Detección de la ocupación de parqueaderos usando redes neuronales convolucionales

Carlos Gustavo Veléz Manco, Felipe Henao Gomez, Maria Isabel Arango Palacio
cgvelezm@eafit.edu.co, fhenaog3@eafit.edu.co, miarangop@eafit.edu.co
Optimización - Departamento de Ciencias Matemáticas
Universidad EAFIT

***Resumen*—Las personas pueden pasar el promedio 7 minutos buscando un lugar para estacionarse, especialmente en zonas con alto vehicular. Actualmente, muchas de las soluciones brindadas para este problema han sido para espacios cerrados usando sensores con altos costos. Es por esto y por la gran efectividad de las Redes Neuronales Convolucionales (CNN) en la clasificación de imágenes, que en este trabajo se implementará un modelo CNN para determinar la ocupación de un estacionamiento, basados en imágenes de estacionamientos tomadas por diferentes cámaras en ambientes abiertos. Como también, es de gran interés minimizar el error entre la predicción de la red neuronal y la clasificación real. Se reporta una precisión del modelo del 97% y una perdida de 3% lo cual indica que este método de bajo costo para detectar estacionamientos es bastante eficaz y tiene gran potencial para ser aplicado como solución en diferentes zonas de las ciudades.**

***Index Terms*—Redes Neuronales Convolucionales, Estacionamiento, Clasificación, Reconocimiento de Imagenes, Convolución.**

I. Introducción

El hombre en su búsqueda constante de mejorar sus condiciones de vida ha logrado reducir el trabajo en aquellas actividades en donde su presencia juega un papel primordial. Los progresos obtenidos han permitido dirigir estas actividades a otros campos, como por ejemplo, a la construcción de máquinas que permitan resolver problemas de manera rápida y automática. Los desarrollos actuales, se dirigen al estudio de las capacidades humanas como una fuente de nuevas ideas para el diseño de las nuevas máquinas. Así, la inteligencia artificial es un intento por descubrir y describir aspectos de la inteligencia humana que pueden ser simulados mediante máquinas.

Las redes neuronales no son más que una de estas formas de emular características de los seres humanos, como la memoria y la asociación de hechos, inspiradas en la célula más importante del sistema nervioso: la neurona. Existen muchos tipos de problemas a los que se puede enfrentar a través del uso de redes neuronales. Una de las principales aplicaciones de las redes neuronales es el reconocimiento de imágenes, una subcategoría de "computer vision", que tiene como objetivo principal reunir, procesar y analizar datos del mundo real extraídos de imágenes. Por tanto, el problema que se va a resolver es de clasificación, en donde para cada entrada, que para este caso será una imagen, se debe decidir a qué categoría pertenece, parqueadero con carros o vacío.

II. Objetivos

A. *Objetivo General*

Diseñar un programa que reconozca, mediante el uso de redes neuronales, si un parqueadero está vacío o no de manera que se pueda mejorar la precisión y reducir el costo.

B. *Objetivos Específicos*

- Comprender a detalle los procesos que realizan algunas funciones de Python para el uso de redes neuronales.
- Realizar una red neural convolucional con gran capacidad de aprendizaje para la categorización de múltiples imágenes.
- Utilizar técnicas de optimización para mejorar la precisión de la red.

III. Marco teórico

Las redes neuronales hacen parte de un subconjunto de Machine Learning llamado Deep learning (DL). Los algoritmos de aprendizaje profundo son aquellos más se aproximan al funcionamiento del cuerpo humano, en especial, al cerebro. Estos, tienen como principal característica que son capaces de aprender patrones de ocultos dentro de los datos, más claramente, en gran cantidad de datos.

Así pues, una Red Neuronal Convolucional (CNN) es una clase de redes neuronales profundas y artificiales la cual es aplicada comúnmente a el análisis de imágenes[1]. Estas redes se caracterizan principalmente por su uso del proceso matemático de convolución para la extracción de características y patrones en el conjunto de datos. Las CNN constan de una capa de entrada, capas ocultas y una capa de salida. En las capas intermedias u ocultas las entradas y salidas están sujetas a una función de activación y a un proceso de convolución[2].

