Escuela
Politécnica
Superior

Sistema de reconocimiento de emociones

Grado en Ingeniería Robótica

Trabajo Fin de Grado

Autor:
Carlos Murcia Ferrándiz
Tutor/es:
Miguel Ángel Cazorla Quevedo

Universitat d'Alacant
Universidad de Alicante

Julio 2021

Sistema de reconocimiento de emociones

Autor
Carlos Murcia Ferrándiz

Tutor/es
Miguel Ángel Cazorla Quevedo
Ciencia de la omputación e Inteligencia Artificial

Grado en Ingeniería Robótica

Escuela
Politécnica
Superior

Universitat d'Alacant
Universidad de Alicante

ALICANTE, Julio 2021

Preámbulo

"La razón principal que me ha llevado a realizar este trabajo es mi creciente fascinación por el mundo de la IA (Inteligencia Artificial) a lo largo de la carrera, así como la posible implementación de esta dentro de la robótica. La principal finalidad de este proyecto es conseguir desarrollar una red neuronal robusta capaz de reconocer las emociones humanas."

Agradecimientos

Primeramente me gustaría agradecer a mi tutor Miguel Ángel Cazorla Quevedo todo el gran apoyo y consejo prestado durante la realización de este TFG.

También me gustaría agradecer a mi familia y novia todo el apoyo recibido que me ha servido para continuar en los momentos más difíciles. Me gustaría dedicarle este trabajo especialmente a mi abuela que por desgracia no va a poder ver en lo que me he convertido este último año, nunca te olvidaremos y siempre estarás en nuestros pensamientos.

Finalmente, agradecer a mis amigos toda la compañía y buenos momentos vividos durante estos cuatro años.

Sin todos ellos esto seguramente no habría sido posible y es a ellos a quien dedico este trabajo. Gracias de todo corazón.

El aspecto más triste de la vida
es que la ciencia reúne el
conocimiento más rápidamente
que la sociedad la sabiduría.

Isaac Asimov.

Si hiciésemos todas las cosas
de las que somos capaces,
nos asombraríamos.

Thomas Edison.

Índice general

Índice de figuras

Índice de tablas

1 Introducción

El avance de ciertos campos como la Inteligencia Artificial o el *Internet of Things* (IoT), entre otros, han sido motivados en gran medida por un incremento de la capacidad computacional de la tecnología hardware, así como un desarrollo de reconocimiento de patrones más sofisticados, o la capacidad de recolectar grandes cantidades de datos de múltiples dominios de información. Estos avances han permitido en la actualidad el auge de sistemas de Inteligencia Artificial capaces de realizar tareas humanas con cierto grado de complejidad.

La Inteligencia Artificial es ampliamente usada en sistemas reales en una gran variedad de problemas como la gestión y el control, fabricación, ingeniería, educación, equipamiento, cartografía, software, sistemas de armamento, proceso de datos, finanzas. Así como multitud de aplicaciones comerciales como la configuración, el diagnosis en problemas médicos, la interpretación y el análisis de problemas matemáticos complejos, la monitorización durante la fabricación y gestión de procesos científicos, las interfaces inteligentes, los sistemas de lenguaje natural, los sistemas de visión computerizada en controles de calidad, el desarrollo de software, entre muchos otros.

Una de las áreas que ha despertado un gran interés en la investigación en IA durante los últimos años es la *affective computing*, cuyo objetivo es el de desarrollar sistemas capaces de reconocer, interpretar, e incluso expresar emociones y sentimientos humanos [1]. Esta área abarca un amplio abanico de aplicaciones, mejorando la experiencia del usuario en aplicaciones *human-centric* [2], para la detección de polaridad en los comentarios en la red (e.g. comentarios "positivos" o "negativos") [3], incluso para aplicaciones que permiten obtener la opción pública en eventos, movimientos políticos o campañas publicitarias [4].

Además, esta disciplina permite llevar a cabo un reconocimiento emocional a partir de las expresiones faciales a través de técnicas como *Deep Learning* y el mundo del *computer vision* (CV), hasta el procesamiento del lenguaje natural (NLP), donde es utilizado el *sentiment analysis* también conocido como minería de emociones y que se utiliza para determinar si los datos son positivos, negativos o neutros.

En este trabajo fin de grado se persigue el objetivo de aplicar una de las técnicas de mayor importancia en la actualidad de *Deep Learning*, las redes convolucionales al problema de reconocimiento de emociones en rostros humanos para obtener un modelo de representación robusto y fiable. En este sentido, se investigará la utilización de diferentes tipos de técnicas de aprendizaje profundo para el reconocimiento de diferentes tipos de emociones en imágenes de rostros. Para poder llevar a cabo este proyecto se han de cumplir los siguientes objetivos:

- Recopilar una base de datos de imágenes de expresiones faciales.

- A partir de esta base de datos, se han de obtener diferentes parámetros para la extracción de características para su posterior utilización como entrada en un sistema de reconocimiento de emociones.
- Implementar y evaluar diversos modelos de aprendizaje profundo, para el reconocimiento de emociones basado en Python, Sklearn, Keras, OpenCV, Dlib y Tensorflow.
- Analizar, implementar y evaluar diversos sistemas de aprendizaje profundo para el reconocimiento de expresiones con Keras y modelos convolucionales.

2 Marco Teórico

El ser humano al igual que muchas especies son capaces de reconocer el estado emocional a través de diversos estímulos como los sonidos, los gestos, la escritura, las expresiones faciales, entre otros. Es por ello, que existen diversos sistemas de reconocimiento de emociones:

- Sistemas de reconocimiento de emociones a través del **habla**: El uso de la voz puede llegar a ser una forma de expresar un estado emocional, y es que a través de la velocidad de habla, su intensidad, la calidad de la voz, entre otros, el ser humano es capaz de distinguir entre un estado emocional u otro. Estas variables son utilizadas en tres vertientes de investigación, en una de ellas se procesa la señal digital que ha sido grabada, para encontrar algunas características, en otra de ellas se entrenan algoritmos de *machine learning* sobre la señal de voz y por último la otra vertiente combina ambas técnicas. Este método es utilizado en aplicaciones como:

  Moodies: Se trata de un software que consiste en una aplicación que mide en 15 segundos lo que siente una persona con la que estamos hablando por teléfono. La voz es analizada a través de algoritmos que registran cambios en los intervalos de la voz. De esta forma, a través de un análisis del discurso, la elección de palabras y el volumen de voz detecta el estado de ánimo.

  Empath: Se trata de una compañía japonesa que ha desarrollado un programa de reconocimiento de emociones a través de las propiedades físicas en la voz. El algoritmo desarrollado a través de decenas de miles de muestras de voz de Smartmedical, es capaz de detectar la emoción de una persona con solo escuchar su voz, en tiempo real y en cualquier idioma. Algunas empresas que utilizan este sistema de reconocimiento de emociones son Fujitsu o Philips.

  Xpression: Se trata de un algoritmo que identifica cinco estados de ánimo, entre ellos felicidad, tristeza, actitud neutra, miedosa y enfado, ya que ampliar el espectro de emociones conllevaría reducir el porcentaje de acierto. Lo hace reconociendo algunos matices clave en el discurso del interlocutor y cruzándolos con una base de datos en la que busca referencias.

- Sistemas de reconocimiento de emociones en **texto plano**: a través de un procesado de texto plano reconoce el estado emocional según estructura de frases, vocabulario utilizado, relación entre palabras. En este sentido, este tipo de sistemas son capaces de procesar opiniones de consumidores de un producto para valorar la experiencia global. Este método es utilizado en aplicaciones como:

  Tone Analyzer: Se trata de una aplicación, diseñada para analizar el tono de los que escribimos y ayudarnos a corregirlo y matizarlo, de manera que se transmite el mensaje más eficientemente. Esta aplicación, analiza nuestro texto, palabra a palabra,

y lo categoriza, mostrado el número de palabras que transmiten concisión, amabilidad, franqueza, capacidad analítica o negatividad, entre otras.

Sallee: Se trata del motor de análisis de emociones de Receptiviti, se ha utilizado para medir la emoción en deportes electrónicos e incluso en debates presenciales. Su precisión proviene de la capacidad de comprender la estructura gramatical y las pistas contextuales, teniendo en cuenta intensificadores como muy o nunca, así como la capacidad de comprender las diferentes formas en las que una persona puede utilizar palabras malsonantes, comprende emojis y hashtags.

Bitext: Se trata de una empresa que procesa grandes cantidades de texto generado por consumidores, empresas o empleados, pudiendo determinar el sentimiento, el tono o intensidad de los textos. Analiza relaciones entre palabras, frases, etc y realiza una puntuación y clasificación en colores de las diversas opiniones.

- Sistema de reconocimiento de emociones a través de **rostros**: Se trata del tipo más extendido, ya que es uno de los indicadores que mejor definen el estado emocional de la persona. Algunas de las características son la dirección de la mirada, la posición de los pómulos, la apertura de la boca, la aparición de arrugas en la piel del rostro, etc.Este método es utilizado en aplicaciones como:

  Bismart Face and Emotion Recognition: Desarrollada por la compañía Bismart, se trata de una aplicación móvil que detecta el estado de ánimo mediante un escaneo de la cara de la cara del usuario y un posterior análisis a través de algoritmos de inteligencia artificial, por último crea un informe que muestra el análisis del estado de ánimo y como se puede mejorar.

  Emotion API: Desarrollada por la empresa Microsoft,esta API trabaja con imágenes estáticas y con vídeos realizando un barrido de las posibles caras, para más tarde calcular la emoción y clasificarla, asignando una puntuación a cada emoción.

  Emovu: Desarrollada por el grupo de Eyeris, se trata de una aplicación basada en redes convolucionales, incorpora un complejo sistema de filtrado de decisiones y etapas de verificación.

Existen diversos tipos de reconocimiento de emociones, todos se basan fundamentalmente en adquirir una serie de señales (imágenes,texto,sonido, etc), extrayendo a continuación unas características determinadas y posteriormente realizando una clasificación. En este trabajo únicamente se trabajará con sistemas que reconocen las emociones a través de imágenes.

Un sistema de reconocimiento de emociones debe estar dotado de una serie de características, entre ellas, la capacidad de funcionar correctamente a través de vídeos e imágenes, ser capaz de trabajar en tiempo real, además de ser robusto frente a variaciones lumínicas, con independencia de la etnia, género, etc de la persona. Este tipo de sistemas se subdividen en varios módulos: detección y seguimiento del rostro, extracción de características y clasificación de las expresiones.

Los sistemas sofisticados utilizan un software que reconoce el rostro y traza una serie de puntos característicos en zonas de relevancia de la cara tales como ojos, labios, cejas, mejillas, tal como se puede observar en la Figura 2.1 donde se puede ver un ejemplo de este software

desarrollado por [5] genera un volumen 3D del rostro y lo divide en 16 volúmenes Bezier, extrayendo las características que se esperan más relevantes para la tarea de reconocimiento de la emoción.

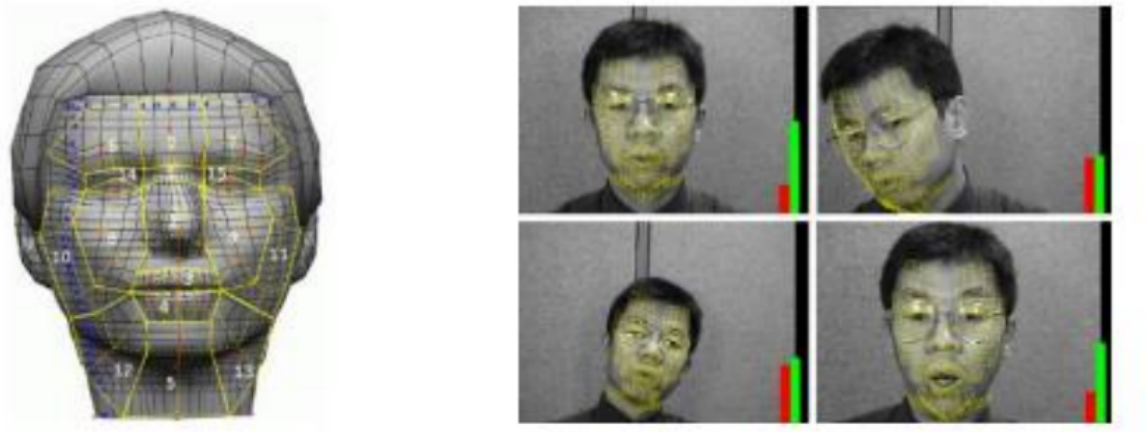


Figura 2.1: Los 16 volúmenes Bezier calculados en tiempo real.

Los movimientos de los diferentes volúmenes o regiones Bezier son cuantificados y asociados a movimientos de la cara. En este sentido, se han realizado numerosos estudios para catalogar expresiones en función de los movimientos del rostro de la cara como los realizados en [6], estudio realizado por Paul Ekman creador del Sistema de codificación facial (FACS, *Facial Action Coding System*). El sistema FACS utiliza *Action Units* que son movimientos de un músculo que produce un cambio en el rostro, por ejemplo la apertura de la mandíbula o levantar las cejas.

Los parámetros extraídos, se introducen en un clasificador que determina la emoción que representan, en este sentido, se ha analizado como estos diferentes clasificadores realizan la tarea. Cohen et al. en [7] hace un estudio de diversos clasificadores estáticos, es decir, aquellos clasificadores que toman las características a través de frames individuales como Naïve Bayes, Tree Augmented Naïve Bayes y Stochastic Structure Research y algunos clasificadores dinámicos, es decir, aquellos que tomas una secuencia de características a partir de diversos frames a lo largo del tiempo como los Modelos Ocultos de Markov. Los resultados muestran que los clasificadores Naïve Bayes dan mejores resultados a pesar de que estos admiten que no hay una dependencia entre diferentes características extraídas en los rostros, lo cual no es del todo cierto en problemas de la vida real ya que existe una similitud entre movimientos faciales y muestra de emociones).

También existen otros tipos de clasificadores como los *Support Vector Machines, SVM*, *Hidden Markov Model, HMM* y las redes neuronales. También es posible la combinación de distintos clasificadores y realizar un proceso de votación, es decir, la emoción más votada para un dato de entrada es la que dará como resultado el sistema.

Para este proyecto se utilizarán las conocidas como redes neuronales convolucionales. En este sentido, no será necesario la extracción de características relevantes del rostro ya que el modelo las aprenderá automáticamente.

3 Metodología

En este capítulo, se explicarán brevemente todas las herramientas que han sido utilizadas, estas son en gran parte librerías, datasets y herramientas que han permitido llevar a cabo este proyecto.

3.1 Setup Hardware

A continuación, se detallan las herramientas que se han utilizado:

- Ordenador portátil con sistema operativo Ubuntu 18.04.5 LTS.
- Servidor jackson.rovit.ua.es con versión de Ubuntu 18.04.5 LTS.
- Tarjeta gráfica Intel UHD Graphics 620.

3.2 Librerías

En esta sección se hará un breve resumen de aquellas librerías que han permitido llevar a cabo el proyecto, así como una breve descripción de para que han sido utilizadas dentro del mismo.

3.2.1 OpenCV

OpenCV (*Open Source Computer Vision*) [8] es una librería software *open-source* de visión artificial y machine learning, que propone una infraestructura destinada a aplicaciones de visión artificial. Esta librería incluye algoritmos de *machine learning* que permiten identificar objetos, caras, clasificar acciones humanas en vídeo, hacer un tracking de movimientos de objetos, extraer modelos 3D, etc. Se usa en aplicaciones como la detección de intrusos en vídeos o, monitorización de equipamientos, ayuda a navegación de robots, inspeccionar etiquetas en productos entre otros. En este proyecto, OpenCV será utilizado para redimensionar las imágenes de los diferentes datasets utilizados a un tamaño de 48x48 píxeles, así como para pasar las imágenes a escala de grises. También se ha utilizado para realizar un alineamiento y recorte de los rostros tras haber sido detectados mediante DLib, así como para la realización de la aplicación que a través de la cámara del ordenador permite detectar la emoción.

3.2.2 DLib

DLib [9] es un moderno kit de herramientas C++ que contiene algoritmos de *machine-learning* y herramientas para crear software complejo en C++. Es una librería utilizada en la industria y en el mundo académico, robótica, dispositivos integrados, teléfonos móviles y grandes entornos de computación de alto rendimiento. Las licencias de código abierto permite

que pueda ser utilizado de forma gratuita en cualquier aplicación. En este proyecto ha sido utilizado para detectar los rostros de las imágenes.

3.2.3 TensorFlow

TensorFlow [10] es una plataforma de código abierto de extremo a extremo para el aprendizaje automático. Cuenta con un ecosistema integral y flexible de herramientas, bibliotecas y recursos de la comunidad que les permite a los investigadores innovar con el aprendizaje automático y, a los desarrolladores, compilar e implementar con facilidad aplicaciones con tecnología de AA. TensorFlow ha sido utilizado en el proyecto para el entrenamiento y la ejecución de los distintos modelos de redes convolucionales para el reconocimiento de emociones.

3.2.4 Keras

Keras [11] es una biblioteca de redes neuronales artificiales de código abierto. Está desarrollada en Python y puede ejecutarse sobre diferentes plataformas como TensorFlow o Theano. Keras está diseñado para ir construyendo por bloques la arquitectura de cada red neuronal, incluyendo redes convolucionales y recurrentes, que son las que permiten, junto a los bloques "más tradicionales", entrenar modelos deep learning.

3.3 Datasets

Para comprender y detectar emociones, el primer y principal requisito de los modelos de aprendizaje automáticos es disponer de un conjunto de datos. A continuación, se hará una referencia a los conjuntos de datos principales para la detección de emociones.

3.3.1 AffectNet

AffectNet [12] es uno de los conjuntos de datos más grandes para el afecto facial en imágenes fijas que cubre modelos categóricos y dimensionales. El conjunto de datos se recopila mediante el uso de 1250 etiquetas relacionadas con las emociones en seis idiomas diferentes, que son inglés, alemán, español, portugués, árabe y farsi. El conjunto de datos contiene más de un millón de imágenes con rostros y puntos de referencia faciales extraídos.

3.3.2 Ascertain

Asertain [13] es una base de datos multimodal para el reconocimiento de la personalidad implícita y el afecto que se puede utilizar para detectar rasgos de personalidad y estados emocionales a través de respuestas fisiológicas. El conjunto de datos contiene escalas de personalidad de los cinco grandes y autoevaluaciones emocionales de 58 usuarios junto con electroencefalograma (EEG), electrocardiograma (ECG), respuesta galvánica de la piel (GSR) y datos de actividad facial registrados de forma sincrónica, registrados utilizando sensores estándar mientras ver clips de películas afectivos.

3.3.3 Dreamer

Dreamer [14] es una base de datos multimodal que consta de señales de electroencefalograma (EEG) y electrocardiograma (ECG) registradas durante la provocación del afecto mediante estímulos audiovisuales. En este conjunto de datos, las señales de 23 participantes se registraron junto con la autoevaluación de los participantes de su estado afectivo después de cada estímulo, en términos de valencia, excitación y dominancia.

3.3.4 Extended Cohn-Kanade Dataset (CK+)

El conjunto de datos extendido de Cohn-Kanade (CK +) [15] es un conjunto de datos de referencia público para unidades de acción y reconocimiento de emociones . El conjunto de datos comprende un total de 5876 imágenes etiquetadas de 123 individuos, donde las secuencias van desde la expresión neutra hasta la máxima. Todas las imágenes del conjunto de datos CK + se presentan con fondos similares, en su mayoría en escala de grises y 640 × 490 píxeles.

3.3.5 EMOTIC

El reconocimiento EMOTIC o EMOTIon in Context [16] es una base de datos de imágenes con personas en entornos reales, anotadas con sus emociones aparentes. El conjunto de datos EMOTIC combina dos tipos diferentes de representación de emociones , que incluye un conjunto de 26 categorías discretas y las dimensiones continuas valencia, excitación y dominio. El conjunto de datos contiene 23, 571 imágenes y 34, 320 personas anotadas. De hecho, algunas de las imágenes se recopilaron manualmente de Internet utilizando el motor de búsqueda de Google.

3.3.6 FER-2013

El conjunto de datos FER-2013 [17] consta de 28.000 imágenes etiquetadas en el conjunto de entrenamiento, 3500 imágenes etiquetadas en el conjunto de desarrollo y 3500 imágenes en el conjunto de prueba. El conjunto de datos se creó mediante la recopilación de los resultados de una búsqueda de imágenes de Google de cada emoción y sinónimos de las emociones. Cada imagen en FER-2013 está etiquetada como una de las siete emociones, como feliz, triste, enojado, asustado, sorpresa, disgusto y neutral, siendo feliz la emoción más prevalente, lo que proporciona una línea de base para adivinar al azar del 24,4%.

3.3.7 Google Facial Expression Comparison Dataset

El conjunto de datos de comparación de expresiones faciales de Google [18] es un conjunto de datos de expresiones faciales a gran escala que consta de tripletes de imágenes faciales junto con anotaciones humanas. El conjunto de datos ayuda a especificar qué dos caras de cada triplete forman el par más similar en términos de expresión facial. El conjunto de datos está destinado a ayudar en temas relacionados con el análisis de expresiones faciales, como la recuperación de imágenes basadas en expresiones, el resumen de álbumes de fotos basados en expresiones, la clasificación de emociones, la síntesis de expresiones, etc.

3.3.8 K-EmoCon

K-EmoCon [19] es un conjunto de datos multimodal adquirido de 32 sujetos que participan en 16 debates emparejados sobre un tema social. El conjunto de datos consta de datos de sensores fisiológicos recopilados con tres dispositivos portátiles listos para usar, imágenes audiovisuales de los participantes durante el debate y anotaciones continuas de emociones.

A partir de los datasets descritos anteriormente, se realizó un estudio de aquellos más viables para el proyecto. Se pretende utilizar datasets que contengan únicamente imágenes que representen las 7 emociones principales, estas son: felicidad, tristeza, asco, neutro, enfado, sorpresa y miedo. Por este motivo, se seleccionaron los dataset de FER_2013 [17] y el dataset AffectNet [12] explicados anteriormente en los apartados 3.3.6 y 3.3.1 respectivamente. Cabe mencionar que el datset Affectnet incluye también imágenes etiquetadas con la clase desprecio, desconocido, sin rostro y ninguna. Estas cuatro clases han sido despreciadas a la hora de descargar las imágenes.

3.4 Redes neuronales

Las redes neuronales son una de las técnicas más populares de *Deep Learning*. Al igual que otras técnicas de IA se basa en copiar el modelo biológico humano, en este caso, del funcionamiento del cerebro, formado por la interconexión de neuronas. La idea principal se apoya en que la interconexión una gran cantidad de elementos simples pueden formar un modelo complejo y, por tanto, con operaciones entre elementos simples se pueden resolver problemas complejos.

En esta sección se tratará la base teórica de las redes neuronales y las redes convolucionales para el paradigma del aprendizaje supervisado.

3.4.1 Redes Neuronales Artificiales

Una Red Neuronal Artificial es un modelo matemático formado por una serie de operaciones que, para un vector de entrada $\mathbf{x}$ ofrece un vector de salida distinta $\mathbf{o(x)}$. En concreto, la tipología de RNA conocida como Perceptrón Multicapa, está constituida por 3 tipos de capas completamente conectadas, como se puede observar en la Figura 3.1:

- **Capa de entrada**: Se puede entender como el vector de datos de entrada antes de realizarse ninguna operación. Si se tiene un vector x de datos, cada valor desde (x1...xn) constituye una neurona de entrada. En el caso de una imagen se tiene tantas entradas como píxeles tiene la imagen.
- **Capa Oculta**: Está formado por las neuronas ocultas y contiene todos los cálculos intermedios de la red. Cada neurona oculta contiene un peso y bias que son los elementos que modificarán los datos de entrada a «algo distinto». El número y tamaño de capas ocultas es variable y no determinado.
- **Capa de salida**: Es la capa en la que se realiza la clasificación y tiene tantas neuronas como posibles clases que presenta el problema asociado. Al igual que la capa anterior

cada neurona tiene pesos y bias.

Una vez se conoce la estructura y la división en capas, es importante conocer como funcionan internamente (se deja lado la capa de entrada, la cuál se utiliza exclusivamente como ejemplo para la introducción del vector de datos).

La transformación que se aplica al vector de datos, en cada capa oculta, sigue la siguiente fórmula:

$$h(x) = s(Wx + b) \tag{3.1}$$

donde b es el vector de bias, W es la matriz de pesos y s la función de activación.

