Entrenamiento de una Red Neuronal Artificial para Reconocimiento de Carros en una Raspberry PI 3 como Asistencia a los Motociclistas

Technical Report · June 2021

DOI: 10.13140/RG.2.2.28362.90565

5 authors, including:

Kevin Ricardo Gomez Muñoz
Sergio Arboleda University
1 PUBLICATION **0** CITATIONS
SEE PROFILE

Omar Alvarez
Central University (Colombia)
4 PUBLICATIONS **0** CITATIONS
SEE PROFILE

Erney David Garcia Vergara
Sergio Arboleda University
1 PUBLICATION **0** CITATIONS
SEE PROFILE

Kevin Ricardo Hernandez Beltran
Sergio Arboleda University
1 PUBLICATION **0** CITATIONS
SEE PROFILE

UNIVERSIDAD SERGIO ARBOLEDA

Escuela de Ciencias Exactas e Ingeniería
Ingeniería Electrónica

Entrenamiento de una Red Neuronal Artificial para Reconocimiento de Carros en una Raspberry PI 3 como Asistencia a los Motociclistas

Kevin Ricardo Gómez Muñoz – Erney David García Vergara – Kevin Ricardo Hernández Beltrán – Richard Anderson Suan Yara – Omar Alvarez Herrera

Resumen: El presente informe muestra el entrenamiento y funcionamiento de un modelo de red neuronal artificial en lenguaje Python que tiene como objetivo reconocer carros a cualquier hora del día, este se ha implementado en la placa de desarrollo Raspberry PI 3B, para esto se utiliza una biblioteca de código abierto de computación numérica llamada TensorFlow, así mismo se utiliza otra biblioteca llamada Keras que se integra con TensorFlow como backend. Para comprobar el funcionamiento de la neurona se realiza una prueba con dos videos, en los cuales uno contiene un carro y el otro no, así mismo en cada uno se presentan dos escenarios "si hay carro" y "no hay carro", obteniendo así en el primer video una precisión de 94.81% para el primer escenario, y una de 5.19% en el caso contrario; Para el segundo video en el que no se encuentra ningún vehículo se obtuvo una precisión de 67.69% en cuanto al primer escenario y para el segundo se obtuvo una de 32.31%, como se puede observar, se obtuvo una exactitud bastante cercana al 100% en el primer video, sin embargo en el segundo no, esto es porque la formación del dataset, configuración de parámetros, cómo el tamaño del lote, tasa de aprendizaje, numero de épocas, entre otras consideraciones deben ser ajustadas detalladamente para obtener un reconocimiento lo más acertado posible.

I. MARCO TEÓRICO

Entre los conceptos esenciales de este desarrollo se encuentra el modelo de red neuronal artificial, que consta de los siguientes tópicos, la arquitectura de la red, el algoritmo de descenso del gradiente, las funciones de activación, cómo está compuesta una red neuronal, así como que es una neurona, como funciona, entre otros.

Teniendo en cuenta lo anterior, un modelo es una representación matemática de la concepción de la realidad de manera simplificada, es decir representa un conjunto de la realidad de la forma más completa y precisa posible, pero sin llegar a ser una copia exacta de la misma, por ejemplo, el dibujo de un animal podría representar el mismo, a esto se le llamaría modelo. [7]

Ahora, una red neuronal artificial no es más que una simulación artificial de un cerebro humano, así como los cerebros, tiene un elemento muy importante el cual es la neurona, en la computación las neuronas son modelos encargados de hacer un procesamiento con unos estímulos externos en otras palabras con los datos sobre los que se realiza operaciones internas que entregan al final del proceso datos de salida, para esto se utilizan varias neuronas ubicadas en distintas capas permitiendo así un aprendizaje jerarquizado como se muestra en la ***Figura 1*** en donde las primeras capas de la red neuronal reciben los pixeles y son procesados y entregados a las capas ocultas, luego envían sus datos a las capas de salida, por ejemplo, si se tiene una imagen de 10x10 pixeles se va a tener 100 neuronas de la capa de entrada para recibir la imagen, a continuación en las capas ocultas estos datos serán procesados por diferentes filtros.[8]