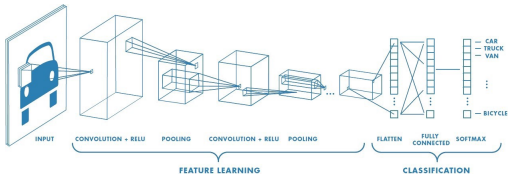

Fig. 1. Representación de las etapas de una CNN

1) **Proceso de convolución**
Antes de mencionar lo correspondiente a la etapa de convolución es importante recordar que, una imagen no es

más que una matriz de valores de píxeles característicos ya sean en 2D o 3D, dependiendo del tipo de imagen que se tenga. En esta primera parte del proceso de la red, se extraen básicamente las características principales de cada imagen que van a permitir hacer una clasificación. En un principio, las características que se extraen son las básicas como los diferentes bordes y formas especiales, y a medida que se va moviendo más adentro de la red, las capas empiezan a detectar características más profundas. Ahora bien, estas características son extraídas usando los filtros kernel, que son aquellos que especifican las características que queremos obtener de las imágenes. Luego de definirlos, se realiza la operación de convolución entre la matriz de entrada(la imagen) y el kernel, que es básicamente el producto escalar entre ambas matrices obteniendo como resultado una nueva matriz con las características filtradas[3]. A esta nueva matriz, se le aplica la función de activación correspondiente, comúnmente la función Rectificadora de Unidades Lineales (ReLU), la cual esta definida como:

$$f(x) = x^+ = max(0, x) \tag{1}$$

Como la nueva matriz corresponderá a la nueva imagen que será el nuevo input de la siguiente capa de convolución, lo que se pretende con esta función de activación es coger cada entrada de la matriz y volver todos los valores negativos en 0, para así obtener las características visuales más claras y sin tanto ruido.

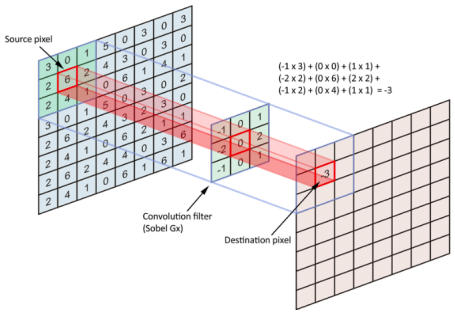

Fig. 2. Proceso de convolución

Este proceso de convolución es realizado tantas veces como capas de convolución de definan en la red neuronal. Es importante resaltar que, aunque no hay un método exacto para calcular cuantas capas deben haber, si debe de tenerse en cuenta que entre más capas, más extracción de características y entre más características más sobreajuste del modelo.

2) **Proceso de agrupación o Pooling**
En esta parte de la red neuronal se tienen capas de agrupación, las cual son responsables de reducir el tamaño de los datos combinando las salidas de los grupos de neuronas en una capa, en una sola neurona en la siguiente capa. Esto se realiza con el fin de disminuir el coste computacional requerido para procesar los datos y para que los datos sean invariantes en el espacio.
Hay dos tipos de agrupación comunes, el primero de ellos es agrupación máxima o Max Pooling, el cual encuentra el valor máximo de cada grupo de neuronas en el mapa de características. Este tipo de agrupación suprime el ruido en gran cantidad. En segundo lugar, esta la agrupación por promedios, la cual devuelve el promedio de los valores de la imagen cubierta por el kernel[3].

Después de ir a través de los dos procesos anteriormente mencionados, se puede decir que el modelo esta lo suficientemente preparado como para reconocer patrones y características. Esto, da paso al proceso final de la red que es el de clasificación. En el cual, se tendrá una capa totalmente conectada, la cual recibirá la forma procesada de la imagen recibida en el input inicial. La idea básicamente es transformar la matriz de entrada en un vector columna unidimensional, el cual agrupe todas las características para poder así mediante el entrenamiento con propagación hacia atrás permitir que el modelo distinga entre características dominantes e irrelevantes para poder hacer su clasificación, la cual esta sujeta a una función de activación (generalmente diferente a la usada en convolución) que le permite obtener la clasificación con el propósito deseado y regresarlo como output.