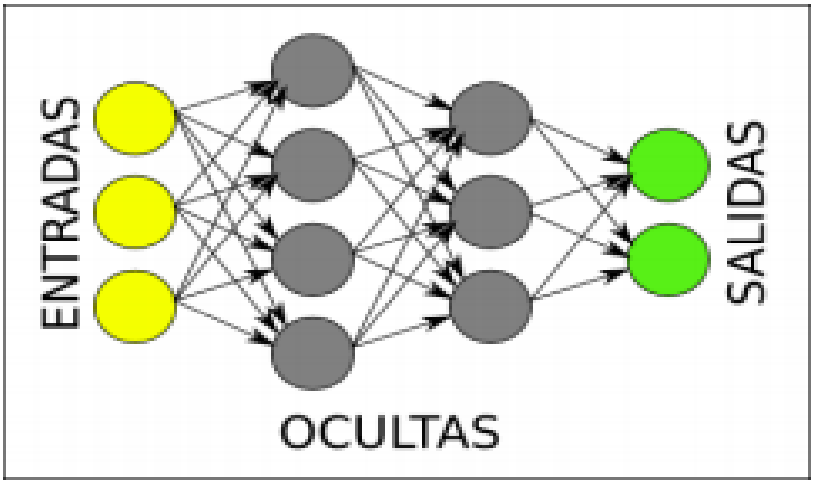


Figura 3.1: Topología Multi-Layer Perceptrón.

3.4.2 Funciones de activación

Una función de activación. se utiliza con el propósito de dar una no linealidad al problema y por tanto que la red sea capaz de resolver problemas más complejos. Existen varios tipos de funciones de activación, entre ellas:

$$tanh(a) = \frac{exp^{a} - exp^{-a}}{exp^{a} + exp^{-a}} \tag{3.2}$$

$$sigmoid(a) = \frac{1}{1 + exp^{-a}} \tag{3.3}$$

$$ReLU(a) = max(0, a) \tag{3.4}$$

La función de activación, debe ser elegida de manera razonada al igual que otros parámetros de la red neuronal ya que cada uno dispone de unas ventajas y unas desventajas. El caso

de 3.4.2 y 3.4.2 es que son más costosas computacionalmente hablando que la función ReLU, ya que las dos primeras se calculan exponencialmente mientras que la otra hace una operación máximo. Otro aspecto a tener en cuenta es que se han de inicializar los pesos de forma distinta según la función de activación escogida ya que se ha de lograr romper la simetría entre las diferentes neuronas y evitar una posible sobresaturación provocada por los valores o una activación excesivamente lineal donde se tardaría mucho en llegar a un resultado bueno.

Glorot y Bengio [20], proponen una serie de métodos para la inicialización de pesos basados en la longitud del vector de entrada (n_{in}) y el número de neuronas en una capa oculta (n_{out}). Los métodos para cada una de las funciones de activación serían:

- Función **tangente hiperbólica** 3.4.2: Se inicializan los pesos de forma aleatoria siguiendo una distribución uniforme en el intervalo (-r,r), donde r:

$$r = \sqrt{\frac{1}{n_{in} + n_{out}}} \tag{3.5}$$

- Función **sigmoide** 3.4.2: Se inicializan los pesos de forma aleatoria siguiendo una distribución uniforme entre el intervalo (-r,r), donde r:

$$r = 4\sqrt{\frac{6}{n_{in} + n_{out}}} \tag{3.6}$$

- Función **ReLU** 3.4.2: No se establece una inicialización fija, ya que lo realiza de forma automática. No obstante, se considera el mismo tipo que para la tangente hiperbólica.

Los parámetros «bias» se puede inicializar a 0 en todos los casos.

Además de este tipo de funciones, en la capa de salida (también llamada de regresión logística) existe una función conocida como *softmax*, que básicamente se encarga de normalizar la salida lineal, a valores entre [0, 1] (cuya suma es 1), el resultado de la cuál es la probabilidad a posteriori de clase. Este resultado se utiliza para calcular la función de coste y, si se realiza la operación a*argmax*, se predice la etiqueta o clase a la que debería formar parte el conjunto de datos de entrada.

3.4.3 Redes neuronales convolucionales

Conocido el funcionamiento básico de las redes neuronales, se explicará de forma breve que son las redes convolucionales y como funcionan. En general una red convolucional no es más que una modificación de una red neuronal básica donde la principal diferencia radica en que los pesos, W, son filtros (de un tamaño deseado, 3x3 por ejemplo) que se convolucionarán directamente con la imagen como se puede observar en la Figura 3.2.

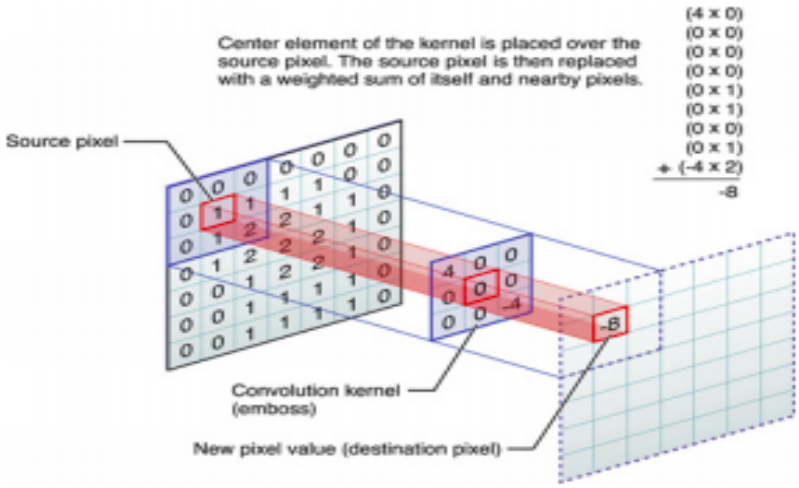


Figura 3.2: Topología Multi-Layer Perceptrón.

3.4.3.1 Capas

Una red neuronal convolucional está formada por las siguientes capas:

- **Capa convolucional**: En extrapolación a un perceptrón multicapa, se puede decir que una capa convolucional es una capa oculta, donde se definen la neuronas con los filtros (W) (para convolucionarlos con los datos, generalmente imágenes), bias (b) y la función de activación escogida.

- **Capa de *pooling* o sub-muestreo**: Esta capa opcional se conecta entre la salida lineal de la capa convolucional y la función de activación. Se utiliza para reducir la dimensionalidad de la imagen, lo que reduce el coste computacional de las siguientes convoluciones, número de parámetros e incluso controlar el sobre-aprendizaje de la red. Se puede realizar *pooling* de varios tipos, siendo lo más común *max-pooling* y *average-pooling* como se puede observar en la Figura 3.3. En el primer caso se elige el valor máximo entre un conjunto de píxeles y en el segundo caso se calcula la media. La dimensionalidad es variable, aunque lo más común es 2x2, 3x3 o 4x4.

- **Capa fully-connected**: Está compuesta por una red neuronal completamente conectada (de ahí el nombre), por ejemplo un perceptrón multicapa, que se conecta a la salida de la última capa convolucional de una red. La capa de conexión se conoce como capa de *reshape*, ya que convierte la matriz de datos en un vector plano.

Al igual que en las redes neuronales anteriormente explicadas, no existe un número fijo de capas que se deben utilizar, esto es una elección que se considera de acuerdo al problema a resolver. Se debe tener en cuenta que una red excesivamente profunda y con pocos datos puede no llegar a aprender nada y de la misma forma cuando se tiene gran cantidad de datos suele venir bien ir más profundo (mayor número de filtros, capas, etc).

Para este proyecto, se han realizado diversas pruebas a través de distintos modelos de redes neuronales convolucionales que se explicarán a continuación más detalladamente.

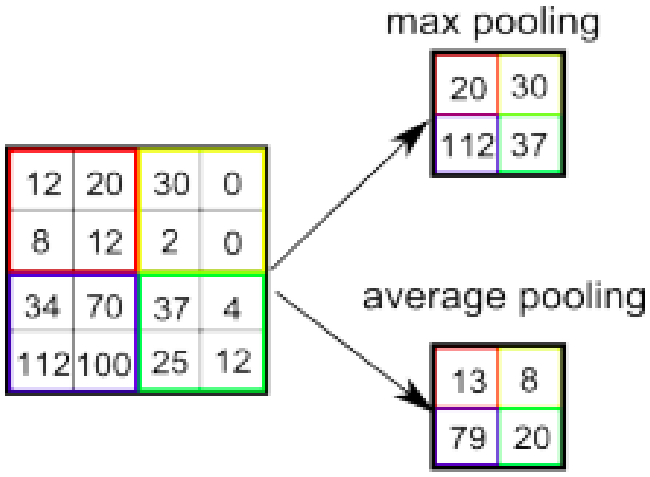


Figura 3.3: Ejemplo *max-pooling* y *average-pooling*

.

3.4.4 Redes convolucionales utilizadas

En este apartado se expondrán las distintas redes neuronales convolucionales utilizadas, entre ellas se han recogido 2 formadas por 4 capas y 8 capas convolucionales. Así como lo modelos de aprendizaje profundo que están disponibles en las aplicaciones de Keras [11], entre ellas el modelo NASNetmobiles y el ResNet50.

3.4.4.1 Red convolucional de 4 capas

El modelo que se puede visualizar en la Figura 3.4 fue extraído de [21], está formado por las siguientes capas:

- Primera capa de convolución a la cual se le aplica un kernel de 32 filtros de 3x3 con función de activación ReLU y que recibe imágenes de tamaño 48x48 en escala de grises.
- Segunda capa de convolución a la cual se le aplica un kernel de 64 filtros de 3x3, con función de activación ReLU.
- A continuación viene un proceso de subsampling para reducir el tamaño de la próxima capa de neuronas, se realizará subsampling con max-pooling de tamaño 2x2.
- Se aplica la técnica de dropout y se eliminan el 25% de los nodos de la red.
- Tercera capa de convolución a la cual se le aplica un kernel de 128 filtros de 3x3, con función de activación ReLU.
- A continuación viene un proceso de subsampling para reducir el tamaño de la próxima capa de neuronas, se realizará subsampling con max-pooling de tamaño 2x2.
- Cuarta capa de convolución a la cual se le aplica un kernel de 128 filtros de 3x3, con función de activación ReLU.

- Seguidamente se realiza *subsampling* para reducir el tamaño de la próxima capa de neuronas, se realizará *subsampling* con *max-pooling* de tamaño 2x2.
- Se aplica la técnica de dropout y se eliminan el 25% de los nodos de la red.
- A continuación se aplica Flatten con el objetivo de cambiar la forma del tensor para que obtenga una disposición igual al número de elementos contenidos en el tensor.
- Se aplica Dense que es la capa regular de la red neuronal y realiza la operación que devuelve una salida de 1024 neuronas con función de activación ReLU.
- Se aplica la técnica de dropout y se eliminan el 50% de los nodos de la red.
- Se aplica Dense que es la capa regular de la red neuronal y realiza la operación que devuelve una salida de 7 neuronas con función de activación softmax, es decir, devuelve el número de clases.

```
model = Sequential()

model.add(Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=(48,48,1)))
model.add(Conv2D(64, kernel_size=(3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Conv2D(128, kernel_size=(3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(128, kernel_size=(3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Flatten())
model.add(Dense(1024, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(7, activation='softmax'))
```

Figura 3.4: Modelo de red convolucional 4 capas

.

3.4.4.2 Red convolucional de 8 capas

El modelo que se puede visualizar en la Figura 3.5 fue extraído de [22], está formado por las siguientes capas:

- Primera capa de convolución a la cual se le aplica un kernel de 64 filtros de 3x3 con función de activación ReLU y que recibe imágenes de tamaño 48x48 en escala de grises.
- Segunda capa de convolución a la cual se le aplica un kernel de 64 filtros de 3x3, con función de activación ReLU.
- Se aplica Batch Normalization con el objetivo de reducir la covarianza interna de los datos reduciendo la frecuente variabilidad que hay entre los datos de un mismo conjunto.

- A continuación viene un proceso de subsampling para reducir el tamaño de la próxima capa de neuronas, se realizará subsampling con max-pooling de tamaño 2x2.
- Se aplica la técnica de dropout y se eliminan el 50% de los nodos de la red.
- Tercera capa de convolución a la cual se le aplica un kernel de 128 filtros de 3x3 con función de activación ReLU.
- Se aplica Batch Normalization con el objetivo de reducir la covarianza interna de los datos reduciendo la frecuente variabilidad que hay entre los datos de un mismo conjunto.
- Cuarta capa de convolución a la cual se le aplica un kernel de 128 filtros de 3x3 con función de activación ReLU.
- Se aplica Batch Normalization con el objetivo de reducir la covarianza interna de los datos reduciendo la frecuente variabilidad que hay entre los datos de un mismo conjunto.
- A continuación viene un proceso de subsampling para reducir el tamaño de la próxima capa de neuronas, se realizará subsampling con max-pooling de tamaño 2x2.
- Se aplica la técnica de dropout y se eliminan el 50% de los nodos de la red.
- Quinta capa de convolución a la cual se le aplica un kernel de 256 filtros de 3x3 con función de activación ReLU.
- Se aplica Batch Normalization con el objetivo de reducir la covarianza interna de los datos reduciendo la frecuente variabilidad que hay entre los datos de un mismo conjunto.
- Sexta capa de convolución a la cual se le aplica un kernel de 256 filtros de 3x3 con función de activación ReLU.
- Se aplica Batch Normalization con el objetivo de reducir la covarianza interna de los datos reduciendo la frecuente variabilidad que hay entre los datos de un mismo conjunto.
- A continuación viene un proceso de subsampling para reducir el tamaño de la próxima capa de neuronas, se realizará subsampling con max-pooling de tamaño 2x2.
- Se aplica la técnica de dropout y se eliminan el 50% de los nodos de la red.
- Séptima capa de convolución a la cual se le aplica un kernel de 512 filtros de 3x3 con función de activación ReLU.
- Se aplica Batch Normalization con el objetivo de reducir la covarianza interna de los datos reduciendo la frecuente variabilidad que hay entre los datos de un mismo conjunto.
- Octava capa de convolución a la cual se le aplica un kernel de 512 filtros de 3x3 con función de activación ReLU.
- Se aplica Batch Normalization con el objetivo de reducir la covarianza interna de los datos reduciendo la frecuente variabilidad que hay entre los datos de un mismo conjunto.

- A continuación viene un proceso de subsampling para reducir el tamaño de la próxima capa de neuronas, se realizará subsampling con max-pooling de tamaño 2x2.
- Se aplica la técnica de dropout y se eliminan el 50% de los nodos de la red.
- A continuación se aplica Flatten con el objetivo de cambiar la forma del tensor para que obtenga una disposición igual al número de elementos contenidos en el tensor.
- Se aplica Dense que es la capa regular de la red neuronal y realiza la operación que devuelve una salida de 512 neuronas con función de activación ReLU.
- Se aplica la técnica de dropout y se eliminan el 40% de los nodos de la red.
- Se aplica Dense que es la capa regular de la red neuronal y realiza la operación que devuelve una salida de 256 neuronas con función de activación ReLU.
- Se aplica la técnica de dropout y se eliminan el 40% de los nodos de la red.
- Se aplica Dense que es la capa regular de la red neuronal y realiza la operación que devuelve una salida de 132 neuronas con función de activación ReLU.
- Se aplica la técnica de dropout y se eliminan el 50% de los nodos de la red.
- Se aplica Dense que es la capa regular de la red neuronal y realiza la operación que devuelve una salida de 7 neuronas con función de activación softmax, es decir, devuelve el número de clases.

3.4.4.3 ResNet50

ResNet es una CNN presentada por Microsoft en 2015 que está basada en bloques residuales. Esta red convolucional basada en bloques residuales se encuentra disponible en diversos tamaños a través de la página Keras Applications, la mayor de ellas de 152 capas consiguió tener una menor complejidad que VGGNet. Con esta nueva arquitectura se batieron récords en clasificación, localización y detección de objetos, ganando el primer lugar en la tarea de clasificación ILSVRC-2015, también gano el primer lugar en tareas de detección de Imagenet, localización de Imagenet, detección de COCO y segmentación de COCO. En la Figura 3.6 se puede observar su arquitectura general.

La idea de ResNet reside en cuestionarse porqué apilar capas y capas, se sabe que la red no mejora a partir de cierta profundidad. Apareciendo en este sentido problemas como el desvanecimiento de gradiente (*vanishing gradient*) y la maldición de la dimensionalidad (*curse of dimensionality*) dejando la red de aprender. También con la profundidad de las capas de precisión llega un momento en el que la red se estanca y comienza a degradarse, se deduce pues que las redes convolucionales menos profundas aprenden en ocasiones mejor que las más profundas. Aquí es donde nace la idea de los bloques residuales, que permiten saltar unas capas como se puede observar en la Figura 3.7 donde se aumenta el número de capas introduciendo una conexión residual (con una capa identidad), mejorando el proceso de aprendizaje.

```
model = Sequential()

model.add(Conv2D(num_features, kernel_size=(3, 3), activation='relu', input_shape=(width, height, 1), data_format='channels_last', kernel_regularizer=l2(0.01)))
model.add(Conv2D(num_features, kernel_size=(3, 3), activation='relu', padding='same'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
model.add(Dropout(0.5))

model.add(Conv2D(2*num_features, kernel_size=(3, 3), activation='relu', padding='same'))
model.add(BatchNormalization())
model.add(Conv2D(2*num_features, kernel_size=(3, 3), activation='relu', padding='same'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
model.add(Dropout(0.5))

model.add(Conv2D(2*2*num_features, kernel_size=(3, 3), activation='relu', padding='same'))
model.add(BatchNormalization())
model.add(Conv2D(2*2*num_features, kernel_size=(3, 3), activation='relu', padding='same'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
model.add(Dropout(0.5))

model.add(Conv2D(2*2*2*num_features, kernel_size=(3, 3), activation='relu', padding='same'))
model.add(BatchNormalization())
model.add(Conv2D(2*2*2*num_features, kernel_size=(3, 3), activation='relu', padding='same'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
model.add(Dropout(0.5))

model.add(Flatten())

model.add(Dense(2*2*2*num_features, activation='relu'))
model.add(Dropout(0.4))
model.add(Dense(2*2*num_features, activation='relu'))
model.add(Dropout(0.4))
model.add(Dense(2*num_features, activation='relu'))
model.add(Dropout(0.5))

model.add(Dense(num_labels, activation='softmax'))
```

Figura 3.5: Modelo de red convolucional 8 capas

.

El input **x** viaja en la red a través de la capas convolucionales con función de activación ReLU que dan como resultado F(x), se llama a H(x)=F(x)+x. En una red convolucional se pretende que F(x) y H(x) tengan el mismo valor. En vez de intentar conseguir el valor de F(x) directamente de x, los bloques residuales calculan el valor añadido x de F(x). El bloque residual añade una pequeña alteración a la entrada x causando una representación alterada, mientras que en las CNN tradicionales al pasar de x a F(x) no se produce ninguna relación. Se obtienen las ventajas de poder optimizar el mapa residual, además de que durante la vuelta atrás con el algoritmo *backpropagation* [23] el gradiente no se desvanece y se mantienen gracias a las operaciones de suma que se van realizando a través de la red, las cuales balancean el gradiente.

Este tipo de arquitectura sobre bloques residuales han sido estudiados en detalle, ya que evitan la profundidad de la red, la hacen más ancha y a su vez la convierten en una red manejable en comparación a aquellas que no presentan bloques residuales.

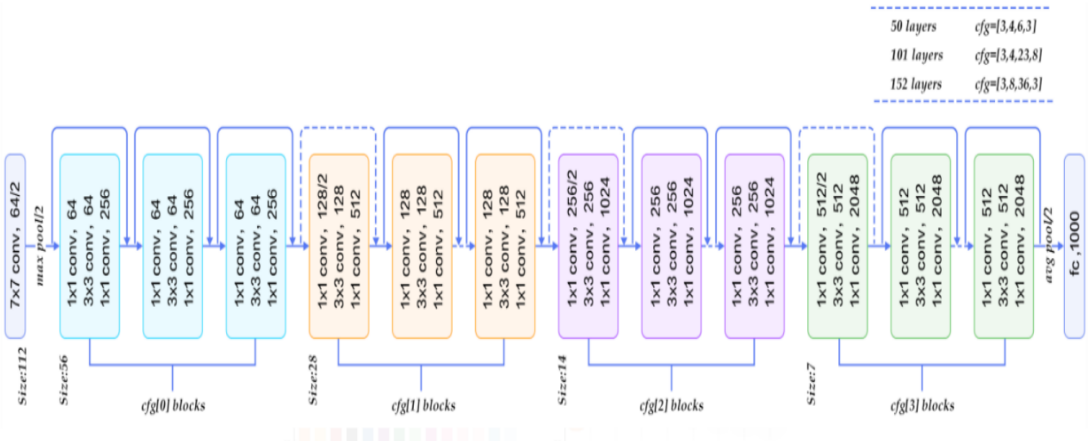


Figura 3.6: Arquitectura ResNet

.

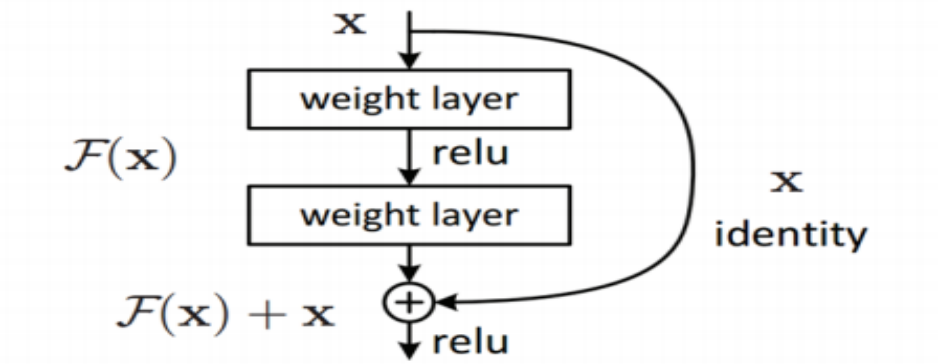


Figura 3.7: Esquema de bloque residual

.

3.4.4.4 NASNetMobile

La investigación de NASNet tuvo como objetivo la búsqueda de una arquitectura CNN óptima mediante el aprendizaje por refuerzo. NAS son las siglas de *Neural Architecture Search* y es una técnica desarrollada en Google Brain para buscar a través de un espacio de configuraciones de redes neuronales. NAS se utilizó con conjuntos de datos estándar como CIFAR10 e ImageNet para optimizar las CNN para diferentes tamaños. La versión reducida se llama NASNetMobile. A continuación, en la Figura 3.8verá la mejor celda convolucional reducida derivada con NAS y CIFAR10.

Arquitectura de las mejores celdas convolucionales (NASNet-A) con B = 5 bloques identificados con CIFAR-10. La entrada (blanco) es el estado oculto de activaciones anteriores (o imagen de entrada). La salida (rosa) es el resultado de una operación de concatenación en todas las ramas resultantes. Cada celda convolucional es el resultado de bloques B. Un solo bloque corresponde a dos operaciones primitivas (amarillo) y una operación de combinación

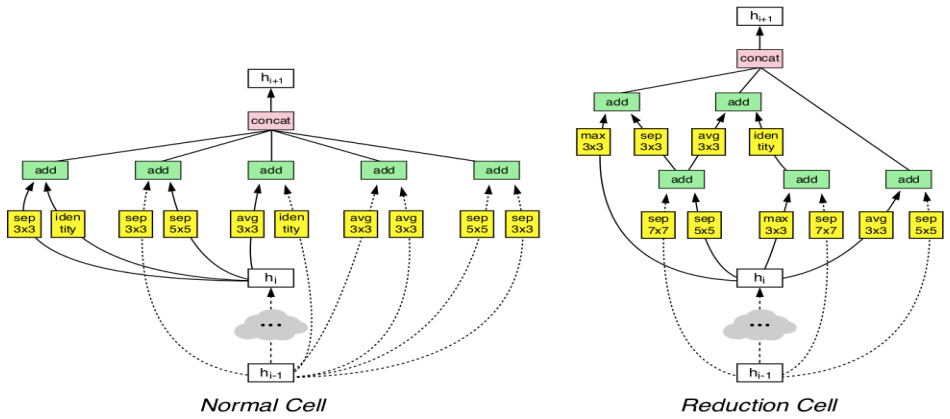


Figura 3.8: Capas de la arquitectura NASNetMobile.

.

(verde). Tenga en cuenta que los colores corresponden a las operaciones.

3.5 Métricas para la evaluación

En esta sección se explicarán los procedimientos utilizados para determinar la calidad del funcionamiento de nuestras redes neuronales ya entrenadas. Se han utilizado varias métricas para analizar la calidad, se ha empleado la matriz de confusión, así como las métricas de precisión.

