APLICACIÓN DE REDES NEURONALES CONVOLUCIONALES EN EL ANÁLISIS DE ESTRUCTURAS

Saúl Jesús Suaste Martínez
sumsaul94@hotmail.com

Eric Manuel Rosales Peña Alfaro
emrosales@ipn.mx

SEPI UPIICSA

Boletín No. 81
1o. de noviembre de 2020

Resumen

En este trabajo se presenta un modelo para realizar un clasificador de imágenes mediante la aplicación de Redes Neuronales Convolucionales, mediante Teachable Machine. El cual clasificara por medio de la carga de imágenes si una estructura está en condiciones normales, tiene grietas u hoyos. La investigación presenta una introducción en la cual se describe el funcionamiento de las Redes Neuronales Convolucionales, además se describe el proceso de entrenamiento del modelo, el cual inicia con la carga de las imágenes para el entrenamiento y para realizar las pruebas. Posteriormente se describen los parámetros que ayudan al entrenamiento de la red para así lograr una precisión alta a la hora de hacer una clasificación. Una vez entrenado el modelo se realizan las pruebas con imágenes que no fueron utilizadas para entrenar el modelo, obteniendo una precisión del 100 % para cada clase creada. Por último, se prueba el modelo entrenado en una computadora y ya no en la web de Teachable Machine, y como resultado se obtiene la misma precisión a la hora de hacer la clasificación.

Introducción

La inteligencia artificial se ha desarrollado enormemente y aunque las técnicas que utiliza no son nuevas, en esta época se puede llevar a cabo con más facilidad. Esto gracias al desarrollo de los equipos de cómputo en especial a las GPUs (Unidad de Procesamiento Gráfico), ya que su procesamiento en paralelo hace que el trabajo sea mucho más rápido. (Rull, 2016) (Rodríguez-Sahagún Alesanco, 2018).

El aprendizaje profundo es una sub rama del Aprendizaje Automático, la cual usa redes neuronales para mejorar tareas automáticas como lo son: Reconocimiento de voz e imágenes, Visión por computadora, etc. (Spencer, Hoskere, & Narazaki, 2019).

Redes Neuronales Convolucionales

Las Redes Neuronales Convolucionales se diferencia de otras debido a que una capa de neuronas no necesariamente recibe conexiones a la entrada de todas las neuronas de la capa anterior, solo recibe algunas de ellas. (Rull, 2016).

Las Redes Neuronales Convolucionales o por sus siglas en ingles CNN, convolucionan las características de los datos de entrada y emplean capas 2D para hacerlo, por lo tanto, este tipo de Redes Neuronales son excelentes para procesar datos 2D como lo son las imágenes. (Rull, 2016) (Vizcaya Cárdenas, 2018) (Shafkat, 2018).

El aprendizaje profundo (Deep Learning) basado en redes neuronales convolucionales es un algoritmo de clasificación, los algoritmos de aprendizaje profundo solamente necesitan imágenes de entrada para clasificar, mientras que otros algoritmos necesitan que se describa la información de entrada al inicio. Los algoritmos infieren las características más relevantes y así pueden implementarlos como filtros de textura, color o forma los cuales son los más utilizados. Gracias a lo mencionado anteriormente, estos algoritmos de aprendizaje profundo se utilizan con antelación para reconocimiento de imágenes. (Arista Jalife, Calderón Auza, Fierro Radilla, & Nakano, 2017).

Esta práctica consiste en realizar operaciones convolucionales a una imagen y así obtener las características más relevantes y destacando las más preponderantes. Las neuronas de la red convolucional tienen una estructura bidimensional a manera de matriz, por lo tanto, la imagen de entrada se considera como una retícula de neuronas, en la cual, cada neurona se relaciona a un pixel de la imagen, además cada pixel contiene información de sí mismo y de los pixeles aledaños. (Arista Jalife, Calderón Auza, Fierro Radilla, & Nakano, 2017).

