

Biometría "Detección de Rostros"

2018

Sebastián Correa Echeverri

1 Introducción

Se propone como ejercicio de práctica de la asignatura Biometría del Master en inteligencia artificial reconocimiento de formas e imágenes digital, implementar un NN-Based Face Detection.

1.1 Datos

Se dispone de una base de datos con caras y no-caras a tamaño fijo (24x24), entregada por el docente. Adicionalmente se extiende la base de datos para obtener una mejor generalización del modelo.

1.1.1 Iniciales

Base de datos inicial (entregado por el docente) contiene 6099 caras y 10.000 no caras de 24x24 pixeles. Los datos están normalizadas con media igual a 0 y desviación estándar a uno.

1.1.2 Datos Extras

Se agrego una base de datos de rostros “in the wild” llamada Large-scale CelebFaces Attributes (CelebA) Dataset [1]. Esta base de datos cuenta con un total de 202,599 rostros con diferentes atributos; gafas, sombreros, diferente estilos de barbas, entre otros.



Ilustración 1 Características CelebA

Para utilizar estos datos con los previamente mencionado en la sección 1.1.2, Los rostro deberían estar recortados como se muestra en Ilustración 2. Para esto se utilizo la librería

Dlib y el modelo shape_predictor_68_face_landmarks para hacer un reconocimiento de los rostros, transformar a escala de grises, normalizar con media=0 y sd=1 y dimensionarlos 24x24. Se obtiene un total de 196.387 rostros.

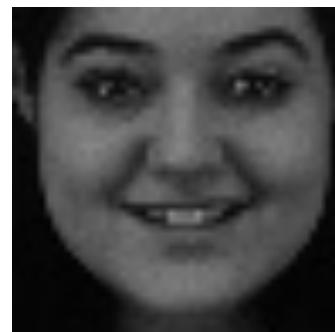


Ilustración 2 Rostro 24x24

Finalmente uniendo ambos dataset se obtiene 202.486 caras y 10.000 no caras, debido a la desproporción entre caras y no caras se agrego el dataset Scene Categories Dataset [2].



Ilustración 3 Ejemplo 8 Scene Categories Dataset

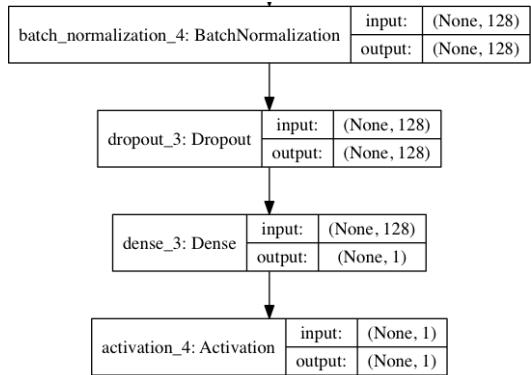
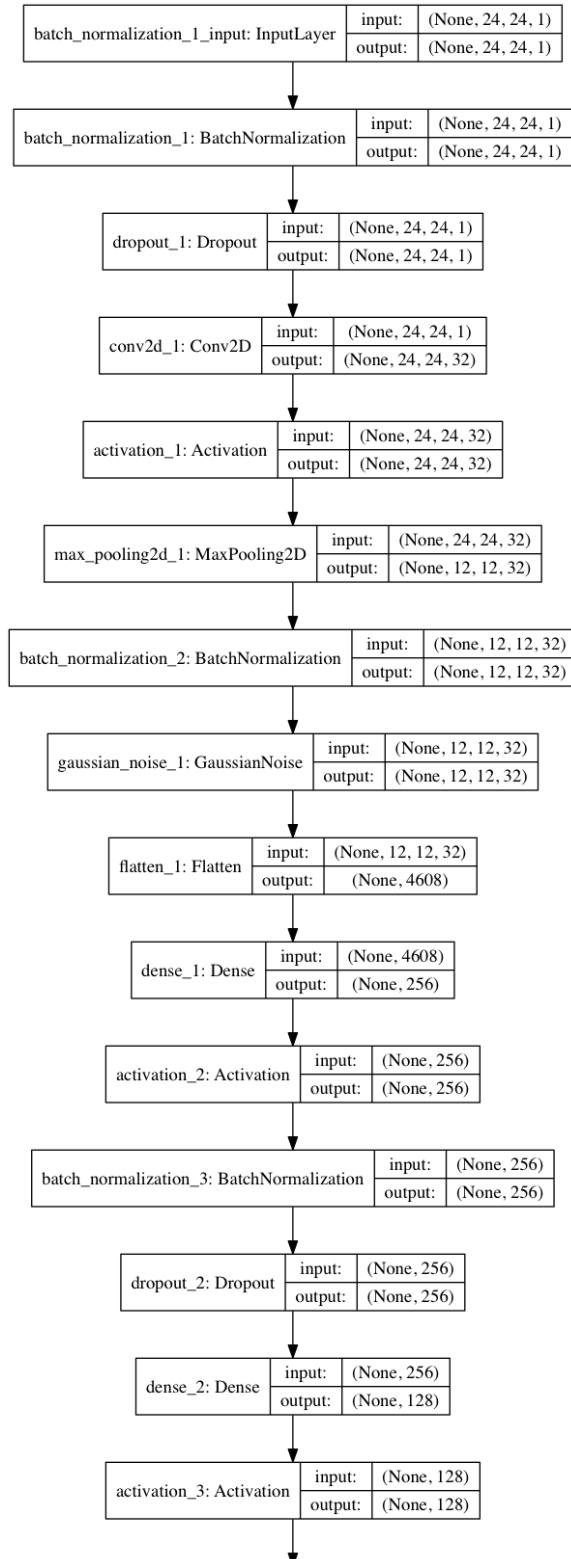
Este dataset contiene 8 categorías de escenas al aire libre: costa, montaña, bosque, campo abierto, calle, ciudad interior, edificios altos y autopistas, con un total de 2600 imágenes de 256x256 pixeles. Cada imagen de este dataset se dividió en 100 cuadros de 24x24, generando un total de 268.800 de no rostros.

