UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH

Reconocimiento de expresiones faciales mediante el uso de redes neuronales convolucionales

Trabajo de Fin de Grado

Presentado en la

Escola Tècnica d'Enginyeria de Telecomunicació de Barcelona

Universitat Politècnica de Catalunya

por

Patricia Sereno Rodríguez

En cumplimiento parcial

de los requisitos para el grado en

CIENCIAS Y TECNOLOGIAS DE LAS TELECOMUNICACIONS

Supervisor: Josep Ramon Morros Rubió

Barcelona, Enero 2017

Abstract

In recent years has appeared the need to be able to classify a person's facial expression automatically, that is, without making use of human intervention.

For that task, this project is based on the research and development of multiple systems capable of recognizing these expressions and to obtain a certain precision in the classification, always with the aim of trying to match the current state of the art.

Throughout this work, the convolutional neural networks (CNN) is used to extract the characteristics of different images (the different expressions) and then the consequent integration of different classifiers such as the Softmax or the Support Vector Machine (SVM) with the objective to assign a certain probability that the image belongs to one emotion or another.

During the course of the project a comparison is made of how the different architectures of the system as well as the parameters that define it affect the accuracy of classification.

Each system will be evaluated in a set of different databases such as Radboud and Umea database for still images and Cohn Kanade Plus for sequences.

Resum

En els últims anys ha aparegut la necessitat de ser capaços de classificar les expressions facials d'una persona de manera automàtica, és a dir, sense haver de fer ús de la intervenció humana.

Es per aquest motiu que aquest projecte es basa en la investigació i el desenvolupament de múltiples sistemes capaços de reconèixer aquestes expressions i obtenir d'aquesta forma una determinada precisió en la classificació, sempre sota la premissa d'intentar apropar-nos a l'actual estat de l'art.

Al llarg d'aquest treball es fa un de les xarxes neuronals convolucionals (CNN) per extreure les característiques de diferents imatges (que reflexen les diferents expressions) y després la conseqüent integració de diferents tècniques de classificació com son la funció Softmax o el Support Vector Machine (SVM) amb l'objectiu d'assignar una determinada probabilitat de que aquesta imatge expressi una emoció o una altre.

En l'esdevenir del projecte es realitza una comparativa de com afecten les diferents arquitectures del sistema i els paràmetres que el defineixen.

Cada sistema serà avaluat sota diferents bases de dades com son la Radboud i la Umea per imatges fixes i la Cohn Kanade Plus per seqüencies.

Resumen

En los últimos años ha aparecido la necesidad de ser capaz de clasificar las expresiones faciales de una persona de forma automática, es decir, sin hacer uso de la intervención humana.

Para ello, este proyecto se basa en la investigación y desarrollo de múltiples sistemas capaces de reconocer estas expresiones obteniendo una determinada precisión en la clasificación, siempre con el objetivo de intentar igualar el estado del arte actual.

A lo largo de este trabajo se hace uso de las redes neuronales convolucionales (CNN) para extraer las características de diferentes imágenes (las diferentes expresiones) y luego la consiguiente integración de diferentes sistemas de clasificación como son el Softmax o el Support Vector Classifier (SVC) con tal de asignar una determinada probabilidad de que esa imagen pertenezca a una emoción u otra.

Durante el transcurso del proyecto se realiza una comparativa de cómo afectan las diferentes arquitecturas del sistema así como los parámetros que lo definen.

Cada sistema será evaluado en un set de diferentes bases de datos como son la Radboud y la Umea database para imágenes fijas y la Cohn Kanade Plus para secuencias.

Agradecimientos

Primero de todo me gustaría agradecer a mi tutor, Josep Ramon Morros ya que me ha hecho de guía a lo largo de todo el proyecto y me ha brindado todo el apoyo necesario durante estos meses de trabajo. Él es una pieza fundamental puesto que ha sido el encargado de estar semana tras semana pendiente del proyecto para que este pudiera crecer.

También agradecer a Albert Gil y a Josep Pujal por el fantástico servició técnico en todos los problemas relacionados con los servidores de la escuela y su rápida actuación frente a los problemas surgidos.

Me gustaría agradecer también a todos aquellos docentes que me han estado apoyando durante esta etapa de mi vida y por la confianza que me han dado para cumplir mi meta.

Agradecer a mis amigos, en especial a Carlos Aguilera Martínez y a David Rodríguez Castellón por haberme apoyado, aguantado y comprendido a lo largo de todos estos años.

Finalmente agradecer a mi familia por el voto de confianza que me han dado y por apoyarme en cada paso de mi vida y en especial en esta última etapa de la carrera.

Historial de revisiones y registro de aprobaciones

Revisión	Fecha	Propósito
0	02/01/2017	Creación del documento
1	09/01/2017	Agregación del contenido
2	10/01/2017	Revisión del documento
3	11/01/2017	Revisión del documento
4	13/01/2017	Revisión del documento

LISTA DE DISTRIBUCIÓN DEL DOCUMENTO

Name	e-mail
Patricia Sereno Rodríguez	Patricia_sr92@hotmail.com
Josep Ramon Morros Rubió	Ramón.morros@upc.edu

Escrito por:		Revisado y aprobado por:	
Fecha	02/01/2017	Fecha	10/01/2017
Nombre	Patricia Sereno Rodríguez	Nombre	Josep Ramon Morros Rubió
Posición	Autor del proyecto	Posición	Supervisor del proyecto

TABLA DE CONTENIDO

Lista de Figuras

Lista de Tablas:

1. Introducción

Desde hace años existe la necesidad de ser capaz de clasificar las expresiones faciales de una persona de forma automática, es decir, sin hacer uso de la intervención humana. Una aplicación cada vez más frecuente es la de dictaminar la emoción que está experimentado una persona haciendo uso de un producto o recibiendo un servicio.

El estudio de técnicas o sistemas para realizar este tipo de tareas ha crecido mucho en estos últimos años gracias a la aparición del *Deep Learning* (DL) y en especial a las Redes Neuronales Convolucionales (CNN – *Convolutional Neural Networks*) para temas de clasificación mediante imagenes.

Por eso a lo largo de este proyecto nos vamos a centrar en desarrollar un sistema capaz de reconocer emociones pero aplicando técnicas de *Deep Learning*.

1.1. Objetivos del proyecto

El objetivo del proyecto es explorar diferentes sistemas basados en arquitecturas que hagan uso de las redes neuronales convolucionales con tal de realizar la tarea de clasificación de expresiones faciales.

El trabajo se centrará en identificar a que clase pertenece cada imagen o secuencia de nuestras bases de datos de las 7 posibles (tristeza, enfado, asco, miedo, neutral, contento y sorpresa).

Para realizar esta tarea se seguirán las siguientes etapas reflejadas en el diagrama de bloques:

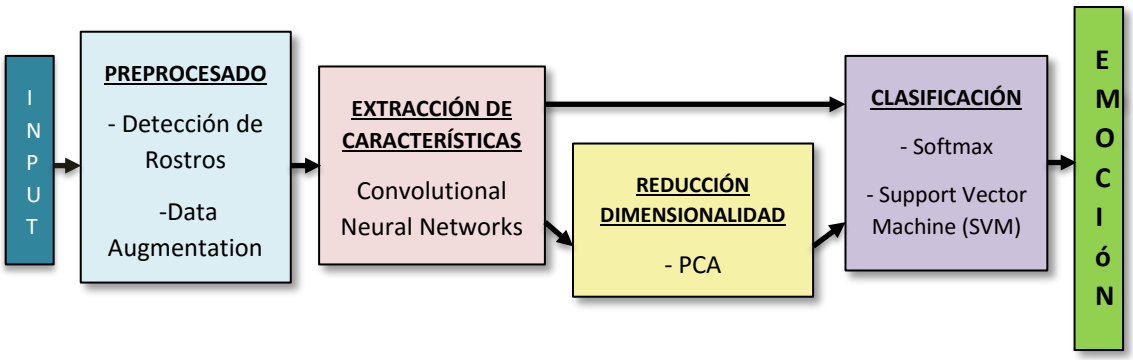

Figura 1 Diagrama de bloques del proyecto

1. Preprocesado: Para cada imagen y *frame* de cada secuencia se realizará un algoritmo de detección de rostros basado en OpenCV. Después se procederá a realizar una técnica de *Data Augmentation* para hacer el sistema más robusto y poder trabajar con un *dataset* más grande a la hora de entrenar nuestro sistema de extracción de características

2. Extracción de características: Se extraerá el vector de características que define cada imagen. Para ello pasaremos cada rostro por un sistema que estará basado en el modelo VGGFace hecho por el *Visual Geometry Group* de la Universidad de Oxford. El cual se desarrolló para hacer el reconocimiento facial (identificar personas) con un *dataset* de gran escala (2.6M de imágenes, 2.6k personas).

3. Reducción de la dimensionalidad: Una vez se tengan el vector de características tendremos dos opciones proceder a la etapa de clasificación directamente o reducir la dimensión de este (hacerlo menos *sparse*) para "eliminar características" que no aportan información relevante a la hora de clasificar.

4. Clasificación: Una vez tengamos el vector de características procederemos a obtener un score o probabilidad para cada emoción. Para ello probaremos dos clasificadores distintos el Softmax y el SVM.

1.2. Requisitos y especificaciones

Requisitos del proyecto:

- Identificar con cierta precisión (cerca del estado actual del arte) las expresiones faciales de una persona en una imagen usando los diferentes sistemas de reconocimientos de expresiones faciales (FER – *Facial Expression Recognition*).
- Identificar con cierta precisión (cerca del estado actual del arte) las expresiones faciales de una persona en una secuencia de video usando los diferentes sistemas de reconocimientos de expresiones faciales.
- Conseguir implementar diferentes algoritmos de clasificación.
- Realizar alteraciones en la red modificando los parámetros que la definen para obtener el mejor sistema posible.