3.5.1 Matrices de confusión

En el campo de la inteligencia artificial y el aprendizaje automático una matriz de confusión es una herramienta que permite visualizar el desempeño de un algoritmo de aprendizaje supervisado. Cada columna de la matriz representa el número de predicciones de cada clase, mientras que cada fila representa a las instancias en la clase real,en términos prácticos nos permite ver qué tipos de aciertos y errores está teniendo nuestro modelo a la hora de pasar por el proceso de aprendizaje con los datos. Se puede ver un ejemplo generalizado de matriz de confusión en la Figura 3.9.

3.5.2 Exactitud (*Accuracy*)

La Exactitud se refiere a lo cerca que está el resultado de una medición del valor verdadero. En términos estadísticos, la exactitud está relacionada con el sesgo de una estimación. También se conoce como Verdadero Positivo (*True Positive Rate*). Se representa como la proporción de resultados verdaderos tanto verdaderos positivos (VP) como verdaderos negativos (VN) dividido entre el número total de casos examinados (verdaderos positivos, falsos positivos, verdaderos negativos, falsos negativos).

$$Accuracy = \frac{VP + VN}{Total} \tag{3.7}$$

		Predicción	
		Positivos	**Negativos**
Observación	**Positivos**	Verdaderos Positivos (VP)	Falsos Negativos (FN)
	Negativos	Falsos Positivos (FP)	Verdaderos Negativos (VN)

Figura 3.9: Matriz de confusión.

.

4 Desarrollo.

En este capítulo se presentará el desarrollo del pipeline que se ha seguido en este proyecto y se explicarán las diferentes partes que lo constituyen. En el proyecto hay dos partes claramente diferenciadas la primera de ellas sería obtener un conjunto de datos robusto para posteriormente entrenar una serie de modelos de redes neuronales convolucionales explicadas en los Apartados 3.4.4.1 y 3.4.4.2, así como dos modelos obtenidos de la librería de Keras que serían ResNet50 y NASNetMobile y que han sido explicados en el Apartado 3.4.4.3 y 3.4.4.4 respectivamente.

4.1 Dataset Fer2013.

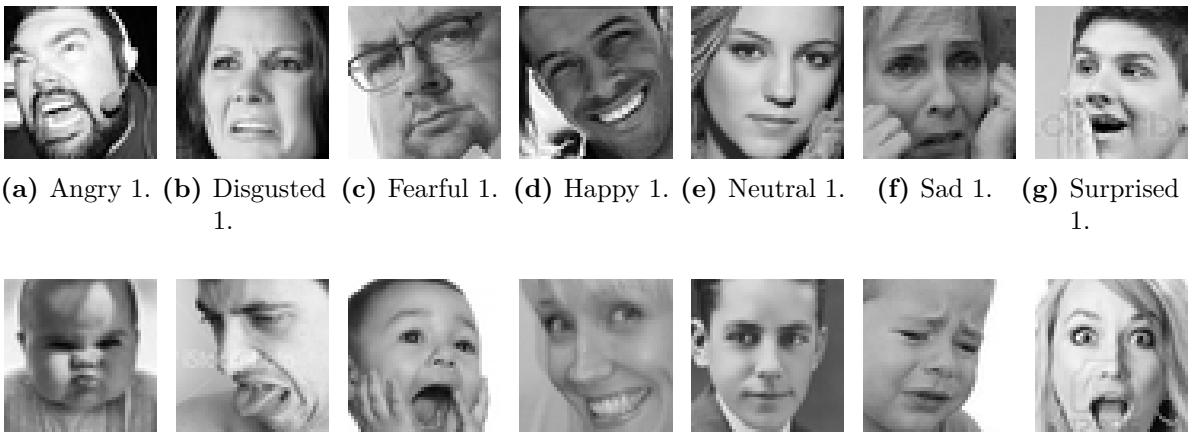
En primer lugar, a partir del archivo fer2013.csv que contiene los píxeles y las emociones con valores del 0-7, se han clasificado las distintas emociones en siete directorios distintos, estos directorios hacen referencia a las clases que son las distintas emociones que se van a intentar reconocer posteriormente, la cantidad de imágenes que se han utilizado se puede visualizar en la Tabla 4.1. Una vez, se obtienen las imágenes se han de procesar para que puedan servir de entrada a nuestros modelos de redes neuronales. Para ello en primer lugar se transforman todas a tamaño 48x48 píxeles y se pasan a escala de grises para trabajar únicamente con un canal y no con los tres canales que proporciona una imagen RGB. Esto produciría un peor entrenamiento de las redes neuronales, así como una mayor tiempo de ejecución. Seguidamente, se han normalizado los valores de los píxeles, se ha realizado una transformación de cada píxel: "valor/255" y nos quedará siempre un valor entre 0 y 1. A continuación, en la Figura 4.1 se pueden visualizar algunas de las imágenes del conjunto de datos, en concreto dos imágenes por cada clase del conjunto.

A continuación, una vez se ha hecho la transformación anteriormente explicada, se han guardado los píxeles en un array que posteriormente se almacena en un archivo. También cabe mencionar que al mismo tiempo se ha guardado en otro archivo la emoción de cada una de las imágenes, de tal manera que el fichero que contiene los píxeles y el que contiene la emoción con un valor del 0-7 quedan relacionados.

Seguidamente, se aplica *train_test_split* que divide el array de entrada en unos subconjuntos de prueba y otros de entrenamiento. Una vez se obtienen los subconjuntos de prueba y entrenamiento se introducen en cada una de las redes neuronales para más tarde plotear tanto la gráfica resultante del entrenamiento como las matrices de confusión tanto normalizadas como sin normalizar. Los resultados de estos entrenamientos se pueden visualizar en el Apartado 5.1.

DATASET FER2013	
Clases	Nº total imágenes
Angry	4953
Disgusted	983
Fearful	5121
Happy	8989
Neutral	6198
Sad	6077
Surprised	4004

Tabla 4.1: Imágenes dataset FER2013.



(a) Angry 1. **(b)** Disgusted 1. **(c)** Fearful 1. **(d)** Happy 1. **(e)** Neutral 1. **(f)** Sad 1. **(g)** Surprised 1.

(h) Angry 2. **(i)** Disgusted 2. **(j)** Fearful 2. **(k)** Happy 2. **(l)** Neutral 2. **(m)** Sad 2. **(n)** Surprised 2.

Figura 4.1: Imagenes Dataset Fer2013.

4.2 Dataset Fer2013 Alineado y Recortado.

En segundo lugar, a partir de las imágenes obtenidas anteriormente del dataset Fer2013, se ha realizado un alineamiento y posterior recorte de las imágenes. Para ello, se han realizado los siguientes pasos:

1. Recorremos las imágenes de los directorios que contienen las distintas emociones. Redimensionamos a tamaño 48x48 píxeles y convertimos a escala de grises.

2. A continuación, utilizamos el detector facial de dlib (basado en HOG) anteriormente inicializado, detectando los rostros presentes en la imagen. En este caso, se trata de un rostro por imagen.

3. Seguidamente, si no se detecta rostro se almacena la imagen sin alinear ni recortar

debido a que se perderían muestras para posteriores entrenamientos, en caso de detectar el rostro se realiza un alineamiento del rostro a través de la imagen en escala de grises y el rectángulo del cuadrado delimitador producido por el detector facial HOG de dlib:

a) Aplicamos el predictor de puntos de referencia facial de dlib y convertimos los puntos de referencia en coordenadas (x, y) en formato NumPy.

b) Extrae las coordenadas del ojo izquierdo y derecho (x, y) a través del diccionario *FACIAL_LANDMARKS_68_IDXS*.

c) A continuación, se ha calculado el centro de cada ojo, así como el ángulo entre los centroides del ojo. El ángulo sirve como componente clave para alinear nuestra imagen. Para ello, se calcula el centroide de cada ojo promediando todos los puntos (x,y) de cada ojo, para más tarde una vez conocidos los centros de los ojos se calculan las diferencias de las coordenadas (x,y) y se toma el arco-tangente para obtener el ángulo de rotación entre los ojos.

d) Seguidamente, se ha calculado el ojo derecho deseado en función de la coordenada x del ojo izquierdo. El valor debe ser equidistante del borde derecho de la imagen, ya que la coordenada x del ojo izquierdo correspondiente es desde su borde izquierdo. Este paso tiene el objetivo de determinar la escala de la cara tomando la relación entre la distancia entre los ojo en la imagen actual y la distancia entre los ojos de en la imagen deseada.

e) A continuación, se ha de encontrar el punto medio entre los ojos, así como calcular la matriz de rotación y actualizar su componente de traslación. Para ello primero se ha calculado el punto medio que se encuentra en la parte superior de la nariz y es el punto en el que se ha girado el rostro, más tarde se ha utilizado *cv2.getRotationMatrix2D* que recibe los parámetros:

   - **eyesCenter**: Punto medio entre los ojos y a partir del cual se ha de girar el rostro.
   - **ángulo**: El ángulo en el que se ha de girar el rostro para que los ojos queden en la misma línea horizontal.
   - **escala**: Porcentaje que se ha de escalar arriba o hacia abajo en la imagen, asegurando que la imagen se escala al tamaño deseado.

f) Seguidamente, se ha actualizado la componente translación de la matriz para que la cara esté en la imagen después de la transformación afín.

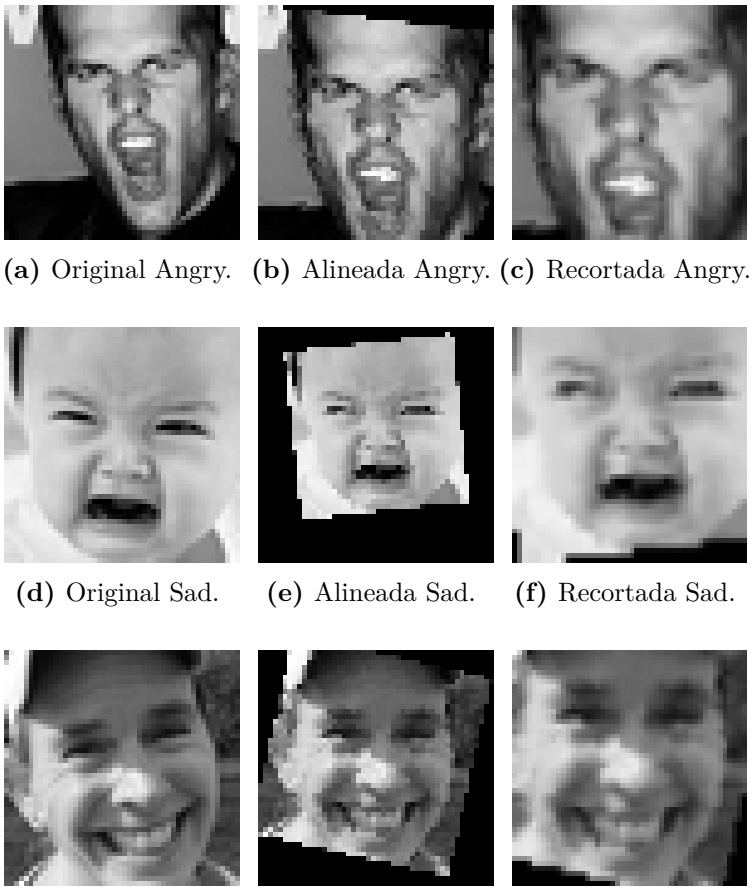
g) A continuación, se hace una llamada a la función *cv2.warpAffine* que requiere de tres parámetros:

   - **imagen**: La imagen del rostro.
   - **M**: Matriz de traslación, rotación y escala.
   - **(w,h)**: Ancho y alto deseados de la cara de salida.
   - **Flags**: El algoritmo que se ha de utilizar para la interpolación, se ha utilizado *INTER_CUBIC*.

h) Por último, se devuelve la cara alineada.

4. Una vez se tiene la cara alineada, nos encontramos con que algunas de las imágenes alineadas tienes zonas en negro, esto supondría un problema a la hora de realizar los entrenamientos. Es por ello que se vuelve a detectar el rostro de las imágenes alineadas y se realiza un recorte. Finalmente se almacenan las imágenes.

Los resultados de este alineamiento y recorte de los rostros se pueden observar en la siguiente Figura 4.2 donde se puede apreciar tanto la imagen original, la alineada y el posterior recorte que se realiza al rostro para diferentes imágenes de distintas clases del dataset Fer2013.



(a) Original Angry. **(b)** Alineada Angry. **(c)** Recortada Angry.

(d) Original Sad. **(e)** Alineada Sad. **(f)** Recortada Sad.

(g) Original Happy. **(h)** Alineada Happy. **(i)** Recortada Happy.

Figura 4.2: Imagenes Dataset Fer2013 Alineadas y Recortadas.

También cabe mencionar que hay imágenes que el detector facial de dlib (basado en HOG) no detecta, estas imágenes se han guardado en el nuevo dataset sin ningún retoque. Algunas de las imágenes en las que no se ha detectado rostro se pueden visualizar en la Figura 4.3.

La cantidad de rostros detectados y que no han sido reconocidos de cada clase se pueden observar en la Tabla 4.2.

(a) Happy. **(b)** Fearful. **(c)** Neutral. **(d)** Surprised.

Figura 4.3: Imagenes Dataset Fer2013 no detectadas.

DATASET FER2013 ALINEADO Y RECORTADO			
Clases	Nº imágenes detectadas	Nº imágenes no detectadas	Nº total imágenes
Angry	3454	1499	4953
Disgusted	765	218	983
Fearful	4751	370	5121
Happy	7017	1918	8989
Neutral	4677	1521	6198
Sad	3407	2670	6077
Surprised	2999	1005	4004

Tabla 4.2: Imágenes detectadas y no detectadas FER2013.

A continuación, una vez se ha hecho la transformación anteriormente explicada, se han guardado los píxeles en un array que posteriormente se almacena en un archivo. También cabe mencionar que al mismo tiempo se ha guardado en otro archivo la emoción de cada una de las imágenes, de tal manera que el fichero que contiene los píxeles y el que contiene la emoción con un valor del 0-7 quedan relacionados.

Seguidamente se aplica *train_test_split* que divide el array de entrada en unos subconjuntos de prueba y otros de entrenamiento. Una vez se obtienen los subconjuntos de prueba y entrenamiento se introducen en cada una de las redes neuronales para más tarde plotear tanto la gráfica resultante del entrenamiento como las matrices de confusión tanto normalizadas como sin normalizar. Los resultados de estos entrenamientos se pueden visualizar en el Apartado 5.2.

4.3 Dataset Affectnet.

En tercer lugar, se han extraídos las imágenes de unos diretorios que hay que descargar y que contienen las imágenes de este dataset. A partir del archivo training.csv y validation.csv ya proporcionados que contienen tanto la ruta a los directorios como la emoción entre otros

valores como se puede observar en la Figura 4.4 se han guardado las imágenes en siete directorios distintos, uno por cada emoción que se pretende reconocer. Cabe mencionar que hay 4 clases dentro del dataset que no han sido utilizadas entre ellas las clases: contempt, non-face, uncertain y none. A todas las imágenes que han sido guardadas se les ha hecho un procesamiento y han sido convertidas a escala de grises y escaladas a tamaño 48x48 píxeles.

	subDirectory_filePath	face_x	face_y	...	expression	valence	arousal
0	689/737db2483489148d783ef278f43f486c0a97e140fc...	134.0	134.0	...	1	0.785714	-0.055556
1	392/c4db2f9b7e4b422d14b6e038f0cdc3ecee239b5532...	20.0	20.0	...	0	-0.017253	0.004313
2	468/21772b68dc8c2a11678c8739eca33adb6ccc658600...	11.0	11.0	...	0	0.174603	0.007937
3	944/06e9ae8d3b240eb68fa60534783eacafce2def60a8...	40.0	40.0	...	1	0.153401	0.038890
4	993/02e06ee5521958b4042dd73abb444220609d96f57b...	22.0	22.0	...	8	0.783972	-0.551684

Figura 4.4: Cabecera archivo training.csv.

A continuación, en la siguiente Tabla 4.3 se pueden visualizar el número de imágenes que se han extraído del dataset Affectnet. El dataset Affectnet está formado por muchas más imágenes de las que fueron extraídas, por falta de espacio en el ordenador solo se pudieron extraer unas cuantas.

DATASET AFFECTNET	
Clases	Nº total imágenes
Angry	2843
Disgusted	702
Fearful	985
Happy	15494
Neutral	8155
Sad	3196
Surprised	2039

Tabla 4.3: Imágenes dataset Affectnet.

Los resultados de esta extracción se puede observar en la siguiente Figura 4.5 donde se pueden apreciar las diferentes clases del dataset Affectnet.

A continuación, una vez se ha hecho el procesamiento previamente mencionado, se han guardado los píxeles en un array que posteriormente se almacena en un archivo. También cabe mencionar que al mismo tiempo se ha guardado en otro archivo la emoción de cada una de las imágenes, de tal manera que el fichero que contiene los píxeles y el que contiene la emoción con un valor del 0-7 quedan relacionados.

Seguidamente se aplica *train_test_split* que divide el array de entrada en unos subconjuntos de prueba y otros de entrenamiento. Una vez se obtienen los subconjuntos de prueba

(a) Angry 1. **(b)** Disgusted 1. **(c)** Fearful 1. **(d)** Happy 1. **(e)** Neutral 1. **(f)** Sad 1. **(g)** Surprised 1.

(h) Angry 2. **(i)** Disgusted 2. **(j)** Fearful 2. **(k)** Happy 2. **(l)** Neutral 2. **(m)** Sad 2. **(n)** Surprised 2.

Figura 4.5: Imagenes Dataset Affectnet.

y entrenamiento se introducen en cada una de las redes neuronales para más tarde plotear tanto la gráfica resultante del entrenamiento como las matrices de confusión tanto normalizadas como sin normalizar. Los resultados de estos entrenamientos se pueden visualizar en el Apartado 5.3.

4.4 Dataset Affectnet+Fer2013.

En cuarto lugar, se ha creado un nuevo directorio donde se han unido tanto las imágenes del datasaet FER2013 sin alinear ni recortar como las del dataset Affectnet con el objetivo de incrementar el número de imágenes con las que relaizar el entrenamiento de los modelos.

Seguidamente, en la siguiente Tabla 4.4 se pueden visualizar el número de imágenes que se han utilizado para entrenar los 4 modelos de redes neuronales convolucionales. Cabe mencionar que al igual que en los anteriores datasets todas las imágenes han recibido un procesamiento y han sido convertidas a escala de grises y escaladas a tamaño 48x48 píxeles.

A continuación, una vez se ha hecho el procesamiento previamente mencionado, se han guardado los píxeles en un array que posteriormente se almacena en un archivo. También cabe mencionar que al mismo tiempo se ha guardado en otro archivo la emoción de cada una de las imágenes, de tal manera que el fichero que contiene los píxeles y el que contiene la emoción con un valor del 0-7 quedan relacionados.

Seguidamente se aplica *train_test_split* que divide el array de entrada en unos subconjuntos de prueba y otros de entrenamiento. Una vez se obtienen los subconjuntos de prueba y entrenamiento se introducen en cada una de las redes neuronales para más tarde plotear tanto la gráfica resultante del entrenamiento como las matrices de confusión tanto normalizadas como sin normalizar. Los resultados de estos entrenamientos se pueden visualizar en el Apartado 5.4.

DATASET FER2013+AFFECTNET	
Clases	Nº total imágenes
Angry	7796
Disgusted	1685
Fearful	6106
Happy	24483
Neutral	14353
Sad	9273
Surprised	6041

Tabla 4.4: Imágenes dataset FER2013+Affectnet.

4.5 Data Augmentation.

El aumento de datos en el análisis de datos son técnicas que se utilizan para aumentar la cantidad de datos agregando copias ligeramente modificadas de datos ya existentes o datos sintéticos recién creados a partir de datos existentes. Se ha realizado data augmentation de los datos obtenidos al juntar el dataset Fer2013 junto con Affectnet ya que es el que mejores resultados nos ha proporcionado, para ello se adquieren las imágenes de cada una de las clases y se han rotado -5º, 5º, -10º, 10º, -15º, 15º sucesivamente hasta completar la cantidad seleccionada.

En primer lugar se han igualado todas las clases de los datos de entrenamiento a 5000 imágenes. A continuación, una vez se ha hecho el procesamiento previamente mencionado, se han guardado los píxeles en un array que posteriormente se almacena en un archivo. También cabe mencionar que al mismo tiempo se ha guardado en otro archivo la emoción de cada una de las imágenes, de tal manera que el fichero que contiene los píxeles y el que contiene la emoción con un valor del 0-7 quedan relacionados.

Seguidamente se aplica *train_test_split* que divide el array de entrada en unos subconjuntos de prueba y otros de entrenamiento. Una vez se obtienen los subconjuntos de prueba y entrenamiento se introducen en cada una de las redes neuronales para más tarde plotear tanto la gráfica resultante del entrenamiento como las matrices de confusión tanto normalizadas como sin normalizar. Los resultados de estos entrenamientos se pueden visualizar en el Apartado 5.5.

4.6 Aplicación.

Finalmente, se ha realizado una aplicación que permite cargar los diferentes modelos entrenados y a través de la cámara del propio ordenador reconocer el rostro de la persona para identificar la emoción. Para ello se han realizado los siguientes pasos:

1. Cargar el modelo de red neuronal convolucional que ha sido guardado previamente durante el proceso de entrenamiento.

2. Crear un diccionario que nos permitirá asignar una etiqueta a cada emoción así como poder situar un pequeño emoticono en la parte superior derecha del cuadro delimitador.

3. Cargar la cámara del ordenador a través de OpenCV y dibujar un cuadrado delimitador alrededor del rostro convertido a escala de grises a través del detector de rostros de Haar Cascade.

4. A continuación, se obtiene la imagen delimitada en escala de grises, se escala a tamaño 48x48 píxeles y se le pasa al modelo para que determine la emoción mayoritaria presente en la imagen.

5. Seguidamente, se muestra en la parte superior izquierda la emoción detectada a través de texto, esto es posible a través del diccionario comentado anteriormente. También se carga un emoticono que se situará en la parte superior derecha en función de la emoción detectada.

6. Finalmente se muestra la cámara del ordenador y se delimita el rostro junto con la emoción en texto y mediante el emoticono.

Los resultados de esta aplicación se pueden visualizar en el Aparatado 5.6.

5 Resultados

En esta capítulo se presentarán los resultados obtenidos en las diferentes tareas realizadas hasta llegar al objetivo final del proyecto, basado en la identificación de las emociones a través del rostro.

5.1 Dataset Fer2013.

A continuación, se pueden visualizar los resultados del entrenamiento realizado a los cuatro modelos de redes neuronales convolucionales utilizados a lo largo del proyecto para el conjunto de datos de Fer2013. Se ha realizado el entrenamiento con diversos valores de *learning rate* entre ellos lr = 0.0001, lr = 0.001 y lr = 0.01, con el objetivo de determinar cuál es el valor que nos proporcionará unos mejores resultados a posteriori.

5.1.1 Red convolucional 4 capas, lr = 0.0001.

Los resultados del entrenamiento se pueden visualizar en la Figura 5.1.

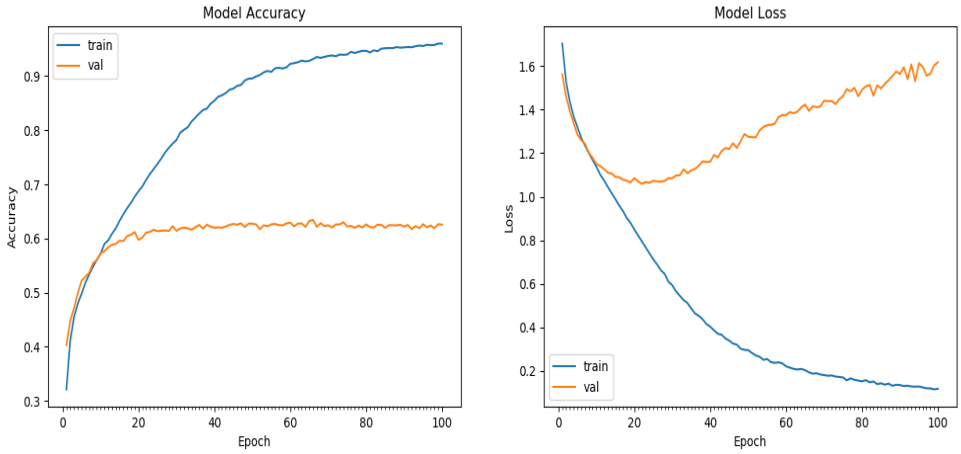
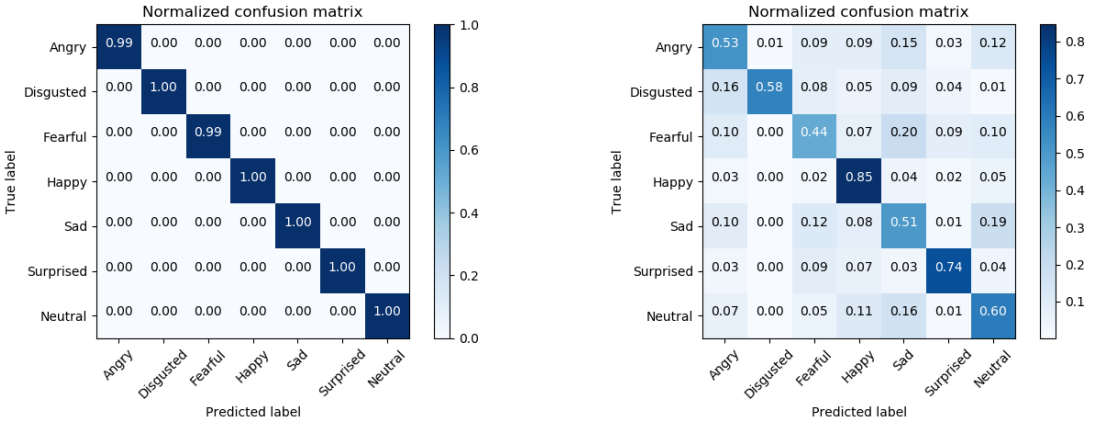


Figura 5.1: Accuracy vs loss red convolucional 4 capas, lr = 0.0001.