Expresado matemáticamente, en una red neuronal convolucional comenzando por una entrada x, cada capa que le sigue x_j, se calcula como [4]:

$$x_j = \rho W_j x_{j-1} \tag{2}$$

Donde W_j es un operador lineal y ρ es uno no lineal. Por lo general en una red neuronal convolucional, W_j es una convolución y ρ es una función de activación, usualmente se escoge el rectificador $\max(x, 0)$, como fue mencionado anteriormente, pero también hay otros como la sigmoide $1/[1 + \exp(-x)]$. Como W_j es lineal, se reescribir como la suma de convoluciones [5], esto es,

$$\begin{aligned} W_j x_{j-1}(u, k_j) &= \sum_k \sum_v x_{j-1}(v, k) w_{j,k_j}(u - v, k) \\ &= \sum_k (x_{j-1}(\cdot, k) * w_{j,k_j}(\cdot, k))(u) \end{aligned} \tag{3}$$

La variable u es usualmente submuestreada. De esta manera,

$$x_j(u, k_j) = \rho(\sum_k (x_{j-1}(\cdot, k) * w_{j,k_j}(\cdot, k))(u)) \tag{4}$$

Donde $*$ es el operador discreto convolución, el cual se define como:

$$(f * g)(x) = \sum_{u=-\infty}^{\infty} f(u) g(x - u) \tag{5}$$

Los operadores ρW_j propagan la entrada $x_0 = x$ hasta la última capa x_J. Luego, la salida $x_J = \phi_J(x)$ se manda a un clasificador, el cual usualmente se compone de capas de redes neuronales conectadas completamente [5], como se menciono anteriormente.

IV. PLANTEAMIENTO DEL PROBLEMA

Para el uso de redes neuronales, el problema que hay es con respecto a la precisión de estas, para esto es necesario hacer uso de un conjunto de datos para poder entrenar la red. En este proceso de entrenamiento lo que se busca es crear la estructura de la red para maximizar la precisión de esta, comparando el resultado obtenido por la red y el resultado real, y se busca que la diferencia sea lo menor posible.
Se busca optimizar los filtros $w_{j,k_j}(u,k)$ para minimizar el error promedio de clasificación en un conjunto de entrenamiento $\{(x_s, y_s) : \ s = 1, 2, ..., n\}$ y $x_s = (x_{1s}, x_{2s}, ..., x_{Is})$. Esto lo podemos escribir de la siguiente manera:

$$
\begin{aligned}
&min: \ \sum_{s=1}^{n} (f(z_s) - y_s)^2 \\
&s.a: \\
&\qquad x_J^s = \rho W_J(\rho W_{J-1}(...\rho W_1 x_s)...) \quad s = 1, ..., n \\
&\qquad z_s = \sum_{j=1}^{K} w_j x_{J,j}^s \quad s = 1, ..., n
\end{aligned}
\tag{6}
$$

Donde x_J^s denota la última capa de convolución para la muestra s, z_i denota la capa final de clasificación después de conectar todos los nodos de x_J^s, $x_{J,j}^s$ representa el j-ésimo nodo de la capa x_J^s, K es la cantidad de nodos en la capa final de convolución, W_j es la convolución para la capa j, w_j es el peso para el nodo $x_{J,j}^s$ para la capa final y $f(\cdot)$ es la función de activación utilizada para clasificar, luego las variables del problema son las entradas de cada W_j y cada w_j.

Por lo general, este problema de alta dimensionalidad no es convexo, sin embargo, la actualización de las convoluciones y pesos se hace mediante un algoritmo de propagación hacia atrás, que puede ser calculado mediante el gradiente descendiente. Para esto se necesita una gran cantidad de datos de entrenamiento [5].

V. METODOLOGÍA

Para poder diseñar la red neuronal que categorice las imágenes de los parqueaderos claramente se requiere una base de datos con suficientes imágenes de parqueaderos tanto llenos como vacíos. Se utiliza entonces una con alrededor de 12.000 imágenes para que el modelo se entrene (6171 para entrenamiento y 6413 para la evaluación)[6].