Finalmente se trabajo con un dataset de 202.486 caras y 278.800 no rostros.

2 Entrenamiento del Modelo

Para el entrenamiento del modelo se realizo un división de train y validation del 80/20 porciento de los datos. El set de datos de entrenamiento quedo con un total de 385.028 y el de validación con un total de 96.258.

Como modelo de detección se uso un modelo CNN con la siguiente arquitectura:



Se agrego un batch normalization y un dropout al inicio para que modelo y tuviera una entrada con un poco de ruido. Se agrego una convolución y dos capas fully connected y una función de activación de sigmoid. Para las pruebas del modelo se utilizo un threshold de 0.5.

Adicionalmente para el entrenamiento se utilizó data-augmentation para hacer el modelo mas capaz de generalizar casos, se aplico un 20% de movimiento horizontal y vertical en la imagen, mas un flip en el eje vertical.

Se uso binary crossentropy como función loss. Como optimizador se utilizo Adama con un learning rate de 0.01, un batch_size de 128 y se realizó un entrenamiento de 100 épocas. El modelo tienen un total de 1.214.083 parámetros entrenables.

Como resultado se obtiene un accuracy de validación de 99.82% y de entrenamiento de 99.53%.



Gráfica 1 Accuracy durante entrenamiento

3 Implementación

Para la ejecución del programa se necesitan de dos parámetros uno obligatorio y el segundo opcional, el primero hace referencia la ruta donde se encuentra la imagen a

analizar y el segundo permite probar diferentes pixeles de recorrido de ventana (ver sección 3.1.1).

- python facedetection.py -i data/test5.jpg -p 1

La implementación del sistema puede dividirse en tres secciones: división de imágenes, predicción y delineado de rostros detectados.

3.1.1 División de imágenes

La primera sección del algoritmo recorre la imagen propuesta para detección con una venta de 24x24 pixeles a diferentes escalas realizando un crop para posteriormente identificar si en dicha venta hay o no un rostro. Inicialmente se planteo un valor fijo para el barrido realizado por la venta 1,2, 3 pixeles, entre menor mas precisión pero igualmente hacia mas lento el sistema para recortar las imágenes y para predecirlas ya que podía generar 400mil crops en algunos casos.