Las neuronas de capas posteriores tienen visión de cierto rango de la capa anterior, a esto se llama campo local receptivo, y estas neuronas son la entrada de la neurona oculta, y si se brinda una activación adecuada a las neuronas contribuirán a la entrada de la capa posterior, así como se observa en la figura 1. (Rull, 2016) (Arista Jalife, Calderón Auza, Fierro Radilla, & Nakano, 2017).

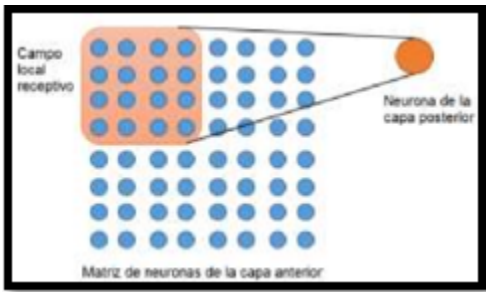

Figura 1. Campo local receptivo.

Estos también pueden superponerse. La cantidad de pixeles que se visualizan en el campo local receptivo se le conoce como paso o stride, y el objetivo de las neuronas de la capa posterior es detectar características en la imagen. (Arista Jalife, Calderón Auza, Fierro Radilla, & Nakano, 2017) (Shafkat, 2018)

Las CNN se componen de varias capas para realizar el proceso, las cuales son: Capa convolucional, Pooling, Capa completamente conectada y capa de clasificación. Las cuales se describen a continuación:

- Capa convolucional. En esta capa se le aplica a la imagen de entrada una operación convolucional con algún filtro determinado. Estas son ventanas deslizantes que recorren la imagen aplicando la operación. Su operación está definida por:

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(m,n)K(i-m, j-n) \quad \dots(1)$$

Donde:

I = La imagen.

K = Filtro o kernel de tamaño mxn que se le aplicara. Por lo regular m = n. (i,j) = Es la posición de los pixeles en la imagen. (Rull, 2016) (Vizcaya Cárdenas, 2018) (Arriola Oregui, 2018) (Rodríguez-Sahagún Alesanco, 2018).

El kernel se puede representar como en la figura 2, donde se tiene la imagen (en este caso binaria), el kernel 3x3 y el resultado. El salto que realiza el kernel es de un pixel. (Arriola Oregui, 2018) (Rodríguez-Sahagún Alesanco, 2018).

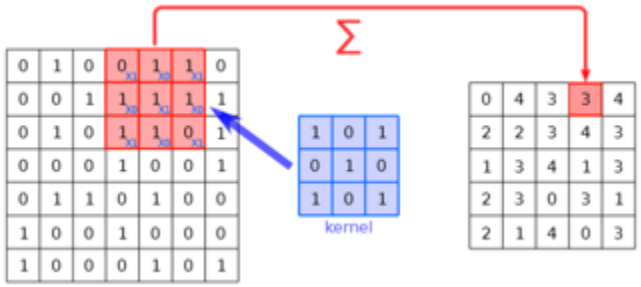

Figura 2. Convolución a una imagen binaria.

Si la entrada es una imagen a color, la profundidad de los filtros seria de 3, debido a los tres colores RGB. Estos filtros se aplican desde la esquina superior izquierda hasta la inferior derecha. A la par de esto se realizan las operaciones convolucionales, que son operaciones de producto punto y sumas, se realizan entre los pixeles de la imagen y los pesos de los filtros. El resultado de esto da un volumen de datos más pequeño, representado por (pxqxd) dónde: p y q es el nuevo tamaño y d es la cantidad de filtros usados. El tamaño cada vez se va reduciendo más y más. (Rull, 2016) (Vizcaya Cárdenas, 2018) (Shafkat, 2018).

El objetivo de este procedimiento es obtener mapas de activación o características, los cuales se activan al pasar los filtros por las regiones de interés de una imagen, las cuales tiene las características que busca cada capa, así como: bordes, líneas, contornos, sombras, etc. tal como lo muestra la figura 3. (Vizcaya Cárdenas, 2018) (Shafkat, 2018) (Contreras Zaragoza, 2018).

$$K = \begin{matrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{matrix}$$

Figura 3. Imagen original. Imagen con filtro

El volumen de los datos que se tengan a la salida dependerá directamente de la cantidad de filtros que se usan como entrada, el paso del kernel, el relleno de ceros y el campo visual del filtro. El rellenar de ceros como bordes a la imagen es para que se ajuste el tamaño de la imagen al filtro que hará el barrido. (Vizcaya Cárdenas, 2018).

