People Recognition

David Suárez

September 24, 2016

1 Introducción

Este documento describe tanto el proceso de desarrollo como el proceso de investigación previo para poder realizar un proyecto basado en una tecnología innovadora como es Tensorflow junto con diferentes técnicas aplicadas para obtener el resultado más eficiente. Tras un proceso previo de investigación que a continuación explicaré, el proyecto finalmente podría dividirse en tres partes claramente diferenciadas.

- Identificación de personas utilizando redes neuronales multicapa.
- Identificación de personas utilizando redes neuronales convolucionales.
- Procesamiento en streaming de imágenes utilizando la red neuronal.

2 Objetivo

El objetivo de este proyecto es el reconocimiento e identificación de personas concretas básandonos en su rostro. Para ello hemos hecho un proceso de investigación que explicaremos a continuación y que ha estado dividio en diferentes fases finalizando con una conclusión en la que se explican los diferentes resultados obtenidos.

3 Fases de investigación y desarrollo

Distinguimos diferentes fases de investigación y desarrollo.

Con respecto a la parte de investigación podemos diferenciar las siguientes:

- Resolución de problemas relativo a la identificación de una cara en una imagen
- Diferentes métodos para obtener información de la cara relativa a la posición de los ojos, la boca, etc.

- Diferentes algoritmos posibles para la obtención del mayor porcentaje de acierto realizando una predicción.

Con respecto a la parte de desarrollo podemos diferenciar las siguientes:

- Procesado de imágenes con diferentes librerías como dlib, Haar Cascades para identificación de caras y puntos de referencia.
- Entrenamiento y modelado de redes neuronales utilizando Tensorflow.
- Creación de un flujo de imágenes en streaming que haga uso del modelo entrenado para dar una predicción.

4 Identificacion de caras

En este punto nos referimos al estudio y procesado de imágenes para la identificacion de caras en ella, es decir, no queremos identificar a la persona aun, simplemente distinguir entre la cara y el resto del cuerpo.

La solución a este problema está ya bastante optimizada, existen varias librerías en el mercado que lo solventan. Primero ha sido testeado mediante el uso de openCV, utilizando haar cascades cuyos resultados fueron buenos y posteriormente he terminado utilizando Dlib, ya que tiene un método creado específicamente para ello y los resultados eran mejores.

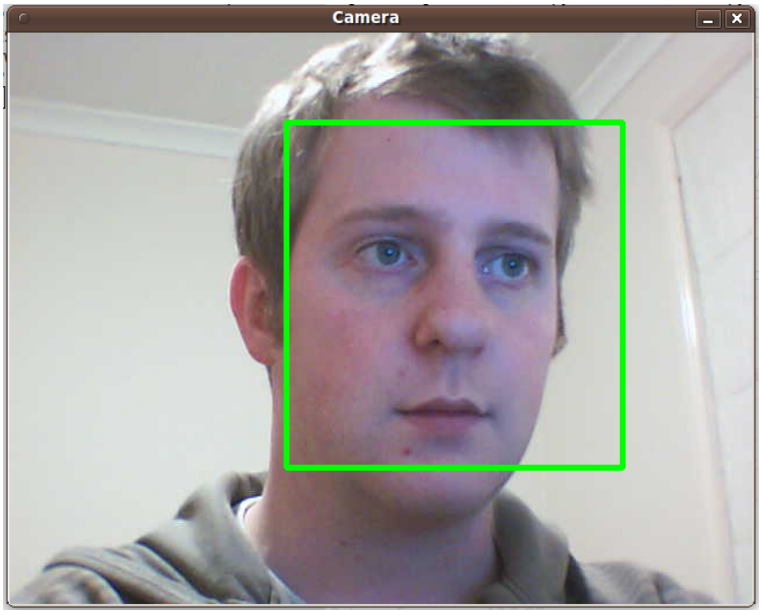

Figure 1: Imagen 1

5 Landmarks

Para este problema hemos probado dos soluciones diferentes, ambas basadas en openCV.

