

# Detección de caras basada en Redes Neuronales

*Francisco Rubin Capalbo*  
Universidad Politécnica de Valencia

17 de mayo de 2018

## 1. Introducción

Este trabajo consiste en el diseño y la implementación de un detector de caras basado en redes neuronales. El detector se puede dividir en dos partes igual de importantes: el clasificador de caras y el extractor de crops a clasificar.

## 2. Dataset

El dataset necesario para entrenar el clasificador de caras consiste en un conjunto de imágenes que contienen una cara (centrada, con un pequeño margen) y otro conjunto igual de grande que no contiene ninguna.

### 2.1. No-caras

Para obtener el conjunto de no-caras[1], he empleado varias herramientas. Por un lado, he descargado imágenes en masa de una API de google imágenes. He utilizado keywords con una gran probabilidad de representar imágenes sin caras, como 'carretera', 'espacio', 'texturas', etc.

A continuación, he utilizado un detector de caras de OpenCV basado en filtros Haar, para descartar las imágenes contenedoras de una o más caras. De las imágenes resultantes he extraído crops de ancho y alto aleatorios en posiciones aleatorias. En total, el dataset de no-caras está compuesto por unas 150 mil imágenes.

### 2.2. Caras

Para el dataset de caras he utilizado CelebA[2]. El problema de este dataset es que las caras que contiene no están centradas, y en algunas imágenes aparece el cuerpo entero. Para resolver este problema, he vuelto a recurrir a el detector de caras de OpenCV[3], para detectar una cara por imagen y extraerla a un archivo a parte. En total, he conseguido unas 150 mil caras.

### 2.3. Nota

Un detalle que me preocupa del método utilizado para obtener el dataset, es que depende completamente de el detector implementado en OpenCV con filtros Haar. Esto podría implicar que nuestro modelo no puede aprender de caras extrañas que los filtros Haar no detectan, así poniendo un techo a la precisión obtenida. Por otro lado, si los filtros Haar detectan caras dónde no las hay, éstas serán utilizadas incorrectamente en nuestro modelo. A pesar de esto, no he encontrado otra manera más efectiva de obtener el Dataset.

### 3. Clasificador

El objetivo del clasificador[4] es decidir si la imagen que recibe como input es una cara o no. Es un clasificador binario (cara / no cara), por lo cual utilizamos la función de pérdida 'Binary Cross Entropy'.

#### 3.1. Data Augmentation

Una manera muy efectiva de que el clasificador generalice mejor es haciendo data augmentation. Para este problema he aplicado rotación, traslación, escalado y flip horizontal de manera aleatoria. De especial importancia es la rotación, ya que la mayoría de las caras del dataset aparecen en posición vertical, y queremos que el clasificador sepa reconocer caras en cualquier orientación.

#### 3.2. Normalización de datos

Todas las imágenes pasadas al clasificador son normalizadas de la misma manera. Primero se convierten a blanco y negro y se aplica una normalización de color ( $x - \text{mean} / \text{std}$ ). Por último, se escalan a un tamaño de 24x24 píxeles.

#### 3.3. Modelos

- diferentes modelos / arquitecturas - bce loss - descripciones modelos - resultados de cada uno de ellos (tabla) - mencionar que todas las pruebas fueron realizadas en un dataset pequeño, incluyendo sólo un 5 % de los datos

Cuadro 1: Evaluación Modelos

| Modelo          | Accuracy      | Tiempo (segundos) |
|-----------------|---------------|-------------------|
| Modelo 1        | 99.766        | 0.316             |
| Modelo 2        | 99.732        | 0.295             |
| Modelo 3        | 99.465        | 0.277             |
| <b>Modelo 4</b> | <b>99.699</b> | <b>0.271</b>      |
| Modelo 5        | 99.532        | 0.293             |
| Modelo 6        | 99.766        | 0.393             |

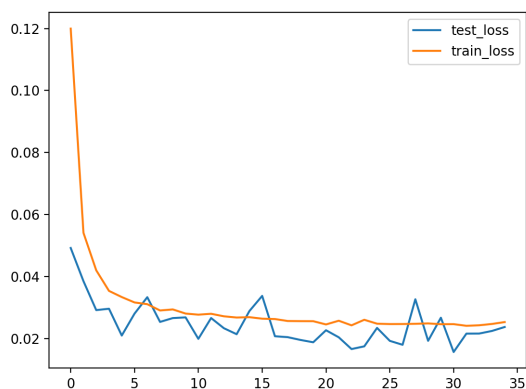


Figura 1: Modelo 4: Loss por epoch

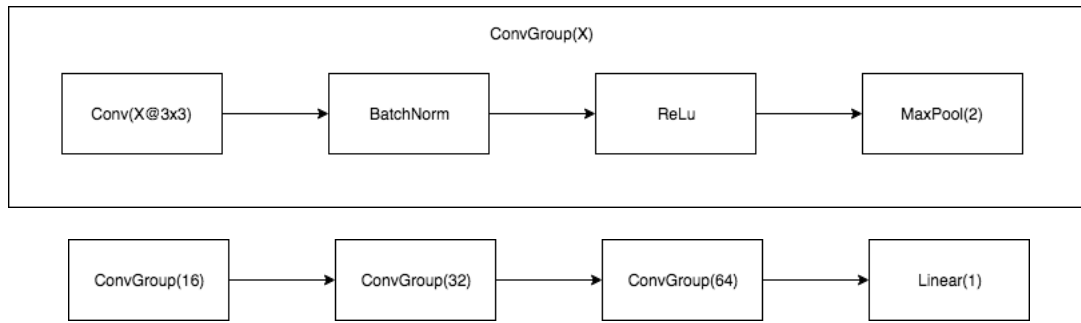


Figura 2: Modelo 1



Figura 3: Modelo 2

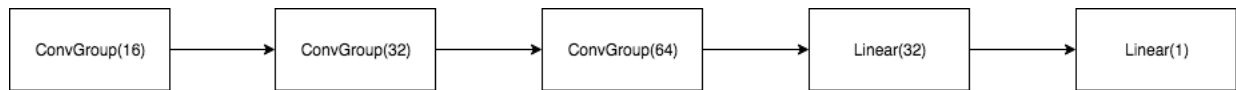


Figura 4: Modelo 3

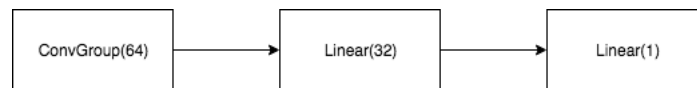


Figura 5: Modelo 4

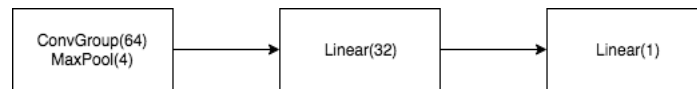


Figura 6: Modelo 5



Figura 7: Modelo 6

- detector - downscale image to a max width and height to reduce number of crops - extracción de crops - mencionar que al pasar los crops a memoria antes de clasificar mejoró de 180 segundos a 10 segundos - clasificación de crops en cara/no cara - pintar rectángulos en imagen final - tutorial de como usar el detector (para q lo pueda probar el profe) - Trabajo futuro - user selective search para selección de crops - probar con depthwise separable convolutionals para mejorar velocidad - aumentar los saltos de zoom y translation para las seleccion de crops usando el modelo con data augmentation potenciada

## Referencias

- [1] Script obtención no-caras
- [2] CelebA Dataset
- [3] Script obtención caras

- [4] Entrenamiento clasificador
- [5] name
- [6] name
- [7] name
- [8] name
- [9] name
- [10] name