el campo de la Inteligencia Artificial (IA), los computadores también son capaces de aprender a reconocer emociones faciales a través de varios tipos de software de reconocimiento y librerías creadas para este fin. El presente trabajo aborda la clasificación de emociones en tiempo real en base a la expresión facial humana de cada una de las siete emociones universales: ira, disgusto, miedo, felicidad, neutralidad, tristeza y sorpresa mediante el desarrollo de una app móvil nativa Android de visión en tiempo real que puede clasificar las emociones. El uso de Python, OpenCV, Keras, Tensorflow y Redes Neuronales Convolucionales (CNN) han hecho posible construir un algoritmo que realiza la detección, extracción y evaluación de estas expresiones faciales para el reconocimiento automático de emociones. Para la construcción de la app se utilizó el conversor de Tensorflow lite, el cual convierte un modelo de Keras guardado, en un modelo compatible con dispositivos móviles. Se consideran las principales características de la cara para la detección de expresiones faciales. Para determinar las distintas emociones se utilizan las variaciones en cada uno de los rasgos principales. Para detectar y clasificar diferentes clases de emociones, se utilizan algoritmos de aprendizaje automático entrenando diferentes conjuntos de imágenes. El propósito de este trabajo es desarrollar mediante un modelo tflite una aplicación móvil que pueda, en tiempo real, detectar las 7 emociones universales.

Palabras clave: detección de emociones, Tensorflow, Redes Neuronales Convolucionales.

Abstract: The ability to recognize facial emotions is not only an ability of human beings, with technological advances in the field of Artificial Intelligence (AI), computers are also capable of learning to recognize facial emotions through various types of software of recognition and libraries created for this purpose. The present work deals with the classification of emotions in real time based on the human facial expression of each of the seven universal emotions: anger, disgust, fear, happiness, neutrality, sadness, and surprise through the development of a native Android mobile app real-time

Cómo citar: Avilés Valencia, J. H., Centeno Alomoto, M. G., Encarnación Umatambo, M. L., & Trujillo Quinto, W. A. (2022). Construcción de una app nativa Android para la detección facial de emociones usando técnicas de inteligencia artificial. Pro Sciences: Revista De Producción, Ciencias E Investigación, 6(45), 52-61. https://doi.org/10.29018/issn.2588-1000vol6iss45.2022pp52-61

view that can classify emotions. The use of Python, OpenCV, Keras, Tensorflow and Convolutional Neural Networks (CNN) have made it possible to build an algorithm that detects, extracts, and evaluates these facial expressions for automatic emotion recognition. For the construction of the app, the Tensorflow lite converter was used, which converts a saved Keras model into a model compatible with mobile devices. The main features of the face are considered for the detection of facial expressions. Variations in each of the main traits are used to determine the different emotions. To detect and classify different classes of emotions, machine learning algorithms are used by training different sets of images. The purpose of this work is to develop a mobile application using a tflite model that can, in real time, detect the 7 universal emotions.

Keywords: emotion detection, Tensorflow, Convolutional Neural Networks.

Introducción

Las emociones son la reacción del ser humano frente a su relacionamiento con otros o las reacciones que tiene frente a los eventos que se enfrentan a diario, por lo que pueden considerarse como componentes básicos de la personalidad de los seres humanos (Esquivel, 2015). Por lo tanto, analizar las emociones de los seres humanos producidas por la realización de una determinada tarea, puede ayudar a comprender la experiencia de esta. (Jaiswal, 2020)

La clasificación de la expresión facial es cómo se diferencian y asocian las emociones. Cada ser humano identifica cada una de las 7 emociones universales (“enojado”, “disgusto”, “miedo”, “feliz”, “triste”, “sorpresa” , "neutral") con una precisión del 65 % ± 5 % (Arriaga, Plöger, & Valdenegro, 2017).