- **Pooling.** Esta función reemplaza la salida (es una "rebanada" de la capa anterior, la cual es la convolución) por un valor que resume de alguna manera los valores tomados. Max pooling (figura 4) es una función común la cual asigna en la matriz de salida el valor máximo de la región dada. Pero así como Max pooling existen también para calcular la media o el promedio. (Rull, 2016) (Arista Jalife, Calderón Auza, Fierro Radilla, & Nakano, 2017) (Arriola Oregui, 2018) (Shafkat, 2018) (Contreras Zaragoza, 2018).

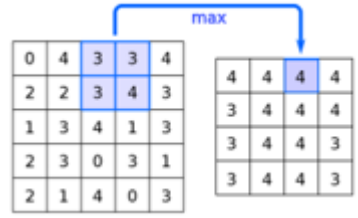

Figura 4. Max pooling con kernel de 2x2.

- **Capa completamente conectada.** Después de las capas anteriores, se van a añadir una o dos capas completamente conectadas (Fully Connected). Aquí se realiza el producto lineal entre un filtro y la salida anterior, la cual debería ser una capa max pooling. Se hace una suma ponderada de sus parámetros por el valor de entrada, con esto el que tenga mayor puntuación es el que tenga una mayor probabilidad, con lo cual se está haciendo una clasificación indicando la probabilidad de cada clase. (Rull, 2016) (Vizcaya Cárdenas, 2018) (Contreras Zaragoza, 2018).
- **Capa de clasificación.** Al final de toda la red es necesario poner una capa de clasificación, con esto se genera un vector que será el resultado donde cada componente aporta información respecto de la probabilidad de una imagen de entrada a pertenecer a una cierta clase. (Rull, 2016).

Finalizando todas las capas, se tiene una red entrenada y completa (figura 5), con los pesos ya calculados, y tomando una gran cantidad de datos a la entrada con una clasificación en específico, la red será capaz de clasificar un dato que se introduzca a la entrada. (Rull, 2016).

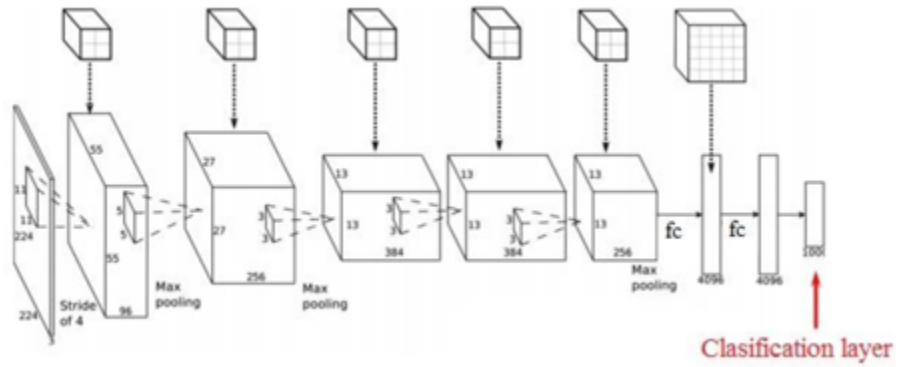

Figura 5. Red Neuronal Convolucional completa.

Desarrollo del modelo

El entrenar un modelo como este requiere mucha capacidad computacional debido al número de capas y grandes volúmenes de datos, al igual que el número de iteraciones del algoritmo (SAS Institute,

2019). Por lo tanto, existen servicios en la nube que realizan este tipo de procesamientos por medio de Plataform as a Services (PaaS), algunos ejemplos de estos son: Haven OnDemand de Hewlett Packard, Watson de IBM, AWS de Amazon y Teachable Machine de Google. (Murch, 2018) (Vaggalis , 2017).

