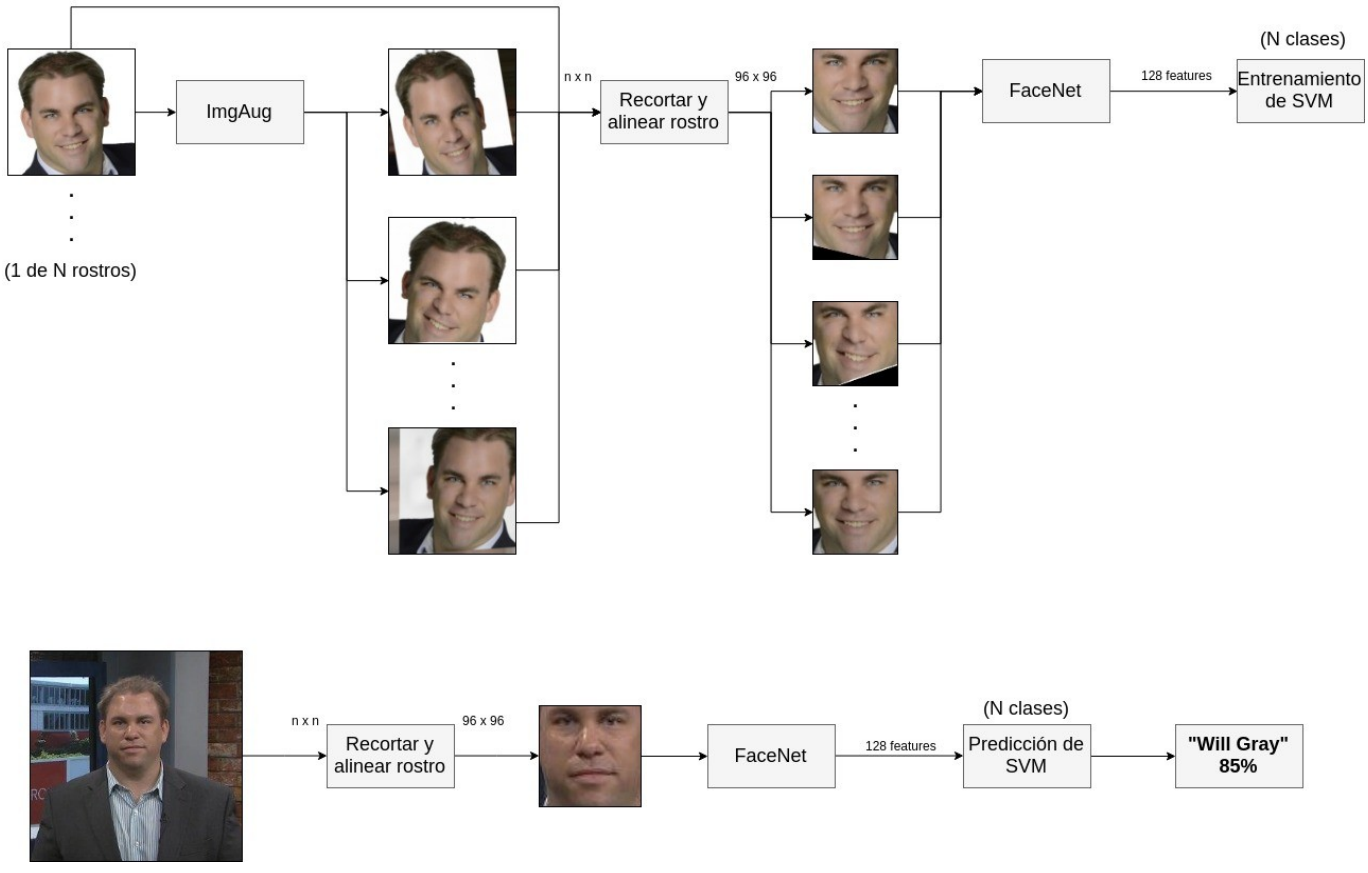
# Descripción de funcionamiento de sistema de reconocimiento facial

Autores:
Jaime Tenorio Ballesteros
Miguel Angel Garduño Melo

---

A continuación se presenta una descripción general del funcionamiento del proyecto de reconocimiento facial, abordando los temas de obtención de características faciales, generación de imágenes adicionales y clasificación de rostros.

