Clasificador de Dibujos

Monreal Gamboa Francisco Manuel
Ramírez García Diana Isabel

12 de junio de 2020

1 Introducción

Existen una gran cantidad de idiomas en el mundo por el cual nos comunicamos a diario, sin embargo estos tienen sus limitaciones. Entonces cuando necesitamos comunicarnos y el idioma se transforma en muros, incluso en un mismo idioma los nombres pueden variar y generar la misma confusión, para eso necesitamos buscar otras opciones tal como las señas y los dibujos. Apesar de que muchos no son los grandes dibujantes logran de manera exitosa expresar su mensaje. ¿Esto a que se debe?
Si lo analizamos vemos que al querer representar algún objeto lo hacemos de la manera más general y ponemos únicamente los detalles más escenciales para dar a entender la idea. Debido a que existen más características que pueden ser distintas para cada uno como puede ser el color, sin embargo todos coincidimos en la idea general, por ello somos capaces de entendernos por medio de dibujos.

2 Objetivo

El objetivo que buscamos es generar una red neuronal convolusional tal que tenga la capacidad de identificar distintos objetos por medio de dibujos escenciales o en otras palabras garabatos.

3 Métodos

En primer lugar obtuvimos la base de datos del sitio Quick, Draw! The Data, del cual decidimos tomarnos 120 dibujos diferentes y descargar la base de datos de cada uno, en formato numpy_bitmap. Para el manejo de los datos y posteriormente para la creación de la red utilizamos tensorflow junto con Keras. Posteriormente cargamos los datos, creamos un arreglo con los datos de las imágenes y otro con con una etiqueta que nos dice que es esa imagen. Decidimos tomar 4000 imágenes de cada categoría para repartir de una manera uniforme los datos y no hubiera mas imágenes de una categoría que de otra, finalmente creamos un conjunto aleatorio de estos tal que estamos listos para

dividirlos en datos de entrenamiento y de prueba, decidimos tomar un 20% de los datos para el conjunto de prueba. Sin embargo estos no se encuentran listos para su uso, es necesario realizar un procesamiento en el cual normalizamos las imágenes y convertimos los vectores de datos en matrices.

El siguiente paso es la creación de nuestro modelo. Como se mencionó en el objetivo, deseamos una red neuronal convolusional debido a que nuestra base de datos está conformada por dibujos y sabemos que este tipo de red va tomando piezas de información para posteriormente combinarlas y poder hallar coincidencias con los patrones obtenidos. A continuación vamos a profundizar en ello.
Un red convolusional esta conformada por 2 partes: las capas ocultas(capas de convolución y pooling) y la clasificación en donde tenemos a todas las neuronas conectadas como una red neuronal tradicional. En la arquitectura propuesta tenemos nuestras capas de convolución en donde cada capa aumenta el número de kernel convolusionales que utiliza debido a que en cada capa intentamos obtener más detalles de la imagen y generar algo más complejo. Además se utiliza la función de activación ReLu ya que su comportamiento nos sirve para que valores negativos no pasen a las siguientes capas, ya que estos no son de utilidad en cuanto a imágenes. Posteriormente utilizamos capas de pooling para reducir el número de parámetros en la red, de tal manera que se mejora el entrenamiento y vamos evitando un sobreentrenamiento. Además en la red se utiliza max pooling porque mantiene la información más significativa mientras que minimiza el mapa de caraterísticas que generan las capas convolusionales.
Ahora para la parte de clasificación es necesario es uso de una capa Flatten ya que necesitamos nuestros datos en una dimensión para poderlos procesar. Después utilizamos capas densas totalmente conectadas, en las cuales utilizamos la función de relu y softmax para que nos regrese un vector con todas las probabilidades de cada una de las 120 categorías.

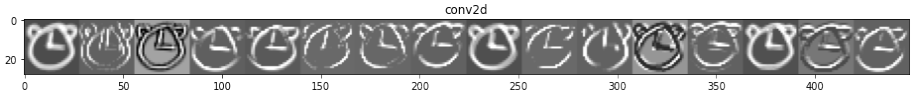
Posteriormente realizamos el entrenamiento de la red con el conjunto previamente dividido, en el cual asignamos 7 épocas en donde la certeza de la red fue mejorando. Seguidamente procedimos a la fase de prueba en la cual obtuvimos una certeza del 90%. Realizamos una prueba en donde elegimos una imagen aleatoria del conjunto de prueba en donde la pudimos visualizar y ver las posibles opciones de reconocimiento que arroja la red.
Sin embargo al momento de probar nuestra red con fotos tomadas de nuestro celular, descubrimos que el rendimiento era muy pobre. Probablemente tuviera que ver la luz o el simple hecho de que todas las imágenes pasadas son dibujos digitales.
Debido a este problema, decidimos hacer nuestros propios dibujos digitales, recordamos un software de desarrollo visual que utilizamos en semestres pasados llamado Processing, que perminte crear un canvas en donde se puede dibujar de una manera fácil y rápida, usamos la versión de javascript llamada p5.js.