El uso de computadoras en el reconocimiento de emociones es un área de investigación ampliamente discutida. Hasta la fecha, los investigadores se han centrado principalmente en la clasificación de emociones utilizando el modo de texto y voz. Se han llevado a cabo años de investigación para desarrollar y evaluar las diferentes formas que se pueden utilizar para automatizar la clasificación de emociones. (Jaiswal, 2020). Se han desarrollado una amplia gama de técnicas a partir de varios recursos, como el Machine Learning (ML) para ayudar con el reconocimiento de emociones usando rasgos faciales. (Arriaga, Plöger, & Valdenegro, 2017)

El enfoque estadístico hacia este proceso requeriría el uso de algoritmos de aprendizaje automático supervisados que se ejecutan a través de un gran conjunto de datos a partir del cual el modelo aprende y predice la expresión facial más precisa. (Jaiswal, 2020)

Los algoritmos de aprendizaje automático brindan una alta eficiencia de clasificación; además, hoy en día con la evolución de las redes neuronales y los procesadores, las soluciones como la IA en tiempo real pueden analizar las expresiones faciales en tiempo real desde la entrada en vivo a través de una cámara conectada. (Arriaga, Plöger, & Valdenegro, 2017)

Para esta investigación se utilizaron dos modelos: *faster rcnn inception v2 coco* (Faster) y *ssd inception v2 coco* (SSD). Estos dos modelos son usados con frecuencia en la detección e identificación de objetos dentro de una imagen debido a su gran nivel de precisión, tiempo de detección y tiempo de entrenamiento las cuales son menores en comparación a otros modelos. (Avilés & Toapanta, 2019).

METODOLOGÍA

El presente trabajo fue realizado mediante la aplicación de un enfoque cuantitativo ya que en una primera instancia se realizó la recolección de datos, que posteriormente servirán para el entrenamiento de la CNN.

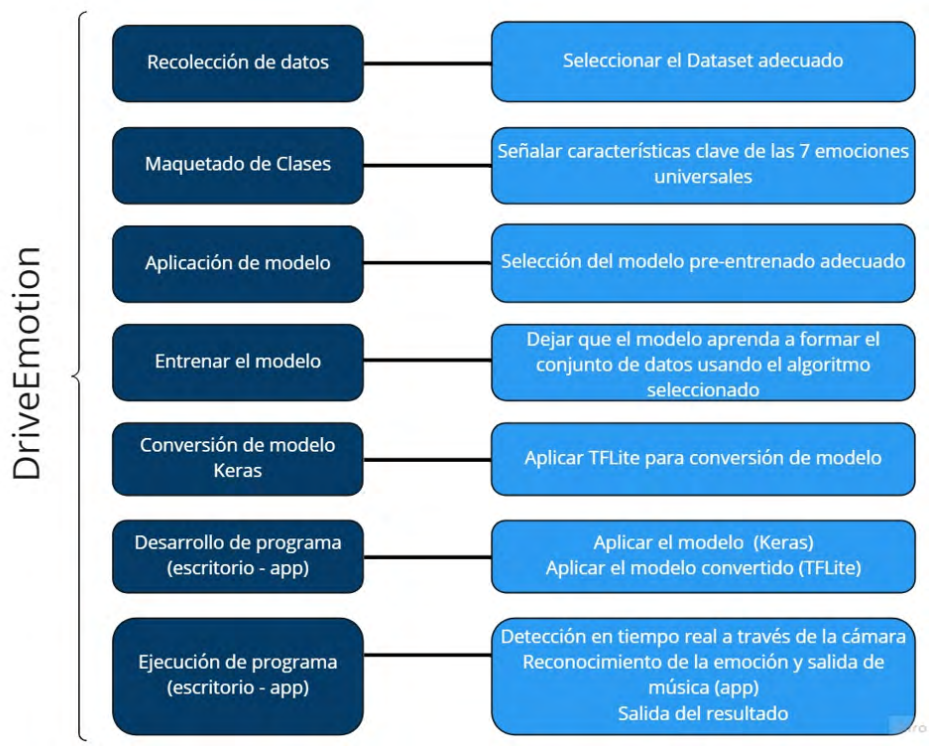

Ilustración 1. Etapas del desarrollo de DriveEmotion
Elaborado por: los autores

Recolección de datos

El desarrollo de aplicaciones de Machine Learning (ML) y Artificial Vision (AV) depende principalmente de la cantidad y calidad de datos (texto, imágenes, videos, audio) que utilizan en su entrenamiento para ser precisos. (G. Webb, 2001)