Especificaciones del proyecto:

- Implementación completa del proyecto realizada en Python sobre el Framework de Keras con la librería de Google Tensorflow como *Backend* [11].
- Uso de CNN para extraer características
- Hacer uso del *Transfer Learning*, es decir, reutilizar redes ya entrenadas.
- Para la detección de rostros se ha usado la librería de OpenCV.
- Scikit-learn es librería que nos ofrece herramientas para el análisis y procesado de la información como es el algoritmo de PCA y la implementación de algoritmos de clasificación como el SVM.
- Programado en el IDE de Pycharm y ejecutado en el servidor del grupo de Procesado de imagen de la UPC.
- Bases de datos utilizadas: Radboud y Jaffe para el entrenamiento. Radboud, CK+ y Umea para el entreno.

1.3. Plan de trabajo

Durante el proyecto se ha intentado ser los más fiel posible a la planificación que se estipulo al inicio de este en los documentos de *Project proposal plan* y *critical review.*

Por lo general se ha seguido la planificación establecida exceptuando algún reajuste en la finalización de las tareas.

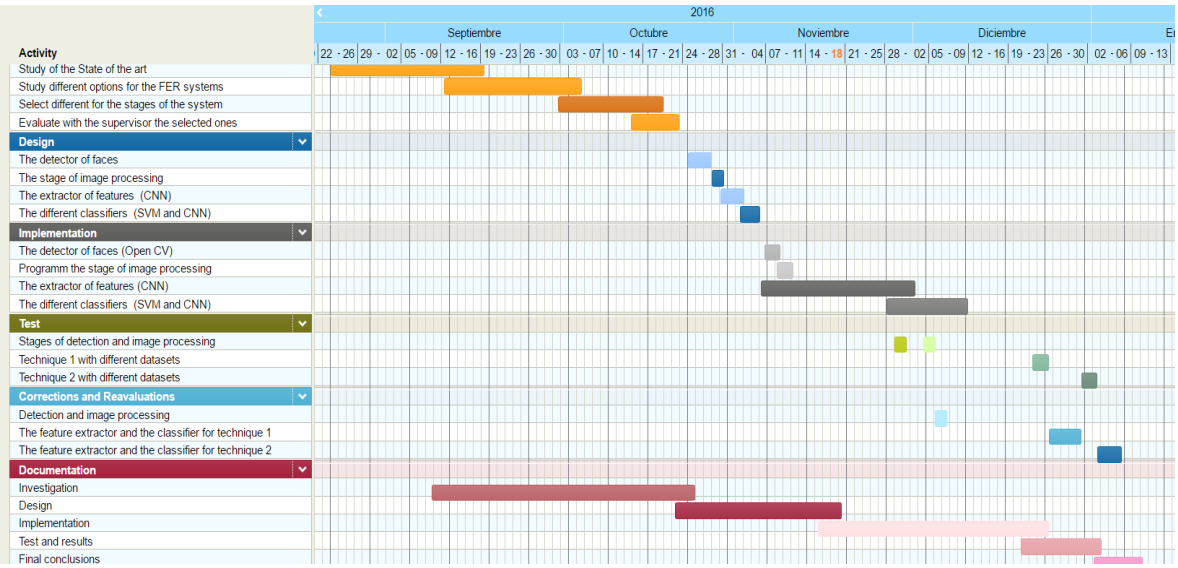

Figura 2 Diagrama de Gantt

A continuación se describen los diferentes bloques del diagrama de Gantt:

Bloque: Investigación
Descripción: Estudiar los sistemas existentes y las técnicas que utilizan para el reconocimiento de expresiones faciales.
Bloque: Diseño
Descripción: Diseñar las diferentes partes del sistema.
Bloque: Implementación
Descripción: Implementar los diferentes diseños del sistema.
Bloque: Test
Descripción: Probar los sistemas en las diferentes bases de datos y comparar resultados.
Bloque: Correcciones y reevaluaciones
Descripción: Corregir los sistemas ajustando los parámetros y volver a reevaluarlo.

2. Background y estado del arte:

2.1. Background

A continuación se hará un breve resumen sobre las diferentes herramientas que se han utilizado para llevar a cabo el sistema de reconocimiento.

2.1.1. Redes Neuronales Convolucionales

Una red neuronal convolucional (CNN) o también conocidas como convnets es un tipo de red neuronal.

Donde una red neuronal es un sistema de aprendizaje automático inspirado en el sistema nervioso biológico. Este sistema se crea a partir de la interconexión de diferentes elementos inteligentes e independientes llamados neuronas.

Estos sistemas siguen una estructura por capas (input layer, hidden layers y output layers), con diferente número de neuronas por cada capa. Estas interconexiones entre neuronas están ponderadas por unos pesos (ponderaciones) los cuales se irán ajustando en la etapa de entrenamiento con el objetivo de minimizar una determinada función de coste. Estos pesos (w_0,w_1, …) se multiplican por la salida de la neurona anterior (x_0, x_1,…) y luego se procede a aplicar un sumador de todas las entradas de la neurona para acabar calculando una determinada función de activación, la cual no es lineal.

Este comportamiento se puede ver en la siguiente figura, extraída del curso de la Universidad de *Stanford Convolutional Neural Networks for Visual Recognition:*

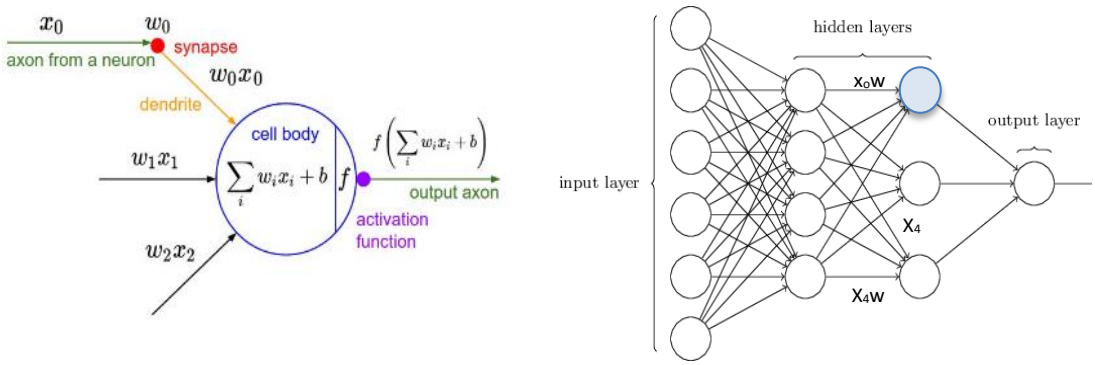

Figura 3 Neurona (izquierda) y Estructura de una Red Neuronal (derecha)

Una red neuronal convolucional es una red neuronal multicapa donde su arquitectura esta optimizada para trabajar con entradas de más de una dimensión, como es el caso de las imágenes las cuales son entradas 2D. El nombre de convolucional le viene puesto que ahora nuestros pesos no serán escalares sino que serán matrices n-dimensionales, también conocidas como filtros mediante los cuales se aplicará una convolución en la entrada para obtener una salida.

Este tipo de red tiene el objetivo de extraer información de alto nivel de las imágenes de entrada para finalmente clasificarlas o extraer los vectores de características. Para realizar todo este proceso, las diferentes capas que conforman la red neuronal convolucional se encargan de procesar la imagen de entrada de manera que la salida de la última capa se obtenga el resultado. Las capas están organizadas de manera que cada una de ellas procesa la salida del anterior excepto la primera capa que procesa directamente la imagen.

2.1.1.1. Tipos de capas

En una red convolucional existen diferentes tipos de capas pero en este apartado vamos hacer una breve explicación de las capas que se han usado para montar nuestra red.

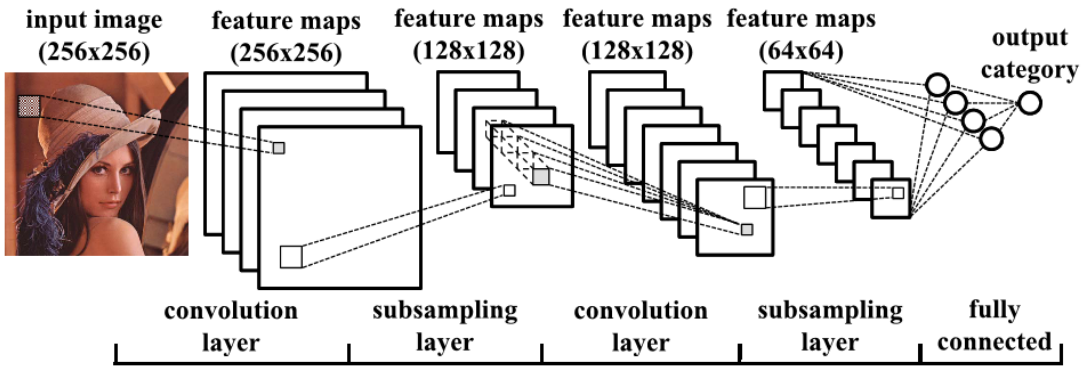

Figura 4: Ejemplo de Red Neuronal Convolucional

- Capa Convolucional

  Es la capa básica que constituye una CNN.

  Los parámetros de esta capa son un conjunto de filtros (las matrices de pesos) que realizan una convolución sobre la información de entrada. De modo que si tenemos K filtros de dimensiones n_{filas} x $n_{columnas}$ x $r_{profundidad}$ que operan sobre unos datos de entrada con dimensión m_{filas} x $m_{columnas}$ x $r_{profundidad}$ obtendremos una salida con dimensiones (m_{filas}-n_{filas} + 1) x ($m_{columnas}$-$n_{columnas}$ + 1) x K.

  Uno de los parámetros de este tipo de capas es el paso de convolución (*stride*), que es equivalente aplicar un diezmado a la salida de la convolución.

  Como vemos esta capa aumenta la profundidad de la salida.

- Pooling o subsampling

  Esta capa fue creada por las dificultades de cálculo cuando la entrada de la red tiene demasiada información. Se encarga de reducir en parámetros la información de su entrada aplicando una operación llamada *pooling*. El tipo de *pooling* más utilizado es el llamado *Max-Pooling* que, de entre diferentes valores de píxeles escoge el de valor mayor como salida, eliminando los restantes.