El funcionamiento se puede dividir en dos partes, antes y después del entrenamiento, como se puede observar en el diagrama siguiente:







---

## Data Augmentation

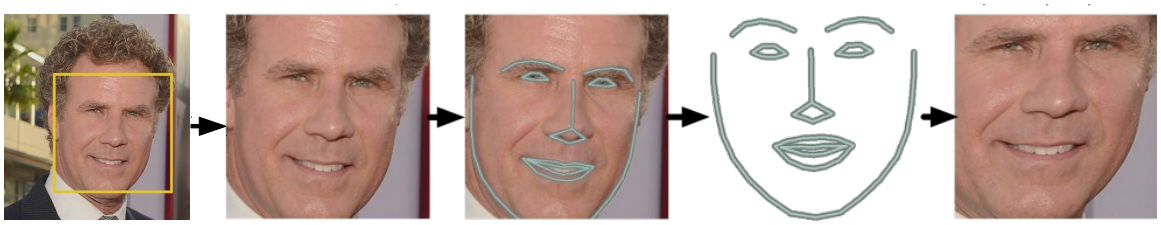
Dadas las condiciones particulares de nuestro proyecto, en la mayoría de los casos se cuenta con una o dos imágenes por cada persona que se desea agregar al sistema, lo cual limita la eficiencia de los algoritmos de clasificación. Por lo tanto, se recurrió a realizar *data augmentation*, es decir, aumentar el número de imágenes modificando las originales, mediante múltiples alteraciones, para así generar más datos de entrenamiento. Para nuestra aplicación utilizaremos un máximo de 10 imágenes (originales más generadas) por persona.

Para esta tarea se hizo uso de ImgAug: https://github.com/aleju/imgaug.

---

**Recorte y Alineación de Imágenes**

Para una mejor obtención de *features* faciales, se debe recortar y alinear la imagen, para que los puntos faciales coincidan con aquellos con los que fue entrenada la red neuronal de OpenFace. Para esto se utiliza un *pipeline* de la manera siguiente:



Primero se detecta un rostro en la imagen, dentro de esa detección se buscan los puntos faciales o *facial landmarks*. Estos son utilizados para hacer una transformación afín, para que los puntos de interés (las esquinas exteriores de los ojos y la punta de la nariz para el caso de OpenFace) se ubiquen siempre en la misma posición.
La detección de rostros y la ubicación de *landmarks* se realiza mediante Dlib: http://dlib.net/
La transformación afín y otras operaciones, se realizan con OpenCV: https://opencv.org/

---

**Obtención de Características Faciales**

Una vez alineados los rostros se procede a la extracción de *features*. Para esto se hace uso de la red neuronal FaceNet, de OpenFace, la cual fue entrenada con FaceScrub y CASIA-WebFace, dos de los datasets más grandes de rostros etiquetados con nombres. Esta red tiene como entrada una imagen que contenga un rostro alineado, y entrega como respuesta un vector de 128 características, que son representativas del rostro.
Para nuestro caso particular, se usó una implementación de este modelo en Python, a través de Keras: https://github.com/iwantooxxoox/Keras-OpenFace.

---

**Clasificación de Características Faciales**

Para clasificar las características faciales y predecir la identidad de una persona, se hace uso de un clasificador SVC (Support Vector Classification), un algoritmo de aprendizaje que genera hiperplanos para clasificar información.
Se hace uso de la librería *scikit-learn (https://scikit-learn.org/stable/)* y se implementa de la manera siguiente:

1. A cada rostro que se desea agregar se le asigna un número de identificación, que fungirá como una clase en el clasificador SVC.
2. Se crea un arreglo que contenga los vectores que corresponden a los rostros.
3. Se crea un arreglo que contenga las clases correspondientes a los vectores anteriores.