Cuando el conjunto de datos es insuficiente o tiene demasiadas inconsistencias, los sistemas basados en ML producirán resultados erróneos poco aplicables a la realidad. (Ikonomakis, Kotsiantis, & Tampakas)

Se analizaron varios datasets disponibles sobre detección de emociones, de entre los cuales se eligió el dataset fer2013 debido a que es un conjunto de datos limpio con más de 28.000 imágenes que se pueden usar para entrenar la CNN, otra de las ventajas de este dataset, es que contiene suficientes imágenes por cada una de las clases a detectar (“enojado”, “disgusto”, “miedo”, “feliz”, “triste”, “sorpresa” , "neutral"). (Sambare, 2020)

De este dataset se utilizaron 200 imágenes por clase, dando un total de 1200. A continuación de enumeran las caracteristicas más relevantes de las imágenes utilizadas para el entrenamiento de la CNN.

- Las imágenes y sus respectivas expresiones faciales se registran de manera que la cara se ajuste al centro aproximadamente.
- El tamaño de cada una de las imágenes es de 48 x 48 pixeles.
- Las imágenes tienen una configuración de color en escala grises.
- Cada una de las imágenes tuvo un peso aproximado de 4,00 KB

Maquetado de Clases

Se tomó en cuenta la fisionomía facial, es decir, el cambio que se produce en el rostro de la persona cuando experimenta alguna de las 7 emociones universales, esto ayuda a identificar las características que posteriormente servirán para detectar si la expresión pertenece a: “enojado”, “disgusto”, “miedo”, “feliz”, “triste”, “sorpresa” , "neutral". (Ekman, 2017)

A continuación se describen las caracteristicas más importantes que definen una emoción:

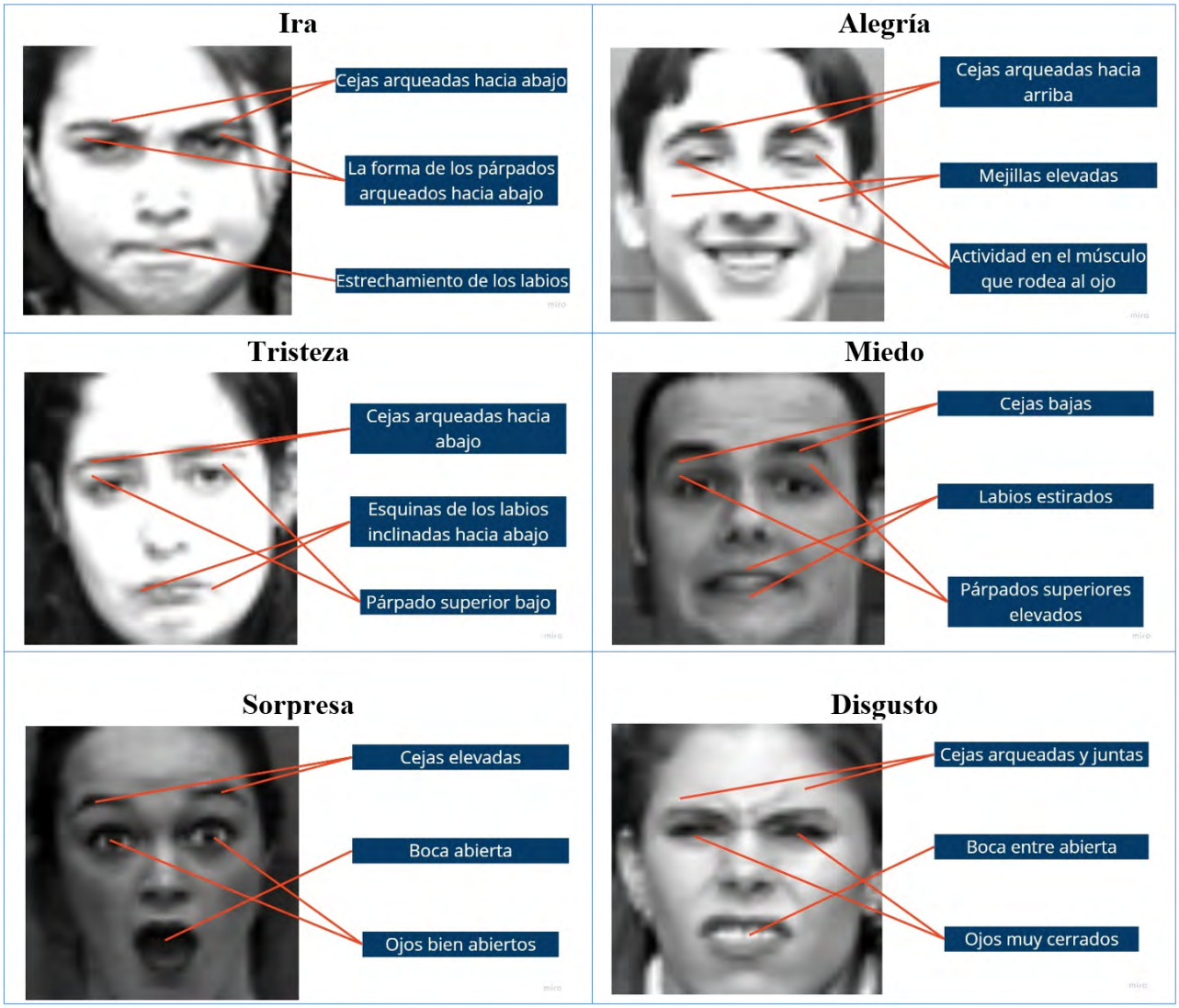

Ilustración 2. Características faciales de las emociones universales

Elaborado por: los autores

Aplicación del modelo

Tensorflow, dispone de varios modelos de detección entrenados previamente con el conjunto de datos COCO 2017. Estos modelos pueden ser útiles para la inferencia lista para usar si está interesado en las categorías que ya están en esos conjuntos de datos.

También son útiles para entrenar nuevos conjuntos de datos ya que al ser modelos entrenados previamente, el tiempo total de iteraciones que nuestro modelo deberá recorrer son menos en comparación a si se intenta entrenar un modelo desde cero.

Tabla 1. Características de los modelos SSD - Faster

Característica	SSD	Faster
Dataset	COCO	COCO
Red	CNN	CNN
Velocidad por iteración	Depende del Hardware	Depende del Hardware
Salida	Cajas/Máscaras	Cajas/Máscaras
TotalLoss recomendado	(1 < 0 < 2)%	(0≤0.05)%

Obtenido de: (Birodkar, 2021)

Entrenar el modelo

SSD y Faster fueron entrenados sobre la siguiente configuración de hardware y software:

Tabla 2. Características de Software y Hardware

Características	Especificaciones
Ram	DDR4, 16GB, 4000MHz
Cpu	I7 10750H, 6 núcleos, 12 hilos, 5,00 GHz, 12MB en caché.
Gpu	RTX 2060, 1920 Cuda Cores, 1365MHz
SO	Windows 11 x64 – Compilación (22000.652)
Tensorflow	v1.12
OpenCV	v3.4.13
Python	v3.7.11
Anaconda	v1.9.6
Protobuf	v3.4.0
CUDA - CUDNN	v10

Elaborado por: los autores

El entrenamiento de la CNN se realizó utilizando la potencia de cómputo de la GPU ya que disminuye el tiempo entre iteraciones, logrando más iteraciones en un menor tiempo.

