
Implementación de una red neuronal convolucional para predecir si una araña posee importancia médica basado en respuestas de una cuenta de Twitter.

José Alejandro López Quel
21001127
Universidad Galileo
Statistical Learning II

Abstract

1 En este paper, se presenta un caso de predicción de la importancia médica de
2 una araña basada en las respuestas de un aracnólogo en su cuenta de Twitter de
3 imágenes de arañas compartidas. Se extrae los datos de la cuenta de Twitter y se
4 etiqueta cada una de ellas para realizar entrenar una red neuronal convolucional. Por
5 lo que se analiza en este trabajo el mejor método para realizar dicha clasificación.

6 1 Introducción

7 La clasificación de imágenes es la tarea de asignar a una imagen de entrada, una etiqueta de un
8 conjunto fijo de categorías. Se trata de uno de los problemas centrales de la visión por computadora
9 que, a pesar de su simplicidad, tiene una gran variedad de aplicaciones prácticas. Las redes
10 neuronales convolucionales (CNN por sus siglas en ingles) se han convertido en una de las técnicas
11 de visión por ordenador más avanzada. Entre los distintos tipos de redes neuronales (otros son las
12 redes neuronales recurrentes (RNN), las de memoria a corto plazo (LSTM), las redes neuronales
13 artificiales (ANN), etc.), las CNN son las más utilizadas.

14
15 En este trabajo se plantea emplear una CNN para resolver el problema planteado de clasi-
16 ficar imágenes de arañas y su importancia médica. Por lo que se cuenta con un dataset de 300
17 imágenes y el texto extraído para cada tweet compartido con la cuenta @Arachno_Cosas. Los
18 datos son muy importantes en lo que respecta a los modelos de aprendizaje profundo. El modelo de
19 clasificación de imágenes tiene muchas más posibilidades de funcionar de manera más óptima si se
20 tiene una buena cantidad de imágenes en el conjunto de entrenamiento. Además, la forma de los
21 datos varía según la arquitectura que se utilice.

22
23 Para este caso se dispone del set de imágenes almacenados en un directorio y un archivo
24 generado a partir de la extracción correspondiente de la plataforma de Twitter, en donde se almacena
25 el nombre de la imagen y el texto del tweet asociado. A continuación, se detalla el procesamiento de
26 dicha información para realizar la clasificación y así encontrar la forma más óptima para tratar este
27 problema planteado.

28 2 Obtención de Datos

29 Para obtener los datos se emplea la librería de python llamada *Tweepy* la cual emplea el API de
30 Twitter para leer los datos de la plataforma, en este caso se analiza la cuenta de @Arachno_Cosas,
31 https://twitter.com/Arachno_Cosas. Se lee las respuestas con mención que realiza el autor
32 de la cuenta que contienen una imagen, por lo que se extrae el texto de la respuesta brindada y la
33 imagen de la interacción.



Figure 1: Ejemplo de texto a extraer de la cuenta Twitter

34 Una vez capturada la información con ayuda de la *Tweepy* se almacenan los datos en un archivo csv
 35 el cual contiene dos columnas, una con el texto del autor de la página y la otra con el nombre de la
 36 imagen correspondiente.

37 3 Pre-procesamiento de Datos

38 Para este caso del texto de los tweets de las personas que interactúan con la cuenta, se clasifican
 39 cuales corresponden a arañas con importancia médica, se observa que a lo largo de estos datos se
 40 emplea un hashtag característico para cada tipo, por lo que #IM corresponde a las arañas que poseen
 41 importancia médica mientras los textos que poseen el hashtag #NIM corresponden a las arañas que
 42 no poseen importancia médica.

0	¡Hola, \n@MaXx_imiliano\n! Gracias por compart...	1.jfif
1	¡Hola, \n@catalina_parr\n! Gracias por tu cons...	2.jfif
2	¡Hola, \n@maxxeff4\n! Gracias por compartir. P...	3.jfif
3	¡Hola, \n@JavierG650ER\n! Gracias por tu consu...	4.jfif
4	¡Hola, \n@headcrusher666\n! Gracias por tu con...	5.jfif

Figure 2: Dataset antes de procesamiento

0	0	1.jfif
1	0	2.jfif
2	0	3.jfif
3	0	4.jfif
4	0	5.jfif

Figure 3: Dataset después de procesamiento

43 Para las imágenes se define una función para normalizarlas y utilizar tamaños pre-establecidos para
 44 su operación, en este caso se transforma a dimensiones de 224×224 píxeles. Después de ello se
 45 dividen los datos en entrenamiento y validación, y por último se emplea la función de Keras llamada
 46 *ImageDataGenerator* para generar más imágenes con los siguientes parámetros:

- 47 • `rotation_range = 30`: Rotar aleatoriamente las imágenes en el rango (grados, 0 a 180)
- 48 • `zoom_range = 0.2`: Ampliar aleatoriamente la imagen
- 49 • `width_shift_range=0.1`: Desplazar aleatoriamente las imágenes horizontalmente (fracción
 50 del ancho total)
- 51 • `height_shift_range=0.1`: Desplazar aleatoriamente las imágenes verticalmente (fracción de
 52 la altura total)
- 53 • `horizontal_flip = True`: Voltar aleatoriamente las imágenes

54 Dichos parámetros aumentan la cantidad de imágenes a utilizar en este caso, ayuda a identificar y
 55 entrenar de mejor manera al modelo, ya que las imágenes que se comparten son tomadas por los
 56 usuarios, haciendo que estas puedan ser transformadas en los distintos escenarios que se pueden
 57 llegar a presentar.