5.1 Haar Cascades

La primera ha sido utilizando haar cascades de los ojos y la sonrisa, lo cual no daba un resultado muy óptimo ya que a veces no era capaz de detectar ambos ojos, o no era capaz de detectar la boca, es decir, la salida de este algoritmo era un vestor cuyo tamaño era variable dependiendo de lo que detectaba. Esto causaba muchos problemas a la hora de introducir los datos en la red neuronal ya que el número de entradas no puede variar. En el caso de streaming esto era especialmente problemático ya que era más dificultoso filtrar los frames que no habían sido correctamente clasificados por lo que esta opción fue desechada.

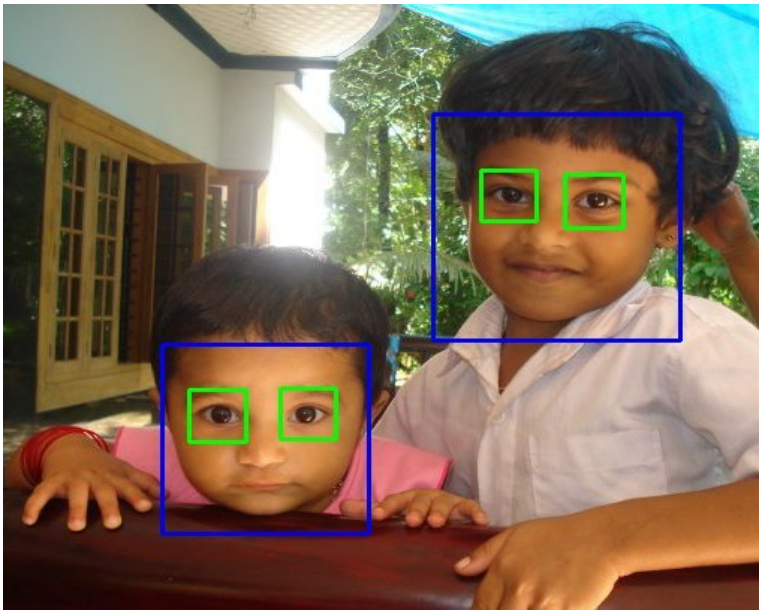

Figure 2: Detección de cara y ojos haar cascades

5.2 Dlib

Esta ha sido mi segunda opción y la definitiva debido a su efectividad. El algoritmo nos devuelve 68 puntos(landmarks) que pueden verse en la imagen inferior y que posteriormente procesamos mediante nuestro algoritmo de cálculo de distancias que explicaré a continuación.

