Deep Age Detector

Martín Alejandro Castro Álvarez

Universidad Internacional de Valencia (VIU)

Calle Pintor sorolla, 21 46002, Valencia (España)

martincastro.10.5@gmail.com

https://github.com/MartinCastroAlvarez/deep-age-classifier

Abstract. Este paper describe la implementación de un estimador de la edad a partir de imágenes digitalizadas, mediante técnicas de Deep Learning y Transfer Learning. Construyendo sobre el modelo pre-entrenado, conocido como *ResNetV2,* elegido por su capacidad de manejo de imágenes, se desarrolla un clasificador capaz de distinguir entre tres categorías de edad: "OLD", "MIDDLE", y "YOUNG". El dataset utilizado, permite la transformación de imágenes en tensores, y su clasificación es evaluada mediante una matriz de confusión.

Keywords: Computer Vision, Deep Learning, Transfer Learning, Supervised Learning, Facial Recognition, Convolutional Neural Networks, Deep Residual Learning, Image Processing, Model Optimization, Confusion Matrix, Image Classification, Dimensionality Reduction, Training and Validation, Model Performance, Accuracy, Age Classifier.

1. Introducción

1. Problema

Detectar la edad de una persona a partir de una imagen es una habilidad casi intuitiva en los seres humanos. Sin embargo, programar a una computadora con esta capacidad es un desafío; no solo es una tarea subjetiva que depende de la percepción del observador, sino que debe, primero, resolver el problema de representación digital de una imagen.

1.2. Deep Learning

Deep Learning es una herramienta muy potente para esta tarea [1], gracias a su habilidad para encontrar patrones en datos de alta dimensionalidad.

1.3. Transfer Learning

La detección de la edad en imágenes es una tarea que se beneficia del aprendizaje previo de modelos computacionales de identificación de objetos. Transfer Learning permite [2] la reutilización de modelos previamente entrenados para tareas similares, tal y como se emplean librerías para acelerar el proceso de creación de software.

1.4 Supervised Learning

Dada la naturaleza del problema, es necesario que un ser humano le indique al ordenador cuál es la clasificación de cada imagen. Como resultado, es necesario utilizar un algoritmo de aprendizaje supervisado.

2. Datos

2.1 Fuente

El dataset, con licencia de dominio público, ha sido descargado desde la plataforma Kaggle [3], y se compone de dos partes: En primer lugar, se descarga un archivo CSV contiene un atributo *"ID"* con la identificación de la imagen, y otro atributo "Class" con la clasificación. En dicho atributo hay 3 valores posibles: "*OLD"*, "*MIDDLE"*, "*YOUNG"*. En segundo lugar, se descarga una carpeta con imágenes, cuyos nombres se encuentran referenciados por el atributo *"ID"* del CSV.

2.2 Limpieza de Datos

No hay duplicados y no hay faltante de datos. La única tarea a realizar es la de cargar las imágenes en memoria en formato tensorial. Cada imagen se corresponde con un vector de dimensiones $(299, 299, 3)$. Los primeros dos componentes corresponden a la altura y anchura de la imagen, y el tercer componente corresponde con los 3 canales de la imagen: *RED*, *GREEN*, *BLUE*. Finalmente, el dataset se divide en 3 partes: Por un lado, el dataset de entrenamiento *"train_set"* que será utilizado en el entrenamiento de la red neuronal. Por otro lado, el dataset de validación *"val_set"* que será utilizado para calcular las métricas de rendimiento al final de cada epoch. Finalmente, el dataset de prueba *"test_set"* que será utilizado para calcular una estimación insesgada del rendimiento del modelo al final de la fase de entrenamiento.

2.3 Reducción de la Dimensionalidad

Se ha utilizado una capa convolucional debido a su eficiencia para reducir la dimensionalidad de las imágenes [4]. Dicha capa permite calcular una suma ponderada de los valores representativos de la intensidad de luz de cada píxel.

3. Optimización

3.1 Deep Residual Learning

El modelo "*ResNetV2"* es utilizado para preprocesar los valores de entrada de la red neuronal, debido a su gran capacidad para procesar imágenes en el contexto de Deep Learning [5]. Dicho modelo ha sido adaptado para formar parte de una red neuronal aún mas grande, que intenta resolver otro problema más específico.

3.2 Input & Output

Las dimensiones de entrada corresponden con las dimensiones de cada imagen, tal y como ha sido indicado en 2.2. Las dimensiones de salida son 3: Una por cada una de las categorías en las que una imagen puede ser clasificada. La capa de activación de la última capa es una *softmax* debido a que esto asegura que la suma de los valores de las 3 salidas es siempre 1.0 [2]. De este modo, podemos utilizar dichas salidas como probabilidades de que la imagen se pertenezca a cada una de las 3 categorías.