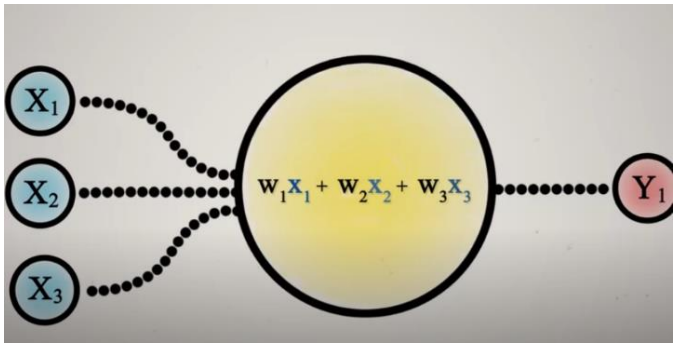

Figura 1. *Capas de una red neuronal artificial. Imagen tomada de [10].*

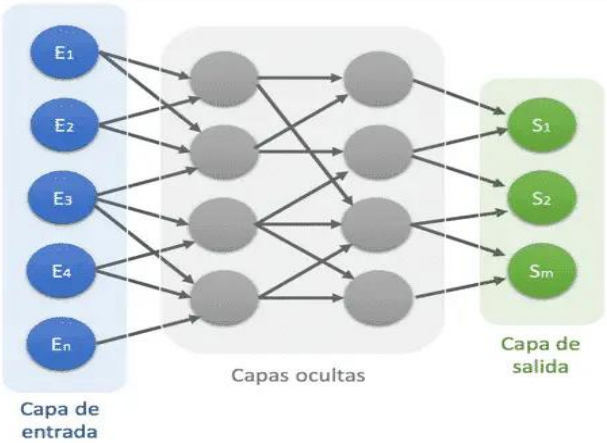

Figura 2*. Diagrama de una Neurona artificial. Imagen tomada de [9].*

En la figura 2 se muestra el funcionamiento de una neurona artificial en donde se realiza un cálculo interno, este es la suma ponderada de todas las entradas, a cada entrada se le asigna un peso sináptico, el cual genera un valor de salida dependiendo del valor de ingreso y sus ponderaciones, al resultado de esta suma ponderada se le aplica una función no lineal con el objetivo de empezar a clasificar los resultados[11], estas funciones pueden ser ESCALONADA, SIGMOIDE, TANH y

la que se usa en este trabajo ReLU (Unidad Rectificada Lineal), la cual convierte a cero los elementos del co-dominio Y en los que su dominio X es negativo, por lo cual esta función es la mas adecuada, en el aprendizaje de redes neuronales enfocadas en imágenes, dado que su uso permite encontrar un patrón en cada capa de convolución, los valores negativos no interesan y los valores positivos deben pasar a la siguiente convolución, cada una puede ofrecer diferentes beneficios según su uso[19]. Por último, con el objetivo de disminuir al máximo la desviación en la salida de las redes neuronales artificiales se aplica el algoritmo del descenso del gradiente, el cual consiste en un vector que entrega la dirección en la cual la función aumenta o disminuye más rápido esto permite ajustar los distintos parámetros de la red según sea necesario. [11]

La arquitectura de red neuronal implementada en este proyecto es la red neuronal convolucional basada en la arquitectura LeNet, la arquitectura tiene varios conjuntos de capas convolucionales, dedicadas a aplicar diferentes filtros sobre las imágenes para identificar diversos patrones que pueda tener y luego está la agrupación "pooling" se usa para reducir el tamaño de la imagen para así en las siguientes capaz procesar una imagen más pequeña. [10]

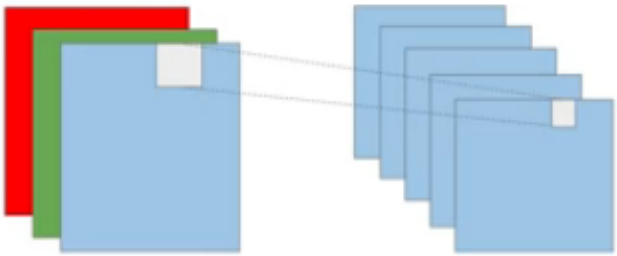

Figura 3. *Profundidad de una imagen. Imagen tomada de [10].*

Básicamente se procesa por partes la imagen como se muestra en la figura 3 para disminuir el tiempo de procesamiento.

Para resumir, una red neuronal artificial es un algoritmo computacional que se usa para predecir o clasificar algún elemento, su funcionamiento se basa en la interconexión de una gran cantidad de neuronas artificiales organizadas en capas de entrada, capas ocultas y capas de salida, todas estas son la unidad base, las cuales reciben estímulos externos con los que se realizara una serie de procesos para obtener la predicción deseada.[11]

Después de tener claro que es, de que se compone y como funciona un modelo de red neuronal artificial es importante explicar los dos FrameWorks de trabajo para inteligencia artificial estos son TensorFlow y Keras ambas bibliotecas de código abierto que facilitan la implementación de la red en lenguaje Python.