Teachable Machine es una herramienta que te permite entrenar una computadora para hacer reconocimiento y clasificación de imágenes. Se puede utilizar imágenes propias, archivos previamente descargados o ejemplos tomados completamente en vivo. Los modelos que genera son de Tensorflow.js la cual hace que funcione en donde sea que se ejecute JavaScript, lo cual lo hace perfecto para funcionar con muchas otras herramientas como P5.js, node.js y muchos más. (Google, 2019).

Se entrena el modelo que se utiliza en el Proyecto varias veces, hasta llegar a los resultados esperados. Se muestra el primer y último de estos entrenamientos para observar cómo va mejorando la precisión de los resultados. El primer entrenamiento se muestra como se hizo paso a paso y para el último entrenamiento los resultados solamente, este además es el que tiene la mejor precisión a la hora de clasificar.

Primer entrenamiento

Cargar Imágenes

Se tienen que cargar las imágenes o tomarlas directamente con la cámara web, al cargar las imágenes Teachable Machine les hace un pre procesamiento haciéndolas cuadradas. Las imágenes cargadas fueron descargadas previamente, con las cuales se crearon tres clases diferentes: Normal, Grietas y Hoyos (figura 6). (Google, 2019).

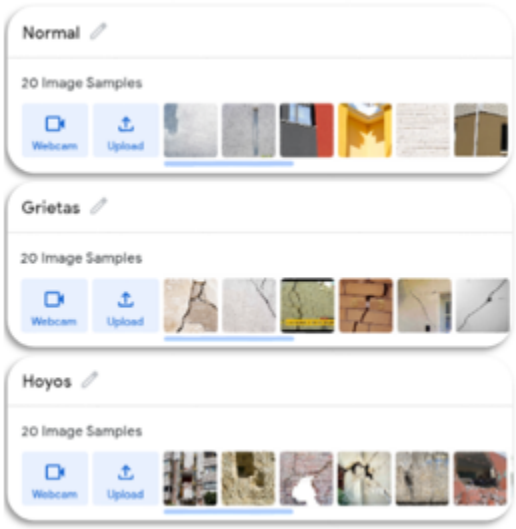

Figura 6. Clases creadas.

Entrenamiento

El segundo paso es entrenar el modelo, donde se tiene la opción de acceder a configuración avanzada y modificar parámetros como: epochs, batch size y learning rate. Donde:

- Épocas. Una época significa que todas las muestras de datos que se cargaron para el entrenamiento se han utilizado por lo menos una vez, esto quiere decir que las imágenes ya alimentaron al entrenamiento por lo menos una vez y lo harán las veces que se indique. Generalmente cuanto mayor sea el número de épocas mejor va a aprender el modelo a predecir los datos. Este se ajusta hasta que el modelo ya tenga buenos resultados a la hora de hacer las predicciones. (Google, 2019).

- Tamaño del lote. El batch size son el conjunto de muestras que se utilizan en una iteración. En este primer entrenamiento se utilizan 60 imágenes en total, por lo tanto 60/16 ≈4 lotes, una vez que los 4 lotes hayan alimentado al modelo se cumple una época. (Google, 2019).
- Tasa de aprendizaje. Es un parámetro que indica cuanto va a cambiar el modelo de acuerdo al error estimado cada vez que se actualicen los pesos del modelo. Un número muy pequeño puede hacer que el aprendizaje sea demasiado tardado y un número grande hará que aprenda demasiado rápido y eso no es óptimo y además hará que el proceso de entrenamiento sea inestable. (Brownlee, 2019).

Una vez que se le da un valor a estos parámetros, se procede a entrenar el modelo, el cual utiliza un 85 % de los ejemplos que le suministramos para el entrenamiento, esto para clasificar correctamente nuevos datos en las clases creadas. El otro 15 % se utilizan para las muestras de prueba, las cuales nunca son usadas para entrenar el modelo y una vez que el modelo fue entrenado estas se utilizan para verificar como se desempeña el mismo. (Google, 2019).