3.3 Entrenamiento

En la fase de entrenamiento se han utilizado 10 epochs. Es decir, se ha procesado el dataset completo 10 veces. Por otra parte, el tamaño del lote es de 64 imágenes. Es decir, se necesita procesar dicha cantidad de imágenes antes de actualizar los pesos mediante el algoritmo de backpropagation. Además, se ha agregado una política de aborto temprano en caso de que el rendimiento caiga por debajo de cierto valor a lo largo de 2 epochs. La función de costo premia las clasificaciones correctas y castiga las clasificaciones incorrectas.

3.4 Métricas

Se ha monitoreado el rendimiento de la red durante la fase de entrenamiento (Ver Fig. 1). Se ha observado que el rendimiento sobre el dataset de entrenamiento se distancia significativamente del rendimiento sobre el dataset de validación (Ver Fig. 2). Dicho comportamiento podría indicar una falla del modelo para generalizar (overfitting). Por otra parte, la evolución de otras métricas de precisión y recall indican si el modelo aumenta su cantidad de falsos positivos, o si hay más casos positivos que no logra predecir correctamente (falsos negativos).

Afortunadamente, a pesar de que la exactitud ha caído con respecto al dataset de entrenamiento, no ha sucedido lo mismo con el rendimiento sobre el dataset de validación y el de prueba.

4. Resultados

4.1 Confusion Matrix

La matriz de rendimiento compara los valores reales con los predichos para cada una de las categorías. (Ver Fig. 3). Los valores en las diagonales deberían ser los que cuenten con mayor número de casos. Esto indicaría que el modelo predice correctamente los resultados. Sin embargo,

4.2 Conclusiones

Este modelo podría ser utilizado para verificar la edad de los usuarios a partir de una imagen capturada por la cámara de cualquier dispositivo moderno. De esta manera, es posible restringir la utilización del software de acuerdo a distintas políticas de edad

5. Referencias

[1] Aggarwal, C. C. (2016). Recommender Systems: The Textbook. Springer.
[2] Géron, A. (2017). Hands-On Machine Learning with Scikit-Learn and TensorFlow. O'Reilly
[3] Kaggle. "Faces: Age Detection Dataset".
https://www.kaggle.com/datasets/arashnic/faces-age-detection-dataset
[4] Ng, A. Deep Learning Specialization. Coursera. https://www.coursera.org/specializations/deep-learning
[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun (2015). "Deep Residual Learning for Image Recognition". ArXiv

6. Anexo

```
Epoch 1/5
38/38 [==============================] - 319s 9s/step - loss: 8.2393 - categorical_accuracy: 0.4409 - precision: 0.4415 - recall: 0.4371 - va
l_loss: 5.0124 - val_categorical_accuracy: 0.5014 - val_precision: 0.5021 - val_recall: 0.5007
Epoch 2/5
38/38 [==============================] - 303s 8s/step - loss: 7.0386 - categorical_accuracy: 0.4513 - precision: 0.4522 - recall: 0.4463 - va
l_loss: 14.2973 - val_categorical_accuracy: 0.5091 - val_precision: 0.5091 - val_recall: 0.5091
Epoch 3/5
38/38 [==============================] - 300s 8s/step - loss: 14.4326 - categorical_accuracy: 0.4421 - precision: 0.4431 - recall: 0.4409 - v
al_loss: 39.0313 - val_categorical_accuracy: 0.3510 - val_precision: 0.3510 - val_recall: 0.3510
Epoch 4/5
38/38 [==============================] - 299s 8s/step - loss: 19.9811 - categorical_accuracy: 0.4367 - precision: 0.4366 - recall: 0.4350 - v
al_loss: 19.1546 - val_categorical_accuracy: 0.5259 - val_precision: 0.5259 - val_recall: 0.5259
```

Fig. 1: Muestra la evolución de las métricas de rendimiento por cada uno de los Epochs