TensorFlow: Es una biblioteca de código abierto desarrollada por Google Brain basada en un sistema de redes neuronales para el aprendizaje profundo, es usada para la visión por computador, el procesamiento de lenguaje natural, entre otros análisis predictivos, por lo que permite analizar patrones y correlaciones. [1][2]

Keras: Es una biblioteca de alto nivel que proporciona bloques para construir redes de aprendizaje profundo, Keras a diferencia de TensorFlow que se basa en diferenciación automática, tiene como base una composición de capas, sin embargo, corre sobre esta o sobre Theano por lo que se podría traducir como una manera más sencilla y rápida de crear modelos de aprendizaje profundo usando la biblioteca TensorFlow. [3][5]

Como se mencionó anteriormente la red neuronal y cada neurona necesita datos externos para su entrenamiento, estos se llaman datasets, para nuestro caso serán todas esas imágenes que tienen y no tienen un automóvil en ellas.

II. RECURSOS UTILIZADOS

A. Software:

- Python v3.7
- Terminal de Linux
- Git
- TensorFlow
- OpenCv
- Keras

B. Componentes:

- Raspberry pi 3B +
- Cámara OV5647 (5 Mp)
- Computador con SO Linux

III. PROCEDIMIENTO

Para llevar a cabo el proyecto se requiere entrenar una red neuronal artificial capaz de reconocer autos en movimiento a cualquier hora del día. Para conseguir que la red neuronal funcione es necesario realizar la instalación de librerías como OpenCv, TensorFlow y Keras tanto en el computador con S.O. Linux como la Raspberry. Dichas instalaciones se realizan de la siguiente forma:

1. OpenCv: Para instalar esta librería en la Raspberry como en el PC se utiliza los siguientes comandos:

```
1. sudo apt-get update
2. sudo apt-get upgrade
3. pip install opencv-contrib-python
```

Para ampliar la información referente a los pasos para instalar esta librería remitirse a la página web [14]

2. TensorFlow: Para realizar la instalación es necesario digitar en el terminal de Linux los siguientes comandos.

```
1. Pip3 install –user –upgrade
   tensorflow==1.14
```

Es importante que se instale la versión 1.14.0 de Tensorflow o inferior en la maquina donde se hace el entrenamiento puesto que el script de Python hace uso de la función **get_default_graph** de Tensorflow la cual fue renombraba en las versionbves superiores de la librería.

3. Keras: Al igual que la anterior librería es importante instalar una version anterior, en este caso se utiliza keras en su version 2.1.5 instalandola con el siguiente comando:

```
4. Pip3 install keras==2.1.5
```

Una vez finalizada la instalación se procedió a buscar información oficial en los sitios web de TensorFlow y Keras en donde se encontró ejemplos de código que sirvieron de base para la elaboración del necesario para necesario realizar el código de entrenamiento y el dataset con el objetivo de encontrar el modelo óptimo para la Raspberry

Por lado, se necesita un dataset adecuado para que al entrenar la neurona esta cuente con una probabilidad entre 90% y 100% al momento de detectar autos. Para esto se realiza un script en donde se descargan todas las imágenes de una búsqueda en Google; una vez descargadas se seleccionan aquellas imágenes en donde el carro se vea completamente de frente, diagonal y de lado. La idea de seleccionar este tipo de imágenes es para tener diferentes ángulos al momento de visualizar el auto. Luego de replicar con diferentes búsquedas el anterior procedimiento, se consigue un dataset de aproximadamente 450 imágenes, con lo cual se realiza el entrenamiento de la red neuronal.

Figura 4. *Pruebas de Dataset creado con 450 imágenes***.**

Al momento de realizar la prueba del dataset anteriormente descrito, mediante de un video donde se muestra una carretera con varios autos en movimiento, se puede visualizar que en momentos puntuales el modelo no detecta los autos, además se comprueba1 con imágenes en donde hay un carro y otra donde hay una calle sola, como se muestra en la figura 5, se puede observar que la probabilidad de que reconozca un carro es de un 85%, la cual es baja teniendo en cuenta que se desea una probabilidad de más del 90% en el reconocimiento.

Figura 5. *Pruebas de reconocimiento de autos.*

También se utiliza la cámara con el fin de determinar bajo una prueba real la eficacia del modelo, para esto se configura el sistema Raspbian, para habilitar la cámara se realiza lo siguiente:

Primero se conecta la cámara a la Raspberry en el puerto CSI (Camera Serial Interface) como se muestra en la figura 6.

***Figura 6**. Conexión física de la Raspberry. Imagen tomada de [17].*

Una vez este la cámara conectada, se ejecuta el siguiente comando en la consola de la Raspberry el cual abre el menú de configuraciones:

```
1. sudo raspi-config
```

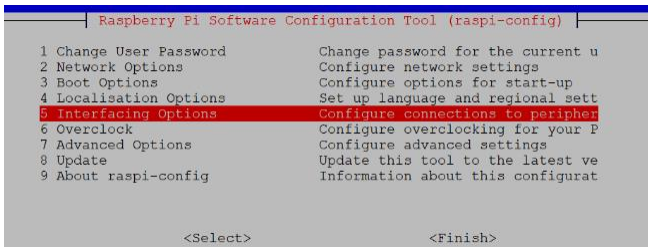

***Figura 7**. Menú de configuración de Raspberry tomado de [18].*

Estando en el menú se ingresa en "Interfaces Options". Allí se encuentra la opción de "camera", la cual debe establecerse en "enable" para el correcto funcionamiento de la cámara.