  Lo que hace es una reducción espacial de la imagen.

- **Rectified Linear Unit (ReLU)**

  Es una función de activación que nos permite normalizar la entrada (*stack* de imágenes filtradas + *maxpooling*). Para ello hace el máximo entre el valor de cada píxel de la entrada y 0, eliminando así cualquier valor negativo que se haya podido crear en las operaciones anteriores.

  Responde a la siguiente ecuación:
  $\boldsymbol{f(x) = \max(0, x)}$

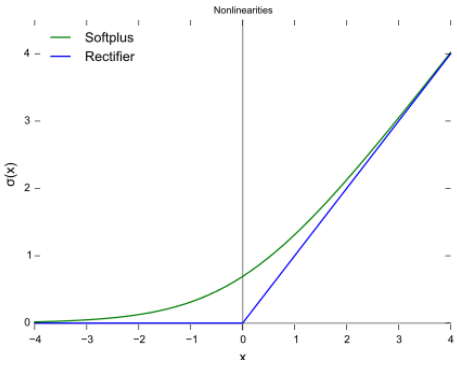

Figura 5 Gráfica ReLU.

- **Fully-Connected (FC)**

  Mediante una convolución 1x1 este tipo de capas se encargan de generar una salida mono-dimensional. Normalmente se posiciona en las últimas posiciones y se encargan de realizar clasificación o razonamientos de alto nivel puesto que todas las neuronas de esa capa con la anterior están conectadas. Este tipo de capas no dejan de ser un caso particular de capas convoluciones. En estás existe un parámetro de gran utilidad, el *Dropout*, que determina con que probabilidad va a existir la conexión entre dos neuronas. Este parámetro permite reducir el *overfitting*, o sobreentrenamiento, de una red [8].

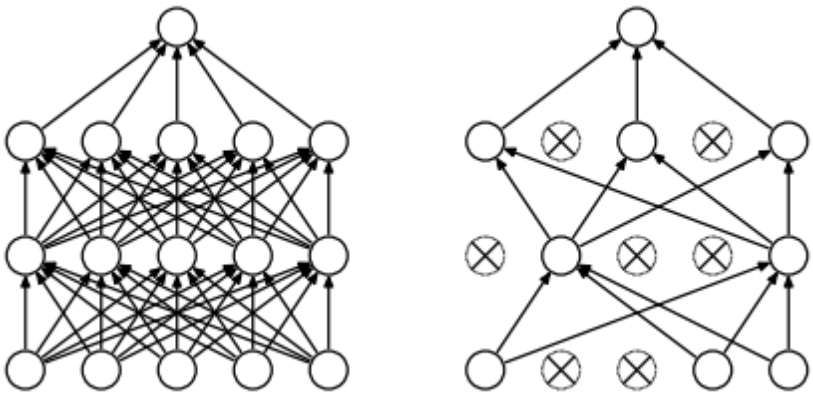

Figura 6: Red neuronal con 2 Hidden Layers vs Red Neuronal con 2 Hidden Layers y Dropout activo.

- **Upsampling**

  Operación inversa al Pooling.

2.1.2. Entrenamiento

Para determinar los parámetros de los filtros de las capas se debe realizar un entrenamiento de la red sobre un conjunto de imágenes, previamente etiquetadas, destinadas al entrenamiento. El entrenamiento se realiza mediante el proceso de *backpropagation* donde para cada imagen de entrada se computa el error que se ha producido en la clasificación y se actualizan los parámetros de los filtros para minimizarlo. Generalmente se usa algún tipo de *Gradient Descent*.

Un parámetro muy importante a la hora de entrenar la red es el algoritmo de optimización a utilizar para el cálculo de los pesos de los filtros. Hay muchas formas diferentes de entrenar la red pero es necesario elegir una buena técnica de cálculo dado al gran volumen de computación que nuestra red necesitará para acabar su entreno. Y además que según el tipo de dataset que tengamos existen ciertos optimizadores que podrán alcanzar un valor de error de clasificación menor.

Dos de los más importantes son el *Stochastic Gradient Descent* y el *Adaptative Moment Estimation* de los cuales se hará una breve explicación.

2.1.2.1. Stochastic Gradient Descent SGD)

El Stochastic Gradient Descent también conocido como gradiente de descenso incremental, es una aproximación estocástica del método de optimización gradiente descendiente usado para minimizar una función objetivo que se escribe como una suma de funciones diferenciables. En otras palabras, SGD trata de encontrar mínimos o máximos por iteración.

El gradiente nos indica la dirección en la que la función tiene la tasa de aumento más pronunciada pero no nos aporta información de hasta donde se debe avanzar en esa dirección. Por eso existe el *step size,* también llamado *learning rate* que nos dice el tamaño de casa paso, es decir, la distancia que recorrerá en cada iteración en la dirección que del gradiente.

Pese que es un optimizador que suele funcionar bien en la mayor parte de los casos, la necesidad de poder fijar un *learnig rate* ha hecho que se estudien otras propuestas ya que el SGD tiene problemas de divergencia si el parámetro es muy elevado y de convergencia si este es muy pequeño.

2.1.2.2. Adaptative Moment Estimation (ADAM)

Como hablábamos en el apartado anterior el SGD tiene problemas puesto que el learning rate (LR) está fijado y fijarlo mal puede a malas optimizaciones.

Para ello existen otros optimizadores que adaptan el rate en función de cómo estén distribuidos los parámetros. Si los parámetros están muy dispersos (*sparse)* el LR aumentará.

En este caso ADAM es una actualización del Optimizador RMSProp (Root Mean Square Propagation) que se basa en LR adaptativo.

2.1.3. Principal Component Analisys (PCA)

Una vez tenemos nuestro vector de características, la salida de la CNN, podemos reducir su dimensionalidad ya sea con el objetivo de disminuir el coste computacional o para eliminar información irrelevante.

Para ello usamos la técnica de análisis de componentes principales (PCA) [21] la cual nos permite reducir el número de dimensiones de un conjunto de datos determinado.

El PCA busca la proyección según la cual los datos queden representados en términos de mínimos cuadrados. Esta convierte un conjunto de observaciones de variables posiblemente correlacionadas en un conjunto de valores de variables sin correlación lineal llamadas componentes principales.

Es decir, el PCA construye una transformación lineal tal que escoge un nuevo sistema de coordenadas para el conjunto original de datos en el cual la varianza de mayor tamaño del conjunto de datos es capturada en el primer eje (primer componente principal), la segunda varianza más grande es el segundo eje, y así sucesivamente.

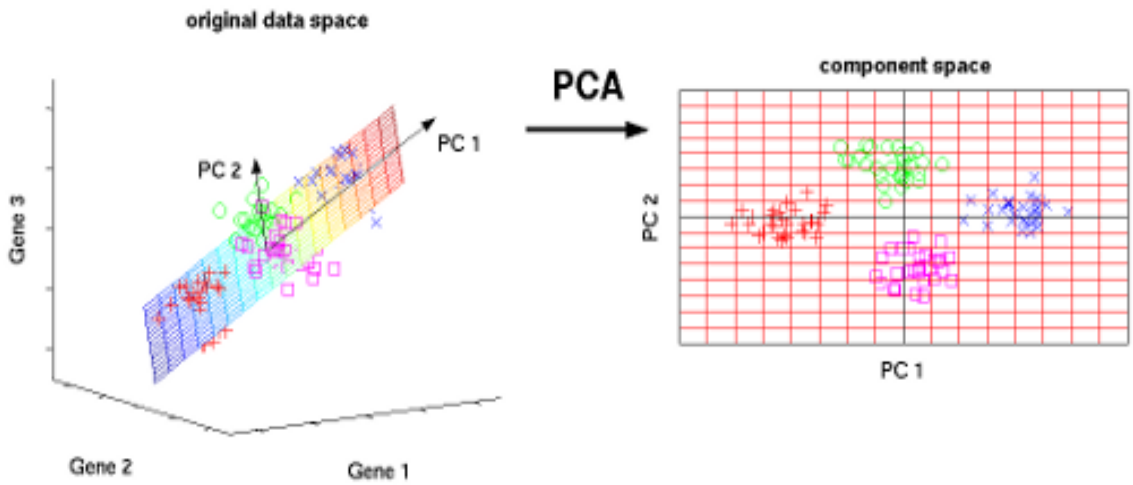

Figura 7: Ejemplo funcionamiento PCA.

Una de las ventajas de usar esta técnica es que las características del conjunto de datos que contribuyen más a su varianza son las que se retienen.

2.1.4. Clasificadores

Una parte fundamental después de haber extraído el vector de características es escoger que clasificador se va a utilizar para realizar la asignación de una clase a una entrada determinada. En este proyecto se han trabajado con las dos opciones más utilizadas por otros investigadores.

2.1.4.1. Softmax

La función Softmax es usada para hacer la regresión logística en casos de clasificación multinomial. En esencia la regresión Softmax es una generalización de la regresión logística pero usada para clasificación multi-clase (bajo la asunción de que las clases son mutuamente excluyentes).

Esta función nos permite mapear un vector de dimensión K (el vector de características) en un vector de probabilidades asignando una probabilidad entre 0 y 1 a cada clase bajo la premisa que la suma de las probabilidades de 1.

La ecuación que define a la función Softmax es: $\sigma(z)j = \frac{e^{Zj}}{\sum_{k=1}^{K} e^{Zk}} \quad para\ j = 1, \ldots, k$

Donde z_j es el valor de una neurona de la capa *fully-connected*. Y el denominador actúa como normalizador.

2.1.4.2. Support Vector Machine (SVM)

Al igual que el Softmax,, el SVM [22] es un modelo discriminativo, es decir, es capaz de trazar las fronteras entre las diferentes clases de un conjunto sin necesidad de modelar la función de distribución de la probabilidad (pdf).

SVM, en nuestro caso con kernel lineal, se basa en buscar diferentes hiperplanos separadores (en el espacio transformado) que maximicen el margen. Ya que tal maximizar este margen minimiza el error de generalización del clasificador.