En el primer entrenamiento como parámetros se estableció lo siguiente:

- Imágenes por clase: 20
- Épocas: 20
- Tamaño lote: 16
- Tasa de aprendizaje: 0.0001

Y como resultado se obtuvo en la clase Normal un 0.67 de precisión, la clase Grietas con 0.67 de precisión, y la clase Hoyos con 1.00 de precisión. (Tabla 1). (Google, 2019).

Tabla. 1.
Precisión por clase, primer entrenamiento.

Clase	Precisión	Numero de ejemplos
Normal	0.67	3
Grietas	0.67	3
Hoyos	1.00	3

En la figura 7 se observa cómo va mejorando el modelo con forme van pasando las épocas, desde la época 5 el entrenamiento ya es bueno, pero las muestras de precisión no alcanzan un nivel óptimo o esperado ya que terminaron en un 0.80 aproximadamente.

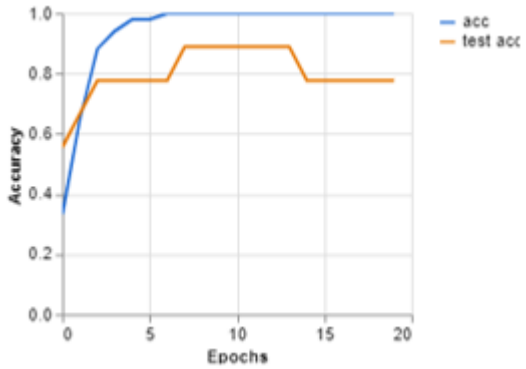

Figura 7. Precisión por época, primer entrenamiento.

En la siguiente grafica (figura 8) se puede observar la pérdida del modelo, donde la pérdida del modelo es casi 0.1 pero la pérdida de las muestras está cerca del 0.4. Por lo tanto el modelo no ha aprendido a predecir las clasificaciones correctas para el conjunto de muestra. Si las pérdidas del modelo se acercan a cero entonces el modelo ya predice de forma correcta, pero en este caso el modelo tiene perdidas al final de las 20 épocas, y la perdida de las muestras de prueba son aún más grandes. (Google, 2019)

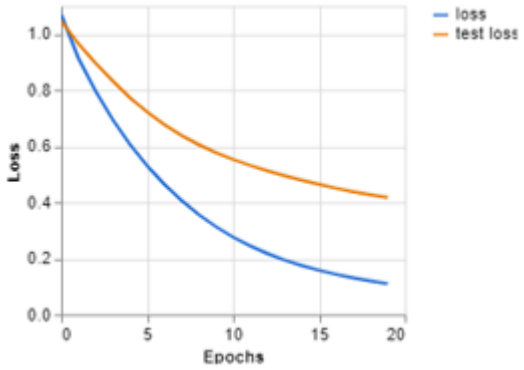

Figura 8. Perdida del modelo por época, primer entrenamiento.

Probando modelo

Ya que se entrenó el modelo y se conocen los resultados del entrenamiento, se sabe que al momento de hacer una predicción con una imagen que no se utilizó en el entrenamiento, el porcentaje de precisión del modelo no será muy alto y no sabrá a que clase pertenece con exactitud.

Se seleccionó una imagen de pared normal, una con una grieta y una con un hoyo. Como se mencionó anteriormente, estas imágenes no fueron parte del entrenamiento.

En la figura 9 se puede ver como el modelo no está seguro de a qué clase pertenece cada imagen que se ingresa como prueba, ya que para la clase normal da una precisión del 50 %, para la clase grietas una precisión del 35 % y para la clase hoyos de un 79 %.

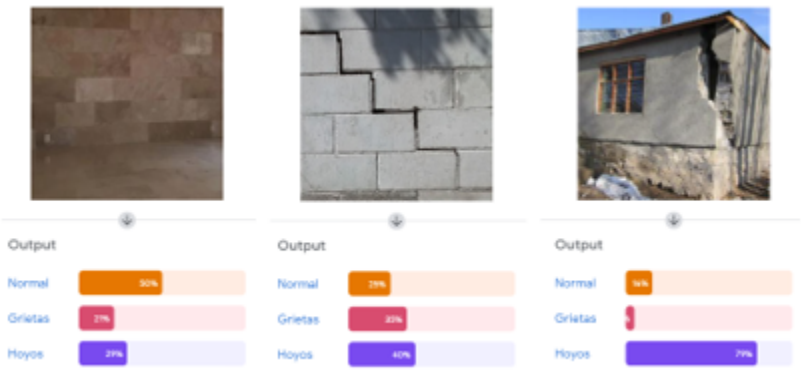

Figura 9. Prueba de clasificación con imagen de pared normal, grieta y hoyo, primer entrenamiento.