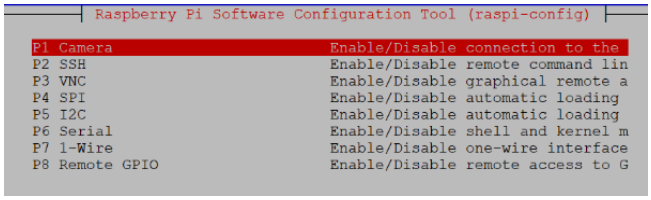

***Figura 8**. Menú de activación de cámara tomado de [18].*

Una vez terminada la configuración den la cámara, se hace la ejecución del script del script que hace el entrenamiento, al analizar el resultado de este entrenamiento no tiene la eficiencia que se desea. Por esto se determina que se debe ampliar el dataset con el fin de buscar un mejor entrenamiento, este análisis se hace atreves de una gráfica genera al momento de realizar el proceso de entrenamiento, como se ve en la ***Figura 9***.

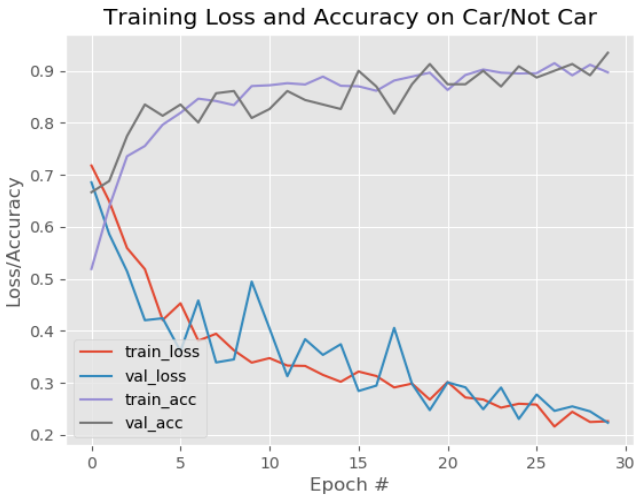

Figura 9. Grafica de pérdidas y precisión del primer entrenamiento.

Para conseguir una mayor cantidad de imágenes, se opto por buscar datasets que contaran con aproximadamente de 3000 a 5000 imágenes (concretamente 5), teniendo como resultado final un total de 20.000 imágenes de carros y 30.000 imágenes que no tienen relación con carros.

Una vez formado el dataset, se entrenan varios modelos variando los siguientes parámetros: las épocas, las cuales son el número de veces que se va a repetir el entrenamiento, el batch size que determina la cantidad de imágenes que se van a usar por época y por último el tamaño del recorte de las imágenes, estos tres parámetros se modifican en el script de entrenamiento. Luego de entrenar aproximadamente 50 modelos se llegó a la conclusión de que los valores ideales para las épocas, batch size y tamaños de imágenes son 200 épocas, 10 batch size y el tamaño de las imágenes sea de 40x40, ya que si son pocas épocas no el modelo no aprende lo suficiente y en caso contrario el tiempo de entrenamiento sube considerablemente a unas 10 horas por modelo o podría sobre ajustar el modelo

Para finalizar, se hace el entrenamiento desde el computador en donde está almacenada la red neuronal artificial con el ultimo dataset creado, lo cual da como resultado un archivo de tipo modelo, este se transfiere a la Raspberry en donde se prueba el modelo ejecutando el script de prueba como se muestra en la siguiente sección.

IV. RESULTADOS

En la figura 10 se muestra el porcentaje de las pérdidas y la precisión que se esperaban y que se ha obtenido durante las épocas de entrenamiento de la red neuronal, allí se representa

en rojo las pérdidas y en azul las perdidas reales calculadas durante el proceso, en morado la precisión esperada y en negro la precisión real obtenida.