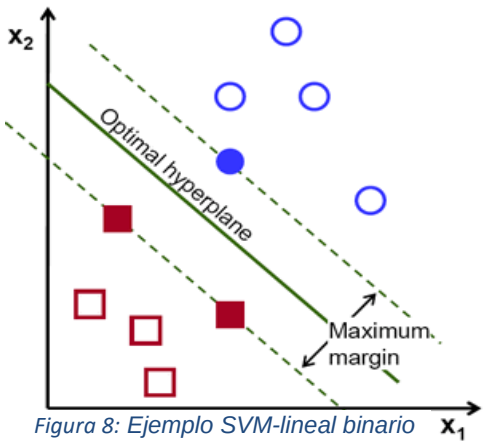

Figura 8: Ejemplo SVM-lineal binario

El SVM partió usándose sólo para clasificación binaria pero actualmente su uso se ha extendido hacia la clasificación multiclase.

Existe más de una vía para atacar este problema pero las dos más comunes son:

1. Uno contra todos (OVA) clasificación: Supongamos que se tienen las clases A, B, C, y D. en Lugar de hacer una de cuatro categorías, cuatro clasificadores binarios: Un A vs no-A, B vs no-B, C vs no-C, y D vs no-D. Cada muestra nueva es asignada a la clase con mayor discriminante. En este caso las regiones de separación no están definidas y es poco simétrico.

2. Todos contra Todos: Clasificación por pares: se consideran c(c-1)/2 pares de clases, y para cada una se diseña un clasificador. Cada nueva muestra es pasada por todos los clasificadores y se asigna a la clase más votada. Hay que resolver más operaciones pero con un menor número de muestras. Las regiones siguen sin estar definidas.

2.1.5. Transfer Learning y Finetune

Dado que hay un gran número de investigadores que han desarrollado diferentes redes para hacer el reconocimiento de personas, sería interesante el poder reutilizar estas redes ya entrenadas y preparadas pero adaptándolas a nuestra aplicación. Este concepto es el llamado Transfer Learning y se centra en adaptar redes ya entrenadas para poder aplicarlo a otros problemas de naturaleza parecida.

Esa operación de adaptación se le conoce como Finetune, es decir, y en casos generales se cogen las últimas capas de la red y se reentrenan con un dataset ya particularizado para nuestra aplicación.

2.1.6. VGGFace

La VGG-Face está basada en la arquitectura CNN de la VGG-Very-Deep-16 descrita en [1]. La red está compuesta por una secuencia de capas convolucionales, de pooling y capas Fully-Connected (FC).

Tiene la siguiente arquitectura:

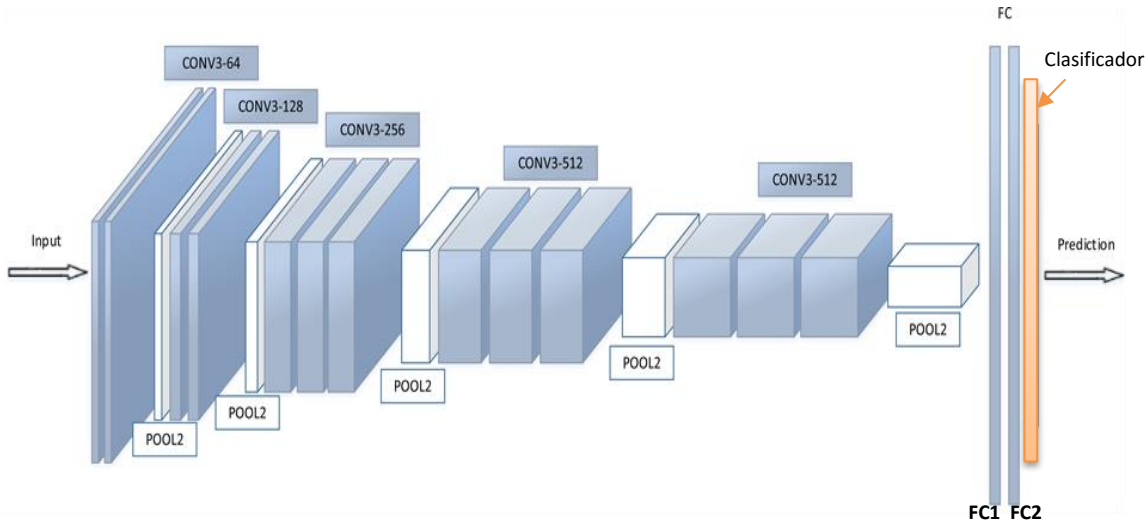

Figura 9 Arquitectura de la VGGFace

Las capas convolucionales CON3-#filtros, 13 en total, utilizan filtros de diferentes dimensiones.
La primera capa convolucional utiliza un filtro de profundidad 3 puesto que la entrada de la red es una imagen (de 224x224 pixeles) con codificación RGB (contiene información en los tres canales de color).
En las otras capas esta profundidad viene dictaminada por la profundidad de la capa anterior. Si la salida de la segunda conv3-64 tiene profundidad 64 (profundidad de salida es igual al número de filtros de la capa convolucional) en la siguiente convolución la Conv3-128 cada uno de esos 128 filtros tendrá una profundidad de 64 (el filtro se replica n veces en profundidad donde n es la profundidad de la salida de la capa anterior).

Tenemos 5 capas de Maxpooling o subsambplig que reducirán la dimensión de la imagen a una de 7x7, cada capa tiene un factor de submuestre de 2. A la salida del último pooling tendremos una salida con dimensión 7x7x512.

Esta salida se conecta a dos capas Fully-connected de dimensión 4096 neuronas.
La salida de la FC2 es el vector de características que procederá a ser conectado al clasificador, en este caso un softmax de dimensión 2622.

En nuestros experimentos, usamos una implementación pre-entrenada de la VGG-Face.
La CNN está entrenada con 982.802 imágenes web de 1622 celebridades y figuras públicas.

La Tabla 1 proporciona a continuación ofrece detalles adicionales sobre las capas de la CNN. La columna de Dimensiones representa la anchura, altura y profundidad de cada capa. La columna de parámetros muestra el número de parámetros que se han aprendido [2].

Tabla 1: Número de parámetros por cada capa en la VGGFace

Capas (1-11)	Dimensiones	Parameters	Capas (12-22)	Dimensiones	Parameters
INPUT	224 × 224 × 3	0	CONV3-512	28 × 28 × 512	(3 * 3 * 256) * 512 = 1,179,648
CONV3-64	224 × 224 × 64	(3 * 3 * 3) * 64 = 1,728	CONV3-512	28 × 28 × 512	(3 * 3 * 512) * 512 = 2,359,296
CONV3-64	224 × 224 × 64	(3 * 3 * 64) * 64 = 36,864	CONV3-512	28 × 28 × 512	(3 * 3 * 512) * 512 = 2,359,296
POOL2	112 × 112 × 64	0	POOL2	14 × 14 × 512	0
CONV3-128	112 × 112 × 128	(3 * 3 * 64) * 128 = 73,728	CONV3-512	14 × 14 × 512	(3 * 3 * 512) * 512 = 2,359,296
CONV3-128	112 × 112 × 128	(3 * 3 * 128) * 128 = 147,456	CONV3-512	14 × 14 × 512	(3 * 3 * 512) * 512 = 2,359,296
POOL2	56 × 56 × 128	0	CONV3-512	14 × 14 × 512	(3 * 3 * 512) * 512 = 2,359,296
CONV3-256	56 × 56 × 256	(3 * 3 * 128) * 256 = 294,912	POOL2	7 × 7 × 512	0
CONV3-256	56 × 56 × 256	(3 * 3 * 256) * 256 = 589,824	FC-1	1 × 1 × 4096	7 * 7 * 512 * 4096 = 102,760,448
CONV3-256	56 × 56 × 256	(3 * 3 * 256) * 256 = 589,824	FC-2	1 × 1 × 4096	4096 * 4096 = 16,777,216
POOL2	28 × 28 × 256	0	FC-3	1 × 1 × 2622	4096 * 2622 = 10,739,712

Mientras que esta red esta entrenada sólo identificar a los sujetos en su conjunto de datos de entrenamiento, se puede utilizar como un extractor de características para cualquier rostro.

2.1.7. Bases de datos

Formada por imágenes Fijas:

Umeå University Database of Facial Expressions [5]**:** Esta consiste en 30 modelos masculinos y 30 femeninos cuya edad está en la franja de los 17 a los 67 años. Cada modelo muestra 7 emociones diferentes (tristeza, enfado, asco, miedo, neutral, contento y sorpresa). La mayor parte de los modelos son suecos pero algunos son de la Europa central, Arabia o Asia.

Las imágenes, se les presentaron a 526 voluntarios en diferente orden y teniendo que puntuar un promedio de 125 imágenes. La precisión de aciertos se elevó hasta el 88%.

Radboud Faces Database (RaFD)[4]:

Es una BBDD hecha por el Behaviour Science Institute of the Radboud University Nijmegen donde tenemos un conjunto de 67 modelos (incluidos hombres, mujeres y niños caucásicos, hombres marroquíes y holandeses) que muestran 8 expresiones faciales diferentes (tristeza, enfado, asco, miedo, neutral, relajado, contento y sorpresa).

Figura 10: Ejemplo de uno de los modelos de la RaFD.

Formada por secuencias:

Cohn Kanade Plus (CK+)[6][7]**:** Es una base de datos que incluye más de 500 secuencias en las cuales tenemos a más de 100 sujetos posando o sin posar (espontáneos). Las secuencias están etiquetadas y validadas. En este caso sólo se definen 67 emociones (tristeza, enfado, asco, miedo, contento, neutral y sorpresa). Todas las secuencias parten de una emoción neutral hasta derivar a otra.