Como se había mencionado en la *Tabla 2*, ambos modelos SSD y Faster tienen un TotalLoss recomendado, esto quiere decir que los modelos deberán alcanzar un TotalLoss dentro de los valores especificados si se desea lograr resultados de detección “justos”. (Vladimirov, 2022) Obviamente, un TotalLoss más bajo es mejor; sin embargo, se debe evitar un TotalLoss muy bajo, ya que el modelo puede terminar sobre ajustando el conjunto de datos, lo que significa que tendrá un rendimiento deficiente cuando se aplique a imágenes fuera del conjunto de datos.

Para el entrenamiento se dividió el dataset de la siguiente forma: 90% de imágenes se utilizaron para el entrenamiento, mientras que el 10% se utilizaron para propósitos de evaluación del entrenamiento de los modelos.

Los resultados del entrenamiento de los modelos se describen con más detalle en la sección RESULTADOS.

Conversión de modelo Keras

El entrenamiento de los modelos SSD y Faster genera un archivo con extensión .h5 el cual deberá comprobarse más adelante para verificar el correcto funcionamiento en la detección de emociones.

Para la comprobación de este modelo, se realizó un pequeño *script* en *python* que carga y prepara una imagen para la detección de la emoción.

```
from tensorflow.keras.preprocessing.image import load_img
from tensorflow.keras.preprocessing.image import img_to_array
from tensorflow.keras.models import load_model


def load_image(filename):
    img = load_img(filename, target_size=(64,64))
    img = img_to_array(img)
    img = img.reshape(1, 64, 64, 3)
    img = img.astype('float32')
    print(img)
    return img

def run_example():
    img = load_image('s.jpg')
    model = load_model('model.h5')
    result = model.predict(img)
    print(result)
run_example()
```

Ilustración 3. Script para evaluar el archivo Keras con extensión h5
Elaborado por: los autores

Una vez comprobado el modelo se deberá convertir el archivo con extensión h5 a un formato tflite, para ello se generó un pequeño script en python que utiliza TFLiteConverter de TensorFlow y toma como entrada el archivo con extensión h5 y lo transforma a un formato tflite compatible con dispositivos móviles.

```
from tensorflow.keras.models import load_model
import tensorflow as tf


model = load_model('model.h5')
converter = tf.lite.TFLiteConverter.from_keras_model(model)


tflite_model = converter.convert()
file = open( 'model.tflite' , 'wb' ).write( tflite_model )
```

Ilustración 4. Script para convertir un modelo Keras a tflite
Elaborado por: los autores

Desarrollo de programa

Escritorio

Para el desarrollo de la aplicación de escritorio se desarrolló un script en python que define las emociones a detectar, carga el model.h5 descrito en la sección anterior y habilita la entrada de datos en tiempo real mediante la cámara del computador.

La salida del script en python, es la siguiente:

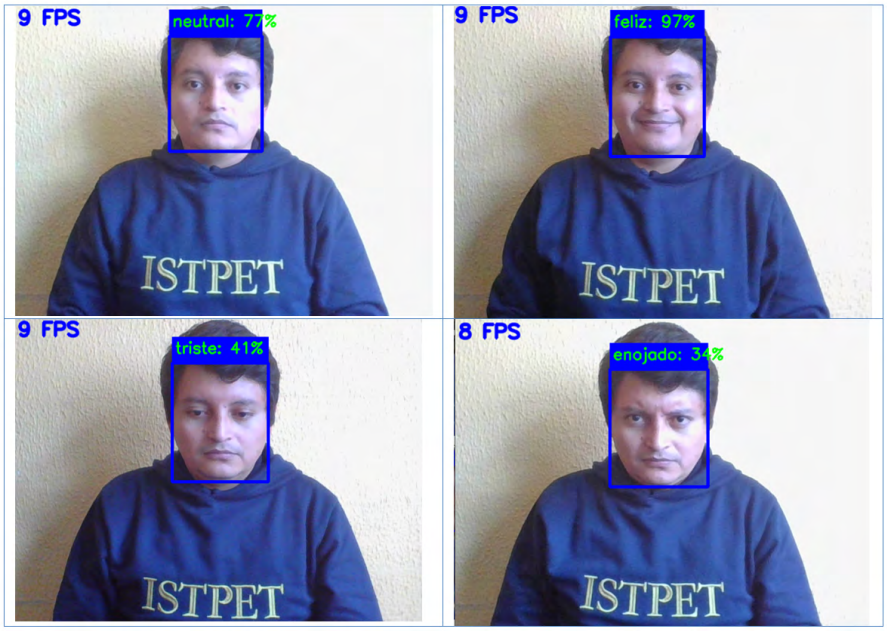

Ilustración 5. Salida detección de emociones (escritorio)
Elaborado por: los autores

Aplicación móvil

Se consideró el desarrollo de una App nativa en Android debido a la necesidad que tendrá el aplicativo en el consumo de recursos del dispositivo móvil para poder ejecutar la codificación necesaria en conjunto con el modelo tflite convertido y entrenado previamente.

La aplicación se desarrolló sobre un Sistema Operativo Android Oreo (v8.0) debido a su gran compatibilidad en la mayoría de los dispositivos, teniendo un 86.7% de compatibilidad.

La salida de la app de detección de emociones es la siguiente:

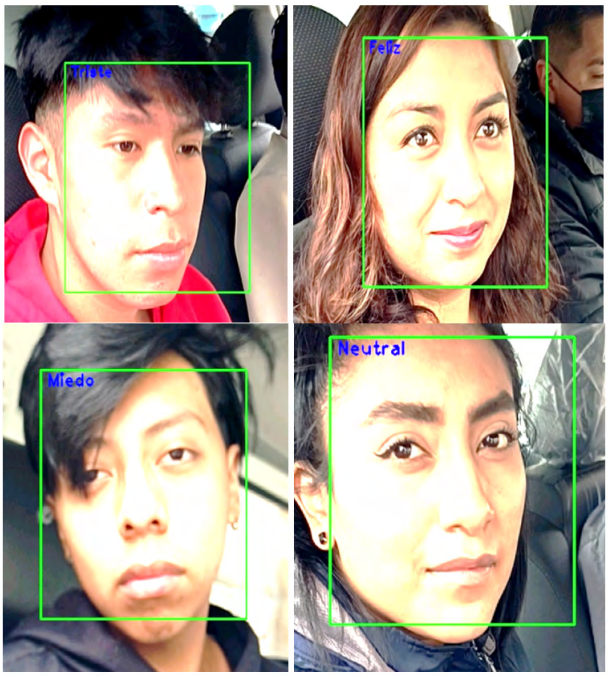

Ilustración 6. Salida de app móvil DriveEmotion
Elaborado por: los autores

Resultados y Discusión

El hiperparámetro learning rate (lr) ayuda a controlar de manera directa el tiempo total de entrenamiento de cada modelo, es decir, un lr demasiado bajo, aumenta el tiempo en que el modelo se adapta a un problema mientras que un lr demasiado alto puede hacer que el modelo se adapte más rápido, pero como consecuencia de ello, el modelo no será 100% confiable, generando soluciones no viables. (Brownlee, 2019)

En el caso de los modelos FASTER y SSD al ser modelos que ya han sido entrenados previamente este hiperparámetro estuvo configurado por defecto.

Durante el entrenamiento, ambos modelos: SSD y Faster obtuvieron entre 110k a 120k iteraciones, alcanzando un TotalLoss aproximado de 1.067 (Ilustración 8) y 0,002433 respectivamente (Ilustración 7).

Para el control del entrenamiento, se utilizó la herramienta llamada TensorBoard dentro TensorFlow.

En el modelo FASTER se usó un lr diferente a medida que aumentan las iteraciones, es decir, a partir de la iteración cero el modelo utilizó un lr de 0,00021 y al llegar a la iteración 105k usó un lr de 2.0001e-3 ~ 0.0306 esto implicó que el modelo se adaptó al problema en un periodo más corto de tiempo (Zulkifli, 2018).