Último entrenamiento

En este entrenamiento como parámetros se estableció lo siguiente:

- Imágenes por clase: 50
- Épocas: 100
- Tamaño lote: 16
- Tasa de aprendizaje: 0.0001

Y como resultado se obtuvo que la precisión o exactitud por clase fue muy buena para las tres clases, en la clase Normal con 1.00 de precisión, la clase grietas con 1.00 de precisión, y la clase Hoyos con 1.00 de precisión (Tabla 2). (Google, 2019).

Tabla 2.
Precisión por clase, último entrenamiento.

Clase	Precisión	Numero de ejemplos
Normal	1.00	8
Grietas	1.00	8
Hoyos	1.00	8

En la figura 10 se observa cómo mejoro el modelo con un mayor número de imágenes y de épocas. Se mantiene en 1 prácticamente desde la época 7 para el modelo y las imágenes de muestra.

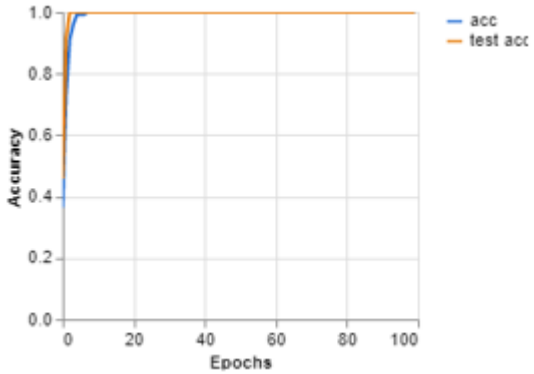

Figura 10. Precisión por época, último entrenamiento.

Para este entrenamiento, la pérdida que se obtuvo fue muy pequeña al terminar todas las épocas, y desde la época 80 prácticamente es cero la perdida para el modelo y las imágenes de muestra, como se puede observar en la figura 11.

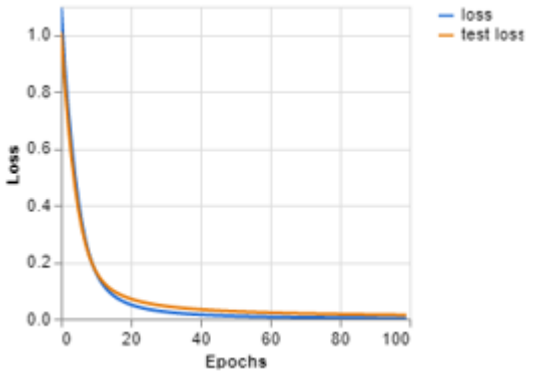

Figura 11. Perdida del modelo por época, último entrenamiento.

Para este entrenamiento del modelo los resultados fueron buenos, por lo tanto se espera que al hacer una predicción con una imagen nueva para cada clase, el resultado de la precisión sea alta. En la figura 12 se puede ver como el modelo arroja para cada resultado un valor de 100 % de precisión en cada clase.

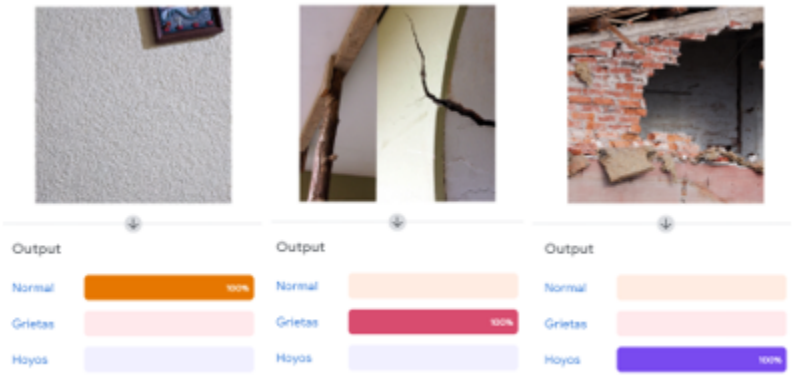

Figura 12. Prueba de clasificación con imagen de pared normal, grieta y hoyo, último entrenamiento.