A continuación en la siguiente Figura 5.2a se pueden visualizar los resultados del entrenamiento, como se puede observar el entrenamiento se ha realizado correctamente y la mayoría de imágenes se encuentran en la diagonal principal de la matriz de confusión para el conjunto de entrenamiento. También, se pueden visualizar los resultados obtenidos a través de las matrices de confusión para el conjunto de validación a través de la Figura 5.2b.



(a) Entrenamiento. **(b)** Validación.

Figura 5.2: Matrices confusión red convolucional 4 capas, lr = 0.0001.

5.1.2 Red convolucional 4 capas, lr = 0.001.

Los resultados del entrenamiento se puede visualizar en la Figura 5.3.

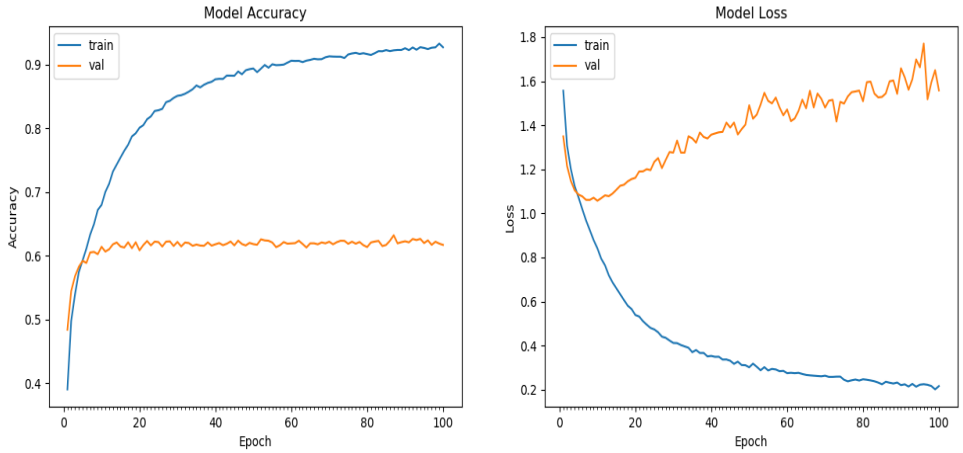
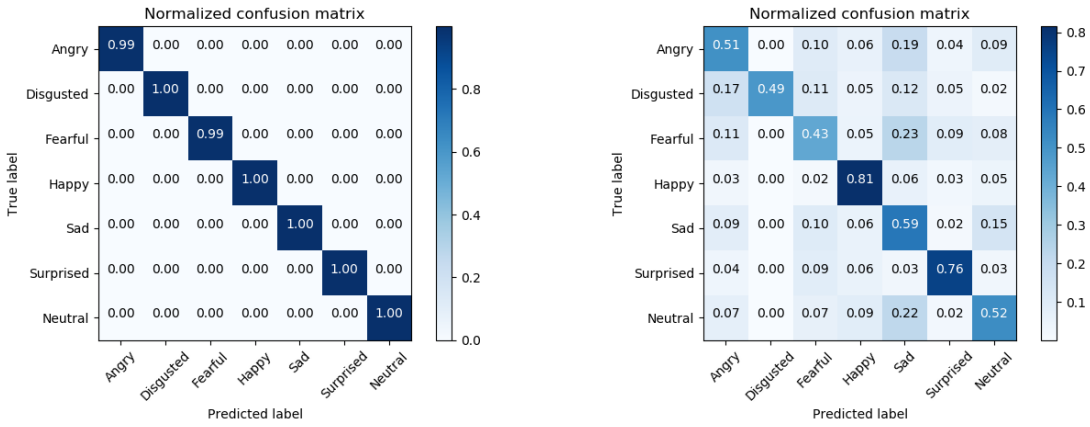


Figura 5.3: Accuracy vs loss red convolucional 4 capas, lr = 0.001.

A continuación en la siguiente Figura 5.4a se pueden visualizar los resultados del entrenamiento, como se puede observar el entrenamiento se ha realizado correctamente y la mayoría de imágenes se encuentran en la diagonal principal de la matriz de confusión para el conjunto de entrenamiento. También, se pueden visualizar los resultados obtenidos a través de las matrices de confusión para el conjunto de validación a través de la Figura 5.4b.



(a) Entrenamiento. **(b)** Validación.

Figura 5.4: Matrices confusión red convolucional 4 capas, lr = 0.001.

5.1.3 Red convolucional 4 capas, lr = 0.01.

Los resultados del entrenamiento se puede visualizar en la Figura 5.5, se puede observar como la red neuronal para un valor de *learning rate* = 0.01 no aprende el conjunto de datos.

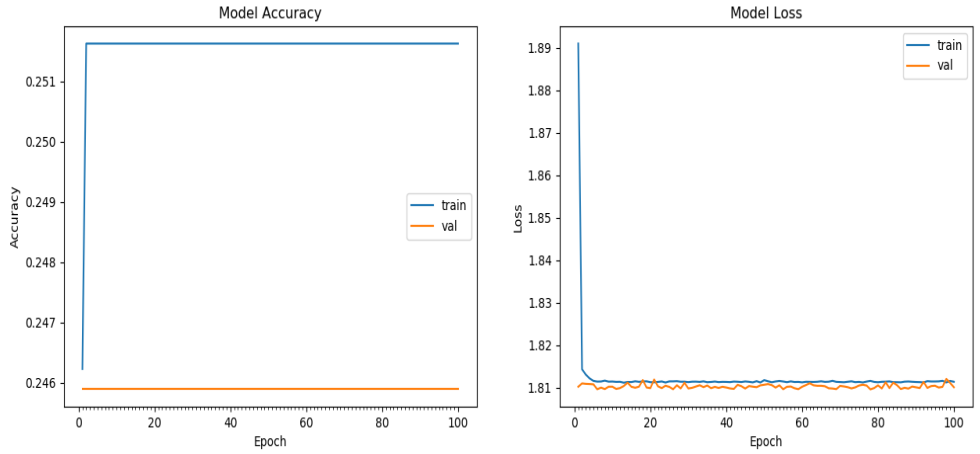
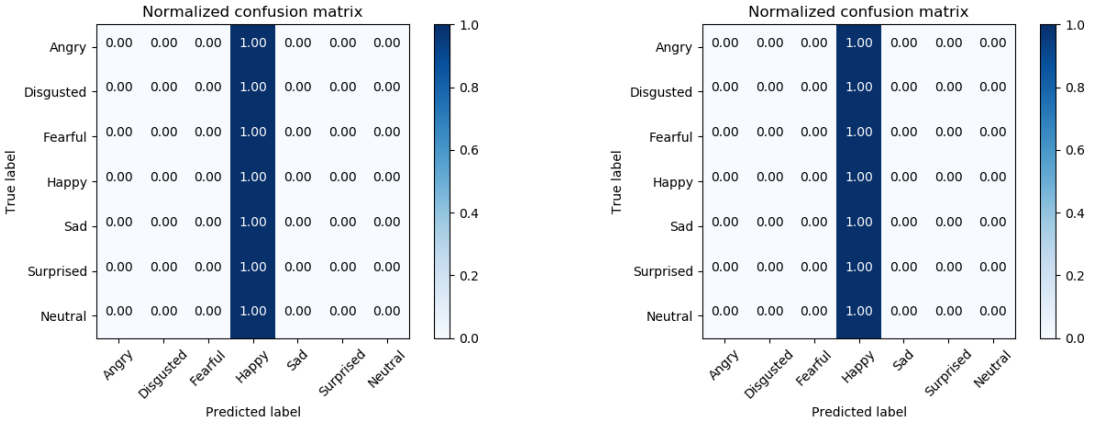


Figura 5.5: Accuracy vs loss red convolucional 4 capas, lr = 0.01.

A continuación en la siguiente Figura 5.6a se pueden visualizar los resultados del entrenamiento, como se puede observar el entrenamiento se ha realizado incorrectamente, se han clasificado todas las imágenes como si fueran de la clase felicidad (*Happy*). También, se pueden visualizar los resultados obtenidos a través de las matrices de confusión para el conjunto de validación a través de la Figura 5.6b, los cuales son mucho peores en comparación con los obtenidos para valores de *learning rate* inferiores. En este sentido, el modelo clasifica todas las imágenes como si pertenecieran a la clase feliz.



(a) Entrenamiento. **(b)** Validación.

Figura 5.6: Matrices confusión red convolucional 4 capas, lr = 0.01.

5.1.4 Red convolucional 8 capas, lr = 0.0001.

Los resultados del entrenamiento se puede visualizar en la Figura 5.7, se puede observar como se alcanza el *overfitting* más tarde que para el modelo convolucional de 4 capas, es decir, alrededor de la época 50.

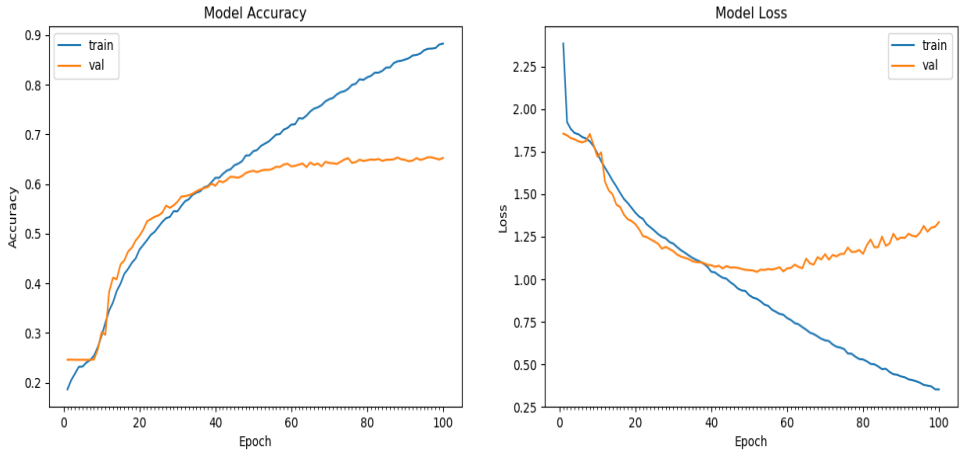
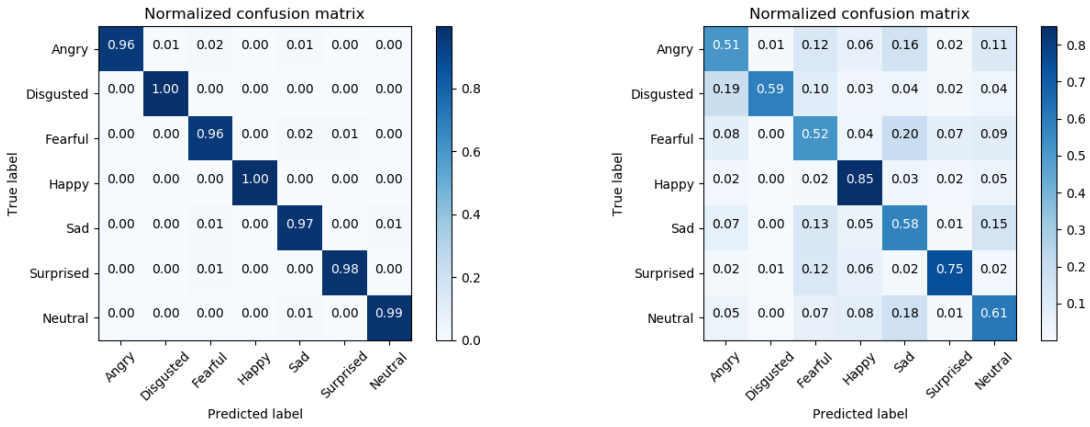


Figura 5.7: Accuracy vs loss red convolucional 8 capas, lr = 0.0001.

A continuación en la siguiente Figura 5.8a se pueden visualizar los resultados del entrenamiento, como se puede observar el entrenamiento se ha realizado correctamente y la mayoría de imágenes se encuentran en la diagonal principal de la matriz de confusión para el conjunto de entrenamiento. También, se pueden visualizar los resultados obtenidos a través de las matrices de confusión para el conjunto de validación a través de la Figura 5.8b.



(a) Entrenamiento. **(b)** Validación.

Figura 5.8: Matrices confusión red convolucional 8 capas, lr = 0.0001.

5.1.5 Red convolucional 8 capas, lr = 0.001.

Los resultados del entrenamiento se puede visualizar en la Figura 5.9.

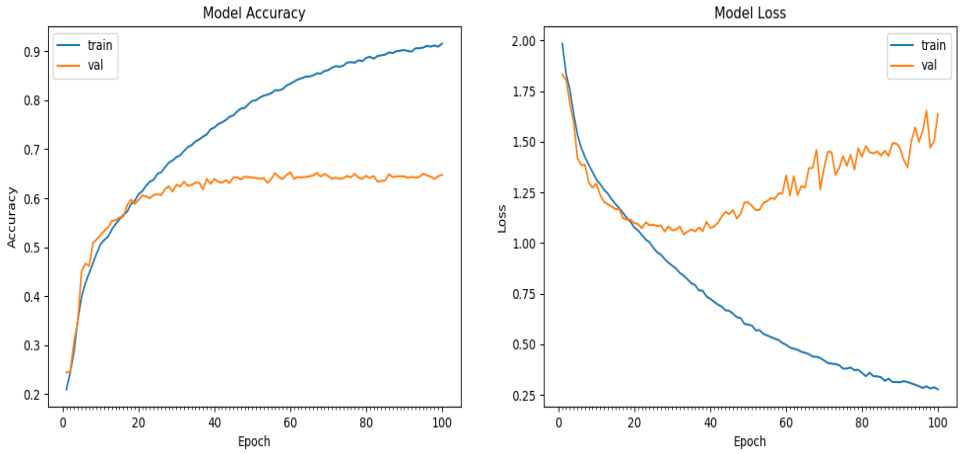
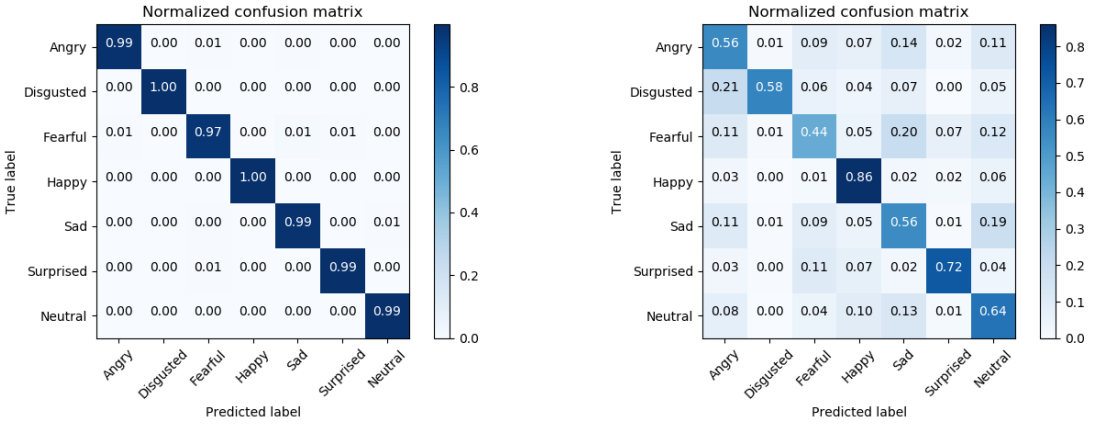


Figura 5.9: Accuracy vs loss red convolucional 8 capas.

A continuación en la siguiente Figura 5.10a se pueden visualizar los resultados del entrenamiento, como se puede observar el entrenamiento se ha realizado correctamente y la mayoría de imágenes se encuentran en la diagonal principal de la matriz de confusión para el conjunto de entrenamiento. También, se pueden visualizar los resultados obtenidos a través de las matrices de confusión para el conjunto de validación a través de la Figura 5.10b.



(a) Entrenamiento. **(b)** Validación.

Figura 5.10: Matrices confusión red convolucional 8 capas, lr = 0.001.

5.1.6 Red convolucional 8 capas, lr = 0.01.

Los resultados del entrenamiento se puede visualizar en la Figura 5.11, se puede observar como la red neuronal para un valor de *learning rate* = 0.01 no aprende el conjunto de datos.

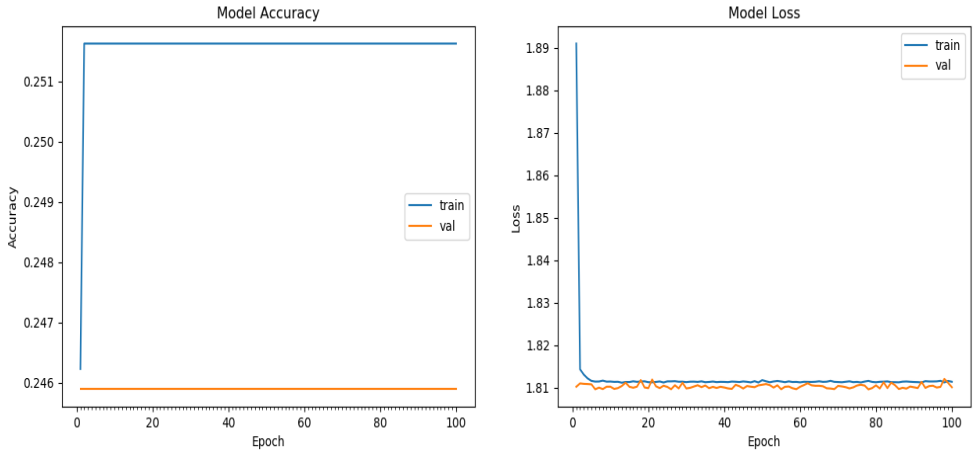
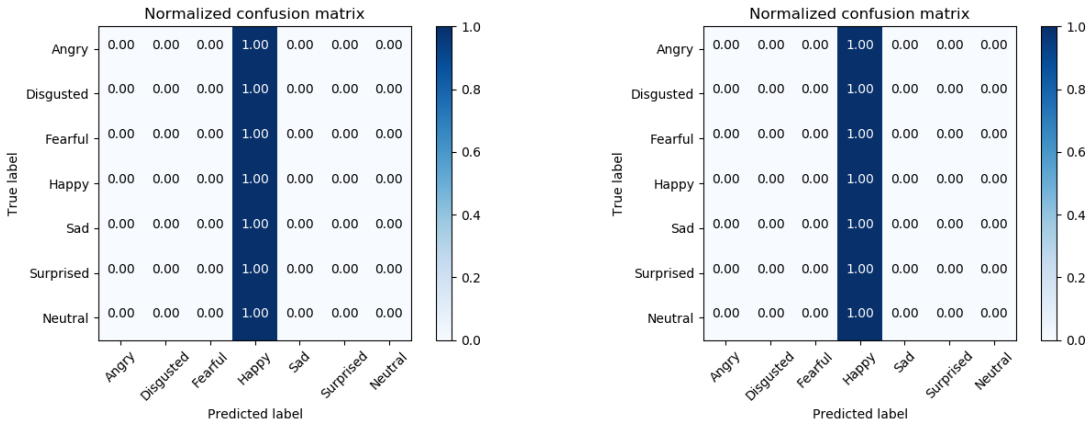


Figura 5.11: Accuracy vs loss red convolucional 8 capas, lr = 0.01.

A continuación en la siguiente Figura 5.12a se pueden visualizar los resultados del entrenamiento, como se puede observar el entrenamiento se ha realizado incorrectamente, se han clasificado todas las imágenes como si fueran de la clase felicidad (*Happy*). También, se pueden visualizar los resultados obtenidos a través de las matrices de confusión para el conjunto de validación a través de la Figura 5.12b, los cuales son mucho peores en comparación con los obtenidos para valores de *learning rate* inferiores. En este sentido, el modelo clasifica todas las imágenes como si pertenecieran a la clase feliz.



(a) Entrenamiento. **(b)** Validación.

Figura 5.12: Matrices confusión red convolucional 8 capas, lr = 0.01.

5.1.7 NASNetmobile, lr = 0.0001.

Los resultados del entrenamiento se puede visualizar en la Figura 5.13.

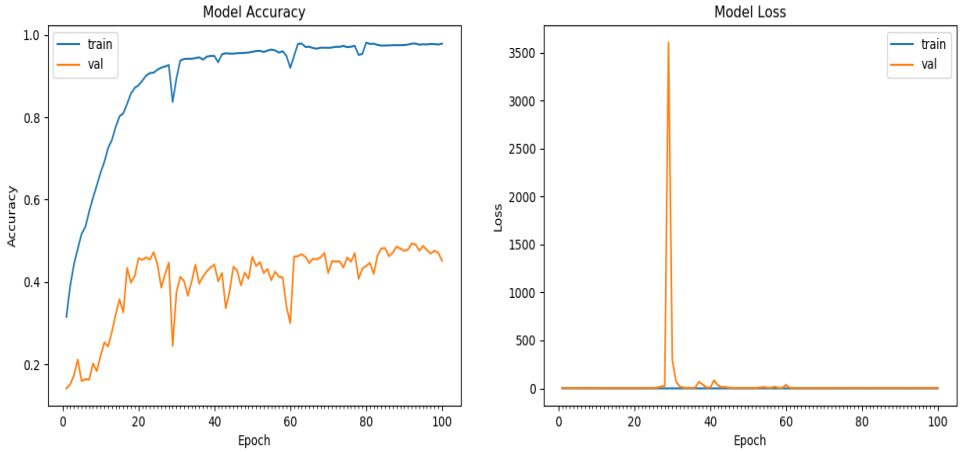
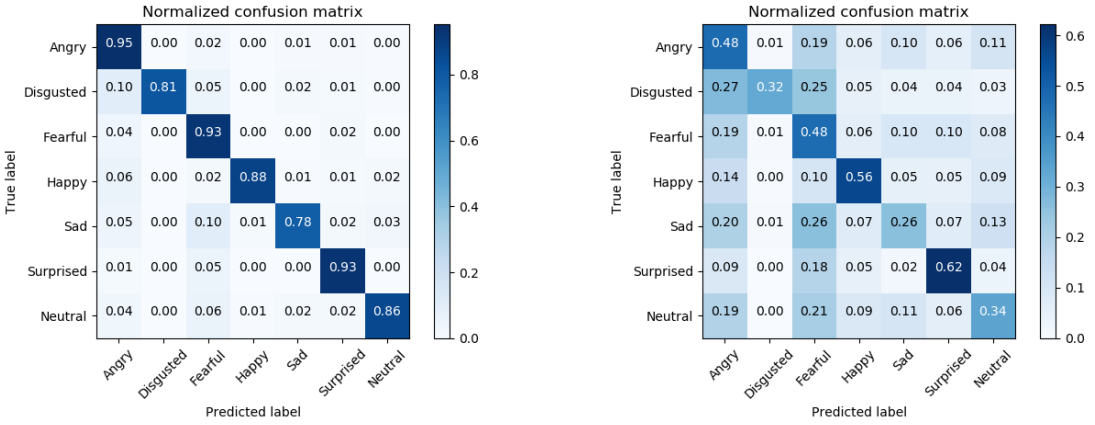


Figura 5.13: Accuracy vs loss NASNetmobile, lr = 0.0001.

A continuación en la siguiente Figura 5.14a se pueden visualizar los resultados del entrenamiento, como se puede observar el entrenamiento se ha realizado correctamente y la mayoría de imágenes se encuentran en la diagonal principal de la matriz de confusión para el conjunto de entrenamiento. También, se pueden visualizar los resultados obtenidos a través de las matrices de confusión para el conjunto de validación a través de la Figura 5.14b.