A diferencia del modelo FASTER, el modelo SSD utilizó un único valor de lr de 0,00450 para todas las iteraciones, obteniendo un Total Loss de 1,067% dentro del rango recomendado.

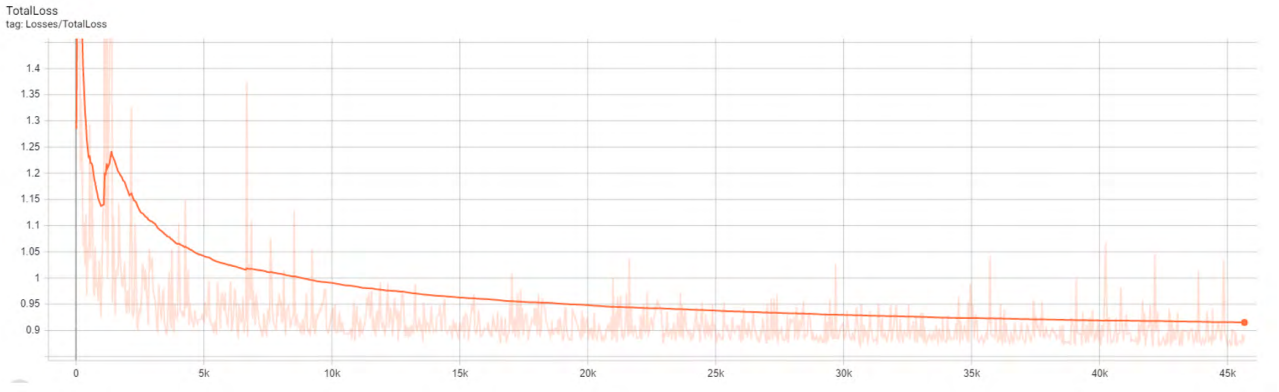

Ilustración 7. TotalLoss de modelo Faster
Elaborado por: los autores

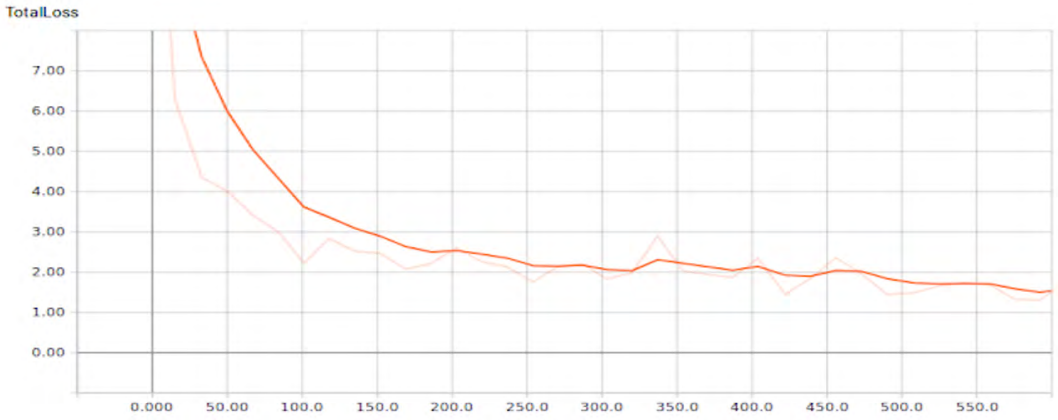

Ilustración 8. TotalLoss de modelo SSD
Elaborado por: los autores

La información de tiempo total utilizado para entrenar los modelos se muestra a continuación:

Tabla 3. Información general de entrenamiento SSD vs Faster

Indicador	FASTER	SSD
Iteraciones	107050	119250
Tiempo total de entrenamiento	2d 20h 05m 15s	6d 12h 32m 08s
TotalLoss alcanzado	0.002433	1.067

Elaborado por: los autores

Referente al rendimiento de detección de los modelos SSD Y Faster, se obtuvieron resultados variados, la emoción con más porcentaje de detección fue la emoción de felicidad, seguida de la emoción neutral, triste y enojado, los detalles se muestran a continuación.