Esta base de datos resulta de tomar fotos de ciertos parqueaderos completos y recortar cada celda, es decir, las aproximadamente 12.000 imágenes son de celdas de parqueadero, ya que si se tienen en cuenta imágenes de un parqueadero completo, sería demasiado difícil predecir lo que se quiere.

Fig. 3. Recopilación de la base de datos

La entrada del modelo que se construyo son las imágenes, y la salida es '0' si el parqueadero está lleno o '1' si el parqueradero se encuentra vacío. Pero para realizar un correcto entrenamiento del modelo, se tienen que tratar un poco las imágenes antes de pasárselas a este. Se harán diferentes procesos para los conjuntos de imágenes de entrenamiento y de evaluación.

Es por esto que, para el conjunto de entrenamiento, primero se normalizan los pixeles de las imágenes entre un rango de 0 a 255 que es básicamente el rango genérico para el uso de redes neuronales, esto se hace para cambiar el nivel de intensidad de los pixeles. También se le hace un zoom a la imagen y por último se hace un efecto espejo girando la imagen de forma horizontal. Para el conjunto de evaluación, sólo se normalizan los pixeles en el mismo rango de 0 a 255, no se transforman de ninguna otra manera.

Además, se reducen todas las imágenes (tanto las de entrenamiento como las de testeo) a un mismo tamaño, en este trabajo, todas las imágenes a tratar quedaran de 64x64 pixeles. También, se elige un tamaño de conjunto de imágenes para que el programa las estudie todas por grupos, en este caso, tomamos conjuntos de 32 imágenes que serán evaluadas en cada ciclo del aprendizaje. Y por último, el arreglo que representa cada imagen será "aplastado" a un vector, es decir, se convierte a un arreglo de una sola columna, y es este el que se le entregará a la red.

Después de procesar las imágenes y que estas queden listas para ser procesadas, se crea una red neuronal convolucional que tendrá dos capas de convolución (aunque también se realizan comparaciones con 1 y 4 capas), las cuales serán las que generen un mapa de características para clasificar las imágenes como parqueaderos vacíos o llenos a través del uso de reconocimiento de los patrones que cada capa pueda reconocer. Cabe resaltar que estas neuronas tendrán como entrada el arreglo unidimensional obtenido luego de tratar todas las imágenes.

Luego, se define el aprendizaje del modelo. Para esto se crea una variable llamada *epochs* la cual será la cantidad de veces que se quiere que el algoritmo pase de la primer imagen de un subgrupo de 32 imágenes ya establecido

hasta la última y se devuelva hasta la primera, tratando de reconocer la mayor cantidad de patrones que ayuden con la categorización. Esta variable para nuestro ejemplo es de 25. Es decir, para cada grupo de 32 imágenes, el programa recorre desde la primera hasta la última y de nuevo hasta la primera en busca de patrones que le ayuden a distinguir a qué categoría pertenece la imagen, teniendo en cuenta, que para el grupo de entrenamiento, el programa sabe si la imagen es un parqueadero lleno o vacío.

Después de esto, se realiza lo mismo para el conjunto de evaluación, y ya el programa determina la etiqueta de la imagen. A partir de esto, se miden 4 variables: pérdida, precisión, pérdida en el conjunto de validación. Estas variables dicen qué tan preciso es el modelo, qué tan rápido está aprendiendo el programa, y la suma de errores que está teniendo, es decir, la suma de imágenes que fueron etiquetadas incorrectamente.

Y finalmente, después de todos los proceso mencionados anteriormente, el modelo esta listo para predecir cualquier conjunto de datos que se quiera.

VI. Resultados y discusión

En primera instancia, durante la realización del modelo se decidió comparar que tal era su comportamiento y las diferentes métricas cuando se tenían distinto número de capas convolucionales. A continuación se presentan los resultados de entrenar y predecir los estacionamientos con los distintos números de capas ocultas.