2.2. Estado del arte

En este apartado tenemos algunos de los avances que han hecho otros investigadores en campo de sistemas de reconocimiento facial [9][10]:

Muchos métodos dependen de la extracción de la región facial. Esto se puede realizar a través de la inferencia manual [16] o con detección automática [15]. Muchos métodos a menudo utilizan el Sistema de Codificación de la Acción Facial (FACS), el cual describe la expresión facial utilizando las Unidades de Acción (AU). Una Unidad de Acción es una acción facial como "levantar la ceja". Múltiples activaciones de AUs describen una expresión [17]. Ser capaz de detectar correctamente AUs es un paso útil, ya que permite evaluar el nivel de activación de la emoción correspondiente.

Las marcas faciales hechas a mano se pueden utilizar como hizo Kotsia [16]. La detección de estos puntos de referencia puede ser difícil, ya que la distancia entre ellos difiere dependiendo de la persona en las cuales se evalúen [16].

No sólo pueden utilizarse AUs para detectar emociones, sino que también se puede hacer mediante la textura. Es decir, cuando una cara muestra una emoción la estructura cambia y se pueden aplicar diferentes filtros para detectar eso [18]

El siguiente paso en la evolución de los sistemas de detección facial fue divergir hacia las Redes Neuronales. A continuación se detallan alguno de estos sistemas.

El primero de ellos es el que introdujeron Yu & Zhang para su sistema FER basado en redes neuronales convolucionales. En este caso ellos en su implementación perturbaron las imágenes de entrada en la secuencia de entrenamiento lo cual produjo un aumento en la precisión entorno al 2-3% *(Data Augmentation)* [12]. Además, en el momento de testeo su modelo genero predicciones para múltiples perturbaciones de cada imagen de test y voto en cada etiqueta de clase para producir una respuesta final.

El segundo, es el que introdujo Kim que además de variar la arquitectura de la red y los parámetros que la componen inicializo los pesos del modelo usando pesos entrenados en otros datasets para posteriormente aplicarles un reentreno [13].

Un avance importante ha sido el de prevenir las redes del overfitting, sobreentrenamiento, cambiando el parámetro de dropout en las capas Fully-Connected [8].

Y por último, el uso de reemplazar el clasificador softmax con un Support Vector Machine lineal aumenta significativamente la ganancia en un rango bastante amplio de datasets como son el MNIST o el ICML 2013 (dataset usado en el facial expresión recognition Challenge) [3].

3. Metodologia/Desarrollo del proyecto

Se han implementado diversos *Facial Expression Reconition Systems* (FER systems) donde las diferencias entre ellos radican en los cambios hechos en los diferentes bloques que lo componen.

Para ello hemos dividido cada sistema FER en 4 bloques donde cada uno tiene una funcionalidad distinta pero que al concatenarlos cumplen la función de identificación de emociones.

Estos bloques son:

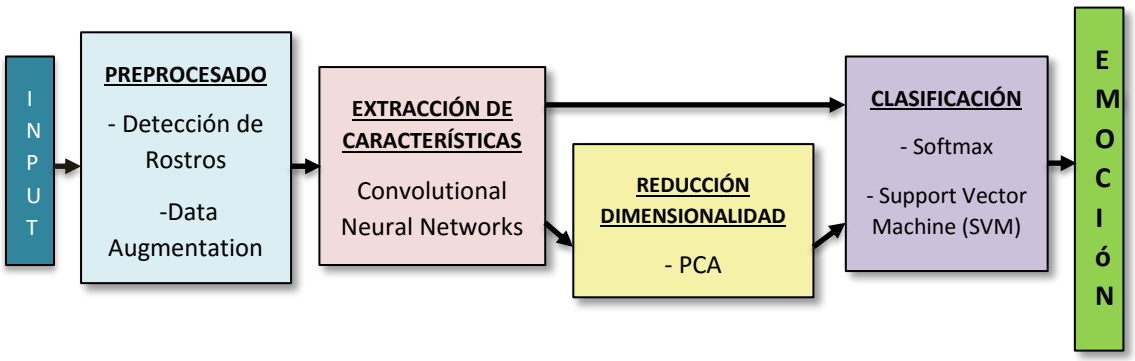

Figura 11 Diagrama de bloques de un Sistema de reconocimiento de expresiones faciales

El primero de ellos es el preprocesado, en ese bloque se ha procedido a la adaptación las imágenes de las diferentes bases de datos para que estas sean aptas para nuestra red. Entre los cambios que se han hecho están: la detección de rostros, ajustar la imagen para convertirlas de escala de grises a color en los casos que fuera necesario y también se ha hecho *data augmentation,* que explicaremos con más detalle en el siguiente apartado.

El siguiente bloque del sistema es el de extracción de características, este se ha basado en la red neuronal convolucional VGGFace. Aquí se ha reentrenado la red con diversas bases de datos, se han hecho alteraciones en las capas que la componen, se ha cambiado el optimizador utilizado en el entrenamiento y se ha modificado el parámetro de dropout en las capas Fully-Connected.

Entre la extracción de características y la clasificación se ha probado la reducción de la dimensionalidad de este vector mediante el uso de la técnica PCA.

El último bloque pero no el menos importante, es el clasificador. La VGGFace viene por defecto con un softmax como clasificador pero después un artículo donde se comentaba el incremento en la precisión al usar un SVM a la salida de una CNN nos hemos decantado por probarlo [3].

A continuación se van a explicar los diferentes cambios en los bloques con más detalles.

3.1. Bloque: Preprocesado:

Este bloque tiene como objetivo extraer toda la información base de las diferentes bases de datos.

Con todas las BBDD con las que trabajamos en este proyecto, ya descritas en apartado 2, tenemos el problema de que sólo alrededor del 30% de toda la imagen representa información útil, es decir, el rostro.

Para solventarlo se ha usado un algoritmo de detección de rostros basado en el método para detectar objetos usando *Haar Feature-based Cascade Classifiers* por Paul viola y Michael Jones ya entrenado para detectar caras [14].

En nuestro caso hemos usado el algoritmo que ya viene implementado en la librería para Python OpenCV [23].

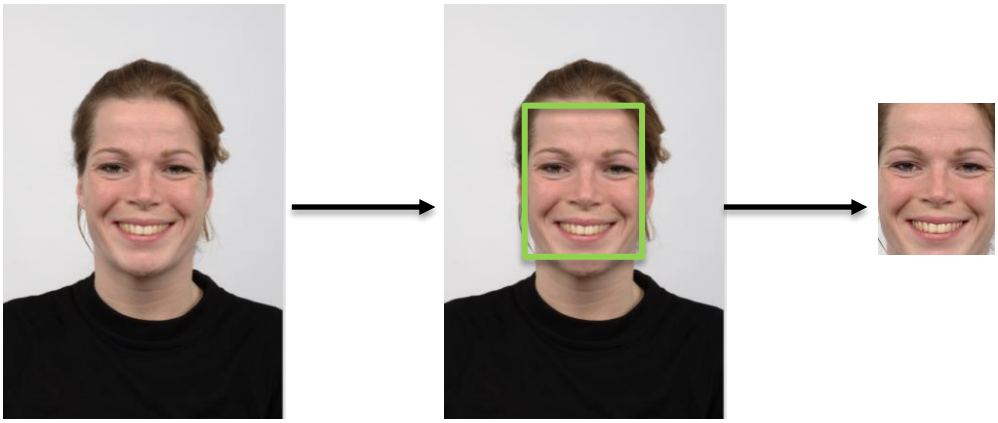

Figura 12: Imagen original, imagen con el rostro detectado, rostro (o imagen recortada)

Además un segundo problema que tenemos es que las bases de datos con las que trabajamos son muy pequeñas y nosotros tenemos que entrenar muchos parámetros. Para ello se ha usado la técnica de *data augmentation* que es implementable con Keras, el Framework que utilizamos para el desarrollo del proyecto, ya tiene una funcionalidad que hace exactamente esa tarea (el ImageDataGenerator).

Data augmentation nos permite incrementar artificialmente el tamaño del conjunto de datos de entrenamiento. Para ello se hacen diferentes transformaciones sobre las imágenes como son las rotaciones, las translaciones, reescalados, giros y recortes, entre otras.

El aumento del conjunto de muestras usando esta técnica nos permite hacer el sistema más robusto e invariante.

En este bloque también se dividen las bases de datos en 2 datasets distintos, uno destinado al entreno de la red y otro para test. En nuestro caso hemos divido la base de datos RadBoud (RaFD) en un 20% para el test y el 80% restante para el entrenamiento. Es importante realizar una buena partición para no obtener resultados falsos, es decir, cuando se destinan demasiados recursos al entrenamiento el sistema puede quedar demasiado particularizado para esa base de datos.

En nuestro caso nuestra red ha sido entrenada utilizando únicamente como base de datos la RaFD y después testeada en esta, la Umeå University Database of Facial Expressions y la Cohn Kanade Plus, por lo tanto debemos entrenar una red capaz de extraer vectores de características de forma correcta para esas tres BBDD's, es decir una red más generalizada.

3.2. Bloque: Extracción de características

El objetivo de este bloque es extraer los vectores de características de las imágenes para así facilitar su clasificación. Esta tarea se realiza mediante redes neuronales convolucionales y la se ha implementado en Keras, un Framework que usa la librería de Google Tensorflow como backend.

En este bloque se han diseñado diferentes extractores de características pero todos ellos basados en la red VGGFace debido a la complejidad que presenta entrenar una CNN desde cero.