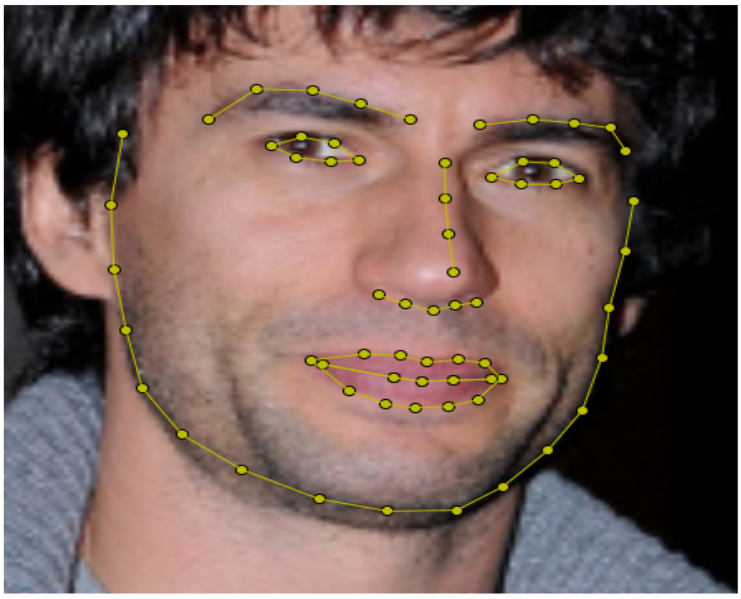

Figure 3: Detección de landmarks Dlib

6 Algoritmo de cálculo de distancias

Para nuestra primera rama de investigación en la que utilizaremos una red neuronal multicapa he diseñado un algoritmo que calcula distancias entre diferentes puntos de la cara(landmarks) creando una matriz de entrada a la red multicapa basada en estas mediciones. A continuación figuran varias imágenes que representan con líneas las mediciones tomadas. Todas estas medidas son posteriormente normalizadas utilizando la distancia total del tabique nasal, para evitar errores en las mediciones en caso de que las fotos sean tomadas desde diferentes distancias con respecto a la cámara.