A. Red neuronal con una capa de convolución:

Lo que se hizo en este red fue solamente construir una capa convolucional la cual tendría 32 filtros kernel y una capa para el proceso de agrupamiento. Al evaluar la precisión y las perdidas del modelo en el proceso de entrenamiento y validación, se encontraron los siguientes resultados:

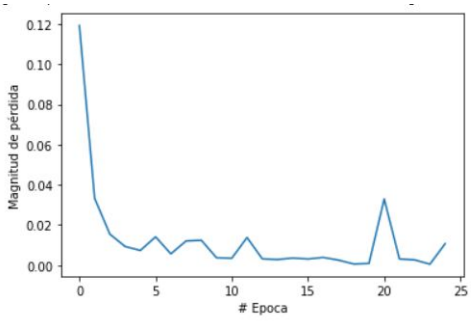

Fig. 4. Gráfico de pérdida con una capa de convolución.

De las anteriores gráficas, podemos ver entonces como fue el aprendizaje del modelo con 25 *epochs*, esto significa que el modelo iba a realizar los algoritmos de forwardpropagation y bacwardpropagation 25 veces. En un principio presenta gran perdida, sin embargo, después de llegar a perdida mínima posible, su comportamiento se mantiene relativamente constante,

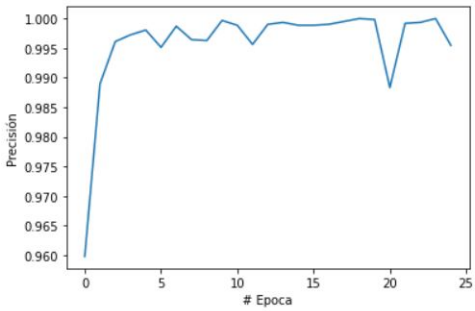

Fig. 5. Gráfico de precisión con una capa de convolución

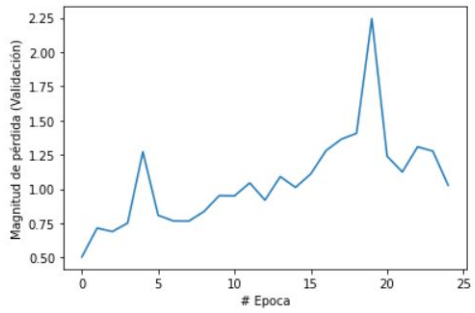

Fig. 6. Gráfico de precisión de la validación

esto da una buena señal de que el algoritmo esta teniendo un buen aprendizaje.

De la misma manera se puede observar que la precisión creció muy rápido y que entre menos se presentaban perdidas más el modelo mejoraba su precisión tanto en los datos de entrenamiento como en los datos de validación.

B. Red neuronal con dos capas de convolución:

En esta CNN se agrego una capa más para la convolución la cual tendría 64 filtros kernel, esto debido a que esta hará una segunda extracción de características más profunda y entre más filtros específicos se tengan, mejores abstracciones se obtendrán. EL comportamiento de esta red esta ilustrado por las siguientes imágenes.

Aquí, se encontró que las perdidas que se tenían en el entrenamiento también disminuían rápidamente, sin embargo se toma dos epochs más en comparación con la anterior para llegar a su perdida mínima y luego de ella se mantiene constante cerca de cero.

En segundo lugar, se evidencia que el comportamiento de la precisión en el entrenamiento y la validación se mantienen muy similares al proceso con una capa de convolución.

El comportamiento de la validación de los datos en el proceso de entrenamiento se presenta siempre ascendente, es decir, que a medida que va recorriendo una y otra vez la red, la precisión en estos datos aumenta considerablemente.

C. Red neuronal con cuatro capas:

Por ultimo, se aplicaron dos capas convolucionales con su respectiva capa de agrupamiento. Los resultados fueron los siguientes:

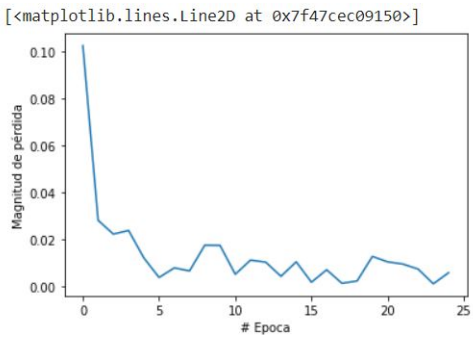

Fig. 7. Gráfico de pérdida con dos capas de convolución

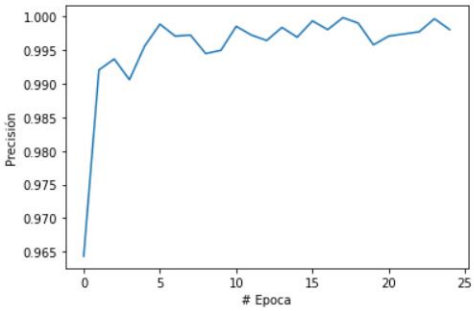

Fig. 8. Gráfico de precisión con dos capas de convolución

Para perdidas, se observa que no se mantienen contantes, tienen variaciones altas y muy abruptas, además que se demora un poco en el proceso de ir y volver a través de la red para dejar de perder considerablemente.

Por el lado de la precisión, es de notar que tampoco tienen un comportamiento deseable, ni similar a los resultados anteriores, se hace dudosa la precisión del modelo con estas características, es bastante variable.

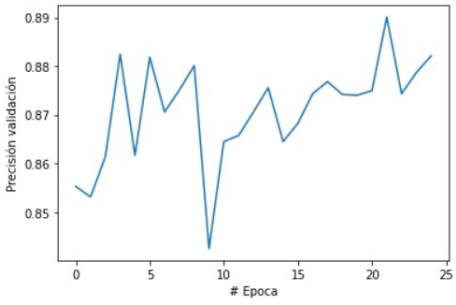

Fig. 11. Gráfico de precisión con cuatro capas

Y para la validación, el comportamiento permanece similar a las otras redes, va creciendo a medida que avanza en los ciclos de aprendizaje. Sin embargo, también presenta variaciones notorias, en un momento es muy precisa y al siguiente es lo menos precisa posible.

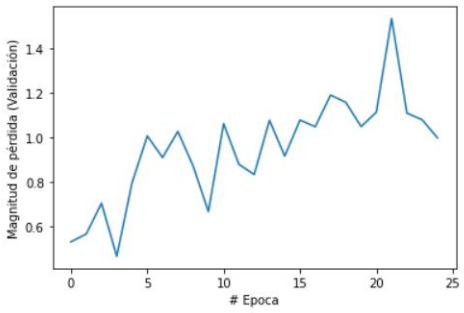

Fig. 9. Gráfico de precisión de la validación con dos capas

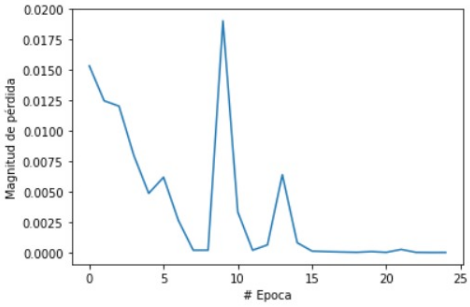

Fig. 10. Gráfico de pérdida con cuatro capas

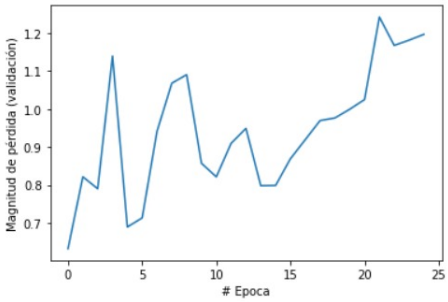

Fig. 12. Gráfico de precisión en la validación con 4 capas

Luego de estos análisis y de las diferentes pruebas, se evidenció que el tener dos capas convolucionales en el modelo ayudaría primero a minimizar el error de la precisión; segundo, tendría una gran precisión a la hora de entrenarlo y tercero, se tendrían más características significativas e importantes para la clasificación que con una sola capa de convolución.