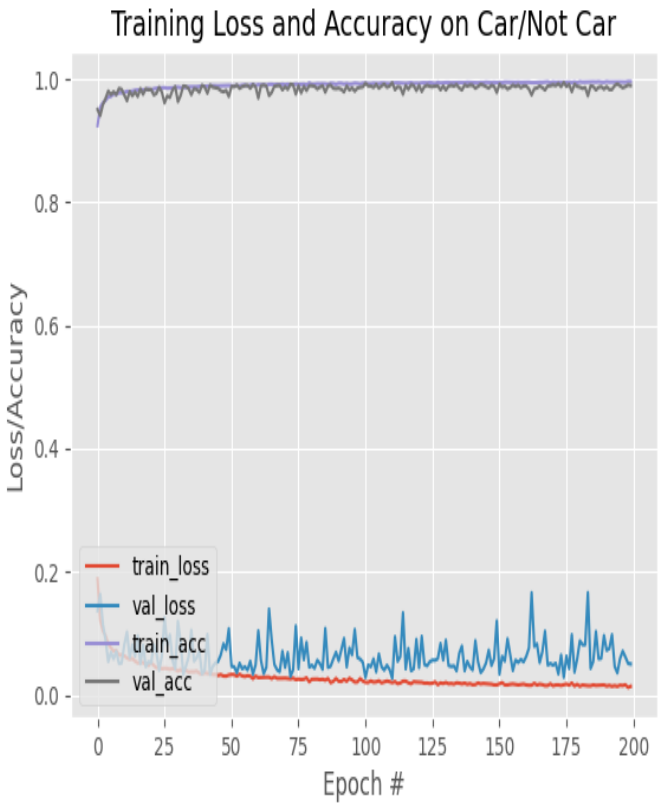

Figura 10. *Perdidas y precisión de entrenamiento.*

Como se muestra en la anterior gráfica, la red entrenada cuenta con excelentes valores de precisión y bajas perdidas ya que dichos valores son muy cercanos al 100% y 0% respectivamente, esto producto en gran parte del dataset de imágenes con la que está entrenado, mediante esta grafica también se puede determinar si la red puede llegar a obtener predicciones acertadas o no, cabe aclarar que esto no determina la funcionalidad requerida de la red neuronal.

Por otro lado, se pone a prueba la red neuronal dentro de la Raspberry PI 3, para esto se usaron dos videos, el video [13] muestra una grabación de una cámara en la parte trasera de un auto conducido en el tráfico por lo que en el 100% del video se logra observar al menos un carro y el video [12] no tiene relación alguna con autos es decir que en la totalidad del mismo no se muestra ninguno, los resultados se muestran en las siguientes graficas de cada uno de los escenarios mencionados.

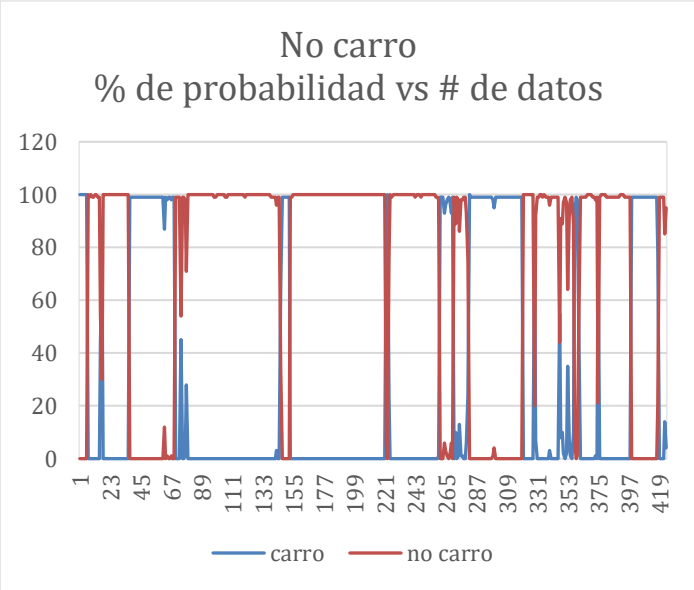

Figura 11. *Probabilidades VS datos en escenario sin carros.*

La figura 11 representa un escenario sin carros, en el cual la red neuronal presenta un comportamiento no esperado durante el transcurso del video, a pesar de esto, cumple su función la mayor parte de la prueba, obteniendo que, en promedio, la probabilidad de no detección de carro fue de 70.19% lo que quiere decir que en promedio el 29.28% fue el porcentaje de probabilidad de detectar un carro, lo cual es muy elevado dado que no existe ningún auto en el video. Cabe aclarar que dichos porcentajes y los que se mencionaran a continuación en posteriores gráficas, hacen referencia al porcentaje de predicción obtenidos por la red neuronal a los cuales se les calculo el promedio luego de 424 frames analizados por la red.