5.1.8 NASNetmobile, lr = 0.001.

Los resultados del entrenamiento se puede visualizar en la Figura 5.15.



(a) Entrenamiento. **(b)** Validación.

Figura 5.14: Matrices confusión NASNetmobile, lr = 0.0001.

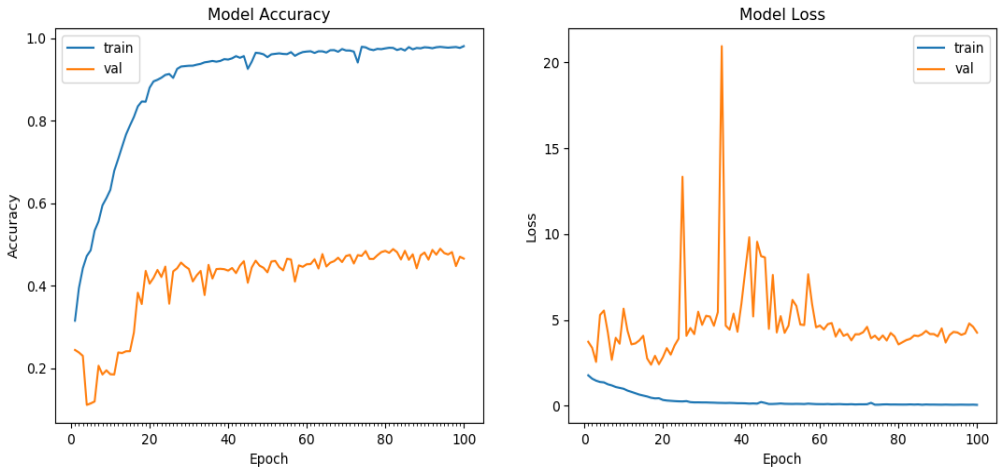


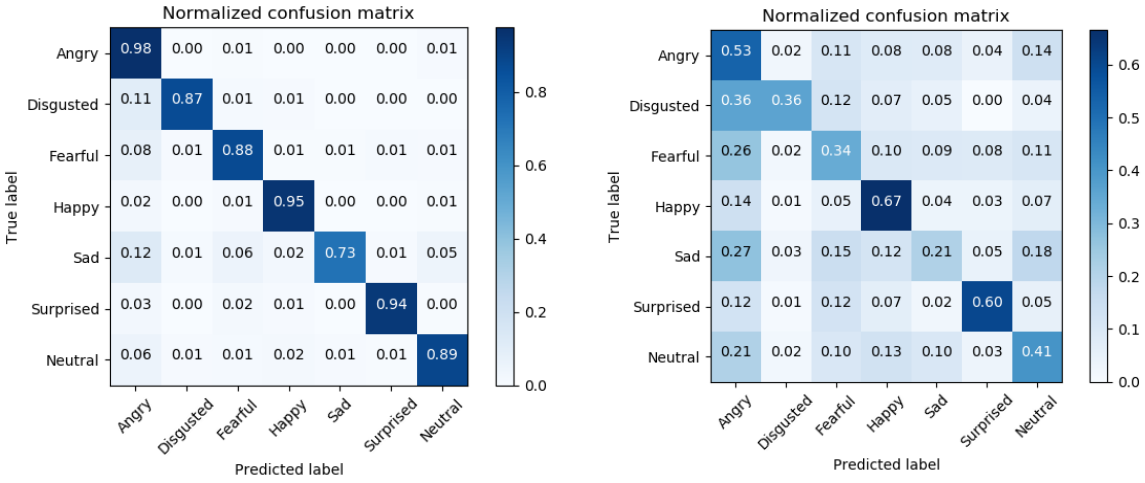
Figura 5.15: Accuracy vs loss red NASNetmobile, lr = 0.001.

A continuación en la siguiente Figura 5.16a se pueden visualizar los resultados del entrenamiento, como se puede observar el entrenamiento se ha realizado correctamente y la mayoría de imágenes se encuentran en la diagonal principal de la matriz de confusión para el conjunto de entrenamiento. También, se pueden visualizar los resultados obtenidos a través de las matrices de confusión para el conjunto de validación a través de la Figura 5.16b

5.1.9 NASNetmobile, lr = 0.01

Los resultados del entrenamiento se puede visualizar en la Figura 5.17.

A continuación en la siguiente Figura 5.18a se pueden visualizar los resultados del entrenamiento, como se puede observar el entrenamiento se ha realizado correctamente y la mayoría de imágenes se encuentran en la diagonal principal de la matriz de confusión para el conjunto de entrenamiento. También, se pueden visualizar los resultados obtenidos a través de las



(a) Entrenamiento **(b)** Validación

Figura 5.16: Matrices confusión red NASNetmobile, lr = 0.001

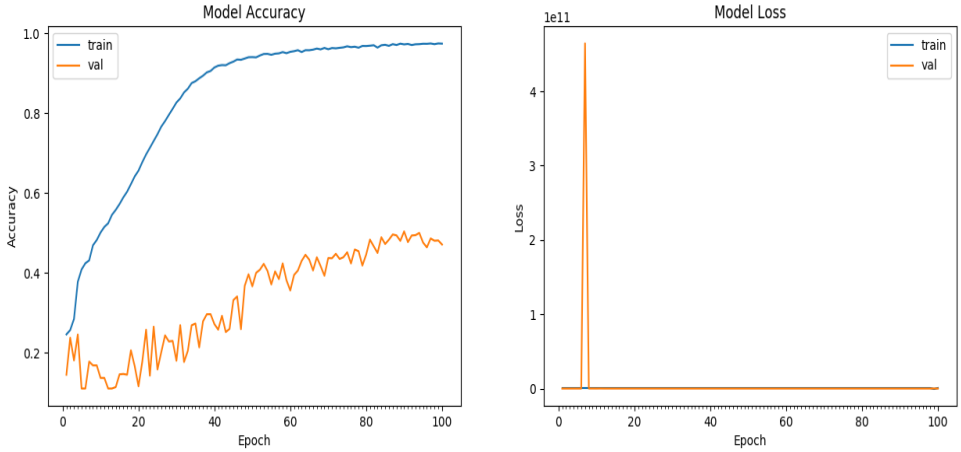


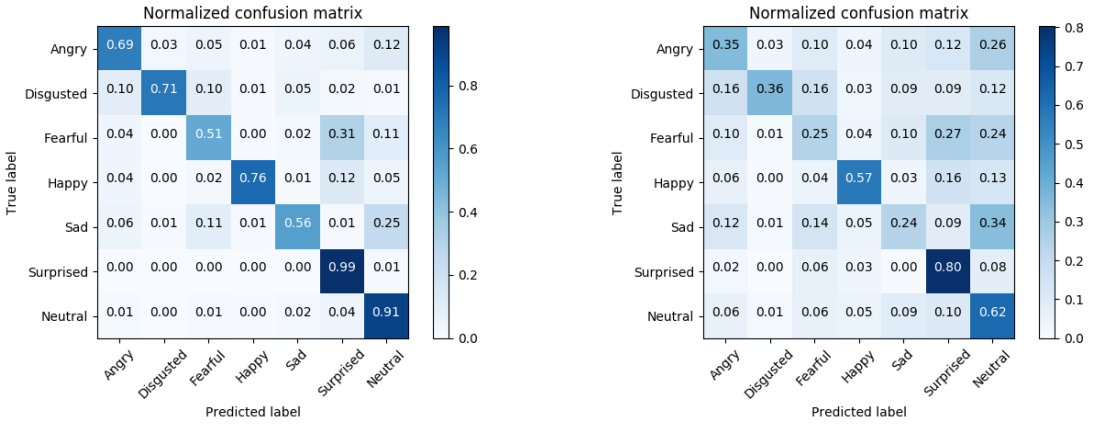
Figura 5.17: Accuracy vs loss NASNetmobile, lr = 0.01.

matrices de confusión para el conjunto de validación a través de la Figura 5.18b

5.1.10 ResNet50, lr = 0.0001

Los resultados del entrenamiento se puede visualizar en la Figura 5.19.

A continuación en la siguiente Figura 5.20a se pueden visualizar los resultados del entrenamiento, como se puede observar el entrenamiento se ha realizado correctamente y la mayoría de imágenes se encuentran en la diagonal principal de la matriz de confusión para el conjunto de entrenamiento. También, se pueden visualizar los resultados obtenidos a través de las matrices de confusión para el conjunto de validación a través de la Figura 5.20b.



(a) Entrenamiento **(b)** Validación

Figura 5.18: Matrices confusión NASNetmobile, lr = 0.01

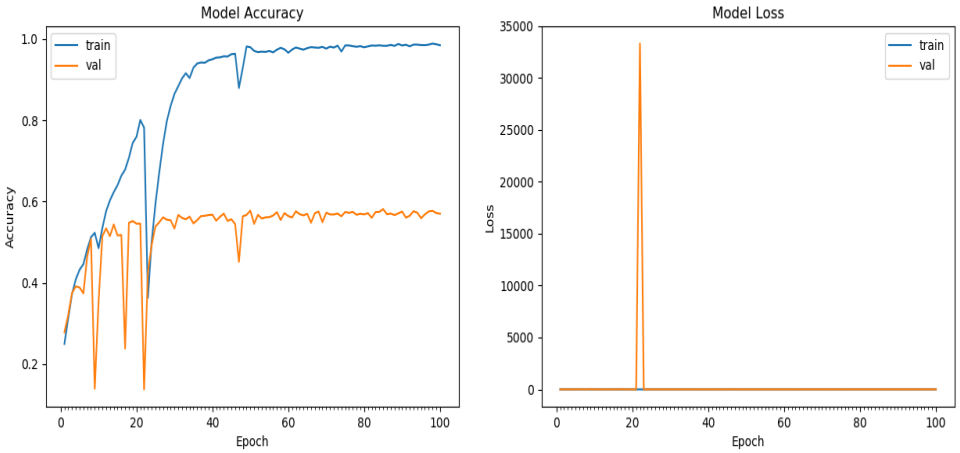


Figura 5.19: Accuracy vs loss ResNet50, lr = 0.0001.

5.1.11 ResNet50, lr = 0.001.

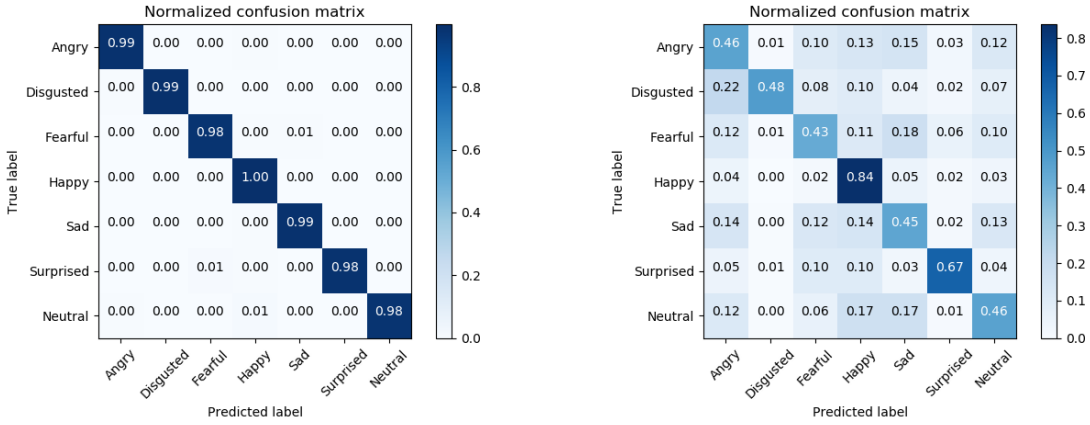
Los resultados del entrenamiento se puede visualizar en la Figura 5.21.

A continuación en la siguiente Figura 5.22a se pueden visualizar los resultados del entrenamiento, como se puede observar el entrenamiento se ha realizado correctamente y la mayoría de imágenes se encuentran en la diagonal principal de la matriz de confusión para el conjunto de entrenamiento. También, se pueden visualizar los resultados obtenidos a través de las matrices de confusión para el conjunto de validación a través de la Figura 5.22b.

5.1.12 ResNet50, lr = 0.01.

Los resultados del entrenamiento se puede visualizar en la Figura 5.23.

A continuación en la siguiente Figura 5.24a se pueden visualizar los resultados del entrena-



(a) Entrenamiento. **(b)** Validación.

Figura 5.20: Matrices confusión ResNet50, lr = 0.0001.

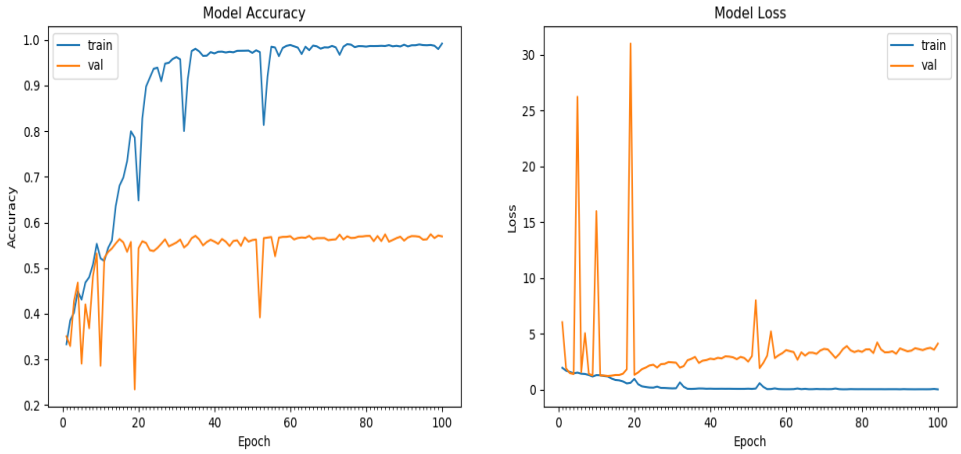
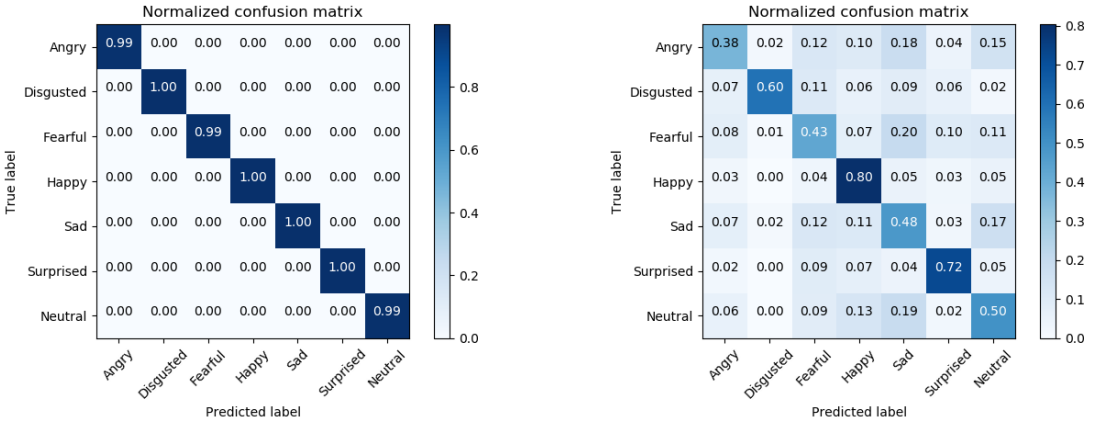


Figura 5.21: Accuracy vs loss red ResNet50, lr = 0.001.

miento, como se puede observar el entrenamiento se ha realizado correctamente y la mayoría de imágenes se encuentran en la diagonal principal de la matriz de confusión para el conjunto de entrenamiento. También, se pueden visualizar los resultados obtenidos a través de las matrices de confusión para el conjunto de validación a través de la Figura 5.24b.

5.1.13 Tabla comparativa FER2013

En este apartado se analizarán los diversos modelos de redes neuronales convolucionales que han sido entrenados con el dataset FER2013 a partir de la diagonal principal de la matriz de confusión y la respectiva métrica de exactitud. En la siguiente Tabla 5.1 se pueden visualizar las diagonales principales de cada uno de los modelos a través de los diferentes valores de *learning rate*, así como los respectivos valores de exactitud.



(a) Entrenamiento. **(b)** Validación.

Figura 5.22: Matrices confusión red ResNet50, lr = 0.001.

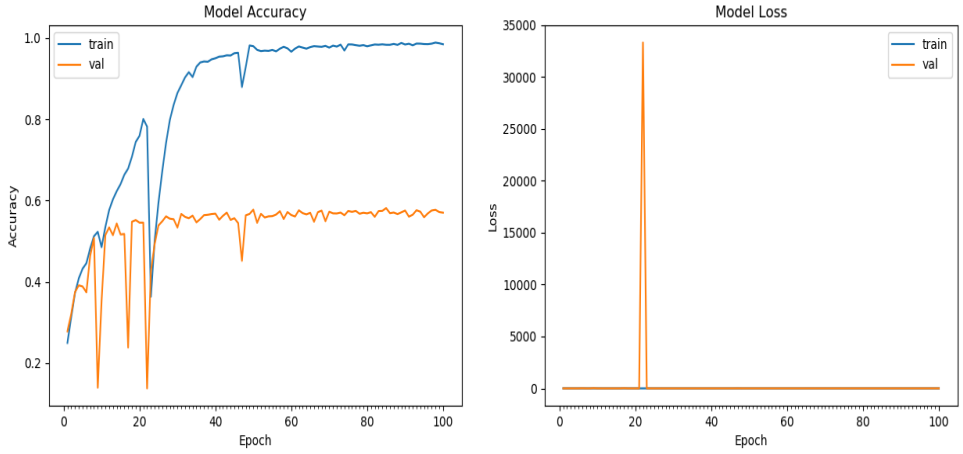
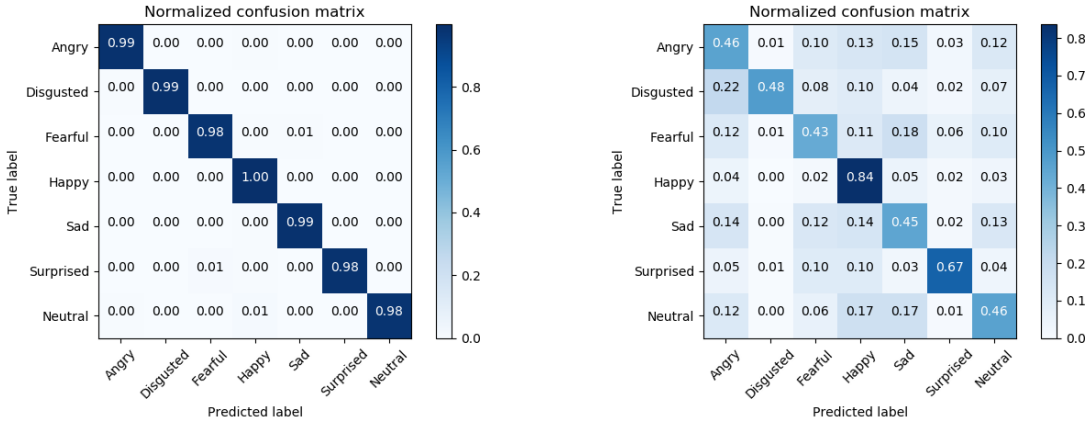


Figura 5.23: Accuracy vs loss ResNet50, lr = 0.01.

Como se puede observar en la Tabla 5.1 los mejores resultados se obtienen para la red neuronal convolucional de 8 capas, por otro lado los peores resultados se han obtenido para el modelo de las aplicaciones de Keras NASNetmobile. También cabe destacar que con un *learning rate* = 0.0001 es decir el más bajo empleado los modelos de las aplicaciones de Keras NASNetmobile y ResNet50 obtienen peores resultados que con valores de *learning rate* mayores, mientras que los 2 modelos de redes neuronales de 4 capas y 8 capas obtienen mejores resultados con valores de *learning rate* mayores. Es por esto, por lo que los siguientes entrenamientos se han realizado únicamente con un valor de *learning rate* = 0.001, ya que se obtienen entrenamientos más estables para todos los modelos que se pretenden estudiar. También cabe mencionar que las clases con mayor número de aciertos del conjunto de pruebas son la clase feliz y sorpresa, mientras que las clases con menos número de aciertos son miedo, enfado y neutro.



(a) Entrenamiento. **(b)** Validación.

Figura 5.24: Matrices confusión ResNet50, lr = 0.01.

MATRICES CONFUSIÓN Y MÉTRICA EXACTITUD DATASET FER2013								
Modelos	Angry	Disgusted	Fearful	Happy	Sad	Surprised	Neutral	*Accuracy*
CNN4 lr = 0.0001	0.53	0.58	0.44	0.85	0.51	0.74	0.60	62.57%
CNN4 lr = 0.001	0.51	0.49	0.43	0.81	0.59	0.76	0.52	61.69%
CNN4 lr = 0.01	0	0	0	1.00	0	0	0	24.59%
CNN8 lr = 0.0001	0.51	0.59	0.52	0.85	0.58	0.75	0.61	65.25%
CNN8 lr = 0.001	0.56	0.58	0.44	0.86	0.56	0.72	0.64	64.79%
CNN8 lr = 0.01	0	0	0	1.00	0	0	0	24.59%
NASNetmobile lr = 0.0001	0.48	0.32	0.48	0.56	0.26	0.62	0.34	45.15%
NASNetmobile lr = 0.001	0.53	0.36	0.34	0.67	0.21	0.60	0.41	46.63%
NASNetmobile lr = 0.01	0.35	0.36	0.25	0.57	0.24	0.80	0.62	47.12%
ResNet50 lr = 0.0001	0.46	0.48	0.43	0.84	0.45	0.67	0.46	56.94%
ResNet50 lr = 0.001	0.38	0.60	0.43	0.80	0.48	0.72	0.50	56.94%
ResNet50 lr = 0.01	0.46	0.48	0.43	0.84	0.45	0.67	0.46	56.94%

Tabla 5.1: Tabla comparativa resultados redes neuronales convolucionales FER2013.

5.2 Alineamiento y recorte del rostro FER2013.

A continuación, se realizó un alineamiento del rostro, posteriormente a través de DLIB se reconoció el rostro y se recortó, de esta forma nos quedamos con la parte más significativa de cada una de las imágenes. En ocasiones, no se detecta el rostro, en este caso se han guardado las imágenes sin recortar ni alinear debido a que si no perderíamos datos para entrenar nuestro modelo.

Seguidamente, se pueden visualizar los resultados del entrenamiento realizado a los cuatro

modelos de redes neuronales convolucionales utilizados a lo largo del proyecto para el conjunto de datos alineados y recortados de FER2013. Se ha realizado el entrenamiento con un valor de *learning rate* lr = 0.001 debido a que era uno de los mejores resultados proporcionados para el dataset FER2013 y porque el tiempo de ejecución respecto a un *learning-rate* = 0.0001 era menor, por lo que merece la pena sacrificar un poco de *accuracy* con tal de entrenar los modelos más rápidamente.

5.2.1 Red convolucional 4 capas, lr = 0.001.

Los resultados del entrenamiento se puede visualizar en la Figura 5.25.