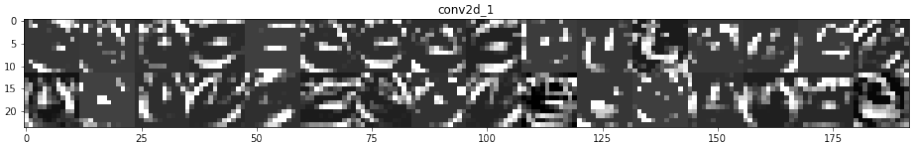
Finalmente utilizamos tensorflow js para almacenar el modelo creado con las extensiones necesarias que nos permitiera su uso en la interfaz gráfica creada por medio de javascript en donde el usuario puede dibujar alguno de los objetos

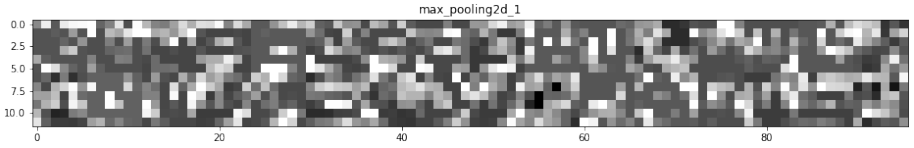
con los que se entrenó la red. Esta nos dirá lo que cree que es con un porcentaje de certeza.

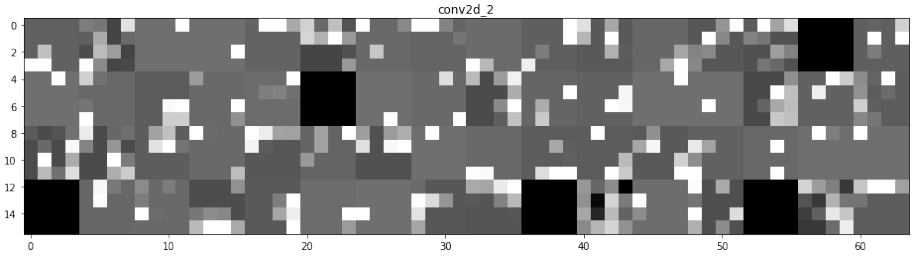
4 Resultados

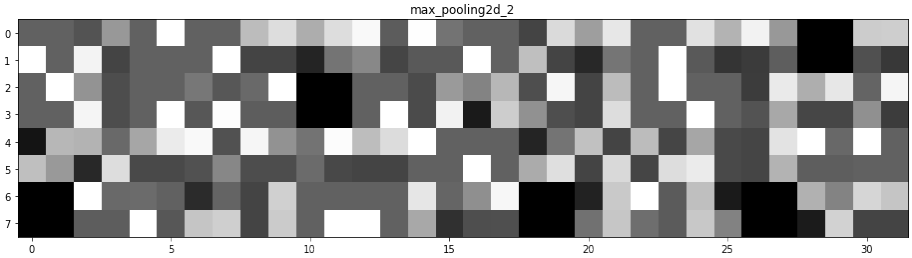
Al momento de realizar pruebas quisimos ver como se iba desarrollando las capas de nuestra red. Podemos ver el comportamiento de las capas convolusionales, como estas van particionando los datos, mientras que las capas de max pooling reducen los parámetros.