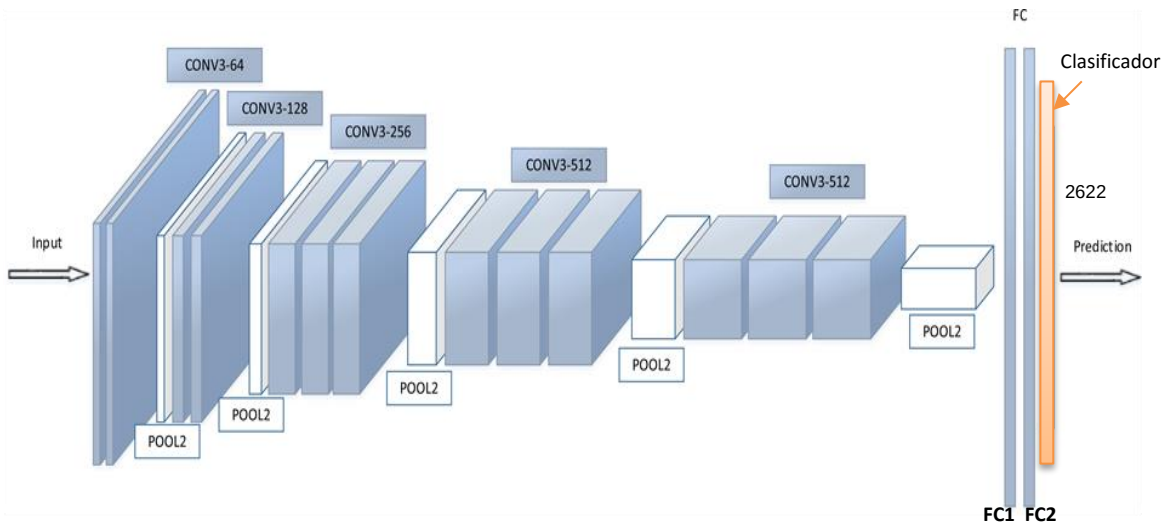

Figura 13 Arquitectura de la VGGFace

Como ya hemos dicho anteriormente, esa red ha sido entrenada para clasificar personas y no emociones. Y es por este motivo por el cual hemos reentrenado las últimas capas, las Fully Connected, para extraer descriptores de la imagen más especializados en la aplicación que nosotros estamos desarrollando.

A lo largo del proyecto hemos planteado diferentes arquitecturas para extraer las características y a continuación se enumeraran los más importantes:

- La primera es el modelo baseline: Directamente usar la red VGGFace ya preentrenada pero como base de datos de test usamos la Radboud. Los resultados como esperábamos no fueron nada satisfactorios (inferior al 15% de precisión) puesto que como ya he dicho anteriormente la VGGFace no está implementada para clasificar emociones.

- En la segunda arquitectura tenemos el modelo baseline pero con la última capa fully-connected (FC2) entrenada con la partición destinada al entreno de nuestra base de datos RaFD. Los resultados que obtuvimos tampoco fueron los esperados pero era porque algo estaba fallando en el reentreno puesto que la precisión no superaba el 20%. En este caso usábamos el stochastic gradient descent (SGD) como optimizador. Pese a cambiar los parámetros que definen al SGD como son el Learning Rate (LR) y el decaimiento de este LR no se consiguió superar esa barrera del 20%.

- En la tercera se cambió el optimizador empleado en la hora de realizar el reentreno de la última capa (FC2). Se pasó de usar un SGD a usar un *Adaptative Moment Estimation* (ADAM). En este caso la precisión en la clasificación aumento del 19.3% al 70%.

- El siguiente cambio que se introdujo para obtener la cuarta fue emplear el bloque 1 de detección de rostros que hasta ahora no se había utilizado en los modelos anteriores. Esto hizo aumentar la precisión a un 93.57%.

- Otra configuración con la que se decidió hacer pruebas era modificar el tamaño de la FC2, se pasó de 4096 a 256 neuronas haciendo escala en 2048 y en 1024. El que mejor resultados dio fue cuando la FC2 tenía un tamaño de 256 puesto que ya no teníamos que reentrenar 16M de parámetros sino 1M de ellos (parámetros a entrenar=neuronas_FC1 * neuronas_FC2). Esto nos dio una precisión de 82.5%.

- La última arquitectura que se hizo con tal de aumentar la precisión fue reentrenar también la primera capa *fully-connected* (FC1). Como en el quinto modelo se pasó de 4096 a 256. En este caso nos quedamos con 2048 neuronas para esta capa y obteniendo una precisión total de 89.28%. En este caso teníamos que entrenar un total de 52M de parámetros (7*7*512*2048 + 2048*256). Pese a que pueden parecer muchísimos, que lo son, ha quedado reflejado que es preferible entrenar las últimas capas de un modelo con un conjunto de datos específico para la aplicación que se quiere utilizar la red ya que estás ultimas capas son las que nos dan los descriptores de la imagen más específicos.

Nota: En todas las configuraciones que hemos mencionado se ha usado el *Softmax* como clasificador donde el tamaño de la capa se ha cambiado de 2622 a 7 (acorde con las 7 emociones que queremos clasificar). Además en los porcentajes de precisión tienen como referencia la base de datos RaFD, en el apartado de resultados se reflejan las otras dos bases de datos utilizadas y los resultados obtenidos usando SVM como clasificador.

3.3. Bloque: Reducción dimensionalidad

El objetivo de este bloque es reducir los vectores de características de las imágenes para así facilitar su clasificación. Esta tarea se realiza mediante una técnica llamada Análisis en componentes principales (PCA). En este caso esta ha sido implementada haciendo uso de la librería para Python orientada para aplicaciones de *Deep Learning Scikit-Learn* [24].

Una vez tenemos nuestro extractor de características listo procedemos a almacenar todos los vectores de las imágenes de entrenamiento para poder mirar el grado de *sparsidad* que tiene nuestro conjunto de datos. En este caso vimos que la dimensión de los vectores de características podía ser reducida puesto que no necesitábamos vectores de 256 posiciones para representar la imagen. Después de hacer varias pruebas aplicando diferentes tamaños al espacio resultante al aplicar PCA y probando el posterior clasificador obtuvimos que un tamaño óptimo que era utilizar tan sólo vectores de características de longitud 60.

El incremento en la precisión de clasificación no es apreciable. Pese a que no es un gran cambio respecto a lo obtenido con anterioridad el hecho de reducir la dimensión del vector ha reducido el coste computacional de la clasificación.

3.4. Bloque: Clasificadores

Después de encontrar la mejor configuración para el bloque de extracción de características y de reducir la dimensión de los vectores se ha procedido a evaluar el comportamiento del sistema bajo diversos clasificadores.

En este caso se han probado dos, el *Softmax* que es que viene por defecto configurado en la VGGFace y el *Support Vector Machine* (SVM), en nuestro caso el Lineal. Escogimos el SVM puesto después de mirar el estado del arte se ha visto que este incrementa la ganancia respecto al *softmax* y decidimos demostrar si eso era cierto en todas las aplicaciones de *Machine Learning* (ML) basadas en CNN como extractoras de características.

Este lo hemos probado de dos formas diferentes:

- SVM directamente a la salida de la CNN, es decir, teniendo como entrada los vectores de características de dimensión 256.
- PCA + SVM, en este caso los vectores son de longitud 60 que es óptimo que habíamos encontrado.

En ambos casos la precisión en la clasificación se vio incrementada respecto al *softmax*. Mientras con el *softmax* como máximo obtuvimos 89.28% de precisión con el SVM conseguimos un 98.14% de precisión con la misma configuración en la CNN y probados sobre el conjunto de test de la base de datos RaFD

3.4.1. Secuencias

El proceso de clasificación para secuencias es diferente que para imágenes fijas. En el caso de imágenes fijas pasamos el vector de características extraídas por la CNN por el clasificador y obtenemos cual es probabilidad tiene esa imagen de estar en cada clase, finalmente escogemos como clase definitiva la que tenga mayor probabilidad.

En el caso de secuencias hemos planteado dos opciones, en ambas partimos del vector de características ya extraído:

- En la primera opción clasificamos *frame* a *frame* y nos quedamos con el vector de probabilidades de cada imagen. Una vez los tenemos todos procedemos a realizar un promedio, es decir, los sumamos todos y dividimos por el total de vectores de probabilidades, obteniendo así un vector que nos dirá que emoción representa esa secuencia, ya que como habíamos dicho con anterioridad la base de datos CK+ pasa de una expresión neutral a otra emoción (siempre hay dos emociones por secuencia con una de ellas siendo la neutral).
- En la segunda opción seguimos la misma estrategia que en el caso 1 pero con la diferencia que se han ponderamos las emociones. Esto se hace porque algunas de las secuencias tienen un número de *frames* de la emoción neutral que a veces es más elevado que el número de *frames* de la emoción que se quiere identificar lo cual daba resultados erróneos en algunos casos.

Las secuencias de la base de datos CK+ han sido probadas únicamente en el mejor sistema, es decir, con el que obteníamos una precisión más elevada.

3.5. Resumen de Modelos

En la siguiente tabla se muestran todos los modelos que se han implementado donde tenemos resaltados los cambios que se han producido respecto al modelo anterior.

Modelo	Procesado	Extracción de Características	Optimizador	PCA	Clasificador
Baseline	DA	VGGFace	SGD	No	Softmax
Baseline	DA + **DR**	VGGFace	SGD	No	Softmax
Mod. 1a	DA	VGGFace + **Finetune FC2**	SGD	No	Softmax
Mod. 2a	DA	VGGFace + Finetune FC2	**ADAM**	No	Softmax
Mod. 3a	DA + **DR**	VGGFace + Finetune FC2	ADAM	No	Softmax
Mod. 4a	DA + DR	VGGFace + **Alteraciones FC2** + Finetune FC2	ADAM	No	Softmax
Mod. 4b	DA + DR	VGGFace + **Alteraciones FC2** + Finetune FC2	ADAM	No	SVM
Mod. 5a	DA + DR	VGGFace + **Alteraciones FC1**-FC2 + **Finetune FC1**-FC2	ADAM	No	Softmax
Mod. 5b	DA + DR	VGGFace + **Alteraciones FC1**-FC2 + **Finetune FC1**-FC2	ADAM	No	SVM
Mod. 6b	DA + DR	VGGFace + Alteraciones FC1-FC2 + Finetune FC1-FC2	ADAM	**Sí**	SVM
Mod. 7b	DA + DR	VGGFace + Alteraciones FC1-FC2 + Finetune FC1-FC2 **con Salida FC1**	SGD	**Sí**	SVM

Tabla 2: Resumen modelos implementados

DA: Data Augmentation
DR: Detección de Rostros
SGD: Stochastic Gradient Descent
ADAM: Adaptative Moment Estimation
SVM: Support Vector Machine

Como podemos observar los cambios se han ido haciendo de forma gradual para así poder ir estudiando los diferentes bloques que componen el sistema y ver como estos afectan a la precisión en la tarea de clasificación.