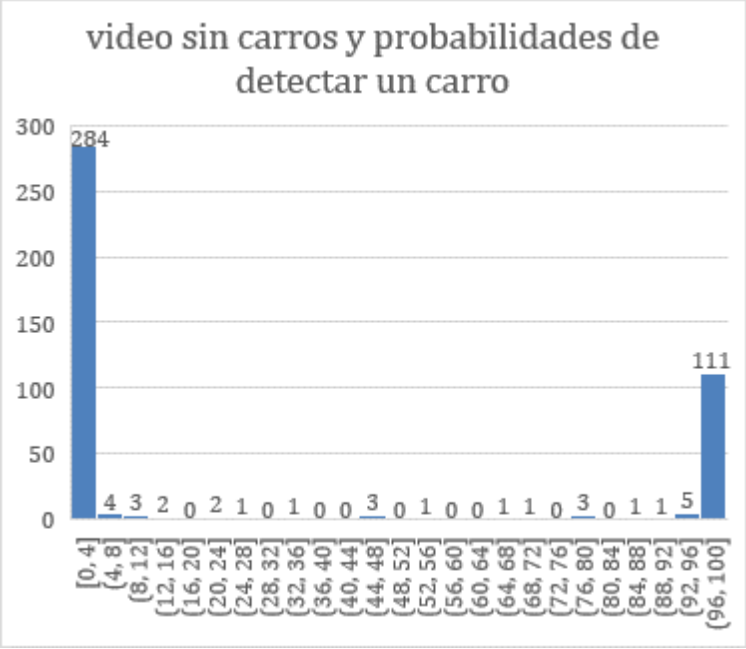

Figura 12. *Histograma de detectar un carro en escenario sin carros.*

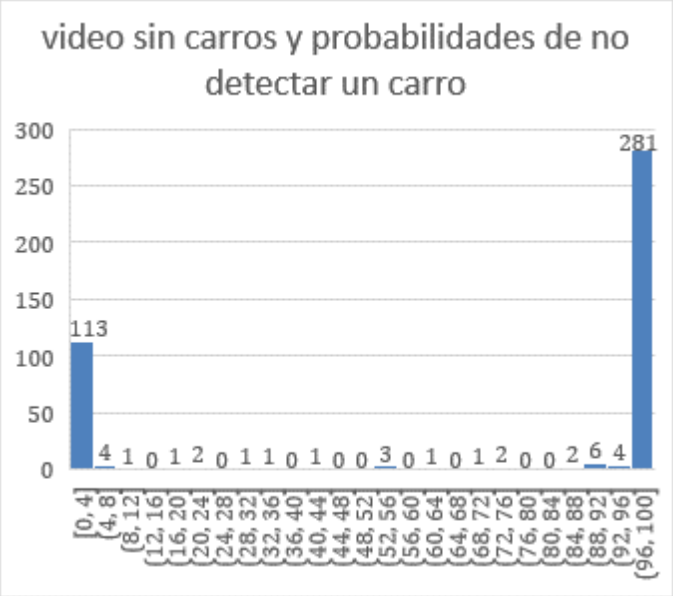

Figura 13. *Histograma de no detectar un carro en escenario sin carros.*

En las figuras 12 y 13 se muestra la distribución de los datos obtenidos durante el escenario sin carros, en estas se logra determinar que la mayoría de datos tomados se encuentran concentrados en los extremos de sus respectivos casos y una pequeña pero considerable concentración en el extremo opuesto lo que indica un comportamiento no esperado por la red neuronal.

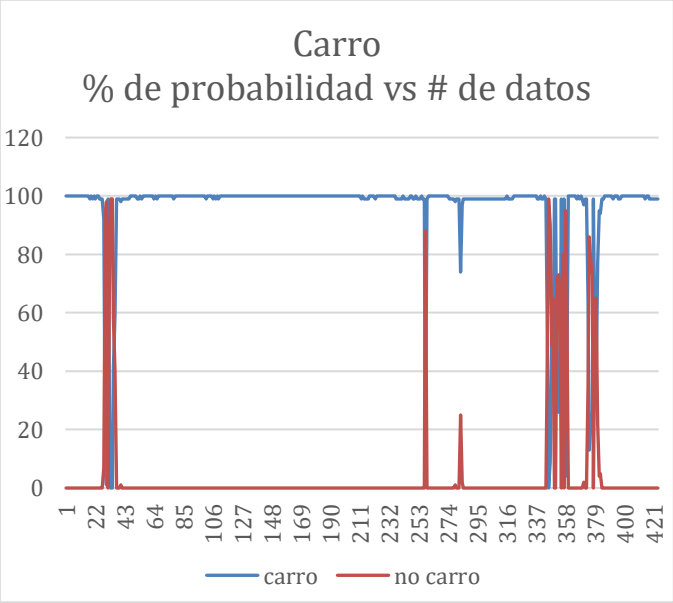

Figura 14. *Probabilidades VS datos en escenario con carros.*

Ahora, la figura 14 muestra el escenario contrario donde se reproduce un video en el cual siempre se observa un auto y como se logra ver en la misma, la predicción es excelente, pues son pocas las veces durante el video que falla la precisión, por lo que se detectó que hubo un auto con una probabilidad del 95,07% en promedio, mientras que tan solo el 4,56% corresponde a la probabilidad promedio de no detectar un carro.