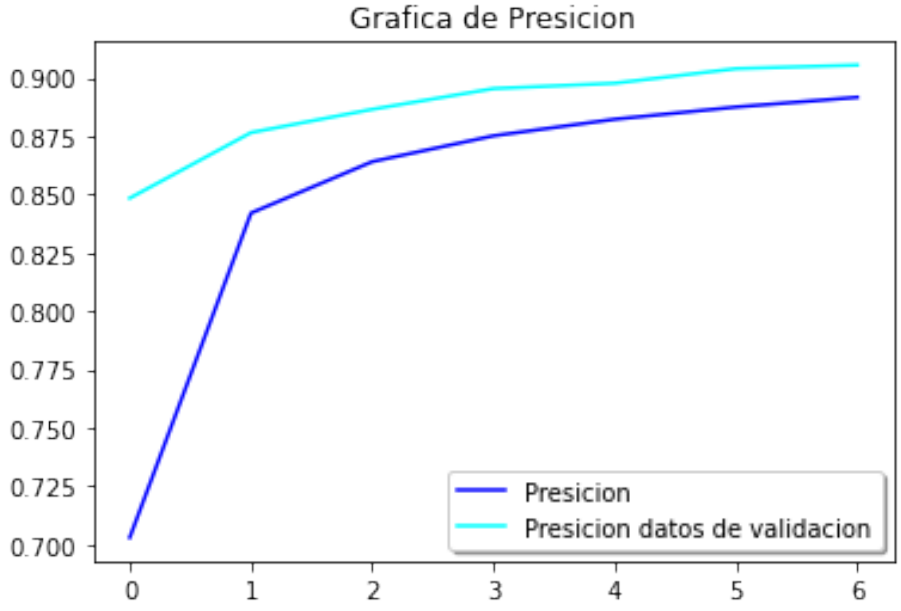


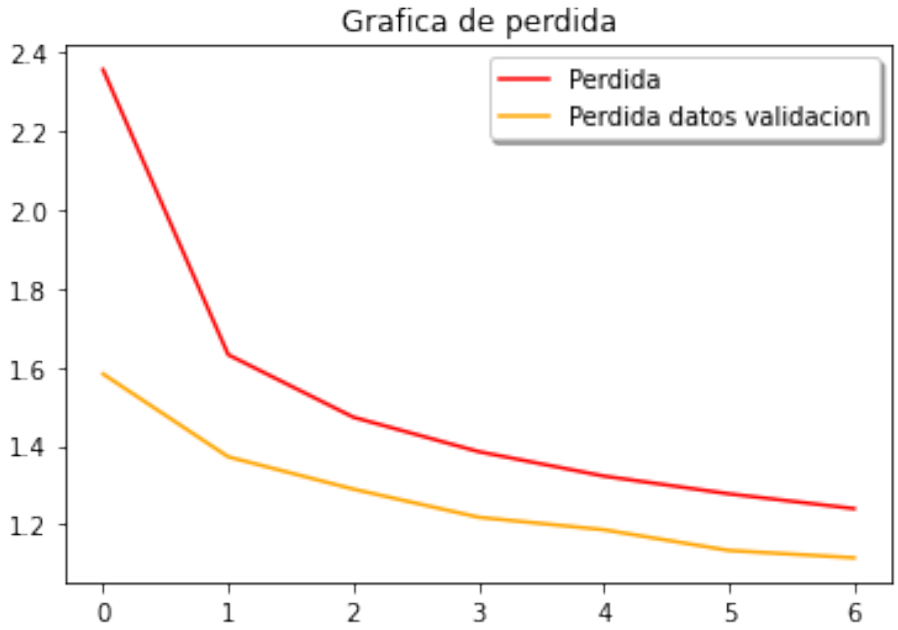






Cuando terminamos la red, la siguiente pregunta que nos hicimos fue analizar su comportamiento, es decir su precisión y su pérdida.





Una vez terminado el modelo realizamos diferentes pruebas en donde notamos que al utilizar fotografías de dibujos no obteníamos los resultados deseados, por ello probamos con processing ,como herramienta, para generar imágenes digitales en donde la red tuvo mucho mejor desempeño.

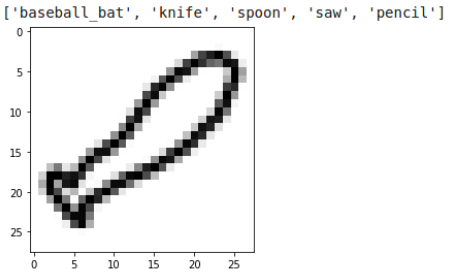


Figure 1: Imagen digital

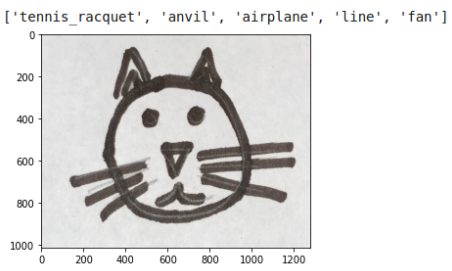
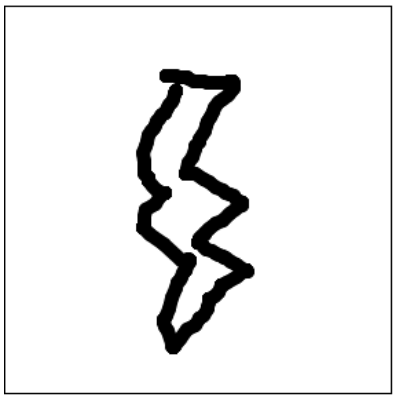


Figure 2: Fotografía

Por último realizamos una interfaz gráfica sencilla para probar con nuestros propios dibujos y comprobar su precisión, en el cual podemos ver que se tiene un buen porcentaje de predicción.

