Reconocimiento Facial con RNA Convolucionales

Rodríguez Hernández Josué

El desbloqueo de los dispositivos móviles por medio del reconocimiento facial es cada vez más común hoy en día pero, ¿Cómo es que se logra esto?. En este trabajo se realiza paso a paso una de las técnicas con la cual el reconocimiento facial es posible.

RNA Convolucional | Reconocimiento facial | Modelo | Transfer-Learning | Base de datos | Características

Introducción

El objetivo principal de este trabajo es entrenar una RNA - red neuronal (desde cero) que reconozca el rostro de una persona en específica. Debido a que no contamos con suficientes imágenes del rostro de la persona a la que queremos identificar, haremos uso de la base de datos **CelebA**, que contiene imágenes de rostros etiquetados con 40 características o atributos. Entonces, primero entrenaremos una red convolucinal capaz de predecir los atributos de esta base de datos, para después utilizarla como base para entrenar una nueva red, la cual será capaz de identificar solo si es el rostros o no de la persona que queremos identificar, haciendo uso de una técnica llamada **Transfer-Learning**.
Es importante mencionar que las redes que se entrenaron en este trabajo, solo son alimentadas con fotos de rostros, es decir, solo se centran en la extracción de características y no en otras étapas del reconocimiento facial como lo son la detección y alineación de rostros.

Estrategia empleada

Durante todo el desarrollo de este trabajo, se hicieron cuatro Scripts de python, cada uno enfocado a una tarea o problema en específico que se fueron presentando conforme a la marcha de este proyecto. A continuación se describe la función de cada uno de ellos.

A. Cargar datos.py. En este script, como su nombre lo menciona se cargan los datos de la base de datos **CelebA**, además de darles un pequeño preprocesamiento para obtener mejores resultados a la hora de entrenar nuestra red.
CelebA se divide en dos archivos principales, el primero es la carpeta 'img_align_celeba', la cual contiene más de $200K$ imágenes de rostros de distintas personas y el segundo es el archivo 'list_attr_celeba.txt', el cual contiene todos los atributos de cada una de las imágenes de rostros identificados con el nombre de cada una de ellas, etiquetados con 1 y -1 para tiene o no tiene el atributo respectivamente. Al convertir el archivo 'list_attr_celeba.txt' en un dataframe surgen problemas con el seperador, ya que, hay dobles espacios en el archivo, por lo que en este script se eliminan esos dobles espacios.

Un vez teniendo el dataframe de los atributos listos, la función principal de este script es obtener un objeto **Dataset** de tenserflow, que conteniene las imágnes cargadas con sus respectivos atributos etiquetados.

B. Red conv - Reconocimiento facial.py. Aquí se vuelven a cargar los datos ya preparados con la técnica empleada en el script anterior, adicionandoles algunos métodos incluidos en tensorflow para optimizar la carga de los mismos en el proceso de entrenamiento. Una vez hecho esto, se crea un modelo de 4 capas convolucionales 2D con 20 filtros, una ventana de atención de 4x4 y un maxpooling de 2x2; seguidas de una capa densa de 128 neuronas y una capa de salida densa de 40 neuronas. La función de activación empleada en cada una de las capas es 'relu', a excepción de la capa de salida donde se emplea una función de activación 'sigmoide'.

```
=================================================================
conv2d (Conv2D)              (None, 177, 177, 20)      980

max_pooling2d (MaxPooling2D  (None, 88, 88, 20)        0
)

conv2d_1 (Conv2D)            (None, 85, 85, 20)        6420

max_pooling2d_1 (MaxPooling  (None, 42, 42, 20)        0
2D)

conv2d_2 (Conv2D)            (None, 39, 39, 20)        6420

max_pooling2d_2 (MaxPooling  (None, 19, 19, 20)        0
2D)

conv2d_3 (Conv2D)            (None, 16, 16, 20)        6420

max_pooling2d_3 (MaxPooling  (None, 8, 8, 20)          0
2D)

flatten (Flatten)            (None, 1280)              0

dense (Dense)                (None, 128)               163968

dense_1 (Dense)              (None, 40)                5160

=================================================================
Total params: 189,368
Trainable params: 189,368
Non-trainable params: 0
```

Fig. 1. Modelo summary –> Ren conv - Reconocimiento facial

Este modelo se configura con una función de costo 'binary_crossentropy', un optimizador 'Adam' y una métrica 'binary_accuracy'. Se entrena durante un total 25 épocas divididas en dos partes una de 10 y otra de 15, con un batch zise de 50. En las figuras 2 y 3, se puede apreciar el aprendizaje de este modelo representado en las gráficas de la función de costos y de la accuracy respectivamente.