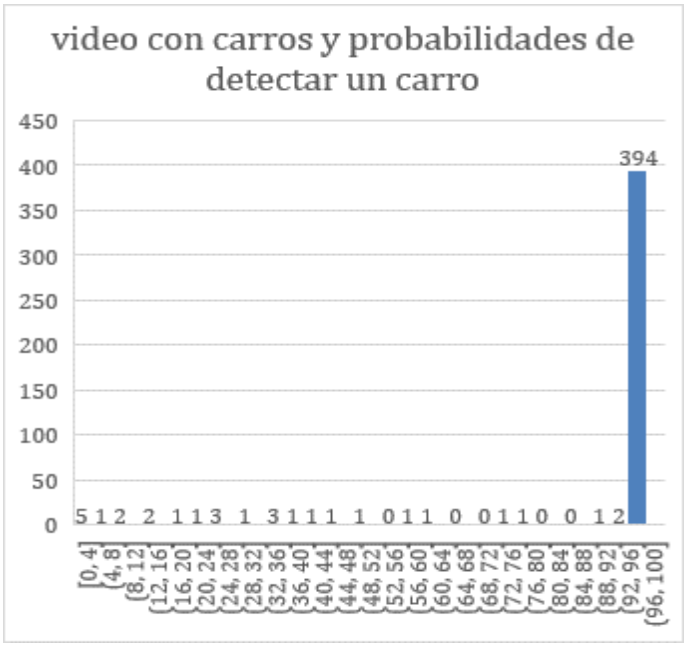

Figura 15. *Histograma de detectar un carro en escenario con carros.*

Figura 16. *Histograma de no detectar un carro en escenario con carros.*

Como se observa en los anteriores histogramas de las figuras 15 y 16, la distribución de los datos es casi ideal, la mayoría de la información se ubican en los extremos según sea el caso por lo que la red neuronal funciona de manera óptima ante la presencia de vehículos aportando un avance importante al resultado final.

A continuación, en la figura 17, se muestra un resumen de las pruebas realizadas, en esta se puede observar la cantidad de datos que se tomaron lo cual se puede traducir a la cantidad de frames predichos durante el video, la cantidad que se predijeron como presencia de carro y los que no dependiendo del escenario del video y su respectivo porcentaje.

	hay carro		no hay carro		Total	
Video	cantidad	%	cantidad	%	cantidad	%
no carro	124	29,25	300	70,75	424	100,00
carro	402	94,81	22	5,19	424	100,00

Figura 17. *Resumen de datos recolectados para las pruebas.*

Por último, aunque no menos importante, a lo largo de las pruebas se mantuvo una frecuencia de visualización de frames de 30 FPS constantes como se puede evidenciar en la figura 18, lo que representa un óptimo comportamiento ya que la diferencia entre el movimiento real y el movimiento proyectado en la Raspberry PI es apenas notable.

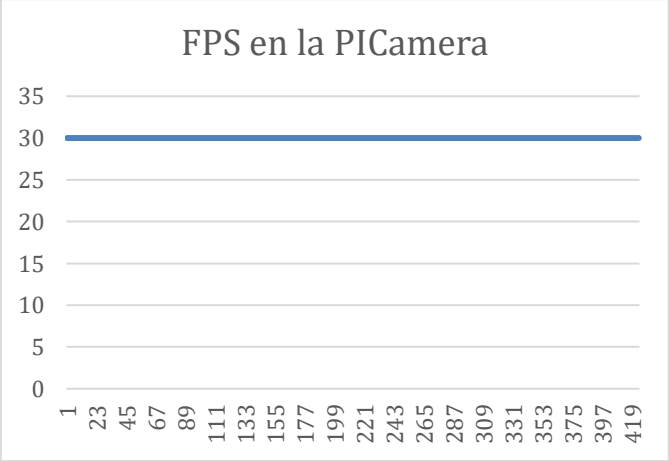

Figura 18. *FPS de la PICamera durante las pruebas*

V. CONCLUSIONES

Se concluye que el dataset debe ser muy variado en cuanto a las diferentes condiciones climáticas y de luz ya que, si se tiene imágenes de carros por el día y luego se prueba la red neuronal con la imagen de un carro de noche, esta entrega una predicción errada. Por otro lado, el dataset debe ser variado con diferentes modelos y marcas de carros ya que si se tienen imágenes de un solo modelo y marca la red neuronal solo va a reconocer ese modelo y marca entonces cuando se pruebe con otro modelo este va a dar una predicción con una baja probabilidad o errada.

Por otro lado, los factores clave de los que depende el correcto funcionamiento de la red neuronal son: tamaño del lote, tasa de aprendizaje y las épocas o iteraciones ya que de estos depende de cómo se va a entrenar la red neuronal.