Por otro lado, también es importante resaltar los errores tipo I y tipo II en la predicción del modelo. Es por esto que, después de entrenarse el modelo y tenerlo con los parámetros deseados (buena precisión y baja perdida), se realizó una predicción sobre 1000 imágenes de estacionamiento, 500 de ellas corresponden a parqueaderos vacíos y las restantes a parqueaderos llenos. Y se construyo la siguiente matriz de confusión:

		Valor predecido	
		Lleno	Vacío
Valor real	Lleno	336	164
	Vacío	11	489

De esta matriz podemos notar que el error tipo II, que es aquel en el que se predice que un estacionamiento esta lleno cuando realidad esta vacío, fue bastante bajo, para la cantidad de datos que se estaban usando. Así mismo, el error tipo I, aquel en el que se predice que esta vacío cuando en realidad esta lleno si es alto. Sin embargo, se evidenció que lo que influyo a este error fue que el conjunto de datos a predecir no estaba totalmente claro ya que contenía imágenes como las siguientes,

Fig. 13. Ejemplo del conjunto de datos a predecir

Fig. 14. Ejemplo del conjunto de datos a predecir

las cuales son confusas en su contenido, por lo que el modelo no logra distinguir completamente lo que se quiere. Pero a pesar de esto, se tiene como resultado final un modelo de Redes Neuronales Convolucionales que permite hacer una clasificación adecuada de imágenes de estacionamientos, con alta precisión y poca perdida.

VII. Conclusiones

- Se logro trabajar en el problema de optimización planteado, esto es, la red busca minimizar la diferencia entre la salida real y la salida de la red. Este se fue logrado al mejorar el proceso de entrenamiento. De esta manera, la red se forma para buscar una predicción más precisa haciéndolo mediante un algoritmo de propagación hacia atrás.

- A pesar de ser un modelo efectivo, al querer que su aprendizaje sea más acertado y subir el número de ciclos para hacer más propagación hacia atrás y hacía adelante, se tiene un alto coste computacional y de tiempo por parte de las redes CNN.

- En conclusión, es importante determinar el numero de capas de convolución que permite que el modelo obtenga las características necesarias y fundamentales para poder reconocer los patrones para la clasificación. Ya que, si se extraen demasiadas características, al final muchas de estas van a resultar innecesarias y van a afectar fuertemente en la clasificación, sobrecargando el modelo con información.

- Por otro lado, el modelo se vuelve muy versátil puesto que existen variables que se pueden alterar de tal forma que se oriente a un objetivo concreto, por ejemplo, si se modifica la variable *epochs*, se tiene que podría aumentarse o disminuirse el tiempo de entrenamiento del modelo, además si se modifica el tamaño común de las imágenes, se modifica el tiempo de ejecución, entre otros.

- También, el aprendizaje del modelo es muy bueno, porque teniendo los altibajos que se ven en las gráficas de pérdida, se tiene esta tiende a ser cero a medida que van aumentando los *epochs*, por tanto, al tener un número muy alto en esta variable, la pérdida tenderá al valor de cero.

Referencias

[1] IBM Cloud Education (2020, Octubre 20). Convolutional Neural Networks. [En linea]. Disponible en: https://www.ibm.com/cloud/learn/convolutional-neural-networks#toc-types-of-c-yL2bT7qZ

[2] Krizhevsky.A, Sutskever.I, Hinton.G (2017, Mayo 24). ImageNet Classification with Deep Convolutional Neural Networks. [En linea]. Disponible en: https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf

[3] Mandal.M (2021, Mayo 1). Introduction to Convolutional Neural Networks (CNN). [En linea]. Disponible en: https://www.analyticsvidhya.com/blog/2021/05/convolutional-neural-networks-cnn/

[4] J.Koushik (2016, Mayo 30). Understanding Convolutional Neural Networks [En línea]. Disponible en: https://arxiv.org/abs/1605.09081

[5] S. Mallat (2016 Abr 13). Understanding deep convolutional networks [En línea]. Disponible en: https://royalsocietypublishing.org/doi/full/10.1098/rsta.2015.0203

[6] CNR. [En linea]. Disponible en: http://cnrpark.it/

Figuras:

[1] Saha.S (2018,Diciembre 15). A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way. [En linea]. Disponible en: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-\&eli5-way-3bd2b1164a53

[2] Du.s (2020,Junio 8). Understanding Deep Self-attention Mechanism in Convolution Neural Networks. [En linea]. Disponible en: https://medium.com/ai-salon/understanding-deep-self-attention-mechanism-in-convolution-\neural-networks-e8f9c01cb251

[3] [13] [14] Las respectivas figuras fueron obtenidas del dataset usado para el trbajo