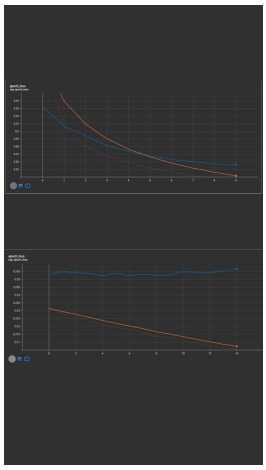

Fig. 2. Graficas de la función de costo vs epochs del entrenamiendo del modelo. Arriba se muestra el aprendizaje durante las épocas del 1-10 y abajo el aprendizjae durante las épocas de la 11-25.

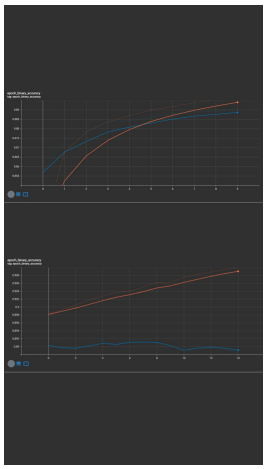

Fig. 3. Graficas de la accuracy vs epochs del entrenamiendo del modelo. Arriba se muestra el aprendizaje durante las épocas del 1-10 y abajo el aprendizjae durante las épocas de la 11-25.

Es importante mencionar que este modelo empezó a obtener buenos resultados hasta que se optó por ocupar la función de costo y el optimizador mencionados anteriormente, y se cambiaron las etiquetas de los atributos de -1 a 0. Además, se ocuparon solo el 10 por ciento de los datos de **CelebA** y se usaron solo el 80 por ciento para entrenar y el restante para validación, gracias a esto el tiempo de entrenamiento se redujo considerablemente.

C. Red conv - Fotos mias.py. En este Script se entrena la RNA principal de este trabajo, es decir, la que es capaz de distinguir si es o no la imagen del rostro de la persona elegida. Para ello, se construye un nuevo conjunto de datos llamado 'Dataset', conformado de un total de 422 imáganes, de las cuales solo 206 son de la persona elegida y las demás de distintas personas extraidas de **CelebA**.
Se cargan los datos haciendo uso de **ImageDataGenerator** (objeto de tensorflow), ya que de esta manera no solo podemos cargar imágenes a partir de un derectorio previamente preparado, sino que también es posible aumentar el número de imágenes que tenemos cintéticamente, haciendo transformaciones como reescalamiento, rotaciones, zoom, etc; y sin la necesidad de guardalas en el disco duro. Además también nos permite clasificar nuestras imágenes de acuerdo al número de subdirectorios que tengamos en el directorio principal.

Una vez teniendo los datos ya listos para entrenar, cargamos el modelo entrenado **Red conv - Reconocimiento facial.py** y mandamos a llamar todas sus capas convolucionales ya descritas anteriormente para entrenar nuestra nueva red, es decir, se excluyen las capas densas del modelo anterior. A este nuevo modelo se le agrega una capa densa de 113 neuronas con una función de activación 'relu' y se le asigna una capa de salida densa de una solo neurona con una función de activación 'sigmoid'.