Gracias a Teachable Machine se pudieron hacer las pruebas necesarias y tener un modelo bien entrenado con una precisión para clasificar del 100 %.

Exportar modelo

Como se mencionó anteriormente, Teachable Machine permite que puedas extraer el modelo entrenado, y gracias a esto es posible ejecutar un código que utilice el modelo entrenado, y así se puedan hacer predicciones desde cualquier equipo.

El programa primero procesa la imagen (la que estamos interesados a clasificar) y arroja una vista previa de cómo queda. Posteriormente en forma de vector arroja el resultado de la clasificación. De acuerdo a cuando se entrenó el modelo, la clasificación es la siguiente [0 = Normal, 1 = Grieta, 2 = Hoyo]. En la figura 13 se puede ver que el programa muestra la imagen procesada y el resultado de la clasificación en forma de vector [0.00000019, 0.9999925, 0.0000073], dando como resultado un 99.99 % de precisión que la imagen es de una grieta.

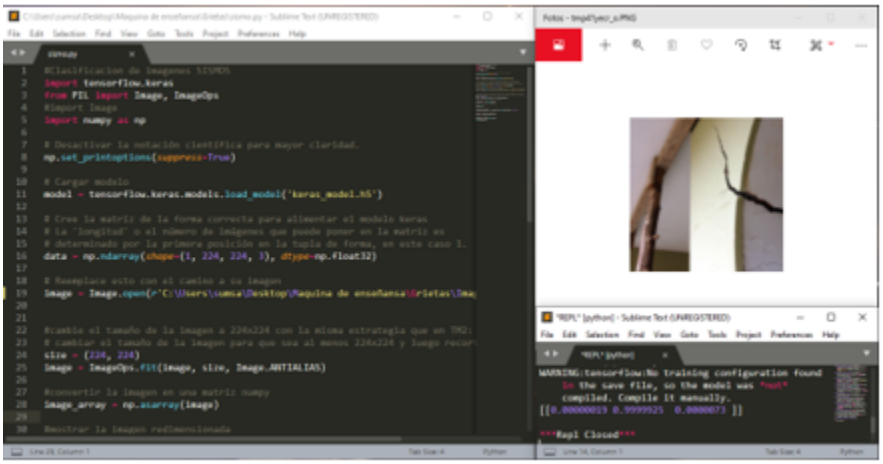

Figura 13. Resultado de la clasificación.

Para la clase Normal y clase Hoyos se compilo el código nuevamente con su respectiva imagen. El resultado para la clase Normal fue [0.99742055, 0.00253926, 0.00004016] que es un 99.74 % de precisión, y para la clase Hoyos el resultado fue [0.00003843, 0.00062527, 0.9993363] que es un 99.93 % de precisión (figura 14 y figura 15).

Figura 14. Clasificar clase Normal.

Figura 15. Clasificar clase Hoy

Por lo tanto el código sigue dando buenos resultados en nuestro equipo, al igual que en la web de Teachable Machine.

Conclusiones

Al utilizar las CNN podemos automatizar procesos que la visión humana realiza, algunos ejemplos de esto son: reconocimiento óptico de números y letras, para reconocimiento de matrículas, escaneo de códigos, codificación de texto, etc.

Gracias a esto se desarrolló un clasificador de imágenes con tres clases, la cual clasifica con un gran porcentaje de acierto la imagen cargada al modelo previamente entrenado. Se buscó tener una gran precisión para que el sistema no fallara y diera un resultado muy certero, y así sea una herramienta de confianza para los especialistas encargados del análisis de estructuras.