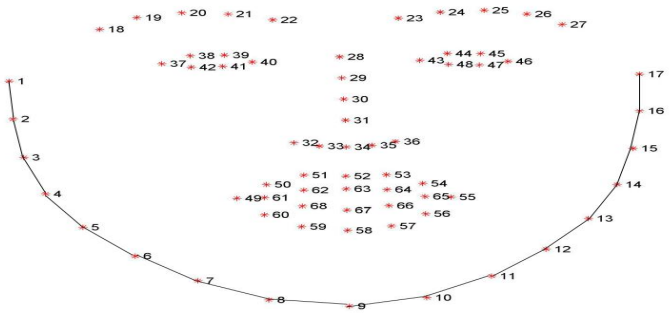

Figure 4: Perímetro

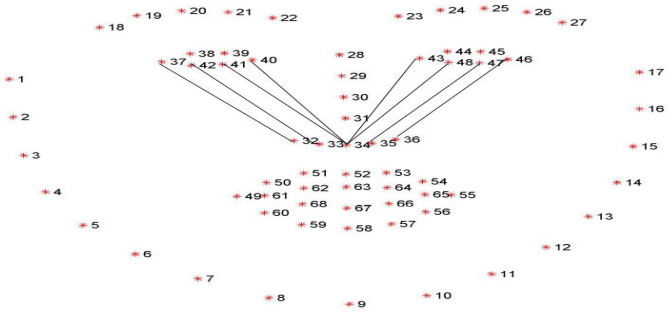

Figure 5: Ojos - Naríz

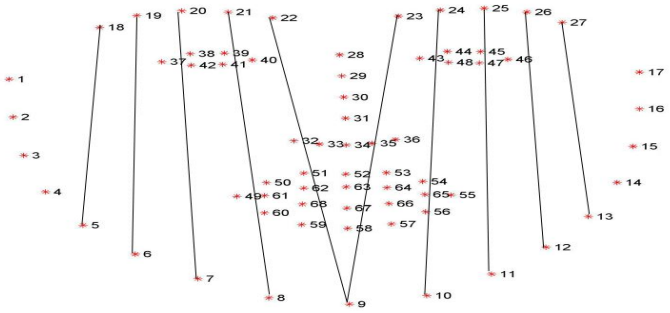

Figure 6: Cejas - Barbilla

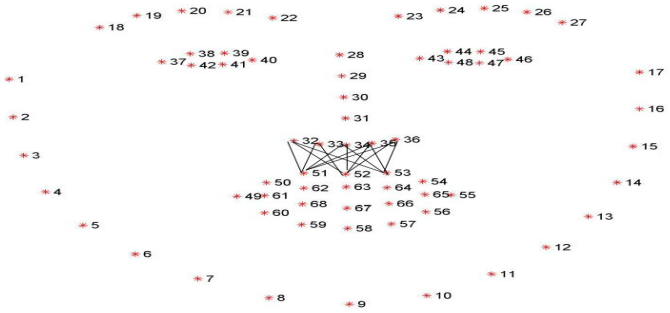

Figure 7: Naríz - Boca

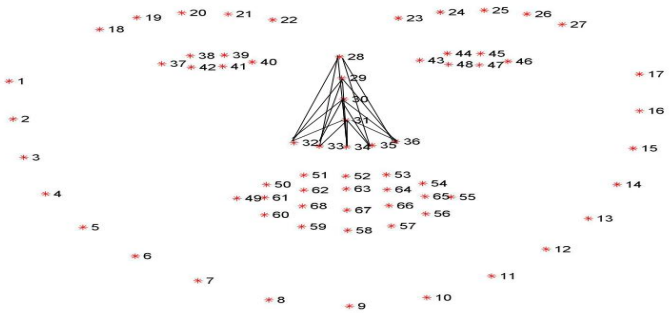

Figure 8: Naríz

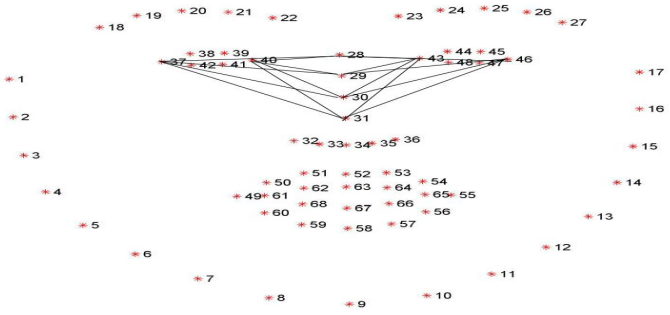

Figure 9: Ojos - Naríz

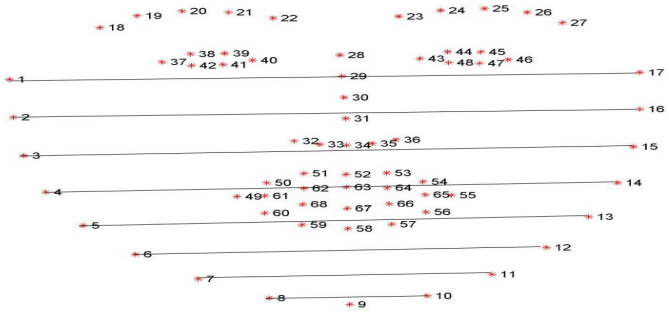

Figure 10: Ancho

7 Red Neuronal Multicapa

Para el desarrollo de las redes neuronales como ya he nombrado anteriormente he usado la librería TensorFlow [1], en concreto su API para Python.

Las Redes Neuronales Artificiales (RNA) [**?**] son sistemas de procesamiento de la información cuya estructura y funcionamiento están inspirados en las redes neuronales biológicas. Consisten en un gran número de elementos simples de procesamiento llamados nodos o neuronas que están organizados en capas(entrada, ocultas y salida). Cada neurona está conectada con otras neuronas mediante enlaces de comunicación, cada uno de los cuales tiene asociado un peso. En los pesos se encuentra el conocimiento que tiene la RNA acerca de un determinado problema.