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 177, 177, 20)	980
max_pooling2d (MaxPooling2D)	(None, 88, 88, 20)	0
conv2d_1 (Conv2D)	(None, 85, 85, 20)	6420
max_pooling2d_1 (MaxPooling2D)	(None, 42, 42, 20)	0
conv2d_2 (Conv2D)	(None, 39, 39, 20)	6420
max_pooling2d_2 (MaxPooling2D)	(None, 19, 19, 20)	0
conv2d_3 (Conv2D)	(None, 16, 16, 20)	6420
max_pooling2d_3 (MaxPooling2D)	(None, 8, 8, 20)	0
flatten (Flatten)	(None, 1280)	0
dense (Dense)	(None, 113)	144753
dense_1 (Dense)	(None, 1)	114

Total params: 165,107
Trainable params: 144,867
Non-trainable params: 20,240

Fig. 4. Model summary –> Red conv - Fotos mias

Como se muestra en la figura 4 hay algunos parametros de la red que no son entrenables, esto es debido a que se congelan los parámetros las capas convolucionales del modelo anterior, ya que, estos ya fueron entrenados en la sección anterior, porque únicamente llos parámetros de las últimas de capas de este nuevo modelo son ajustados durante el entrenamiento.

Este modelo se configura con una función de costo 'binary_crossentropy', un optimizador 'Adam' y una métrica 'accuracy'. Se entrena durante un total 15 épocas con un batch zise de 20. En figura 5, podemos apreciar los buenos resultados que obtuvo esta red, ya que desde la primera época obtuvo un accuracy de 1 en los datos de validación, es decir, clasificó perfectamente cada una de las imagenes de validación.

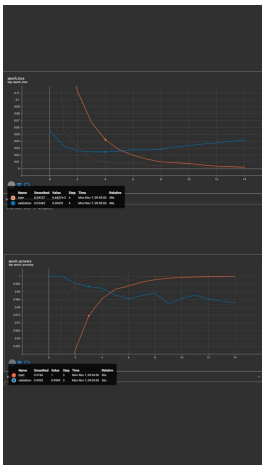

Fig. 5. Grafica de la función de costo vs epochs y de la accuracy vs epochs del entrenamiento del modelo repectivamente.

Es importante mencionar que no hubo la necesidad de hacerle más pruebas a este modelo debido a los resultados obtenidos.

D. Evaluacion del modelo.py. En este último script se ponen a prueba los resultados obtenidos del modelo entrenado en la sección anterior, ya que, aunque por muy buenos que estos parezcan, el comportamiento visto en la gráfica de la accuracy vs epochs (figura 5) no es muy común al ahora de entrenar una RNA. Por lo tanto, debemos de estar seguro de si lo que predice este modelo es realmente lo que queremos (reconocer el rostro de una persona en especifica). Para ello, se crea un pequeño conjunto de datos llamado 'test_set', en el cual se pondrán aprueba 10 imágenes de la persona elegida contra 20 fotos de extraños.
Se cargan los datos de este nuevo directorio haciendo uso de la libreria Pillow de pyton en una matriz numpy de python, etiquetando con 1 las fotos de la persona elegida y 0 para el caso contrario. Una vez hecho esto, se carga el modelo entrenado en **Red conv - Fotos mias.py** y se evalua con este nuevo conjunto de imágenes haciendo uso de **model.evalute**. El modelo logra casificar nuevamente bien todas las imágenes de 'test_set' (ver figura 6), lo que nos comprueba que realmente es capaz de reconocer el rostro de la persona elegida.

```
1/1 [==============================] - 0s 340ms/step - loss: 5.3361e-38 - accuracy: 1.0000
Test loss: 5.3361198892959 31e-38
Test accuracy: 1.0
[[1.]
 [1.]
 [1.]
 [1.]
 [1.]]
[1. 1. 1. 1. 1.]
```

Fig. 6. Salida de la terminal al ejecutar Evaluacion del modelo.py

Por último, solo para verificar nuevamente el modelo se hace uso de **model.predict**, pasandole 5 imágnes de la persona elegida y viendo si capaz de reconocerla. Lo cual nuevamente lo logra, por lo que todas nuestras dudas y sospechas de nuestro modelo son descartadas, dejandolo listo para desbloquear nuestro dispositivo móvil.

Conclusión

El reconocimiento facial sin duda es una tecnologia muy interesante y poderosa, gracias a este trabajo se pudo apreciar que las RNA se ajustan perfecto este tipo de tareas, logrando resultados bastante buenos a pesar de no contar con conjuntos de datos lo suficientemente grandes. Además, se logra ver lo poderosas que son las RNA convolucionales para la clasificación de imágenes y lo útil que es el **Transfer-Learning** para entrenar modelos con objetivos o funciones similares a los de otros modelos pre-entrenados.