Para solucionar este inconveniente se decidió inicialmente reducir el tamaño de la imágenes siendo el máximo 480 de ancho, se eligió este valor ya que es múltiplo de 24 el tamaño de la ventana, la altura se ajusta dependiendo de la imagen para no afectar el aspect ratio de esta. A si mismo al tratar con diferentes imágenes se presentaba el caso que pixeles muy altos se ajustaban a algunas y valores bajos a otras, buscando un valor optimo para todos se decidió usar el entero mas pequeño entre la altura y el ancho divido 100, en imágenes inferiores a 100 pixeles 1 pixel no representa mucho pero una imagen 2048 pixeles un pixel es demasiado pero 20 seria un valor mas apropiado. En otras palabras cada imagen seria dividida en 100 partes como mínimo.

```
if template.shape[0]>template.shape[1]:
    swing = template.shape[1]
else:
    swing = template.shape[0]

divh = int(swing/100)
divw = int(swing/100)
```

El crop realizado a la imagen de mayor tamaño (480 pixeles) puede tomar un tiempo 0.2 segundos, pero al realizar la

normalización con media cero y desviación estándar 1 aumenta mucho el tiempo de corte, el usar scale de Sklearn lo hacia muy lento, 20-30 segundos por lo que se opto por usar el “mean” y “std” de numpy optimizando el tiempo a 2-10 segundos.

```
for scale in delta:
    # Resize the image according to the scale
    resized = scipy.misc.imresize(template, scale)

    if resized.shape[0] < 24 or resized.shape[1] < 24:
        break
    h = resized.shape[0]-24
    w = resized.shape[1]-24
    for y in np.arange(0,h,divh):
        for x in np.arange(0,w,divw):
            crop_img = resized[y:y+24, x:x+24]
            if crop_img.std() > 40:
                if crop_img.shape[0] < 24 or crop_img.shape[1] < 24:
                    crop_img = cv2.resize(crop_img, (24, 24))
                crop_image.append((crop_img-np.mean(crop_img, axis=0))/np.std(crop_img, axis=0))
                crop_image.append(crop_img)
                crop_boxes.append([x,y,scale])
```

Para la elección de la escala se probó diferentes conjuntos de 0-1 con delta de 0.05 obteniendo veinte valores y de 0-1 con deltas de 0.1 obteniendo diez. Para el primero se obtiene muy buenos resultados pero toma mas tiempo que el segundo por lo que se llegó a un punto medio donde nos pierde tanta información y se obtienen buenos resultados, de 0-1 con deltas de 0.07 obteniendo 15 valores.

Adicionalmente para no obtener muchos crops, se busco una manera para desechar algunos de forma previa a la predicción. Con esto en mente se decidió desistir de los valores que tuvieran una desviación estándar muy pequeña que significa que todos los pixeles de ese crop tiene valores similares y pueden estar haciendo referencia a fondos no caras donde la desviación estándar era alta ya que hay bastante diferencia entre los pixeles que la componen. De forma empírica se llegó que 40 era un valor que reducía las muestras y que no afectaba el rendimiento del algoritmo. Finalmente se obtiene un vector con los crops y otro con el origen del crop y la escala a la que se detectó.

3.1.2 Predicción

La segunda sección del algoritmo ejecuta un proceso de predicción usando el modelo presentado en la sección 2. Como entrada es un vector con todos los crops previamente mencionados y su salida es un vector con la salida de la sigmoid para cada crop.

3.1.3 Delineado de rostros detectados.

La tercera sección del algoritmo selecciona los crops que superan un threshold y guarda el centro del crop escalado a la imagen original. A este vector centros se le buscan y promedian todos los crops que tenga su centro dentro de un radio de similitud de x pixeles, siendo x el menor valor entre altura y ancho dividido 10. Este proceso se realiza dos veces con el fin de reducir los posibles hipótesis y buscar el rostro.

```
for box in center_faces:  
    x = box[0]  
    y = box[1]  
    count = 1  
    for all_box in center_faces:  
        if abs(box[0]-all_box[0]) < swing/10 and abs(box[1]-all_box[1]) < swing/10:  
            x += all_box[0]  
            y += all_box[1]  
            count+=1  
  
    new_boxes.append([int(x/count), int(y/count)])  
new_boxes = np.unique(np.array(new_boxes), axis=0)
```

Finalmente se dibujan las hipótesis presentes y se muestra la imagen con dichas de color blanco.

4 Resultados

Se realizaron 6 pruebas en total para comparar los diferentes valores y parámetros y probar el detector con dichos parámetros tuneados.