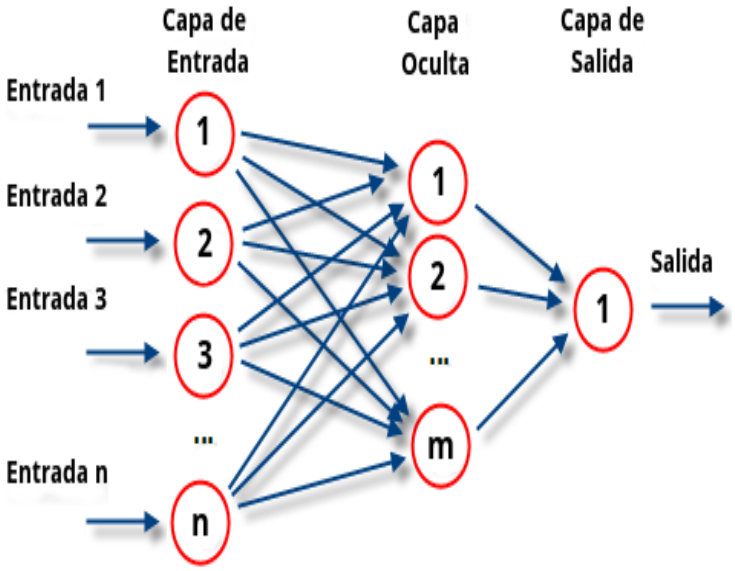

Figure 11: Multilayer Representation

Entrando un poco más en detalle y ayudándonos del gráfico inferior podemos describir el flujo que sigue una red neuronal. En primer plano tenemos la capa de entrada, que tiene tantas neuronas como longitud tiene nuestro vector de entrada a la red. En segundo plano se sitúa la capa oculta(en este caso solo tenémos una pero puede haber más) la cual es la que contiene la información de la red(los pesos). Estos pesos se van calculando utilizando la función de coste(loss function) [**?**] la cual nos mide el error de nuestra predicción para que de una manera iterativa nuestra red pueda ir aprendiendo(aprender es encontrar un mínimo de esta función) y de esta manera obtener los pesos óptimos para los cuales el resultado de nuestra predicción sea efectivo. En la capa de salida se sitúa lo que llamamos la función de activación [**?**]. Esta función se encarga de definir la salida de la red a partir de los datos que obtenemos de la capa oculta. Un ejemplo de esta sería la función sigmoide [**?**] que define la salida de tu red como 0 ó 1.

En nuestro caso hemos utilizado la función softmax [**?**] que es una variante de la sigmoide en la que la salida es un vector de 0s y 1s de tal manera que en caso de que nuestra red quiera diferenciar entre tres personas diferentes la salida siempre será (0, 0, 1) ó (0, 1, 0) ó (1, 0, 0).

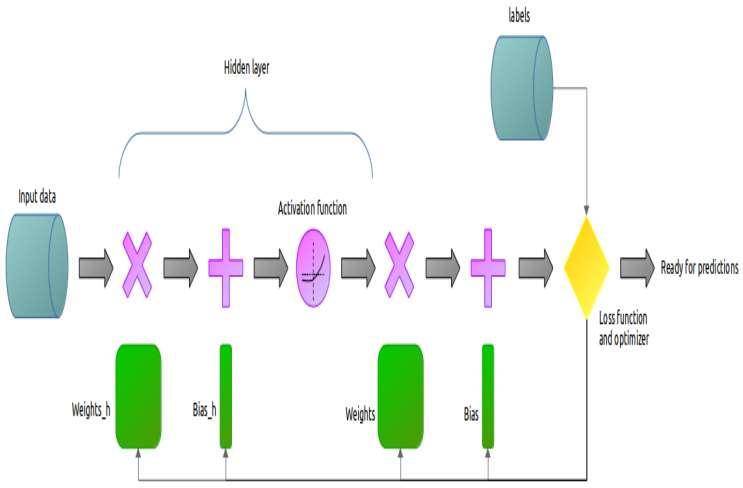

Figure 12: Multilayer Architecture

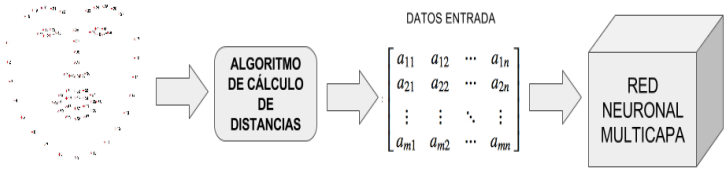

Figure 13: Multilayer Architecture

8 Red Neuronal Convolucional

A diferencia de la arquitectura anterior la red convolucional usa como datos de entrada imágenes.

La estructura de la red también cambia respecto a la anterior. A modo de resumen, la Red Neurononal Convolucional [**?**] aplica sus transformaciones mediante un *kernel* con el que se van obteniendo los píxeles de la imagen que mejor representan características de nuestros datos. Estas transformaciones se aplican capa a capa, reduciendo el tamaño de la imagen pero aumentando el mapa de características de la misma. Después de todas las transformaciones se obtiene la predicción de los parámetros como en los casos anteriores.

