Is this Loss?

Sistema de reconocimiento de objetos en imágenes

Hernández Chiapa David Felipe
López García Gilberto Isaac

Facultad de Ciencias
Universidad Nacional Autónoma de México

13 de abril de 2018

1. Introducción

2. Descripción del problema

3. Redes Neuronales Convolucionales

3.1. Capa de convolución

3.2. Pooling

3.3. Perceptrón Multicapa

3.4. Espacio de Hipótesis

3.5. Ventajas y desventajas

4. Propuesta e implementación

4.1. Datos

El conjunto de datos consiste de una colección de 1735 imágenes (1304 para entrenamiento y 431 para verificación), divididos en dos categorías, `loss_edits` y `not_loss`, donde la primera son edits de Loss, y la segunda memes variados.

Los memes variados se obtuvieron de la colección personal de memes de los desarrolladores, obtenidas con el paso del tiempo ya sea descargándolas de distintos sitios de internet como redes sociales, chats de WhatsApp, Messenger, etc., mientras que los edits de Loss se descargaron principalmente de la galería de KnowYourMeme.com[1] y Google Images (que son miniaturas de distintos sitios, como el subreddit r/lossedits[2] y KnowYourMeme.com por lo que puede haber imágenes repetidas en el dataset).

[1] `http://knowyourmeme.com/memes/loss/photos`
[2] `http://reddit.com/r/lossedits`

4.1.1. Preprocesamiento

El primer paso fue cambiar la resolución de las imágenes para tener un dataset más uniforme (aunque se reescalarán posteriormente de nuevo pues la red neuronal necesita entradas de tamaño fijo). Dado que las miniaturas de edits tienen baja resolución (entre 200 y 300 pixeles de ancho/alto), los memes variados fueron reescalados para tener una resolución similar (250 pixeles de ancho), después se dividió el conjunto completo de imágenes en conjuntos de entrenamiento y verificación de manera aleatoria.

Algunos edits de Loss se basan en el uso de colores, pero en general es la orientación de los polígonos lo que importa como se ve en la Figura 1. Para tratar estos casos se construyen modelos que trabajan con imágenes a color y modelos que trabajan con imágenes en escala de grises.

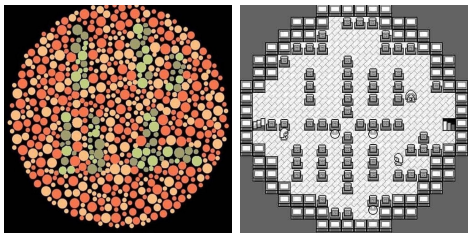

Figura 1: Edit basado en color (izq.) y en orientación (der.)

Procesar las imágenes para obtener las aristas se consideró una propuesta poco viable pues se tiene mucho ruido o se pierde información. Usando el modelo Canny de OpenCV para obtener las aristas presentes en una imagen nos encontramos con mucho ruido, por ejemplo, si un edit se contruyó con fotografías con muchos detalles en ellas y objetos innecesarios no fueron desenfocados, las aristas de dichos objetos se extraen en conjunto con los polígonos que nos interesan. Si modificamos los umbrales inferior y superior del modelo para filtrar aristas podemos eliminar aristas de los objetos que nos interesan realmente. Estos problemas se pueden apreciar en la Figura 2

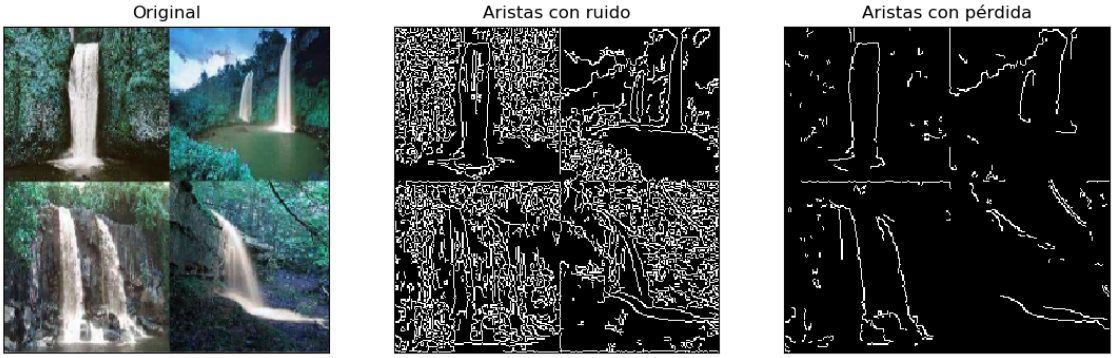

Figura 2: Detección de aristas, algoritmo de Canny.

Además, los umbrales inferior y superior pueden no funcionar en distintas imágenes, entregrando mucho ruido o pérdida de información. La baja resolución de las imágenes también dificulta este proceso.

4.2. Implementación

La implementación se realizó en el lenguaje de programación Python[3]. Para la construcción de la red neuronal convolucional nos apoyamos de los paquetes TensorFlow[4] y Keras[5].

5. Resultados

5.1. Pruebas preliminares

5.2. Primer intento de clasificación

5.3. Segundo intento de clasificación

5.4. Observaciones

6. Conclusiones

Referencias

[1] Haykin, S. S. (2011). *Neural networks and learning machines*. New Dehli: PHI Learning.

[2] Wu, J. (2018). *Convolutional neural networks*. National Key Lab for Novel Software Technology. Nanjing University, China. Obtenido de `http://cs.nju.edu.cn/wujx/teaching/15_CNN.pdf`.

[3] Scherer, D., Müller, A., & Behnke, S. (2010). *Evaluation of Pooling Operations in Convolutional Architectures for Object Recognition*. Artificial Neural Networks-ICANN 2010 Lecture Notes in Computer Science, 92-101. DOI:10.1007/978-3-642-15825-4_10.

[4] Ducji, J., Hazan, E., & Singer, Y. (2011). *Adaptive Subgradient Methods for Online Learning and Stochastic Optimization*. Journal of Machine Learning Research 12, 2121-2159.

[5] Kingma, D. P. & Lei Ba, J. (2014). *Adam: A Method for Stochastic Optimization*. arXiv:1412.6980. Obtenido de `http://arxiv.org/pdf/1412.6980.pdf`.

[3] Versión 3.6.4, `http://www.python.org/downloads/release/python-364/`

[4] `http://www.tensorflow.org/`

[5] `http://keras.io`