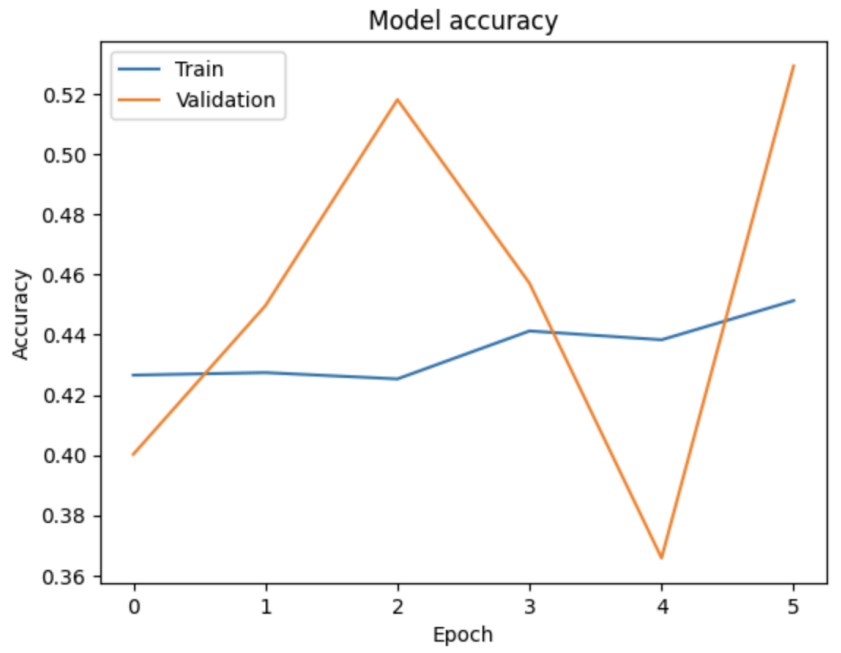

Fig. 2: Representación gráfica de la métrica de Accuracy a lo largo de los Epochs. En azul, el rendimiento sobre el dataset de entrenamiento. En naranja, el rendimiento sobre el dataset de validación. Cuando el rendimiento naranja se encuentra por debajo de la azul, indica overfitting.

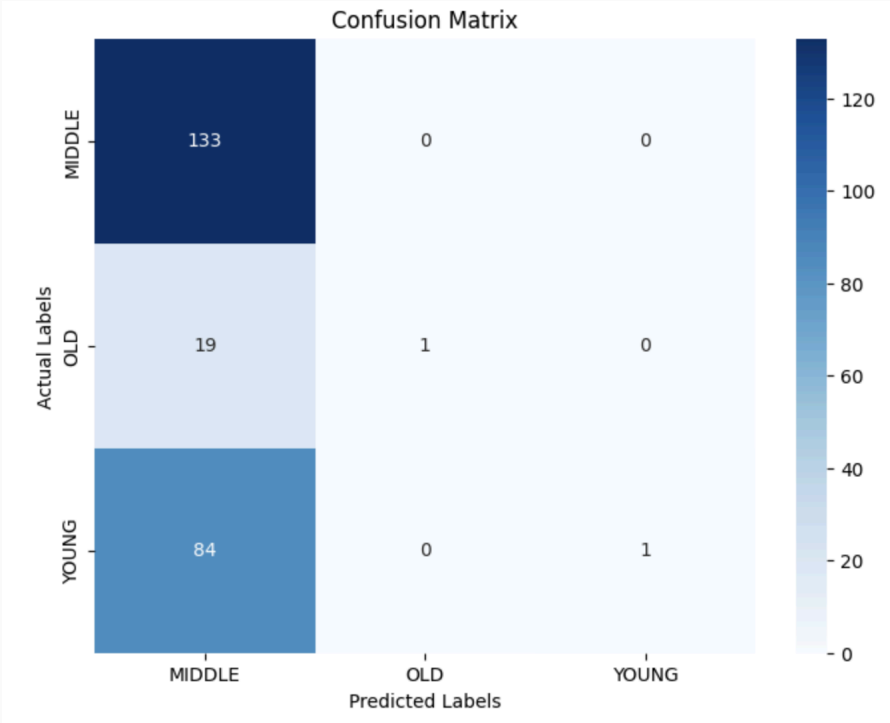

Fig. 3: Matriz de confusión, comparando la clasificación propuesta por el modelo de predicción, y la categoría actual de acuerdo al set de datos.

Make a prediction based on a random image from the training set

[26]:
```
random_id = random.randint(0, raw_df.shape[0])
print('Using image at position:', random_id)
random_row = raw_df.loc[random_id]

path = os.path.join('Data', random_row.ID)
image = load_img(path, target_size=(299, 299))

image_array = np.expand_dims(img_to_array(image), axis=0)
class_probabilities = model.predict(image_array)

for index, probability in enumerate(class_probabilities[0]):
    print(f'Probability of being {classes[index]} is: {int(100 * probability)}')

image
```

```
Using image at position: 19439
1/1 [==============================] - 3s 3s/step
Probability of being MIDDLE is: 100
Probability of being OLD is: 0
Probability of being YOUNG is: 0
```

[26]:

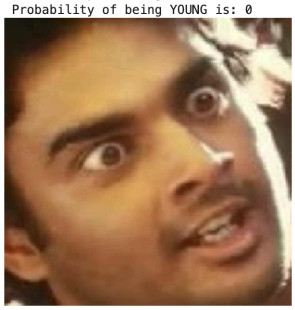

Fig. 4: Ejemplo de una predicción, luego de la fase de entrenamiento. El modelo clasifica correctamente que el hombre de la foto es de mediana edad.