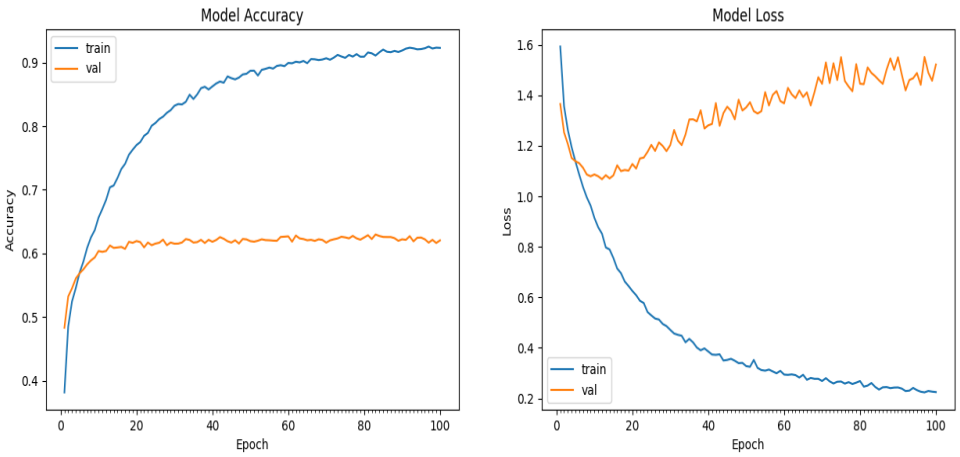
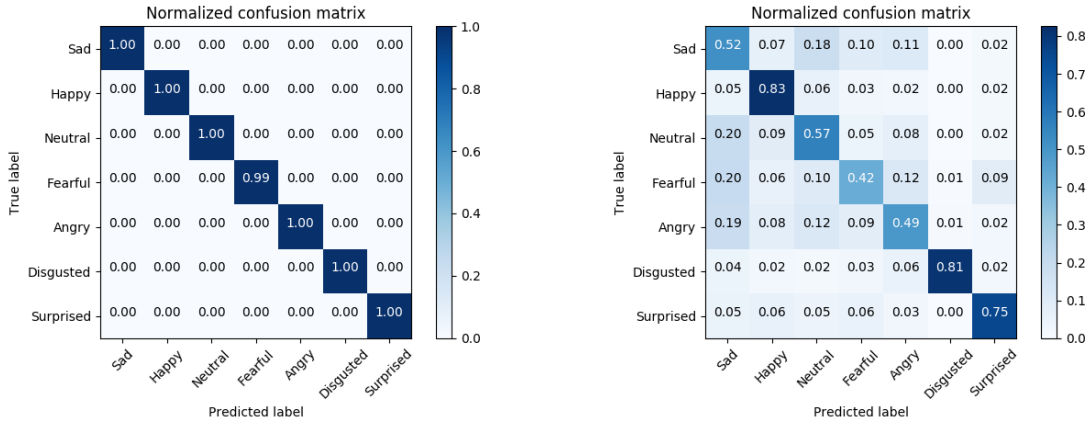


Figura 5.25: Accuracy vs loss red convolucional 4 capas, lr = 0.001.

A continuación en la siguiente Figura 5.26a se pueden visualizar los resultados del entrenamiento, como se puede observar el entrenamiento se ha realizado correctamente y la mayoría de imágenes se encuentran en la diagonal principal de la matriz de confusión para el conjunto de entrenamiento. También, se pueden visualizar los resultados obtenidos a través de las matrices de confusión para el conjunto de validación a través de la Figura 5.26b.

5.2.2 Red convolucional 8 capas, lr = 0.001.

Los resultados del entrenamiento se puede visualizar en la Figura 5.27.



(a) Entrenamiento. **(b)** Validación.

Figura 5.26: Matrices confusión red convolucional 4 capas, lr = 0.001.

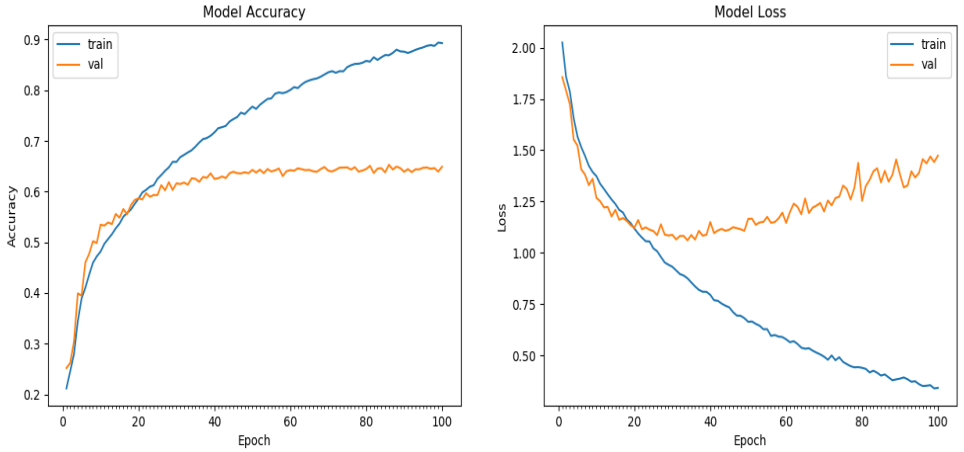
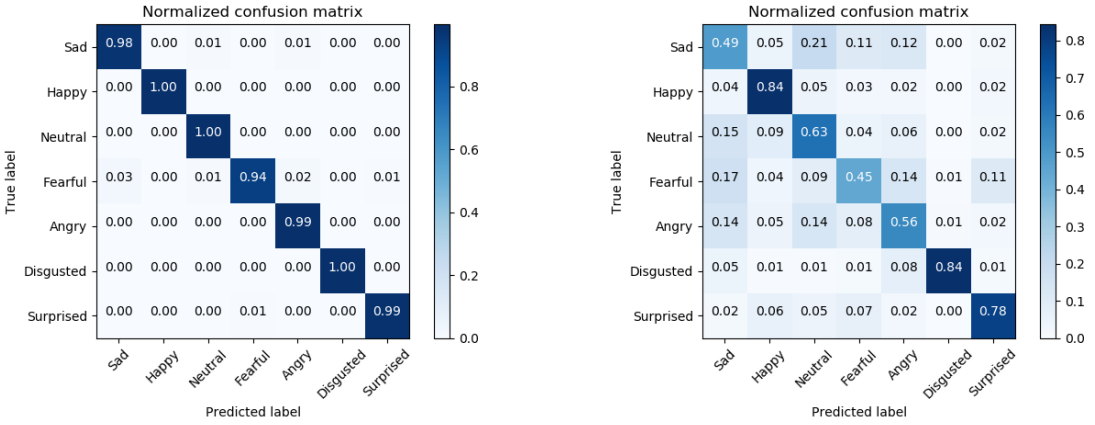


Figura 5.27: Accuracy vs loss red convolucional 8 capas, lr = 0.001.

A continuación en la siguiente Figura 5.28a se pueden visualizar los resultados del entrenamiento, como se puede observar el entrenamiento se ha realizado correctamente y la mayoría de imágenes se encuentran en la diagonal principal de la matriz de confusión para el conjunto de entrenamiento. También, se pueden visualizar los resultados obtenidos a través de las matrices de confusión para el conjunto de validación a través de la Figura 5.28b.

5.2.3 NASNetmobile, lr = 0.001

Los resultados del entrenamiento se puede visualizar en la Figura 5.29.



(a) Entrenamiento. **(b)** Validación.

Figura 5.28: Matrices confusión red convolucional 8 capas, lr = 0.001.

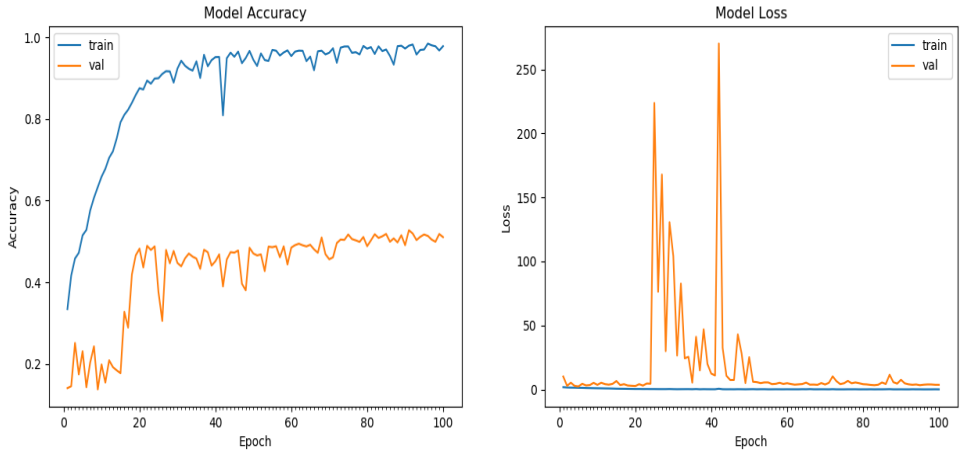


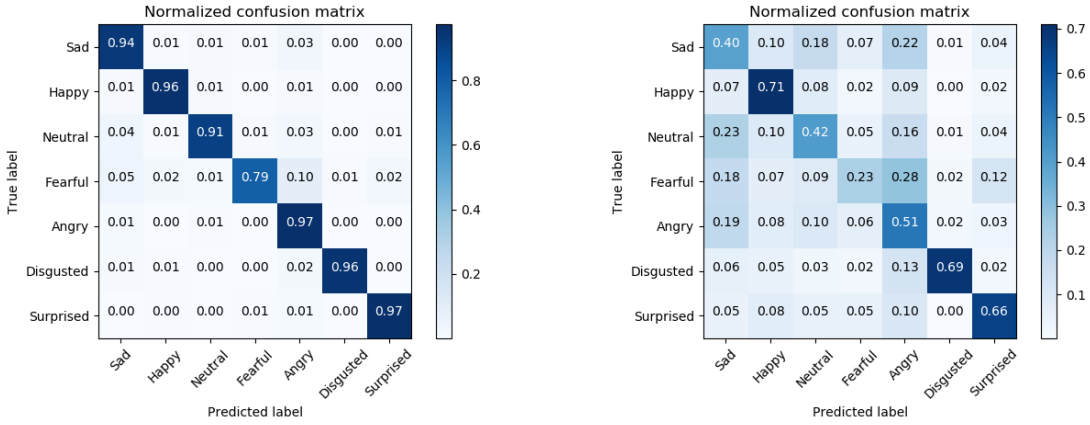
Figura 5.29: Accuracy vs loss NASNetmobile, lr = 0.001.

A continuación en la siguiente Figura 5.30a se pueden visualizar los resultados del entrenamiento, se produce correctamente el entrenamiento y la mayoría de imágenes se encuentran en la diagonal principal de la matriz de confusión. También, se pueden visualizar los resultados obtenidos a través de las matrices de confusión para el conjunto de validación a través de la Figura 5.30b.

5.2.4 ResNet50, lr = 0.001

Los resultados del entrenamiento se puede visualizar en la Figura 5.31.

A continuación en la siguiente Figura 5.32a se pueden visualizar los resultados del entrenamiento, se produce correctamente el entrenamiento y la mayoría de imágenes se encuentran en la diagonal principal de la matriz de confusión. También, se pueden visualizar los resul-



(a) Entrenamiento. **(b)** Validación.

Figura 5.30: Matrices confusión NASNetmobile, lr = 0.001.

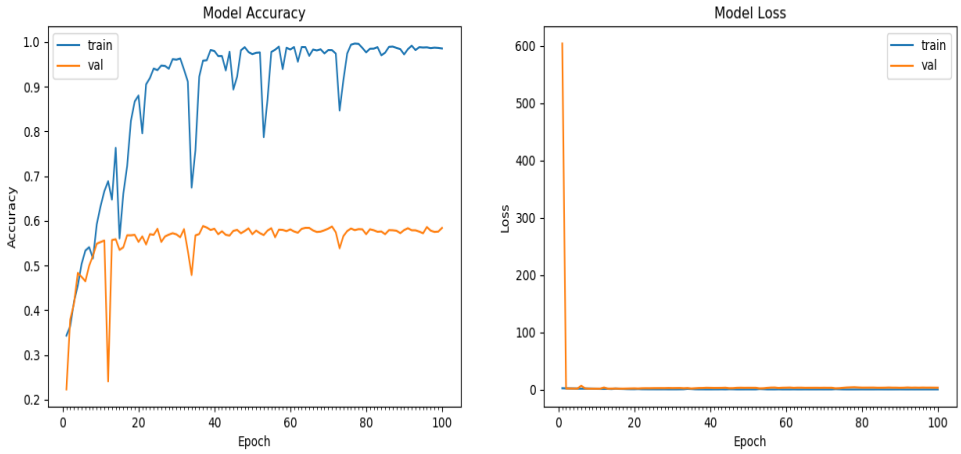


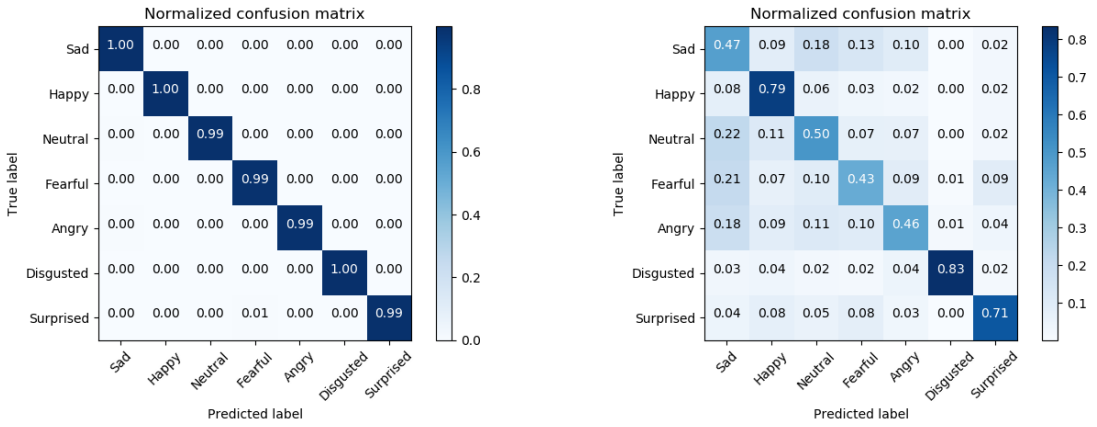
Figura 5.31: Accuracy vs loss ResNet50, lr = 0.001.

tados obtenidos a través de las matrices de confusión para el conjunto de validación a través de la Figura 5.32b.

5.2.5 Tabla comparativa FER2013 Alineado y Recortado

Seguidamente, en la Tabla 5.2 se pueden visualizar las diagonales principales de cada uno de los modelos así como los respectivos valores de exactitud.

Se puede observar que tras realizar el alineamiento y recorte de los rostros los resultados no mejoran significativamente con respecto a los resultados obtenidos en la Tabla 5.1 para los mismos valores de *learning rate*. Es por este motivo, por el cual se ha decidido continuar con los próximos datasets sin alinear ni recortar los rostros de las imágenes. También se puede visualizar como el modelo que obtiene un mayor valor de exactitud es el de 8 capas



(a) Entrenamiento. **(b)** Validación.

Figura 5.32: Matrices confusión ResNet50, lr = 0.001.

DATASET FER2013 ALINEADO Y RECORTADO								
Modelos	Angry	Disgusted	Fearful	Happy	Sad	Surprised	Neutral	*Accuracy*
CNN4 lr = 0.001	0.49	0.81	0.42	0.83	0.52	0.75	0.57	62.05%
CNN8 lr = 0.001	0.56	0.84	0.45	0.84	0.49	0.78	0.63	64.91%
NASNetmobile lr = 0.001	0.51	0.69	0.23	0.71	0.40	0.66	0.42	51.04%
ResNet50 lr = 0.001	0.46	0.83	0.43	0.79	0.47	0.71	0.50	58.42%

Tabla 5.2: Tabla comparativa resultados redes neuronales convolucionales FER2013 alineado y recortado.

convolucionales, mientras que el modelo que obtiene peores resultados es de las aplicaciones de Keras NASNetmobile. La clase asco pasa a ser una de las más reconocidas a pesar del poco número de muestras, a su vez la clase feliz y sorpresa siguen teniendo un gran número de aciertos para el conjunto de prueba. Por otro lado, las clases enfado, miedo y neutro siguen teniendo el menor número de aciertos para el conjunto de prueba.

5.3 Affectnet.

Se hará uso del dataset Affectnet para el entrenamiento de los distintos modelos explicados anteriormente.

5.3.1 Red convolucional 4 capas, lr = 0.001.

Los resultados del entrenamiento se puede visualizar en la Figura 5.33.

A continuación en la siguiente Figura 5.34a se pueden visualizar los resultados del entrenamiento, como se puede observar el entrenamiento se ha realizado correctamente y la mayoría

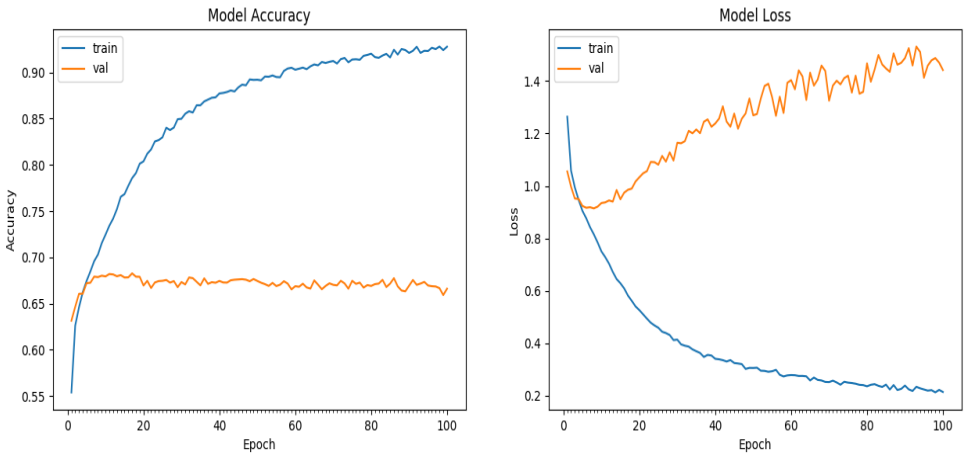
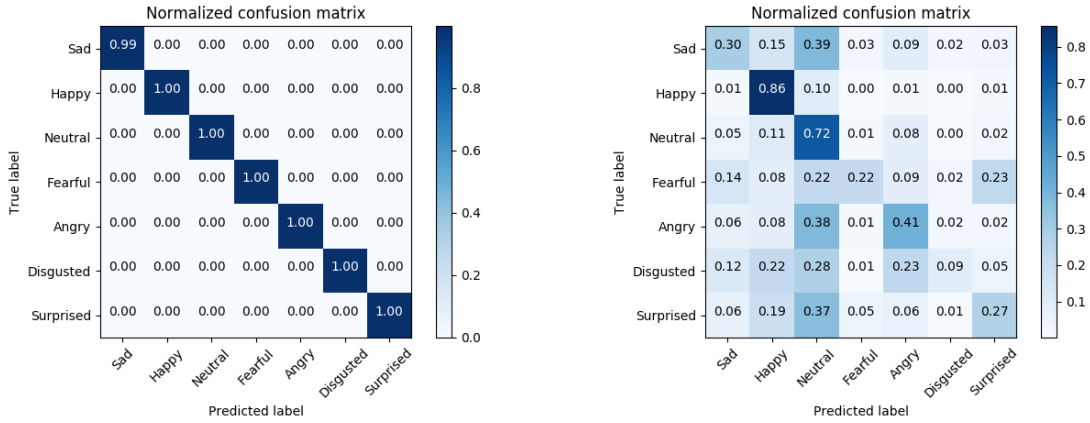


Figura 5.33: Accuracy vs loss red convolucional 4 capas, lr = 0.001.

de imágenes se encuentran en la diagonal principal de la matriz de confusión para el conjunto de entrenamiento. También, se pueden visualizar los resultados obtenidos a través de las matrices de confusión para el conjunto de validación a través de la Figura 5.34b.



(a) Entrenamiento. **(b)** Validación.

Figura 5.34: Matrices confusión red convolucional 4 capas, lr = 0.001.

5.3.2 Red convolucional 8 capas, lr = 0.001.

Los resultados del entrenamiento se puede visualizar en la Figura 5.35.

A continuación en la siguiente Figura 5.36a se pueden visualizar los resultados del entrenamiento, se produce correctamente el entrenamiento y la mayoría de imágenes se encuentran en la diagonal principal de la matriz de confusión. También, se pueden visualizar los resultados obtenidos a través de las matrices de confusión para el conjunto de validación a través de la Figura 5.36b.

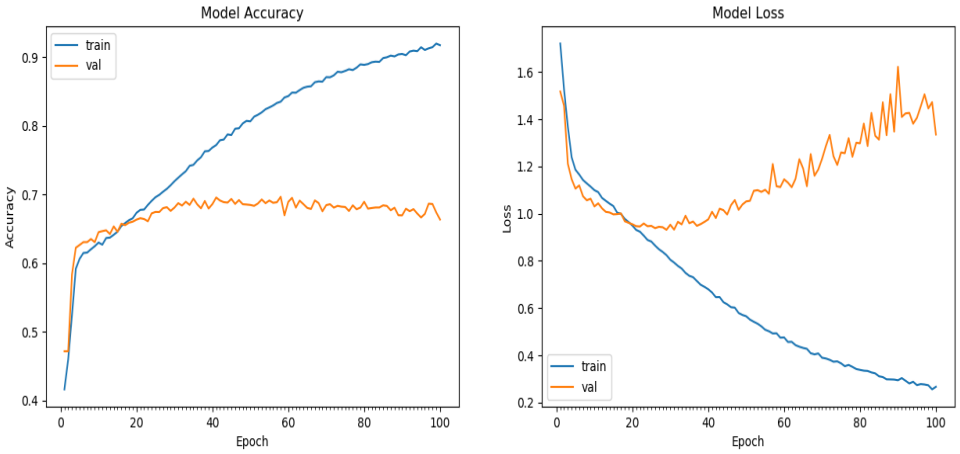
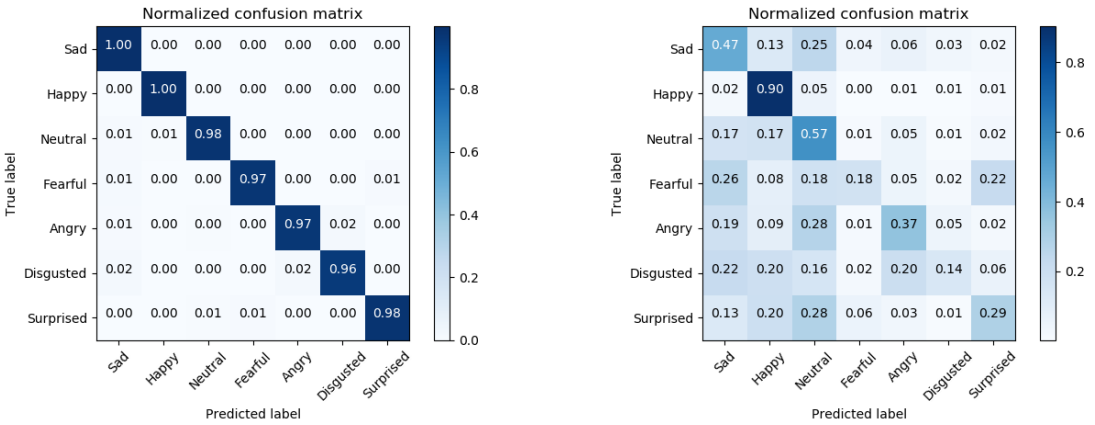


Figura 5.35: Accuracy vs loss red convolucional 8 capas, lr = 0.001.



(a) Entrenamiento. **(b)** Validación.

Figura 5.36: Matrices confusión red convolucional 8 capas, lr = 0.001.

5.3.3 NasNetMobile, lr = 0.001.

Los resultados del entrenamiento se puede visualizar en la Figura 5.37.

A continuación en la siguiente Figura 5.38a se pueden visualizar los resultados del entrenamiento, como se puede observar el entrenamiento se ha realizado correctamente y la mayoría de imágenes se encuentran en la diagonal principal de la matriz de confusión para el conjunto de entrenamiento. También, se pueden visualizar los resultados obtenidos a través de las matrices de confusión para el conjunto de validación a través de la Figura 5.38b.

5.3.4 ResNet50, lr = 0.001.

Los resultados del entrenamiento se puede visualizar en la Figura 5.39.

A continuación en la siguiente Figura 5.40a se pueden visualizar los resultados del entrena-

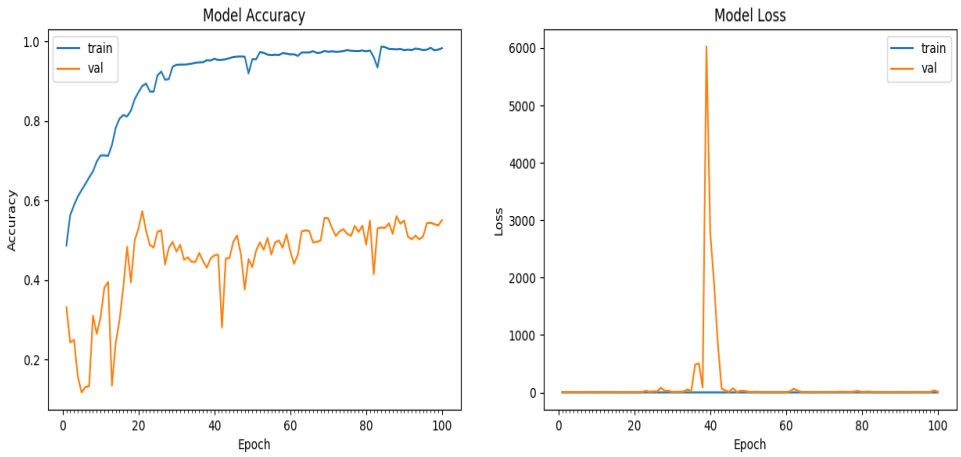
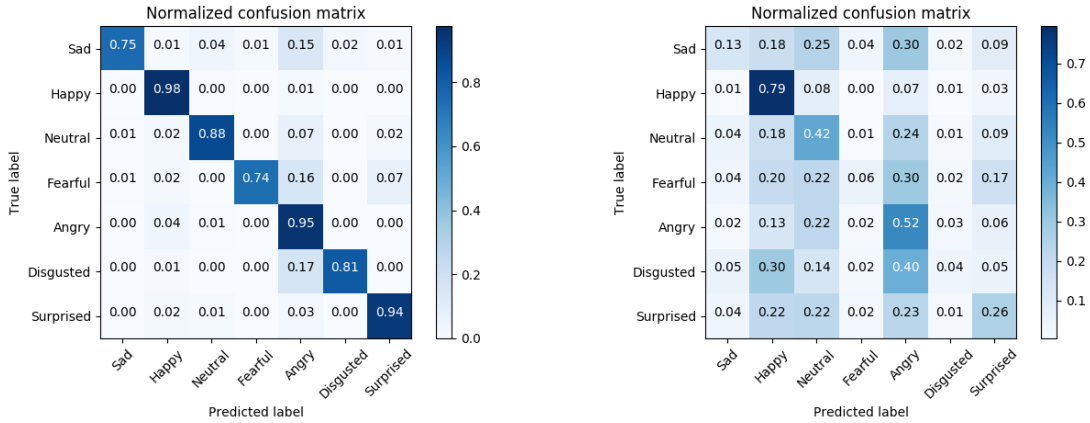


Figura 5.37: Accuracy vs loss red NasNetMobile, lr = 0.001.