Aun así, teniendo en cuenta lo anterior, la principal limitante para llevar acabo el entrenamiento de la red neuronal proviene del computador donde se ejecuta el script ya que antes y durante el entrenamiento de la red neuronal este necesita recursos de procesamiento y memoria , para empezar se necesita un espacio en el disco duro el cual es ocupado por el dataset en el caso del proyecto el dataset pesa 20 GB ,luego cuando se está ejecutando el script se necesita una memoria RAM de mínimo 4 GB ya que Linux necesita mínimo 2GB de RAM para funcionar correctamente y las otras 2GB son usadas en el procesamiento y sí el procesamiento ocupa más de 2GB e intenta usar de la RAM que usa Linux para funcionar este va a cancelar el entrenamiento tambien se necesita un procesador potente para realizar las operaciones necesarias, para evitar el colapso de linux y lograr entrenar una red neuronal de manera adecuada hay dos soluciones posibles la primera es aumentar la memoria RAM a una mayor de 4GB además de agregar una tarjeta gráfica para liberar la carga CPU y la segunda opción es disminuir el tamaño del lote y de las épocas pero esto sería contraproducente ya que disminuiría la precisión del modelo .

En cuanto a los resultados de la red neuronal esta tiene un porcentaje de acierto bajo de aproximadamente 67.69% cuando en el fotograma no hay un carro mientras que por la otra parte cuando hay un carro presente el porcentaje de acierto es de 94% esto se debe a que la red neuronal reconoce dos categorías carro y objetos que no son carros y en el dataset en la categoría de no es un carro no hay un patrón que la red neuronal pueda seguir fácilmente ya que principalmente se compone de calles vacías.

REFERENCIAS

[1] *J. Dillon et al., "TensorFlow Distributions", arXiv.org, 2020. Available: https://arxiv.org/abs/1711.10604*

[2] *J. Buhigas, "Todo lo que necesitas saber sobre TensorFlow, la plataforma para Inteligencia Artificial de Google – Puentes Digitales", Puentesdigitales.com, 2020. Available: https://puentesdigitales.com/2018/02/14/todo-lo-que-necesitas-saber-sobre-tensorflow-la-plataforma-para-inteligencia-artificial-de-google/*

[3] *T. Hope, I. Lieder and Y. Resheff, Learning TensorFlow.*

[4] *Ketkar N. "Introducción a Keras". Deep Learning con Python. Available: https://doi.org/10.1007/978-1-4842-2766-4_7*

[5] *J. Utrera, "Deep Learning básico con Keras (Parte 1)", 2018. Available: https://enmilocalfunciona.io/deep-learning-basico-con-keras-parte-1/*

[6] *J. Torres, "Deep Learning – Introducción práctica con Keras - Jordi TORRES.AI", Jordi TORRES.AI, 2020.*

Available: https://torres.ai/deep-learning-inteligencia-artificial-keras/

[7] *J. Wadsworth, Análisis de sistemas de producción animal. Roma: Organización de las Naciones Unidas para la Agricultura y la Alimentación, 1997.*

[8] *"IBM Knowledge Center", Ibm.com, 2019. Available: https://www.ibm.com/support/knowledgecenter/es/SS3RA7_sub/modeler_mainhelp_client_ddita/components/neuralnet/neuralnet_model.html*

[9] *D. Calvo, Diegocalvo.es, 2017. Available: https://www.diegocalvo.es/definicion-de-red-neuronal/*

[10] AMP Tech, *Redes neuronales convolucionales CNN*. 2020.

[11] Dot CSV, *¿Qué es una Red Neuronal? Parte 1 : La Neurona | DotCSV*. 2020.

[12] *Alejandro Pérez, Este USB es más POTENTE que 7 ORDENADORES minando. 2020.*

[13] *Gregory Marketing Limited, DS301 Driving Recorder - Rear Cam (Day). 2017.*

[14] *"PyImageSearch - You can master Computer Vision, Deep Learning, and OpenCV.", PyImageSearch. Available: https://www.pyimagesearch.com*

[15] *"Instala TensorFlow con pip", TensorFlow. Available: https://www.tensorflow.org/install/pip?hl=es-419*

[16] *"How to Install Keras | Liquid Web", Liquid Web. Available: https://www.liquidweb.com/kb/how-to-install-keras/*

[17]*"Cómo Instalar una cámara en Raspberry Pi: Todo lo que necesitas saber", Descubrearduino.com, 2020. [Online]. Available: https://descubrearduino.com/instalar-una-camara-en-raspberry-pi/.*

[18] *"SPI - Raspberry Pi Documentation", Raspberrypi.org, 2020.[Online].Available: https://www.raspberrypi.org/documentation/hardware/raspberrypi/spi/README.md*

[19] *D. Calvo, "Función de activación – Redes neuronales", Diegocalvo.es, 2020. Available:* https://www.diegocalvo.es/funcion-de-activacion-redes-neuronales/#:~:text=Definici%C3%B3n%20de%20funci%C3%B3n%20de%20activaci%C3%B3n,o%20(%2D1%2C1)