58 4 Metodología y experimentación

59 Para esta investigación se utiliza una red neuronal compuesta por seis capas intermedias, las cuales
 60 se van alternando entre convolución 2D y max pooling 2D, utilizando como función de activación
 61 ReLU y reduciendo el número de unidades a utilizar iniciando en 64 unidades hasta reducirse a 16.
 62 Luego de ello se agrega una capa de dropout con el 45% de los datos para después emplear la capa
 63 de Flatten y por ultimo una capa densa de 128 unidades con activación ReLU para finalizar con una
 64 función sigmooidal.

Model: "sequential_1"		
Layer (type)	Output Shape	Param #
=====		
conv2d_3 (Conv2D)	(None, 222, 222, 64)	1792
max_pooling2d_3 (MaxPooling2D)	(None, 111, 111, 64)	0
conv2d_4 (Conv2D)	(None, 109, 109, 32)	18464
max_pooling2d_4 (MaxPooling2D)	(None, 54, 54, 32)	0
conv2d_5 (Conv2D)	(None, 52, 52, 16)	4624
max_pooling2d_5 (MaxPooling2D)	(None, 26, 26, 16)	0
dropout_1 (Dropout)	(None, 26, 26, 16)	0
flatten_1 (Flatten)	(None, 10816)	0
dense_2 (Dense)	(None, 128)	1384576
dense_3 (Dense)	(None, 2)	258
=====		
Total params: 1,409,714		
Trainable params: 1,409,714		
Non-trainable params: 0		

Figure 4: Modelo utilizado para experimentación

65 4.1 Experimento 1

66 Para el experimento 1 se utiliza un optimizador ADAM con un learning rate de 0.01 y se emplea
67 una función de pérdida binary_crossentropy. Por lo que se obtienen que el modelo converge en la
68 iteración 2, con un accuracy de 0.9 para el conjunto de datos de validación.

```
4/4 [=====] - 12s 3s/step - loss: 4.0485 - accuracy: 0.4583 - val_loss: 0.6722 - val_accuracy: 0.9000
Epoch 2/10
4/4 [=====] - 10s 2s/step - loss: 0.6198 - accuracy: 0.7833 - val_loss: 0.6317 - val_accuracy: 0.9000
Epoch 3/10
4/4 [=====] - 10s 2s/step - loss: 0.6177 - accuracy: 0.7833 - val_loss: 0.6353 - val_accuracy: 0.9000
Epoch 4/10
4/4 [=====] - 10s 2s/step - loss: 0.6566 - accuracy: 0.7833 - val_loss: 0.6038 - val_accuracy: 0.9000
```

Figure 5: Resultados entrenamiento Experimento 1

69 4.2 Experimento 2

70 Para el experimento 2 se utiliza un optimizador ADAM con los parámetros por defecto de Keras
71 y se emplea una función de pérdida binary_crossentropy. Se comprueba que se obtiene el mismo
72 resultado del Experimento 1. Con un accuracy de 0.9 para el conjunto de datos de validación.

```
Epoch 1/10
5/5 [=====] - 10s 2s/step - loss: 0.5255 - accuracy: 0.7833 - val_loss: 0.3904 - val_accuracy: 0.9000
Epoch 2/10
5/5 [=====] - 10s 2s/step - loss: 0.5243 - accuracy: 0.7833 - val_loss: 0.3884 - val_accuracy: 0.9000
Epoch 3/10
5/5 [=====] - 10s 2s/step - loss: 0.5239 - accuracy: 0.7833 - val_loss: 0.3867 - val_accuracy: 0.9000
```

Figure 6: Resultados entrenamiento Experimento 2

73 5 Discusión de resultados

74 Se obtiene que los dos experimentos realizados con el modelo planteado obtienen los mismos
75 resultados de accuracy de 0.9. Por lo que el modelo planteado cumple con identificar los valores de
76 Importancia Médica para las imágenes seleccionadas como validación.

77

78 Se comprueba que al utilizar la función de *ImageDataGenerator* de Keras se pueden llegar
79 a generar más variaciones de las imágenes lo cual ayuda al modelo a tener más datos para entrenar, lo
80 cual hace más robusto su entrenamiento y proporciona mejores resultados.

81

82 Los resultados demuestran que se puede entrenar un modelo empleando CNN para clasificar arañas y
83 predecir si estas son de importancia médica o no.

84 6 Trabajo futuro

85 Se plantea generar un bot el cual analice la cuenta de Twitter @Arachno_cosas y genere la clasificación
86 de las arañas que se comparten en paralelo con las respuestas del autor, y comparar los resultados
87 obtenidos para incrementar la precisión con la que el modelo predice los resultados.

88 Referencias

89 [1] Visualizing CNN filters with keras. (2016, March 23). Jacob Gildenblat. [https://jacobgil.github.io/](https://jacobgil.github.io/deeplearning/filter-visualizations)
90 [deeplearning/filter-visualizations](https://jacobgil.github.io/deeplearning/filter-visualizations)

91 [2] Gautam, T. (2021, July 30). Image Classification in Python with Keras | Image
92 Classification. Analytics Vidhya. [https://www.analyticsvidhya.com/blog/2020/10/](https://www.analyticsvidhya.com/blog/2020/10/create-image-classification-model-python-keras/)
93 [create-image-classification-model-python-keras/](https://www.analyticsvidhya.com/blog/2020/10/create-image-classification-model-python-keras/)

94 [3]