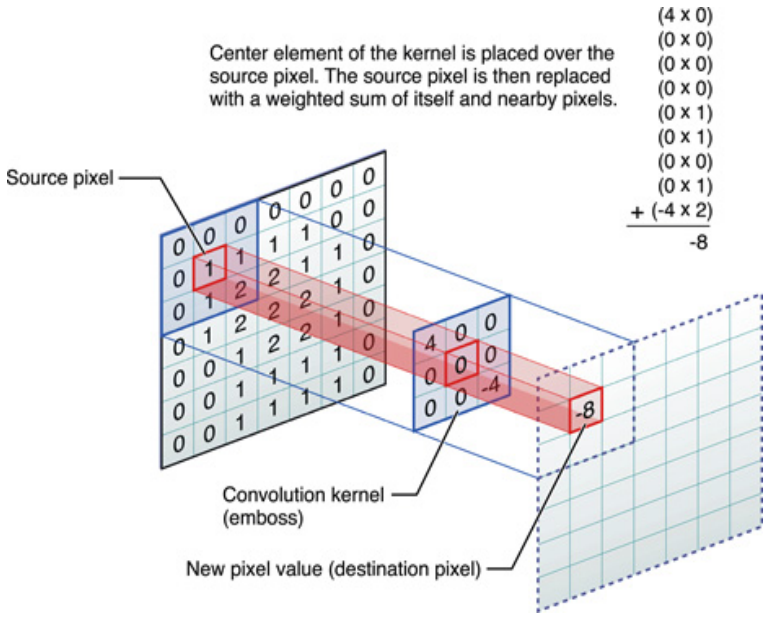

Figure 14: Multilayer Architecture

Nuestra red consistirá de cuatro capas convolucionales, dos capas ocultas y la capa de salida. A continuación podemos ver un gráfico que explica de manera simple el cálculo de la salida final de la red.

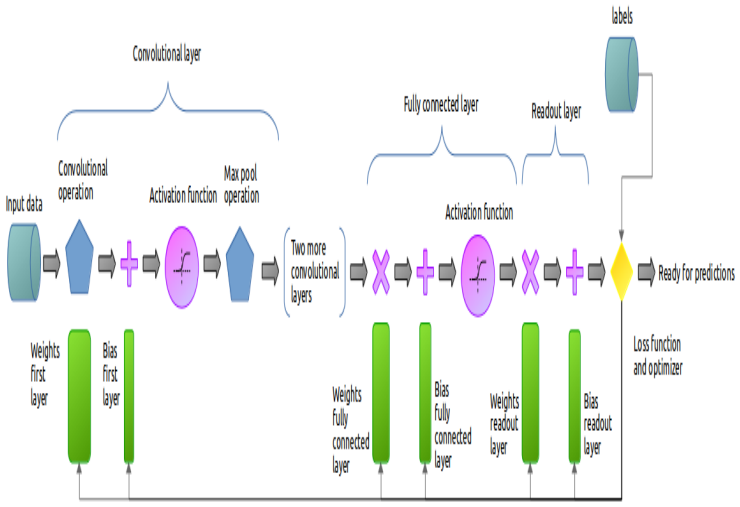

Figure 15: Ancho

En el gráfico inferior podemos observar el proceso de aprendizaje de la red mediante la representación de la precisión de la red y la función de coste en relación al número de iteraciones sobre los datos de entrenamiento.

En este primer gráfico podemos ver representadas dos funciones, la primera(verde) es la precisión sobre el conjunto de entrenamiento y la segunda(azul) es la precisión sobre el conjunto de test.