Primero tenemos el *Baseline*, este se ha hecho para ver desde en qué estado estábamos, es decir, cuál era el grado de precisión en la clasificación de emociones del que partíamos teniendo una red entrenada para reconocer caras. En el siguiente modelo (mod 1a) se quería mirar cual era el grado de mejora en la clasificación si reentrenabamos la última *Fully-Connected* con un conjunto de datos preparado para nuestra aplicación, es decir, cuál era la ganancia que obteníamos si particularizábamos nuestra red.

El siguiente paso fue estimar la influencia del optimizador, en este caso pasando de SGD a ADAM.

Después continuamos particularizando la red entrenando las dos *fully connected* para intentar maximizar la precisión en la clasificación de emociones.

Y finalmente, en los últimos modelos se hace un estudio de cómo afecta el clasificador en el sistema. Y si la reducción en la dimensión del vector de características nos aportaba ganancias.

4. Resultados

En el siguiente apartado se van a comentar dos tipos de resultados, los obtenidos de comparar las implementaciones que se han hecho a lo largo del proyecto y la comparación con otros investigadores.

Para ello hemos usado como métodos de evaluación las matrices de confusión y la *accuracy* de clasificación que tiene nuestro sistema con los conjuntos de test.

Se han usado esos dos métodos puesto que son los más utilizados por otros investigadores [9][10].

La matriz de confusión nos aporta información sobre donde se producen más errores de clasificación y entre que clases producen de forma muy visual. En está matriz está ordenada de tal forma que por filas tenemos las etiqueta real y por columnas la predicción.

La otra metodología de evaluación es la *accuracy* o precisión en la clasificación, esta da el porcentaje de acierto que tiene nuestro sistema clasificando el conjunto de datos de test.

4.1. Sistemas implementados

Como ya se ha comentado en el apartado anterior se han realizado diferentes arquitecturas para definir la red con el objetivo de observar que parámetros o configuraciones de las capas son las más críticas en cuestiones de alcanzar una precisión de clasificación óptima.

En la siguiente tabla se reflejan todas las implementaciones con las precisiones de cada una de ellas en las diferentes bases de datos que se han usado como conjuntos de test. Hay que tener en cuenta que en la tabla sólo se muestran los resultados obtenidos cuando la base de datos de entrenamiento era la RaFD. Más adelante se comentarán algunos experimentos particulares usando las otras dos bases de datos como conjuntos de entrenamiento.

Tabla 3: Resumen diferentes arquitecturas con precisiones

Modelo	Procesado	Extracción de Características	Optimizador	PCA	Clasificador	Acc. RaFD	Acc. Umea	Acc. CK+
Baseline	DA	VGGFace	SGD	No	Softmax	12,86%	Not_tested	Not_tested
Baseline	DA + **DR**	VGGFace	SGD	No	Softmax	16,45%	Not_tested	Not_tested
Mod. 1a	DA	VGGFace + **Finetune FC2**	SGD	No	Softmax	19,28%	Not_tested	Not_tested
Mod. 2a	DA	VGGFace + Finetune FC2	**ADAM**	No	Softmax	70,00%	Not_tested	Not_tested
Mod. 3a	DA + **DR**	VGGFace + Finetune FC2	ADAM	No	Softmax	93,57%	Not_tested	Not_tested
Mod. 4a	DA + DR	VGGFace + **Alteraciones FC2** + Finetune FC2	ADAM	No	Softmax	82,50%	45,23%	Not_tested
Mod. 4b	DA + DR	VGGFace + **Alteraciones FC2** + Finetune FC2	ADAM	No	SVM	86,10%	48,61%	Not_tested
Mod. 5a	DA + DR	VGGFace + **Alteraciones FC1**-FC2 + **Finetune FC1**-FC2	ADAM	No	Softmax	89,28%	36,40%	Not_tested
Mod. 5b	DA + DR	VGGFace + **Alteraciones FC1**-FC2 + **Finetune FC1**-FC2	ADAM	No	SVM	98,14%	37,33%	36,58%
Mod. 6b	DA + DR	VGGFace + Alteraciones FC1-FC2 + Finetune FC1-FC2	ADAM	**Sí**	SVM	98,14%	Not_tested	Not_tested
Mod. 7b	DA + DR	VGGFace + Alteraciones FC1-FC2 + Finetune FC1-FC2 **con Salida FC1**	SGD	**Sí**	SVM	98,14%	Not_tested	Not_tested

A continuación se va a proceder a analizar con más detalle los resultados del sistema con el que se han obtenido mejores resultados.

Este sistema es el que en la *tabla 3* aparece como modelo Mod 5b ya que obtiene la precisión de clasificación más elevada. También podríamos haber seleccionado el 6b donde la única diferencia entre ambos reside en el uso de PCA como reductor de la dimensionalidad del vector de características.

En este modelo (5b o 6b) tenemos como una etapa de preprocesado donde se hace *Data Augmentation* y un proceso de detección de rostros con tal de preparar el conjunto de entrenamiento para nuestra red.

A continuación se alimenta la CNN con estás imágenes y esta se encarga de extraer las características. Si nos fijamos en la tabla 3 la red está configurada de tal forma que la primera *fully-Connected* (FC1), la que nos encontramos después de la última capa convolucional en el modelo VGGFace, ha sido alterada cambiando su tamaño de 4096 neuronas a 2048 y la segunda *Fully-connected* (FC2) también ha visto reducido su número de neuronas en este caso de 4096 a 256.

Como vemos una vez extraídas las características se utiliza un SVM como clasificador obteniendo una precisión del 98.14% en caso de testear el sistema con imágenes del conjunto de test que pertenecen a la base de datos RaFD. Si hubiésemos usado el modelo 4b, que se diferencia del 5b en el clasificador escogido, hubiésemos obtenido unos resultados del 89.28%.

Como vemos esto queda reflejado en la siguiente matriz de confusión, la cual nos aporta información sobre donde se producen más errores de clasificación y entre que clases producen.

Tabla 4: Matriz de confusion modelo 5b

True/Pred	Angry	Disgust	Fearful	Happy	Neutral	Sad	Surprise
Angry	100	0	0	0	0	0	0
Disgust	0	100	0	0	0	0	0
Fearful	0	0	94,12	0	0	5,882	0
Happy	0	0	0	100	0	0	0
Neutral	0	0	0	0	100	0	0
Sad	0	0	6,67	0	0	93,33	0
Surprise	0	0	0	0	0	0	100

Si observamos la tabla (expresada en %) vemos que nuestro sistema acierta con un 100% de exactitud en los casos donde tiene que clasificar las emociones de enfado, asco, felicidad y sorpresa. En cambio comete un ligero error cuando se trata de miedo y tristeza. Si nos fijamos vemos que a veces se confunde entre esas dos.

Como resultados de este modelo también se puede observar que la precisión que da al clasificar imágenes de la BBDD de la universidad de Umea no es demasiado elevada, de un 37,44%, sucede lo mismo al clasificar las secuencias de la BBDD CK+. Esto sucede puesto que se ha entrenado la red con una base de datos cuyas imágenes tienen rasgos en las expresiones faciales bastante diferentes.

Por ejemplo aquí vemos la emoción *'Asco'* para cada una de las bases de datos:

Figura 14: Emoción 'Asco' representada por las BBDD: RaFD, Umea, CK+ (de izquierda a derecha)

Como vemos hay diferencias más que notables en los diferentes rostros aunque todos ellos representen la misma expresión.

Otra factor que hay que tener en cuenta a la hora de evaluar los resultados es que la base de datos con las secuencias, la CK+, no tiene una emoción *'Neutral'* definida en sus secuencias, sino que cada secuencia pasa de esa emoción *'neutral'* a otra emoción. En cambio nuestra red sí que está entrenada para exista una emoción neutral independientemente del resto. Por ejemplo, en la siguiente secuencia:

Figura 15: Secuencia donde se produce el cambio de estado de 'Neutral' a 'Miedo'

Podemos observar que se produce un cambio de emoción pero si nos fijamos atentamente vemos que alrededor del sexto *frame* hasta el noveno hay otra emoción de transición. La existencia de esta transición también nos produce errores en la clasificación.

Otro de los motivos por los que la precisión en la detección de secuencias es baja es debido a que en ambos métodos que planteamos (promedio y ponderación descritos en la metodología) para la detección ninguno resuelve correctamente el hecho de tener la mitad de *frames* de la secuencia, como mínimo, siendo una emoción neutral la cual en algunos casos acaba sobrepasando el peso que aporta la emoción a detectar en la clasificación.

Con tal de intentar mejorar la precisión de clasificación con las bases de datos Umea y la CK+ se han realizado dos experimentos:

- El primero de ellos ha sido entrenar la red del modelo 5b pero usando como base de datos de entrenamiento un conjunto de imágenes de la Umea Database y luego testearlo sobre el conjunto de test de esa misma base de datos.
- El segundo ha sido repetir ese entrenamiento pero con un conjunto de entrenamiento de la CK+ y un conjunto de secuencias también de esa base de datos.

En el primer experimento *(exp1)* hemos obtenido una precisión del casi 93.06%.

En el caso del segundo experimento *(exp2)* la precisión ha aumentado notablemente ya que hemos pasado del 36% al 81.32%. Esta sigue estando por debajo del objetivo que queríamos cumplir pero esto ha sido debido al problema que he comentado antes y es el hecho de tener tantos *frames* de una emoción neutral antes de hacer la transición a la emoción que queremos detectar.

4.2. Sistema propio vs el de otros investigadores

En el siguiente sub-apartado vamos a comparar nuestro sistema con el de otros investigadores.

Partimos de la premisa que el resto de sistemas que vamos a nombrar a continuación entrenan con la misma base de datos con la que luego hacen el test lo cual implicará una mayores precisiones respecto a nuestro sistema, ya que nosotros hemos testeado y entrenado con bases de datos totalmente diferentes.