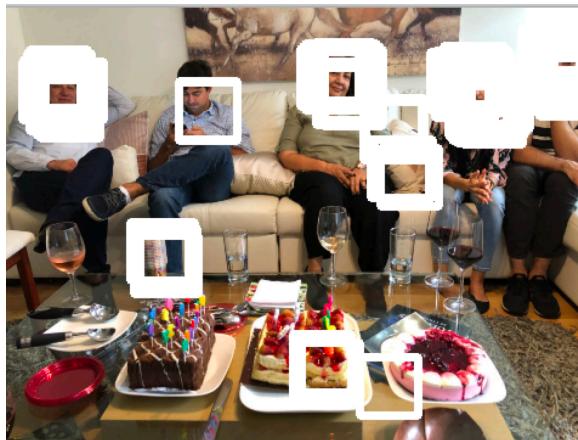


Ilustración 4 Prueba con movimiento de venta de 1 pixel
tiempo 3 min

Al variar el threshold relacionado con la desviación estándar de la imagen se obtuvieron diferentes resultados, con un valor mayor a 80 pixeles no se reconocían rostros. Con un valor menor a 10 pixeles se aumentaba el tiempo un 30 % mas que con 40 pixeles.



Ilustración 5 Threshold std > 10

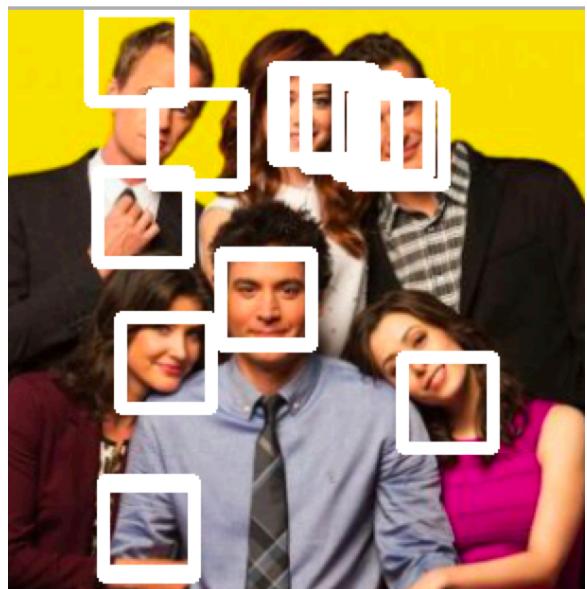


Ilustración 6 Threshold de predicción igual 0.5

A continuación se muestra los resultados para cuatro imágenes obtenidas de forma aleatoria y no usadas para el entrenamiento.



Ilustración 7 Resultado 1, tiempo 11 segundos



Ilustración 10 Resultado 4, tiempo 17 segundos



Ilustración 8 Resultado 2, tiempo 15 segundos

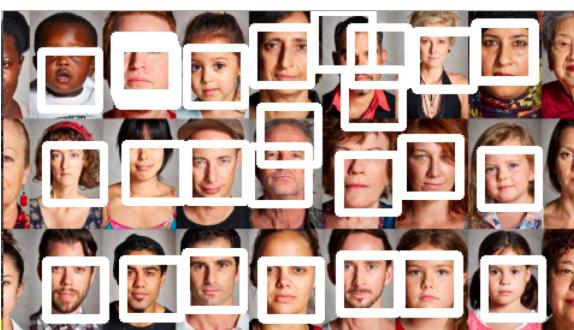


Ilustración 9 Resultado 3, tiempo 45 segundos

5 Conclusiones

Lo que mas tiempo tardaba era en la sección de predicción esto se debe en parte a la dinámica del modelo, puede que un modelo NN y no CNN podría obtener los mismos resultados y con menor cantidad de parámetros y mas rápido para la predicción.

Adicionalmente en el momento del crop la normalización de los crops es lo que mas tiempo toma, hacer un modelo sin normalización media cero y desviación 1 podría mejorar la velocidad.

También se puede mejorar la predicción usando un algoritmo mas complejo para descarta los crops con menor probabilidad de ser rostro.

6 Referencias

- [1] Z. L. P. L. X. W. y X. T. , «Deep Learning Face Attributes in the Wild,» *Proceedings of International Conference on Computer Vision (ICCV)*, 2015.

- [2] A. T. y A. O. , «Modeling the Shape of the Scene: A Holistic Representation of the Spatial Envelope,» 2001.