Clasificador de 100 imagenes

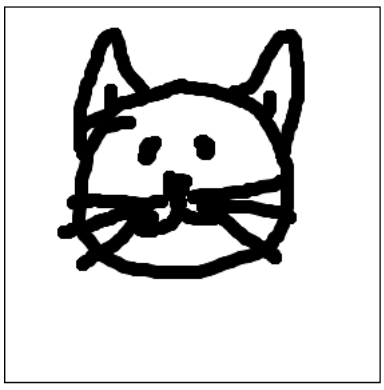
Borrar



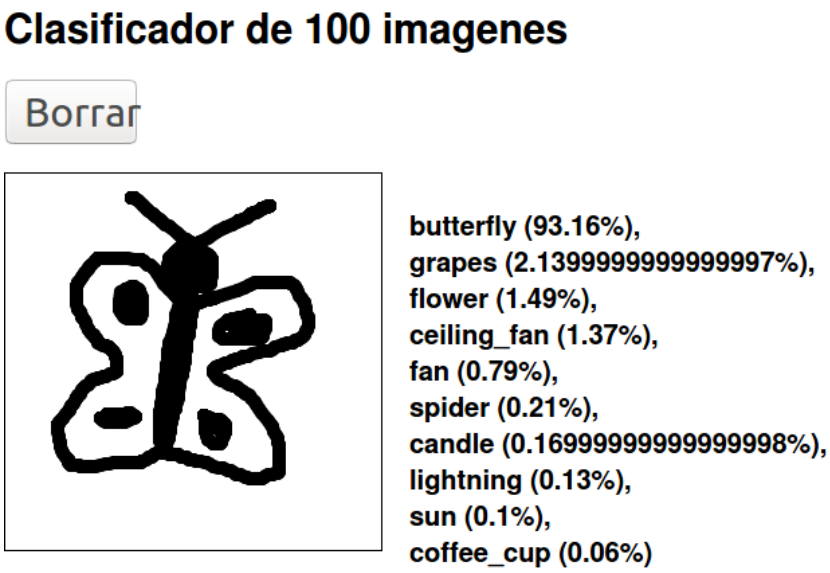
lightning (98.44000000000001%),
syringe (0.61%),
grapes (0.31%),
cloud (0.26%),
sun (0.04%),
screwdriver (0.03%),
tree (0.03%),
microphone (0.02%),
moustache (0.02%),
moon (0.02%)

Clasificador de 100 imagenes

Borrar



cat (95.46%),
lightning (1.1900000000000002%),
cloud (0.69%),
face (0.5%),
flower (0.32%),
cookie (0.25%),
diving_board (0.16%),
anvil (0.15%),
coffee_cup (0.13999999999999999%),
grapes (0.11%)



5 Conclusiones

Podemos concluir que fuimos capaces de construir una red neuronal convolusional tal que fue capaz de reconocer con un gran nivel de certeza ciertas imágenes, sin embargo en algunas otras se confundía fácilmente, ya que una gran variedad de imágenes pueden ser muy similares a pesar de ser distintos objetos o incluso las distintas interpretaciones de un mismo objeto, que al ojo humano no es mucha diferencia porque podemos reconocerlo pero para una computadora sigue siendo de gran dificultad. Además de lo anterior, al tomar una base de datos open-source no tenemos asegurada su limpieza, ya que en algunos casos nos encontramos cosas totalmente distintas a lo esperado como alguien que escribió "I dont know" en lugar de dibujar un pájaro. Entonces para mejorar el desempeño de la red es necesario ser más meticulosos con nuestros datos posiblemente pueden existir cambios dentro de la arquitectura propuesta que nos permitirían una mejora.

6 Bibliografía

- https://relopezbriega.github.io/blog/2016/08/02/redes-neuronales-convolucionales-con-tensorflow/
- https://www.freecodecamp.org/news/an-intuitive-guide-to-convolutional-neural-networks-260c2de0a050/
- https://www.tensorflow.org/

- https://sitiobigdata.com/2019/06/22/relu-funciones-activacion/
- https://sempwn.github.io/blog/2017/04/06/conv_net_intro
- https://stackoverflow.com/questions/48243360/how-to-determine-the-filter-parameter-in-the-keras-conv2d-function