En cuanto a sistemas de detección de secuencias existen diferentes implementaciones que se recogen en la siguiente tabla resumen:

Autor	Método	Precisión
Happy et al [18]	Parches Faciales + SVM	94,09%
Lucey et al [19]	Active Appearance Model (AAM) + SVM	80%
Song et al. [20]	CNN	99,20%
DeXpression [10]	CNN + x-Cross validation	99,60%
Propuesto	CNN + SVM	36,58%

Tabla 5: Comparación sistemas FER clasificando la BBDD CK+

Como vemos todas esas implementaciones superan el 80%. Si usamos los resultados obtenidos cuando entrenábamos con la base de datos RaFD vemos que estamos muy por debajo del objetivo a cumplir, pero si usamos los resultados obtenidos en el segundo experimento *(exp2)* donde entrenábamos con imágenes de la base de datos de la CK+ vemos que estamos más cerca de cumplir el objetivo de tener un sistema que de unos resultados próximos a los existentes en la actualidad.

Otro de los resultados a tener en consideración, pero en este caso sobre imágenes fijas, son los obtenidos por el profesor Perl, desarrollador de la base de datos *Umeå University Database of Facial Expressions* [5] donde las imágenes, se les presentaron a 526 voluntarios en diferente orden y teniendo que puntuar un promedio de 125 imágenes. La precisión de aciertos se elevó hasta el 88%.

En nuestro caso la precisión ha sido de alrededor de 37% esto es debido, y como pasaba con la CK+, el sistema no se ha entrenado con esa base de datos por lo tanto no había tenido la ocasión de ver procesar ese tipo de información hasta la hora del test. Si usamos los resultados obtenidos en el experimento (exp1) tenemos que hemos superado esa precisión alrededor del 5 %

5. Presupuesto

El presupuesto del proyecto recae únicamente en desarrollo a nivel de Software ya que no se ha realizado ninguna implementación física.

La carga de trabajo ha sido equivalente a 400h. Para este presupuesto se ha considerado el salario estándar de becario marcado por la Universidad Politécnica de Catalunya de 8 €/h, con lo cual la cifra asciende a 3200€.

También se ha considerado el coste asociado al supervisor del proyecto, con el cual teníamos una reunión de aproximadamente una hora cada semana con un coste de 50 €/h. Por lo tanto tenemos 12 reuniones a 50 €/reunión un total de 600€.

El lenguaje de programación utilizado ha sido Python y todas las librerías son de código abierto, por tanto, no incrementan el presupuesto mínimo final del proyecto.
Como IDE de programación en Python ha sido utilizado el programa PyCharm de JetBrains cuya licencia profesional asciende a 20€/mes. Dado que el proyecto ha durado alrededor de 4 meses, se le suma al presupuesto 80€ en gastos de programación.

Por último añadir, que como durante el proyecto se ha hecho servir los servidores de la universidad, la contratación del servicio de los servidores equivalente a lo usado estos meses.
Se ha decidido presupuestar el alquiler de un servidor a la empresa Ultrarender durante 5 meses, este servidor tiene hasta 10 GPU's, una Nvidia TITAN X y una memoria RAM de hasta 24 GB.

Por lo tanto el total es:

Trabajo Alumno	3.200 €
Supervisor	600 €
Licencia Pycharm	80 €
Servidor	350 €
Total	**4.230 €**

6. Conclusiones y trabajo futuro

Durante el desarrollo de este proyecto se han podido evaluar los diferentes bloques que forman un sistema de reconocimiento de emociones. Se ha podido observar que las redes neuronales convolucionales como extractoras de características son una alternativa muy potente.

Después de estudiar diferentes arquitecturas y configuraciones del sistema se ha llegado a la conclusión de que montar una red neuronal convolucional basada en la red VGGFace es una opción viable para desarrollar un sistema FER.

Otra de las cosas que se ha podido ver es la importancia de escoger bien el optimizador con el que se va a entrenar la red.

Y lo decisivo que es escoger bien un dataset de entrenamiento determinado para el entorno de datos con los que luego se quiere probar. Ya que si entrenamos con una base de datos diferente a la que queremos luego aplicarle el sistema podemos caer en el caso de que particularicemos demasiado la red para una determinada información y que luego el sistema no sea capaz de generalizar.

También se ha medido el grado de ganancia que se obtiene si añadimos una etapa de preprocesado al sistema, es decir, es fundamental preparar la información con la que se va a alimentar la red para que esta pueda exprimir al máximo sus capacidades.

De cara a un futuro sería interesante poder continuar con la búsqueda de nuevas técnicas para solucionar el problema de la clasificación de secuencias, como por ejemplo el uso de una ventana móvil o utilizar una red neuronal recursiva (RNN) como es el caso de una arquitectura LTSM (Long Short-term Memory) que está optimizada para predecir sobre secuencias temporales [25].

7. Bibliografía:

[1] V.C. Gungor, B. Lu, G.P. Hancke. "Opportunities and challenges of Wireless Sensor Networks in Smart Grid". *IEEE Transactions on Industrial Electronics*, vol. 56, no. 10, pp. 3557-3564, October 2010. DOI: 10.1109/TIE.2009.2039455.

[2] El Khiyari, H. and Wechsler, H. (2016) Face Recognition across Time Lapse Using Convolutional Neural Networks. Journal of Information Security, 7, 141-151. doi: 10.4236/jis.2016.73010.

[3] Yichuan Tang . "Deep Learning using Linear Support Vector Machines". Workshop on Representational Learning, ICML 2013, Atlanta, USA, 2013

[4] Langner, O., Dotsch, R., Bijlstra, G., Wigboldus, D.H.J., Hawk, S.T., & van Knippenberg, A. (2010). Presentation and validation of the Radboud Faces Database. Cognition & Emotion, 24(8), 1377—1388. DOI: 10.1080/02699930903485076

[5] Samuelsson H, Jarnvik K, Henningsson H, Andersson J, Carlbring P, The Umeå University Database of Facial Expressions: A Validation Study, J Med Internet Res 2012;14(5):e136

[6] Kanade, T., Cohn, J. F., & Tian, Y. (2000). Comprehensive database for facial expression analysis. Proceedings of the Fourth IEEE International Conference on Automatic Face and Gesture Recognition (FG'00), Grenoble, France, 46-53.

[7] Lucey, P., Cohn, J. F., Kanade, T., Saragih, J., Ambadar, Z., & Matthews, I. (2010). The Extended Cohn-Kanade Dataset (CK+): A complete expression dataset for action unit and emotion-specified expression. Proceedings of the Third International Workshop on CVPR for Human Communicative Behavior Analysis (CVPR4HB 2010), San Francisco, USA, 94-101.

[8] N Srivastava, GE Hinton, A Krizhevsky, I Sutskever, R Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. Journal of Machine Learning Research 15 (1), 1929-1958

[9] Raghuvanshi Arushi, Choksi Vivek, Facial Expression Recognition with Convolutional Neural Networks. Report for Standford, 2016.

[10] Peter Burkert, Felix Trier, Muhammad Zeshan Afzal, Andreas Dengel, Marcus Liwicki. DeXpression: Deep Convolutional Neural Network for Expression Recognition. 17 Aug 2016. arXiv:1509.05371v2

[11] F. Chollet. keras. https://github.com/fchollet/ keras, 2015

[12] Z. Yu and C. Zhang. Image based static facial expression recognition with multiple deep network learning. ICMI Proceedings

[13] S.-Y. D. Bo-Kyeong Kim, Jihyeon Roh and S.-Y. Lee. Hierarchical committee of deep convolutional neural networks for robust facial expression recognition. Journal on Multimodal User Interfaces, pages 1–17, 2015.

[14] Viola, P., & Jones, M. (2001). Robust real-time object detection. International Journal of Computer Vision, 4.

[15] K. Anderson and P. W. Mcowan, "A real-time automated system for recognition of human facial expressions," IEEE Trans. Syst., Man, Cybern. B, Cybern, pp. 96–105, 2006

[16] I. Kotsia and I. Pitas, "Facial expression recognition in image sequences using geometric deformation features and support vector machines," Image Processing, IEEE Transactions on, vol. 16, no. 1, pp. 172–187, Jan 2007

[17] B. V. Kumar, "Face expression recognition and analysis: the state of the art," Course Paper, Visual Interfaces to Computer, 2009

[18] S. Happy and A. Routray, "Automatic facial expression recognition using features of salient facial patches," Affective Computing, IEEE Transactions on, vol. 6, no. 1, pp. 1–12, Jan 2015

[19] P. Lucey, J. Cohn, T. Kanade, J. Saragih, Z. Ambadar, and I. Matthews, "The extended cohn-kanade dataset (ck+): A complete dataset for action unit and emotion-specified expression," in Computer Vision and Pattern Recognition Workshops (CVPRW), 2010 IEEE Computer Society Conference on, June 2010, pp. 94–101.

[20]] I. Song, H.-J. Kim, and P. Jeon, "Deep learning for real-time robust facial expression recognition on a smartphone," in Consumer Electronics (ICCE), 2014 IEEE International Conference on, Jan 2014, pp. 564–567

[21] http://cs231n.github.io/neural-networks-2/

[22] http://cs231n.github.io/linear-classify/

[23] http://opencv.org/

[24] http://scikit-learn.org/stable/

[25] http://machinelearningmastery.com/sequence-classification-lstm-recurrent-neural-networks-python-keras/

8. Glosario

CNN: Convolutional Neural Network

FER: Facial Expresion Recognition

SVM: Support Vector Machine

PCA: Principal component Analisys.

Convnet: Convolutional Network.

FC: Fully Connected

Conv: Convolutional

SGD: Stochastic Gradient Descent

ReLU: Rectified Linear Unit

LR: Learning Rate.

Pdf= probability function distribution

CK+ = Cohn Kanade Plus

RaFD =Radboud Faces Database

BBDD: base de datos

ML: Machine Learning