(a) Entrenamiento. **(b)** Validación.

Figura 5.38: Matrices confusión red NasNetMobile, lr = 0.001.

miento, se produce correctamente el entrenamiento y la mayoría de imágenes se encuentran en la diagonal principal de la matriz de confusión. También, se pueden visualizar los resultados obtenidos a través de las matrices de confusión para el conjunto de validación a través de la Figura 5.40b.

5.3.5 Tabla comparativa Affectnet.

Seguidamente, en la Tabla 5.3 se pueden visualizar las diagonales principales de cada uno de los modelos así como los respectivos valores de exactitud.

Se puede observar en la Tabla 5.3 que los resultados mejoran levemente respecto a los resultados obtenidos en la Tabla 5.2 a pesar de haber realizado los entrenamientos con un menor número de imágenes que en FER2013. En este sentido todos los modelos incrementan

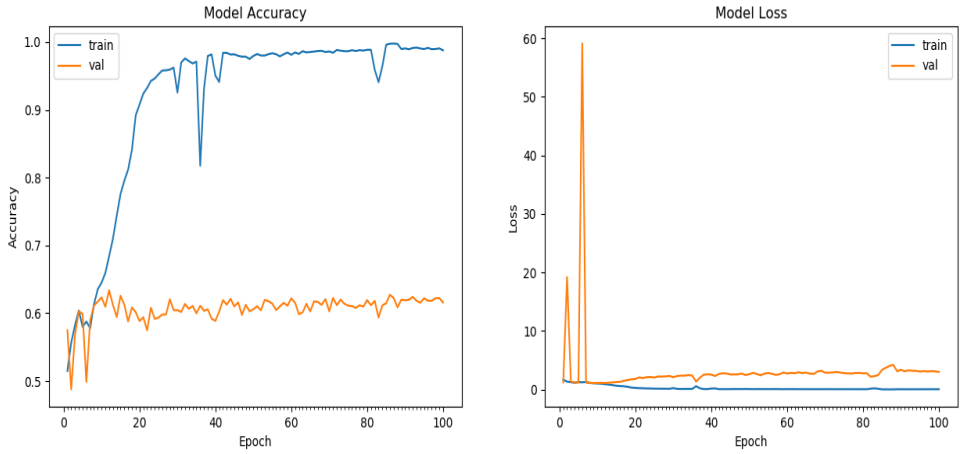
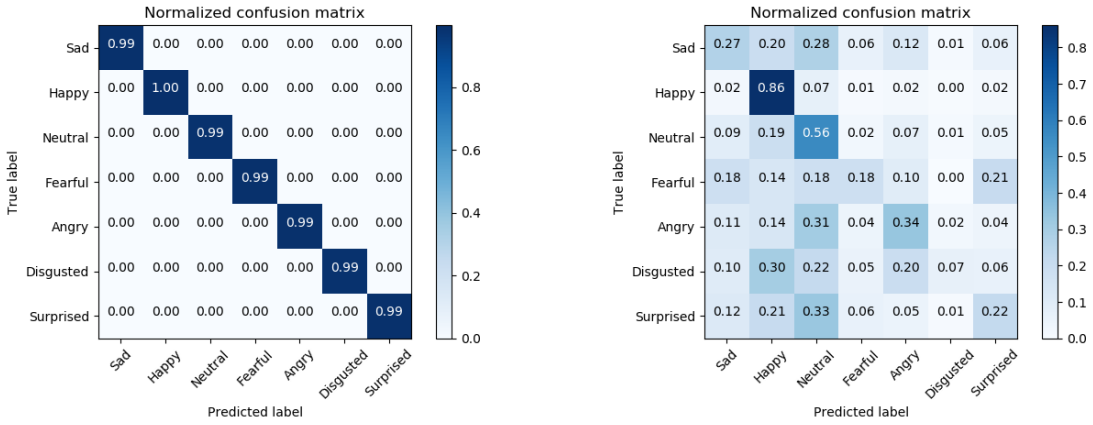


Figura 5.39: Accuracy vs loss red ResNet50, lr = 0.001.



(a) Entrenamiento. **(b)** Validación.

Figura 5.40: Matrices confusión red ResNet50, lr = 0.001.

DATASET AFFECTNET								
Modelos	Angry	Disgusted	Fearful	Happy	Sad	Surprised	Neutral	*Accuracy*
CNN4 lr = 0.001	0.41	0.09	0.22	0.86	0.30	0.27	0.72	66.60%
CNN8 lr = 0.001	0.37	0.14	0.18	0.90	0.47	0.29	0.57	66.35%
NASNetmobile lr = 0.001	0.52	0.04	0.06	0.79	0.13	0.26	0.42	55.10%
ResNet50 lr = 0.001	0.34	0.07	0.18	0.86	0.27	0.22	0.56	61.57%

Tabla 5.3: Tabla comparativa resultados redes neuronales convolucionales Affectnet.

levemente su porcentaje de aciertos respecto a los entrenamiento realizados con los otros dos datasets. También, cabe mencionar que el modelo con mayor porcentaje de acierto sigue siendo el modelo de 8 capas convolucionales mientras que el modelo con menor porcentaje de acierto continua siendo NASNetmobile. Además, las clases más reconocidas son feliz y neutro, mientras que, las clases con menor acierto del conjunto de prueba son asco, miedo y sorpresa que para los otros datasets proporcionaba un buen porcentaje de acierto. Estos resultados, son debidos a que los datos están desbalanceados y hay una gran presencia de imágenes de la clase feliz y neutro mientras que en el resto hay un menor número de imágenes.

5.4 Fer2013+Affectnet.

En esta sección, se mostrarán los resultados de os entrenamientos de los diferentes modelos a través del dataset conjunto de Fer2013 y Affectnet.

5.4.1 Red convolucional 4 capas, lr = 0.001.

Los resultados del entrenamiento se puede visualizar en la Figura 5.41.

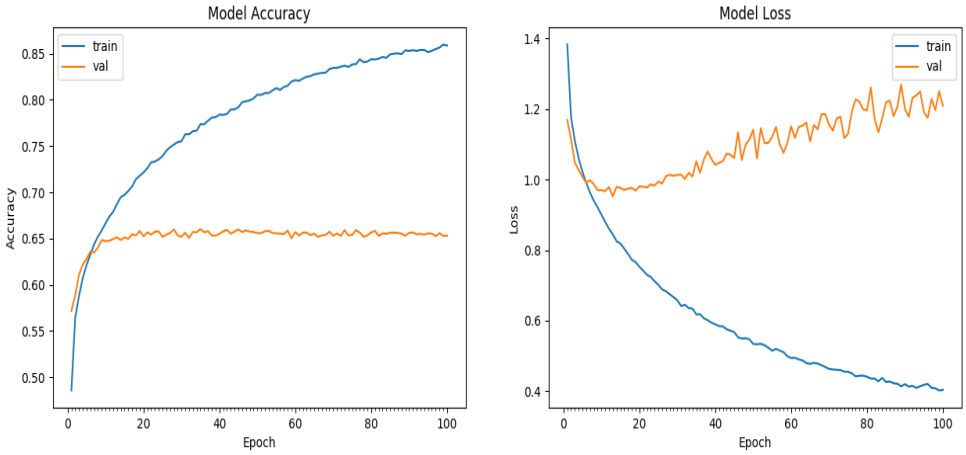


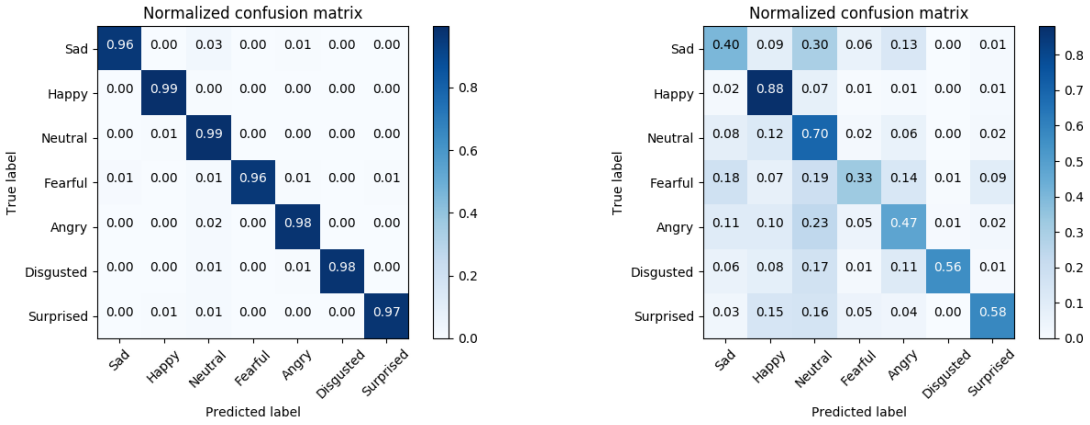
Figura 5.41: Accuracy vs loss red convolucional 4 capas, lr = 0.001.

A continuación en la siguiente Figura 5.42a se pueden visualizar los resultados del entrenamiento, se produce correctamente el entrenamiento y la mayoría de imágenes se encuentran en la diagonal principal de la matriz de confusión. También, se pueden visualizar los resultados obtenidos a través de las matrices de confusión para el conjunto de validación a través de la Figuraa 5.42b.

5.4.2 Red convolucional 8 capas, lr = 0.001.

Los resultados del entrenamiento se puede visualizar en la Figura 5.43.

A continuación en la siguiente Figura 5.44a se pueden visualizar los resultados del entrenamiento, como se puede observar el entrenamiento se ha realizado correctamente y la mayoría



(a) Entrenamiento. **(b)** Validación.

Figura 5.42: Matrices confusión red convolucional 4 capas, lr = 0.001.

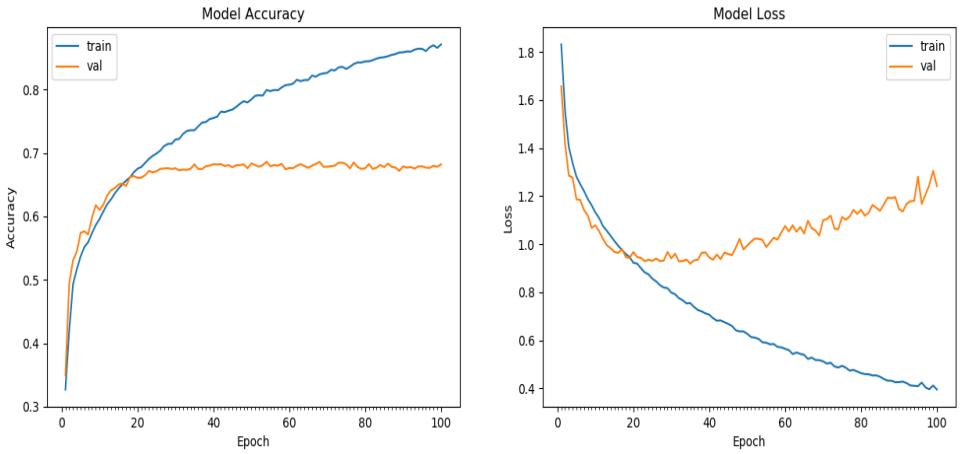


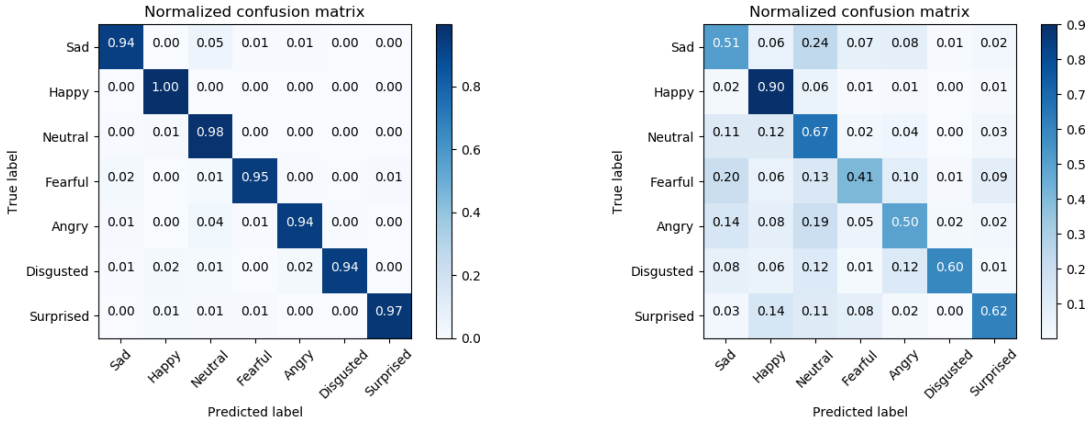
Figura 5.43: Accuracy vs loss red convolucional 8 capas, lr = 0.001.

de imágenes se encuentran en la diagonal principal de la matriz de confusión para el conjunto de entrenamiento. También, se pueden visualizar los resultados obtenidos a través de las matrices de confusión para el conjunto de validación a través de la Figura 5.44b.

5.4.3 NASNetmobile, lr = 0.001.

Los resultados del entrenamiento se puede visualizar en la Figura 5.45.

A continuación en la siguiente Figura 5.46a se pueden visualizar los resultados del entrenamiento, se produce correctamente el entrenamiento y la mayoría de imágenes se encuentran en la diagonal principal de la matriz de confusión. También, se pueden visualizar los resultados obtenidos a través de las matrices de confusión para el conjunto de validación a través de la Figura 5.46b.



(a) Entrenamiento. **(b)** Validación.

Figura 5.44: Matrices confusión red convolucional 8 capas, lr = 0.001.

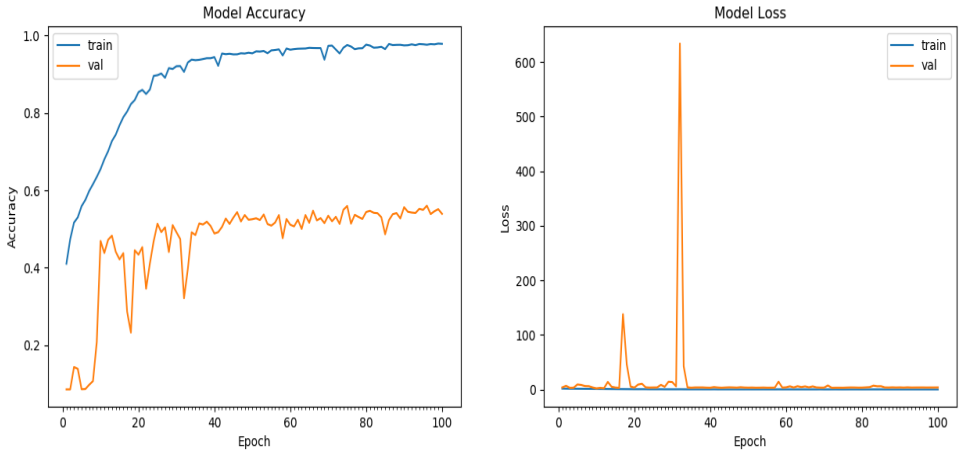
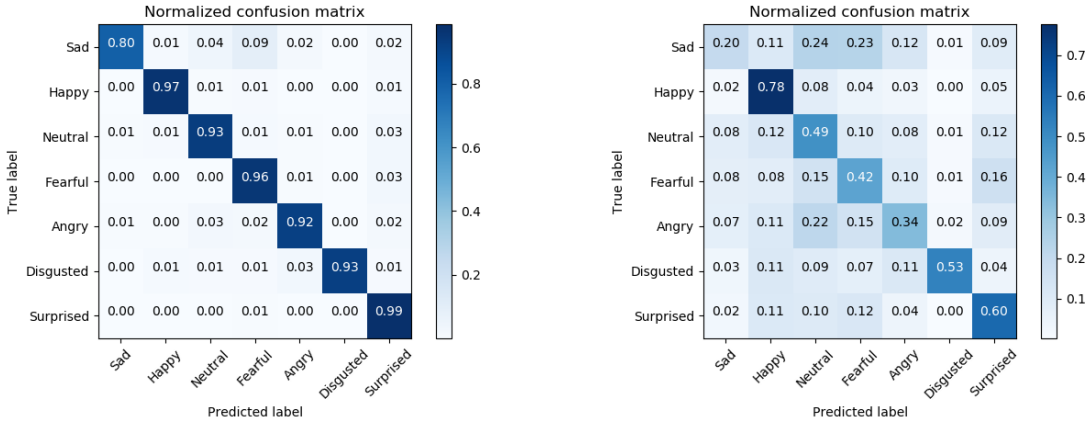


Figura 5.45: Accuracy vs loss red NASNetmobile, lr = 0.001.



(a) Entrenamiento. **(b)** Validación.

Figura 5.46: Matrices confusión red NASNetmobile, lr = 0.001.

5.4.4 ResNet50, lr = 0.001.

Los resultados del entrenamiento se puede visualizar en la Figura 5.47.

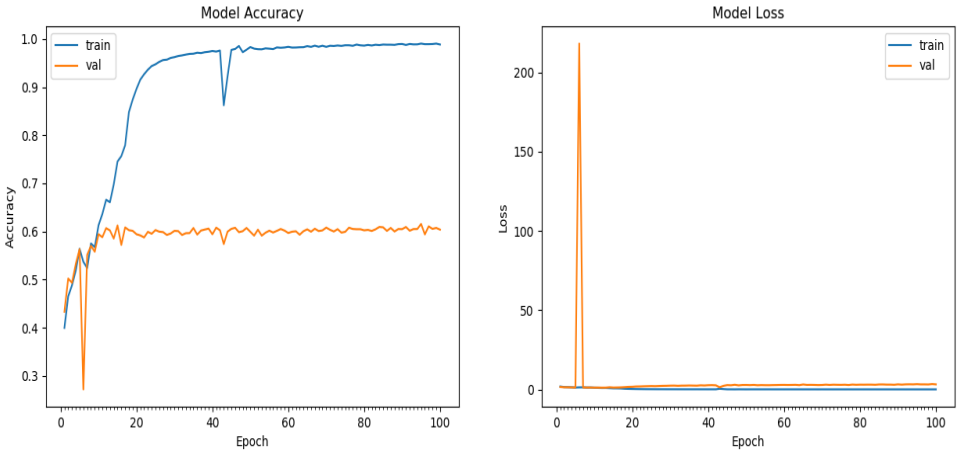
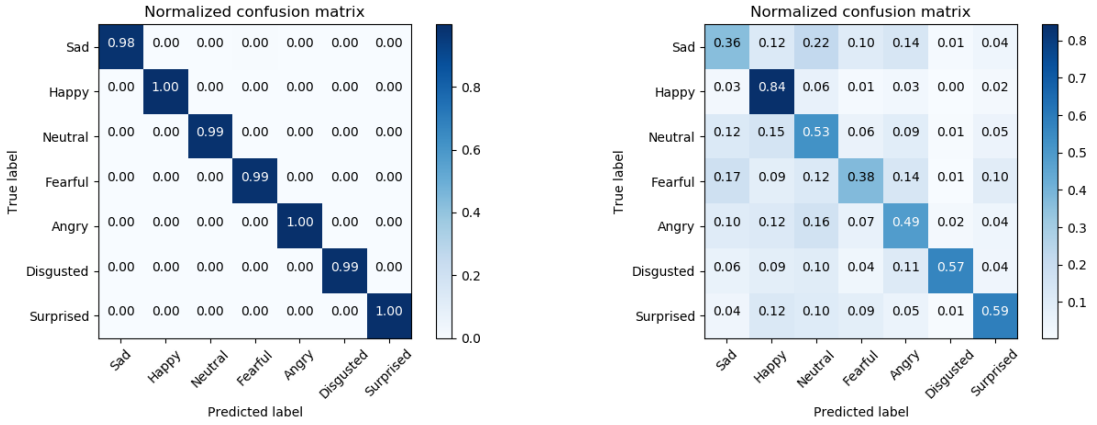


Figura 5.47: Accuracy vs loss red ResNet50, lr = 0.001.

A continuación en la siguiente Figura 5.48a se pueden visualizar los resultados del entrenamiento, como se puede observar el entrenamiento se ha realizado correctamente y la mayoría de imágenes se encuentran en la diagonal principal de la matriz de confusión para el conjunto de entrenamiento. También, se pueden visualizar los resultados obtenidos a través de las matrices de confusión para el conjunto de validación a través de la Figura 5.48b.



(a) Entrenamiento. **(b)** Validación.

Figura 5.48: Matrices confusión red ResNet50, lr = 0.001.

5.4.5 Tabla comparativa FER2013+Affectnet.

Seguidamente, en la Tabla 5.4 se pueden visualizar las diagonales principales de cada uno de los modelos así como los respectivos valores de exactitud.

DATASET FER2013+AFFECTNET								
Modelos	Angry	Disgusted	Fearful	Happy	Sad	Surprised	Neutral	*Accuracy*
CNN4 lr = 0.001	0.47	0.56	0.33	0.88	0.40	0.58	0.70	65.19%
CNN8 lr = 0.001	0.50	0.60	0.41	0.90	0.51	0.62	0.67	68.09%
NASNetmobile lr = 0.001	0.34	0.53	0.42	0.78	0.20	0.60	0.49	53.81%
ResNet50 lr = 0.001	0.34	0.53	0.42	0.78	0.20	0.60	0.49	60.26%

Tabla 5.4: Tabla comparativa resultados redes neuronales convolucionales FER2013+Affectnet.

A pesar del incremento en el número de muestras, los resultados siguen sin mejorar, se puede observar en la Tabla 5.4 que la red neuronal que sigue dando un mayor valor de exactitud es la red neuronal convolucional de 8 capas. Una de las posibles causas de esta ausencia de mejora podría ser la presencia de datos desbalanceados. En este sentido, el siguiente paso que se ha adoptado es el balanceo de los datos quitando muestras de las clases mayoritarias y añadiendo muestras a las minoritarias. También cabe destacar que las clases con mayor porcentaje de aciertos para los datos de prueba son la clase feliz y neutro debido a que presentan un mayor número de imágenes, por otro lado miedo, enfado y tristeza son las que presentan un menor porcentaje de aciertos debido al menor número de imágenes de esta clase respecto de otras.

5.5 Data augmentation.

A continuación, se ha realizado data augmentation de los datos obtenidos al juntar el dataset Fer2013 junto con Affectnet ya que es el que mejores resultados nos ha proporcionado, para ello se han cogido las imágenes de cada una de las clases y se han rotado -5º, 5º, -10º, 10º, -15º, 15º sucesivamente hasta completar la cantidad seleccionada.

En primer lugar se han igualado todas las clases de los datos de entrenamiento a 5000 imágenes. Se ha realizado un entrenamiento a través de las redes definidas anteriormente. A continuación, se han igualado todas las clases del conjunto de entrenamiento a 10000 imágenes en cada una de las clases.

5.5.1 Data augmentation 5000 imágenes.

En este apartado se realizarán los entrenamientos de los diversos modelos de redes neuronales convolucionales con 5000 imágenes en cada clase.

5.5.1.1 Red convolucional 4 capas, lr = 0.001.

Los resultados del entrenamiento se puede visualizar en la Figura 5.49.