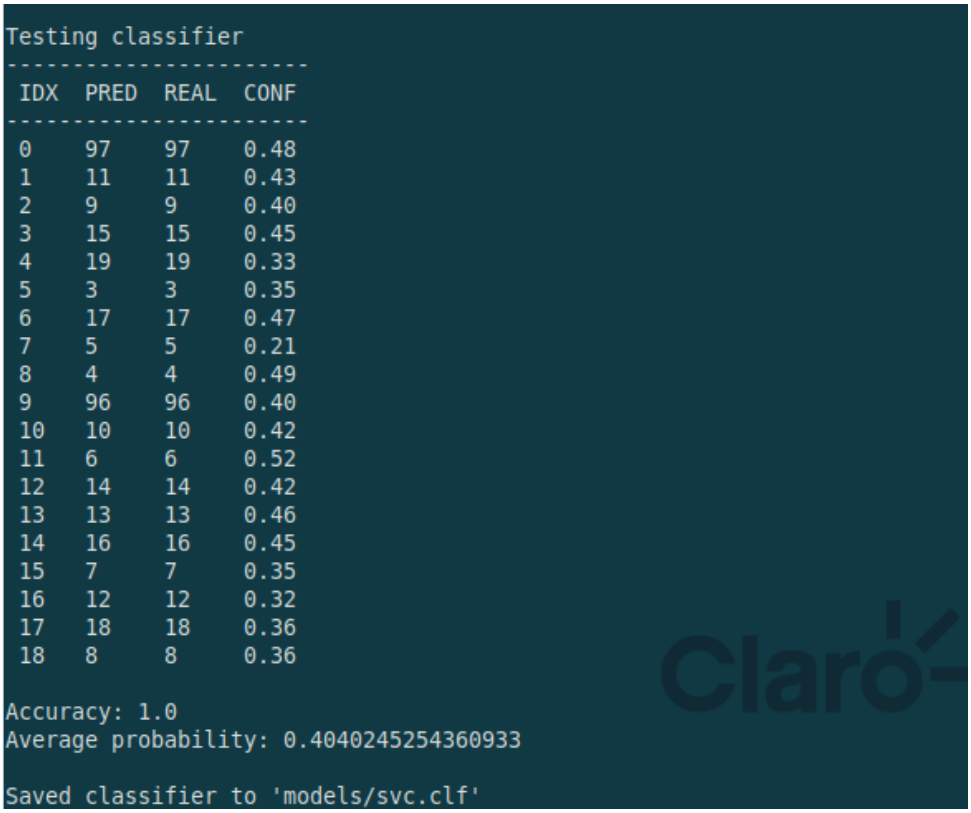
Se entrena el clasificador con un kernel *linear*, ya que se observó que tuvo un mejor desempeño, comparándolo con el *rbf*, *poly* y *sigmoid*. Además, se habilitó la opción *probability* para poder hacer uso de la función *predict_proba* posteriormente. Esta función nos permitirá conocer las probabilidades de que un vector pertenezca a las diferentes clases y de esta manera eliminar las clasificaciones que no superen un umbral determinado.

A continuación se muestra el proceso de entrenamiento y prueba del clasificador en la terminal.

```
Training classifier
[[-0.01892707  0.213447    0.04070387 ... -0.05481184  0.22469606
   0.06707605]
 [-0.01352838  0.09521868  0.06726108 ...  0.12141809  0.09070867
   0.05699403]
 [ 0.00333093  0.1501531   0.04619549 ... -0.00904334  0.24535292
   0.09847492]
 ...
 [-0.06813043  0.05702946  0.11203692 ... -0.02185293  0.11756553
   0.14276803]
 [-0.06200724  0.16717413  0.0808797  ...  0.01872292  0.15449032
   0.12496453]
 [-0.07449927  0.04082309  0.06945984 ... -0.03373935  0.04753967
   0.1591175 ]]

[97 97 97 97 97 97 97 97 97 11 11 11 11 11 11 11 11 11  9  9  9  9  9  9
  9  9  9 15 15 15 15 15 15 15 15 15 19 19 19 19 19 19 19 19 19  3  3  3
  3  3  3  3  3  3 17 17 17 17 17 17 17 17 17  5  5  5  5  5  5  5  5  5
  4  4  4  4  4  4  4  4  4 96 96 96 96 96 96 96 96 96 10 10 10 10 10 10
 10 10 10  6  6  6  6  6  6  6  6  6 14 14 14 14 14 14 14 14 14 13 13 13
 13 13 13 13 13 13 16 16 16 16 16 16 16 16 16 16  7  7  7  7  7  7  7  7
 12 12 12 12 12 12 12 12 12 18 18 18 18 18 18 18 18 18  8  8  8  8  8  8
  8  8  8]

SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape='ovr', degree=3, gamma='auto_deprecated',
  kernel='linear', max_iter=-1, probability=True, random_state=None,
  shrinking=True, tol=0.001, verbose=False)
```




Debajo de *Training classifier* se encuentran dos arreglos, el primero siendo el que contiene los vectores obtenidos con OpenFace y el segundo siendo las clases a las que pertenecen cada uno de ellos. A continuación se muestran los parámetros con los que se creó y entrenó el clasificador SVC, y por último se muestra una etapa de prueba, donde se insertan vectores que no fueron utilizados durante el entrenamiento, para probar el desempeño del clasificador. Por último, se guarda el modelo para poder ser utilizado posteriormente.

---

**Observaciones**

Se había hecho uso anteriormente del modelo de extracción de características de OpenVino, sin embargo, por problemas de compatibildad decidimos hacer uso de OpenFace en Keras, ya que sólo depende de Tensorflow para funcionar con Python.
Al homogeneizar el número de imágenes por persona, la probabilidad de detección correcta disminuyó. Una causa posible es que al generar un mayor número de vectores para el entrenamiento del SVC, este es más preciso, es decir, muy pocas veces genera falsos positivos, pero la probabilidad promedio de una detección correcta fue 0.4.