Aunque en el texto solo se mostró el primer y último entrenamiento, este se realizó varias veces y con diferentes parámetros para lograr que fuera lo suficientemente preciso, gracias a esto en cada entrenamiento se observaba la mejora de la clasificación y a la hora de hacer las pruebas.

Referencias

1. Arista Jalife, A., Calderón Auza, G., Fierro Radilla, A., & Nakano, M. (**2017**). *Clasificación de Imágenes Urbanas Aéreas: Comparación entre Descriptores de Bajo Nivel y Aprendizaje Profundo . Información Tecnológica* 28, 209-224.

2. Arriola Oregui, I. (**2018**). *Detección de objetos basada en Deep Learning y aplicada a vehículos autónomos.*Universidad del País Vasco.

3. Bagnato, J. (**08 de 11 de 2018**). *Clasificación de Imágenes en Python.(Aprende Machine Learning)*Obtenido de https://www.aprendemachinelearning.com/clasificacion-de-imagenes-en-python/

4. Brownlee, J. (**2019**). *Understand the Impact of Learning Rate on Neural Network Performance. (Machine Learning Mastery Pty. Ltd.)*Recuperado el 22 de 05 de 2020, de https://machinelearningmastery.com/understand-the-dynamics-of-learning-rate-on-deep-learning-neural-networks/

5. Contreras Zaragoza, O. E. (**2018**). *Desarrollo de una red neuronal convolucional para el procesamiento de imágenes placentarias.*Ciudad de Mexico.

6. Google. (**11 de 2019**). *Teachable Machine.(Google Creative Lab)*Recuperado el 21 de 05 de 2020, de https://teachablemachine.withgoogle.com/

7. Instinto Programador. (**14 de 06 de 2019**). *Reconocimiento de imágenes en Python con TensorFlow y Keras. (Instinto Programador)* Obtenido de https://www.instintoprogramador.com.mx/2019/06/reconocimiento-de-imagenes-en-python.html

8. Murch, S. (**4 de 12 de 2018**). *Machine Learning/AI for Kids: Resources.(THEMESPHERE)* Recuperado el 23 de 05 de 2020, de https://www.stevemurch.com/machine-learning-ai-for-kids-resources/2018/12

9. Rodríguez-Sahagún Alesanco, P. (**2018**). *Aplicaión de redes neuronales convolucionales y recurrentes al diagnóstico de autismo a partir de resonancias magnéticas funcionales.* Madrid.

10. Rull, M. V. (**2016**). *Reconocimiento de Objetos usando Deep Learning.*Sevilla: Escuela Técnica Superior de Ingeniería .

11. SAS Institute. (**2019**). *Deep Learning. Qué es y por qué es importante.(SAS Institute)*Recuperado el 13 de 05 de 2020, de https://www.sas.com/es_mx/insights/analytics/deep-learning.html

12. Shafkat, I. (**01 de 06 de 2018**). *Intuitively Understanding Convolutions for Deep Learning. (Medium)* Obtenido de https://towardsdatascience.com/intuitively-understanding-convolutions-for-deep-learning-1f6f42faee1

13. Spencer, B., Hoskere, V., & Narazaki, Y. (**2019**). *Advances in Computer Vision-Based Civil Infrastructure Inspection and MonitoringEngineering*5(2), 199-222.

14. TensorFlow. (**06 de 02 de 2020**). *Basic classification: Classify images of clothing.(TensorFlow)* Obtenido de https://www.tensorflow.org/tutorials/keras/classification?hl=en

15. Vaggalis , N. (**18 de 10 de 2017**). *Google's Teachable Machine - What it really signifies.(i-programmer)*Recuperado el 23 de 05 de 2020, de https://www.i-programmer.info/news/105-artificial-intelligence/11219-googles-teachable-machine-what-it-really-signifies.html

16. Vizcaya Cárdenas, R. (**2018**). *DEEP LEARNING PARA LA DETECCIÓN DE PEATONES Y VEHÍCULOS SOBRE FPGA.*Estado de México: Universidad Autonoma del Estado de México.