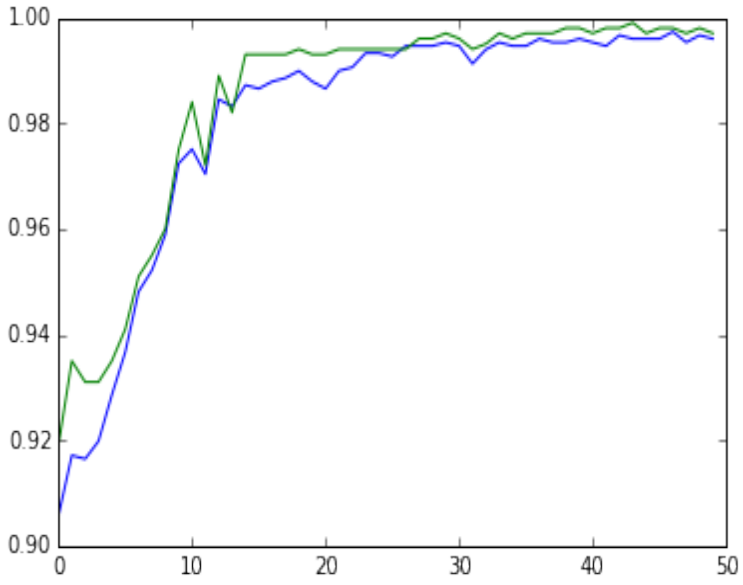

Figure 16: Ancho

En este siguiente gráfico vemos representada la función de coste, que como podemos observar va disminuyendo el error en la predicción conforme vamos iterando.

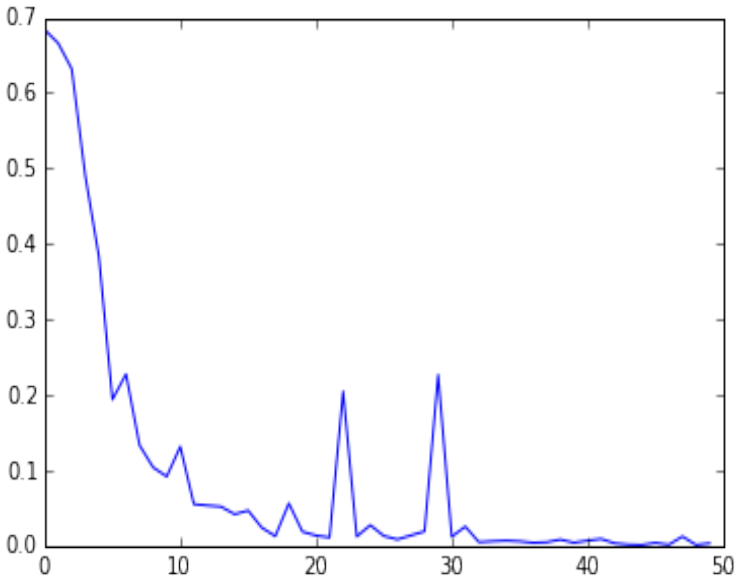

Figure 17: Ancho

La entrada de la imagen consiste en el recorte de la cara detectada en la imagen como se puede ver de manera descriptiva en el grafico inferior. He realizado diferentes pruebas aplicando filtros a la imagen recortada, como podría ser reducir la imagen a blanco y negro pero los resultados obtenidos no han sido buenos por lo que he decidido dejarla en su estado original.

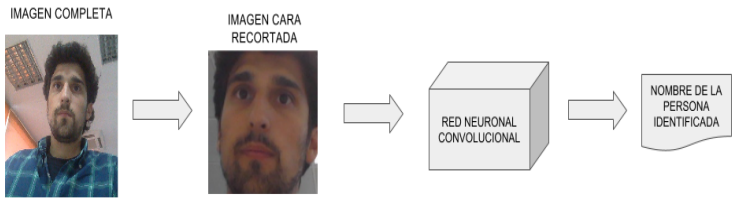

Figure 18: Multilayer Architecture

9 Resultados

Finalmente y a modo de conclusión procederé a la explicación de los resultados obtenidos mediante ambas ramas de investigación:

9.1 Red Neuronal Multicapa y Algoritmo de cálculo de distancias

9.2 Red Neuronal Convolucional

9.3 Demo

Demo: https://youtu.be/RxMD2-Wopdw

References

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya

Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems. `http://tensorflow.org/`, 2015. Software available from tensorflow.org.