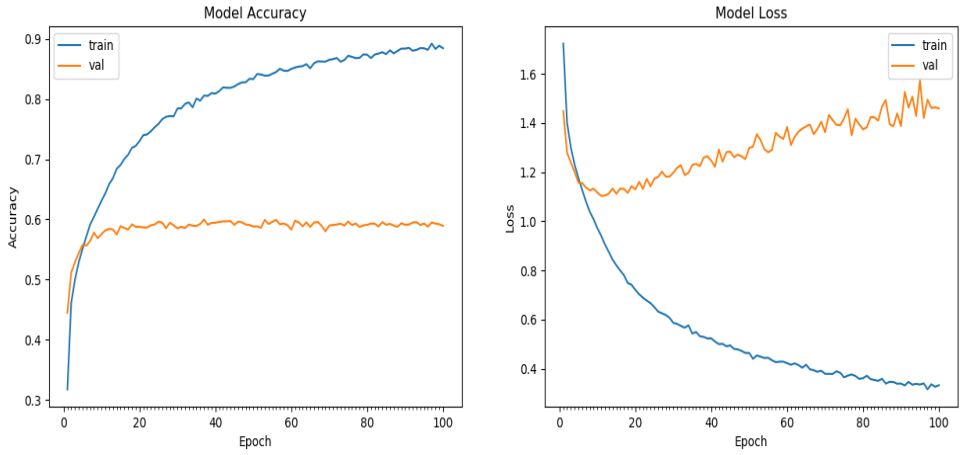
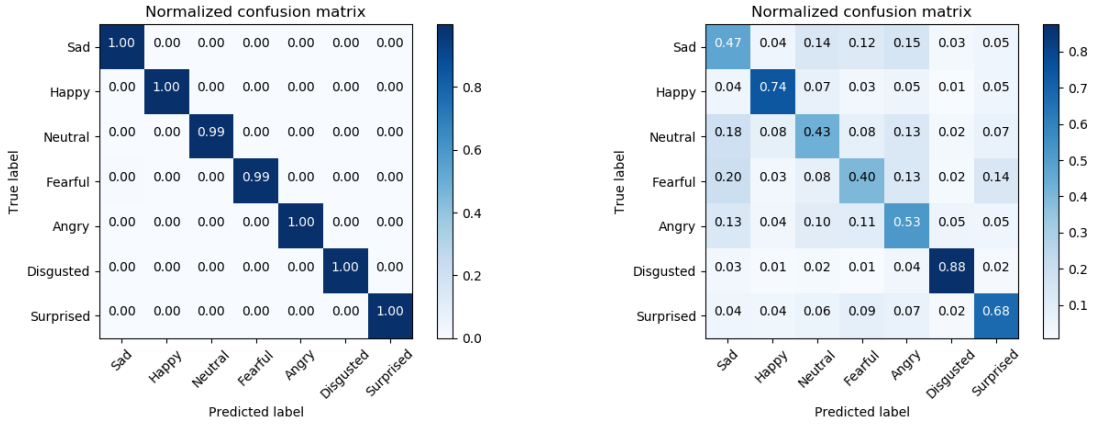


Figura 5.49: Accuracy vs loss red convolucional 4 capas, lr = 0.001.

A continuación en la siguiente Figura 5.50a se pueden visualizar los resultados del entrenamiento, como se puede observar el entrenamiento se ha realizado correctamente y la mayoría de imágenes se encuentran en la diagonal principal de la matriz de confusión para el conjunto de entrenamiento. También, se pueden visualizar los resultados obtenidos a través de las matrices de confusión para el conjunto de validación a través de la Figura 5.50b.



(a) Entrenamiento. **(b)** Validación.

Figura 5.50: Matrices confusión red convolucional 4 capas, lr = 0.001.

5.5.1.2 Red convolucional 8 capas, lr = 0.001.

Los resultados del entrenamiento se puede visualizar en la Figura 5.51.

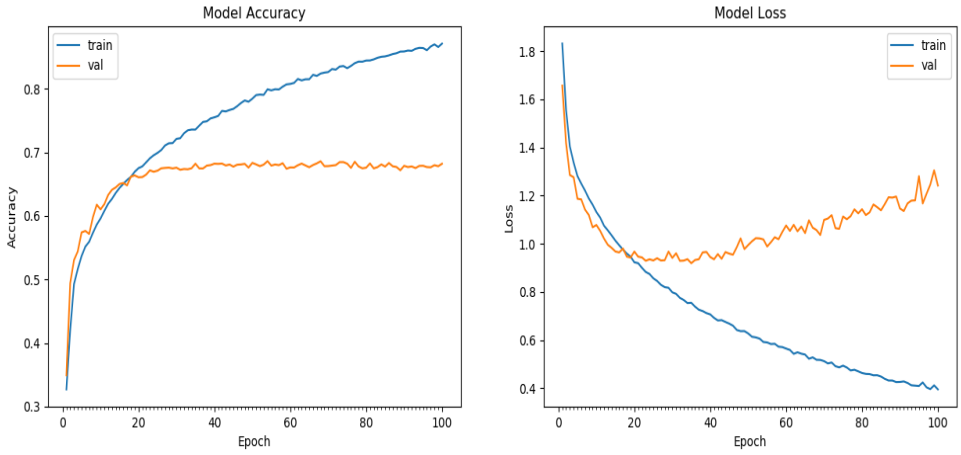
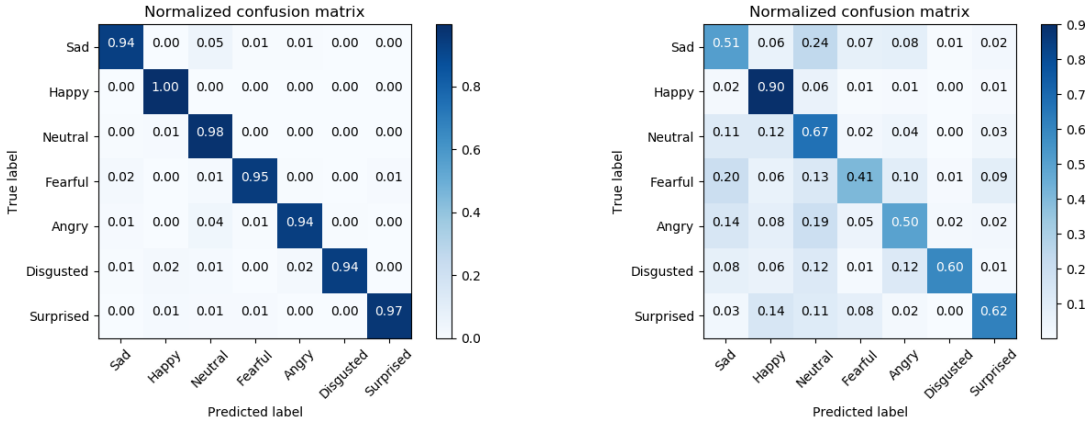


Figura 5.51: Accuracy vs loss red convolucional 8 capas, lr = 0.001.

A continuación en la siguiente Figura 5.52a se pueden visualizar los resultados del entrenamiento, como se puede observar el entrenamiento se ha realizado correctamente y la mayoría de imágenes se encuentran en la diagonal principal de la matriz de confusión para el conjunto de entrenamiento. También, se pueden visualizar los resultados obtenidos a través de las matrices de confusión para el conjunto de validación a través de la Figura 5.52b.



(a) Entrenamiento. **(b)** Validación.

Figura 5.52: Matrices confusión red convolucional 8 capas, lr = 0.001.

5.5.1.3 NASNetmobile, lr = 0.001.

Los resultados del entrenamiento se puede visualizar en la Figura 5.53.

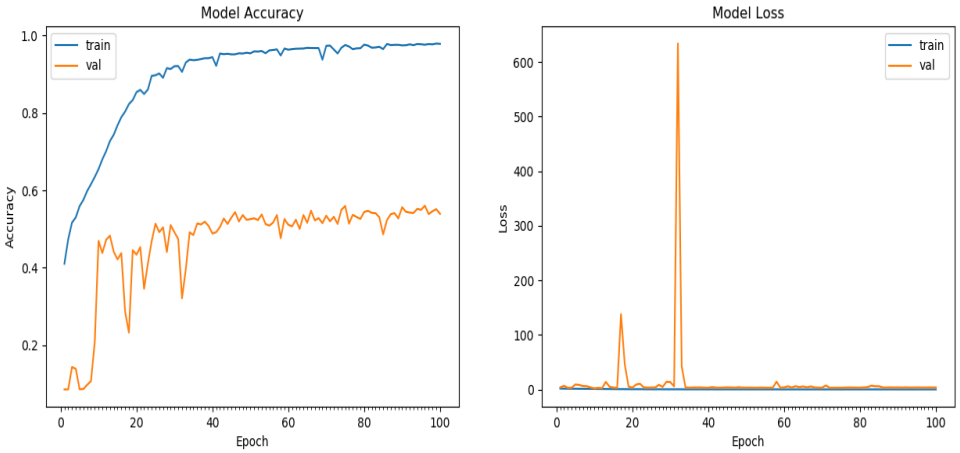
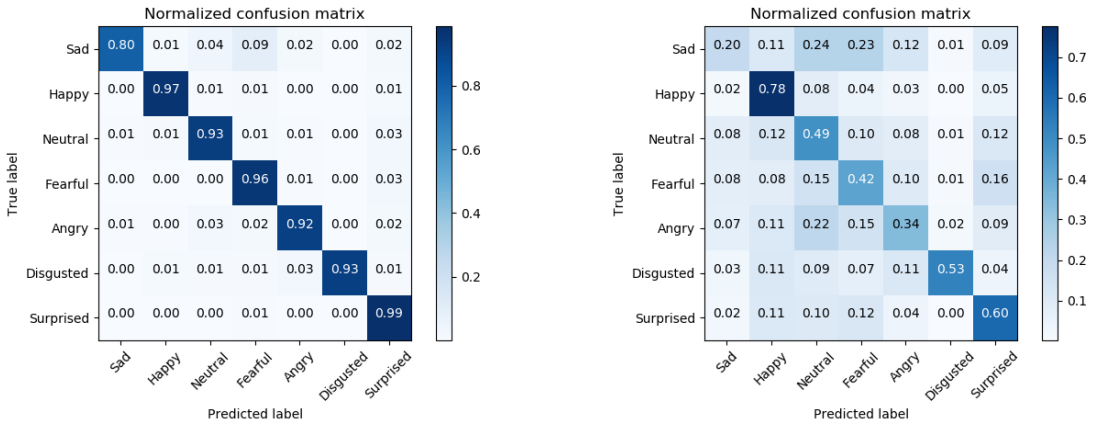


Figura 5.53: Accuracy vs loss NASNetmobile, lr = 0.001.

A continuación en la siguiente Figura 5.54a se pueden visualizar los resultados del entrenamiento, como se puede observar el entrenamiento se ha realizado correctamente y la mayoría de imágenes se encuentran en la diagonal principal de la matriz de confusión para el conjunto de entrenamiento. También, se pueden visualizar los resultados obtenidos a través de las matrices de confusión para el conjunto de validación a través de la Figura 5.54b.



(a) Entrenamiento. **(b)** Validación.

Figura 5.54: Matrices confusión NASNetmobile, lr = 0.001.

5.5.1.4 ResNet50, lr = 0.001.

Los resultados del entrenamiento se puede visualizar en la Figura 5.55.

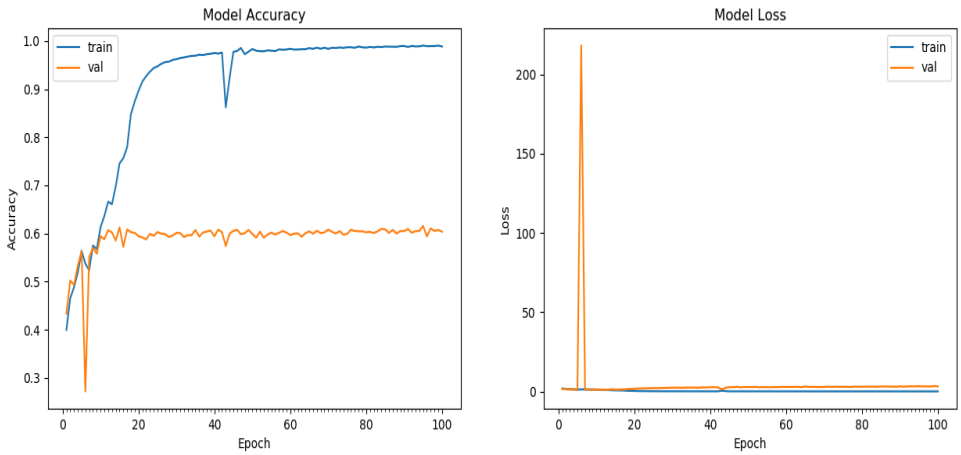
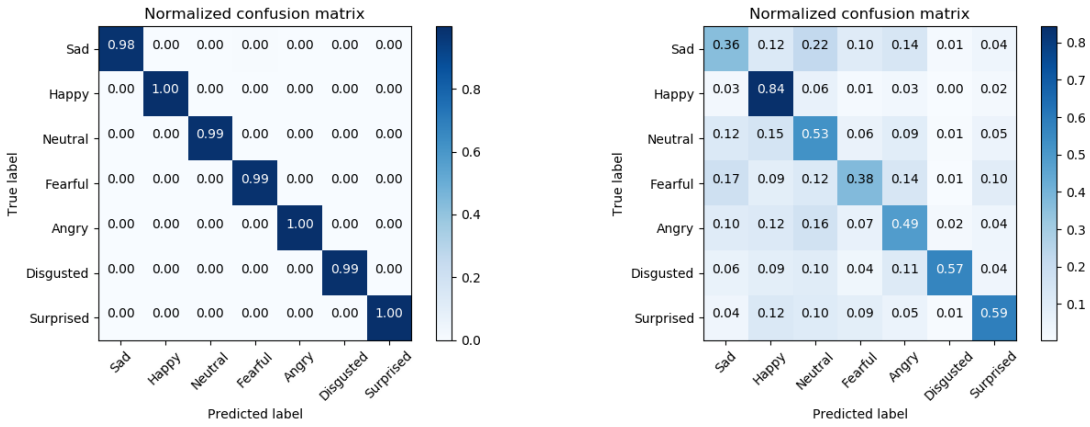


Figura 5.55: Accuracy vs loss ResNet50, lr = 0.001.

A continuación en la siguiente Figura 5.56a se pueden visualizar los resultados del entrenamiento, como se puede observar el entrenamiento se ha realizado correctamente y la mayoría de imágenes se encuentran en la diagonal principal de la matriz de confusión para el conjunto de entrenamiento. También, se pueden visualizar los resultados obtenidos a través de las matrices de confusión para el conjunto de validación a través de la Figura 5.56b.



(a) Entrenamiento. **(b)** Validación.

Figura 5.56: Matrices confusión ResNet50, lr = 0.001.

5.5.2 Tablas comparativas Data Augmentation.

Seguidamente, en la Tabla 5.5 se pueden visualizar las diagonales principales de cada uno de los modelos que han sido entrenados con 5000 imágenes por clase, así como los respectivos valores de exactitud.

DATASET FER2013+AFFECTNET DATA AUGMENTATION 5000								
Modelos	Angry	Disgusted	Fearful	Happy	Sad	Surprised	Neutral	*Accuracy*
CNN4 lr = 0.001	0.53	0.88	0.40	0.74	0.47	0.68	0.43	58.91%
CNN8 lr = 0.001	0.53	0.88	0.44	0.74	0.50	0.73	0.48	61.27%
NASNetmobile lr = 0.001	0.41	0.76	0.38	0.41	0.19	0.58	0.48	45.87%
ResNet50 lr = 0.001	0.41	0.86	0.38	0.68	0.38	0.55	0.42	52.29%

Tabla 5.5: Tabla comparativa resultados redes neuronales convolucionales FER2013+Affectnet Data Augmentation 5000.

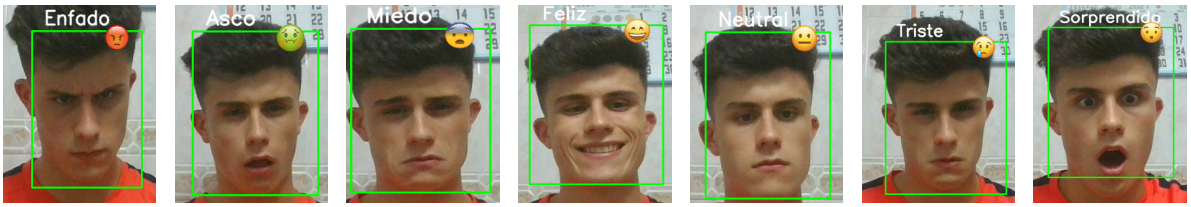
Como se puede observar, los mejores resultados se obtienen para la red convolucional de 8 capas, mientras que el peor porcentaje de precisión va destinado al modelo NASNetmobile. Debido al poco número de imágenes por clase la red no aprende tan bien como en resultados obtenidos anteriormente. También se puede observar como las clases que han obtenido un mayor porcentaje de aciertos para el conjunto de prueba serían asco, feliz y sorpresa, mientras que las que peor resultado obtienen son miedo, triste y neutro.

5.6 Aplicación.

En esta sección, se mostrarán los resultados obtenidos de la aplicación anteriormente explicada en el Apartado 4.6. El modelo de red neuronal convolucional utilizado para la aplicación será el que mayor exactitud nos ha proporcionado para el dataset FER2013+Affectnet, este es modelo es el modelo convolucional de 8 capas.

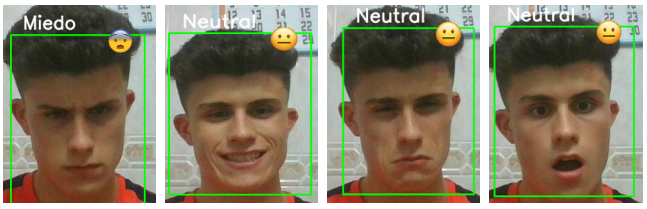
A continuación, en la siguiente Figura 5.57 se pueden visualizar algunas de las emociones correctamente detectadas. No obstante, el modelo a pesar de reconocer en muchos casos la emoción, también tienen un porcentaje de error. En este sentido, se puede observar en la Figura 5.58 las emociones detectadas incorrectamente.

Como se puede observar en la Figura 5.58 la aplicación detecta mal algunas emociones, entre ellas la clase enfado la ha clasificado como miedo, la clase feliz la clasifica en la segunda imagen con neutro, la clase triste la clasifica como neutro y finalmente la clase sorprendido la clasifica como neutro. A través de la Tabla 5.4 podemos observar que la clase Neutral es una de las más detectadas junto con felicidad y este es uno de los motivos por los cuales detecta neutro.



(a) Angry. **(b)** Disgusted. **(c)** Fearful. **(d)** Happy. **(e)** Neutral. **(f)** Sad. **(g)** Surprised.

Figura 5.57: Imágenes correctamente detectadas aplicación CNN 8 capas.



(a) Angry. **(b)** Happy. **(c)** Sad. **(d)** Surprised.

Figura 5.58: Imágenes incorrectamente detectadas aplicación CNN 8 capas.

6 Conclusiones.

En este TFG, se ha conseguido diseñar un sistema de reconocimiento de emociones a través del entrenamiento de diversos modelos de redes neuronales convolucionales para más tarde utilizarlos junto con la cámara del ordenador para predecir la emoción sentida por la persona situada frente a esta.

Cabe mencionar, la importancia de un buen dataset ya que tras las diversas pruebas realizadas se han obtenido unos mejores valores de exactitud empleando unos datasets con un mayor número de imágenes, este es el caso del dataset que combina FER2013 con Affectnet. No obstante, no siempre tener un gran número de imágenes es primordial en una tarea de reconocimiento de emociones, se necesita además que estas sean representativas de un gran número de personas de todas las etnias, edades y que estén bien clasificadas en las distintas clases.

En cuanto a los resultados obtenidos en los entrenamientos, se han obtenido mejores resultados para las redes neuronales convolucionales de diseño específico para la tarea que se pretendía que mediante las redes neuronales de las aplicaciones de Keras. En este sentido, el valor más alto de *accuracy*, se obtuvo para la red neuronal convolucional de 8 capas con un *accuracy* = 68.09%. Aunque estos resultados, no son del todo los mejores esperados, son suficientes para poder discernir entre algunas clases. Otro aspecto a tener en cuenta, es la necesidad de una buena elección del hiperparámetro *learning-rate* ya que un valor demasiado pequeño puede resultar en un proceso de entrenamiento largo que podría atascarse, mientras que un valor demasiado grande puede resultar en aprender un conjunto de pesos subóptimo demasiado rápido o un proceso de entrenamiento inestable.

También se ha realizado una aplicación que permite reconocer a través de modelos ya entrenados la emoción de la persona a través de su rostro. Se realizaron las pruebas de la simulación para las redes neuronales convolucionales de 4 capas y 8 capas, los resultados se pueden visualizar haciendo click en el siguiente enlace . A pesar de que los resultados obtenidos no permiten discernir entre todas las clases y obtener una exactitud del 100%, en parte debido a condiciones de iluminación o posición del rostro, el objetivo del proyecto ha sido salvado.

Finalmente, tras varios meses de trabajo y de continuos cambios en el planteamiento de como realizar los entrenamientos se ha conseguido entrenar modelos de redes neuronales convolucionales a través de *Deep Learning* (DL) y desarrollar una aplicación que permita emplear estos modelos para identificar diferentes emociones en los rostros humanos.

Bibliografía

[1] R. W. Picard, "Affetive computing.," 2000.

[2] F. Eyben and e. a. M. Wöllmer, "Emotion on the road—necessity, acceptance, and feasibility of affective computing in the car," *Advances in Human-Computer Interaction*, p. 10, 2010.

[3] E. Cambria, "Affective computing and sentiment analysis," *IEEE Intelligent Systems*, vol. 31, no. 2, pp. 102—107, 2016.

[4] E. Cambria, B. Schuller, Y. Xia, and C. Havasi, "New avenues in opinion mining and sentiment analysis.," *IEEE Intelligent Systems*, vol. 28, no. 2, pp. 15–21, 2013.

[5] H. Tao and T. Huang, "A piecewise bezier volume deformation model and its applications in facial motion capture, in advances in image processing and understanding: A festschrift.," *The Journal of the Acoustical Society of America*, 2002.

[6] P. Ekman and W. Friesen, "Manual for the facial actioncoding system.," *Consulting Psychologists Press.*, 1977.

[7] I. Cohen, N. Sebe, A. Garg, L. S. Chen, and T. S. Huang, "Facial expression recognition from video sequences: Temporal and static modeling.," *Computer Vision and Image Understanding*, no. 91, pp. 160–187, 2003.

[8] "Opencv.." https://opencv.org/.

[9] "Dlib.." http://dlib.net/.

[10] "Tensorflow.." https://www.tensorflow.org/?hl=es-419.

[11] "Keras.." https://keras.io/.

[12] "Affectnet.." http://mohammadmahoor.com/affectnet/.

[13] "Ascertain.." http://mhug.disi.unitn.it/wp-content/ASCERTAIN/ascertain.html/.

[14] "Dreamer dataset.." https://zenodo.org/record/546113.YNn9pXUzZuR.

[15] "Conjunto de datos extendido de cohn-kanade (ck +).." https://www.kaggle.com/c/visum-facial-expression-analysis%5C.

[16] "Emotico.." http://sunai.uoc.edu/emotic/index.html.

[17] "Fer-2013.." https://www.kaggle.com/deadskull7/fer2013.

[18] "Conjunto de datos de comparación de expresiones faciales de google.." https://research.google/tools/datasets/google-facial-expression/.

[19] "K-emocon.." https://zenodo.org/record/3762962.YNn3vHUzZuR.

[20] X. Glorot and Y. Bengio, "«understanding the difficulty of training deep feedforward neural networks.," *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS'10)*, 2010.

[21] F. P. Navarrete, "Detección de emociones / deep learning." 21 abril de 2019, abril.

[22] N. Sharma, "Facial emotion recognition on fer2013 dataset using a convolutional neural network." 7 noviembre de 2018, noviembre.

[23] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors.," vol. 323, no. 6088, pp. 533–536, 1986.

Lista de Acrónimos y Abreviaturas

CK	Cohn-Kanade.
CNN	Convolutional Neural Network.
CV	Computer Vision.
ECG	Electrocardiograma.
EEG	Electroencefalograma.
FACS	Facial Action Coding System.
GSR	Galvanic Skin Response.
HMM	Hidden Markov Model.
IA	Inteligencia Atificial.
IoT	Internet of Things.
NAS	Neural Arquitecture Research.
NLP	Natural Language Preocess.
OpenCV	Open Source Computer Vision.
RNA	Red Neuronal Artificial.
SVM	Support Vector Machine.
VN	Verdadero Negativo.
VP	Verdadero Positivo.