Tabla 4. Rendimiento de modelos SSD vs Faster

Emoción	Modelo (% de detección)	
	SSD	FASTER
Enojado	35%	30%
Disgusto	15%	13%
Miedo	45%	39%
Feliz	89%	69%
Triste	72%	72%
Sorpresa	10%	15%
Neutral	88%	73%

Elaborado por: los autores

En base a la *Tabla 4*, el modelo seleccionado fue SSD que posteriormente fue convertido a formato h5 y tflite para la implementación de la app nativa Android, la salida de este modelo implementado en escritorio y móvil se lo puede ver en la sección *Desarrollo del programa.*

CONCLUSIONES

Este trabajo logró la construcción de un aplicativo de escritorio y móvil a través de la incorporación de modelos de Redes Neuronales Convolucionales (CNN) que fueron entrenados previamente, usando un conjunto de imágenes de emociones faciales.

Los modelos SSD y Faster seleccionados para la realización de la presente investigación son dos de los mejores modelos de CNN utilizados junto con TensorFlow para la detección y clasificación de imágenes, debido a que presentan un mayor rendimiento/velocidad en comparación con otros modelos.

En promedio, el rendimiento de los modelos utilizados para la detección de las emociones faciales fue de 50,57% para el modelo SSD con niveles de detección superiores al 88%, mientras que para el modelo Faster fue de 44,43% obteniendo niveles de detección superiores al 69%, siendo el modelo SSD superior al modelo Faster.

Las aplicaciones de escritorio y móvil utilizaron el mismo modelo entrenado con SSD pero el rendimiento, en general, fue mucho mejor en la aplicación de escritorio debido a las condiciones de uso. El aplicativo móvil al usarse en condiciones de constante movimiento y cambio de luz tuvo

ciertos inconvenientes al momento de la detección a través de la cámara, en general el rendimiento del aplicativo móvil fue del 72,45%, mientras que la aplicación de escritorio tuvo un rendimiento mayor al 80%.

Referencias Bibliográficas

- Arriaga, O., Plöger, P., & Valdenegro, M. (2017). Real-time Convolutional Neural Networks for.
- Avilés, J., & Toapanta, H. (2019). Construcción de un conjunto de imágenes faciales con metadatos biométricos y rasgos étnicos de ecuatorianos.
- Birodkar, V. (7 de Mayo de 2021). Github. Obtenido de TensorFlow 2 Detection Model Zoo: https://cutt.ly/jLOMzz4
- Brownlee, J. (25 de Enero de 2019). Understand the Impact of Learning Rate on Neural Network Performance. Obtenido de https://bit.ly/2Ng2dJB
- Ekman, P. (2017). El rostro de las emociones. RBA Libros.
- Esquivel, L. (2015). El libro de las emociones. Debolsillo.
- G. Webb, M. P. (2001). Machine Learning for User Modeling. User Modeling and User-Adapted Interaction, vol. 11, nº 1, 19-29.
- Ikonomakis, E., Kotsiantis, S., & Tampakas, V. (s.f.). Text Classification Using Machine Learning Techniques. WSEAS Transactions on Computers, 966-974.
- Jaiswal, R. (2020). Facial Expression Classification Using Convolutional Neural Networking and Its Applications. International Conference on Industrial and Information Systems (ICIIS).
- Sambare, M. (2020). Learn facial expressions from an image. Obtenido de Kaggle: https://cutt.ly/fJOLAVY
- Vladimirov, L. (22 de Marzo de 2022). TensorFlow Object Detection API Tutorial. Obtenido de https://cutt.ly/mLO9xkT
- Zulkifli, H. (21 de Enero de 2018). Understanding Learning Rates and How It Improves Performance in Deep Learning. Obtenido de https://bit.ly/3Qc1dT3