UNIVERSIDAD POLITÉCNICA SALESIANA
SEDE QUITO

CARRERA:
COMPUTACIÓN

Trabajo de titulación previo a la obtención del título de:
Ingeniero en Ciencias de la Computación

TEMA:
SISTEMA DE RECONOCIMIENTO SOBRE LA DISPONIBILIDAD DE ZONAS PARA PARQUEO MEDIANTE REDES NEURONALES CONVOLUCIONALES CON IMÁGENES EN TIEMPO REAL EN EL CAMPUS SUR DE LA UNIVERSIDAD POLITÉCNICA SALESIANA

AUTORES:
ARIAS SAMPEDRO DIEGO PAÚL
ÁLVAREZ CHAMORRO GUÍBED PAÚL

TUTOR:
ZAPATA MOLINA LINA PATRICIA

Quito, noviembre del 2021

CESIÓN DE DERECHOS DE AUTOR

Nosotros, Diego Paúl Arias Sampedro, con documento de identificación N° 1720179082 y Guíbed Paul Álvarez Chamorro, con documento de identificación N° 1724733249, manifestamos nuestra voluntad y cedemos a la Universidad Politécnica Salesiana la titularidad sobre los derechos patrimoniales en virtud de que somos los autores del trabajo de titulación intitulado: SISTEMA DE RECONOCIMIENTO SOBRE LA DISPONIBILIDAD DE ZONAS PARA PARQUEO MEDIANTE REDES NEURONALES CONVOLUCIONALES CON IMÁGENES EN TIEMPO REAL EN EL CAMPUS SUR DE LA UNIVERSIDAD POLITÉCNICA SALESIANA, mismo que ha sido desarrollado para optar por el título de INGENIERO EN CIENCIAS DE LA COMPUTACIÓN en la Universidad Politécnica Salesiana, quedando la Universidad facultada para ejercer plenamente los derechos cedidos anteriormente. En aplicación a lo determinado en la Ley de Propiedad Intelectual, en nuestra condición de autores nos reservamos los derechos morales de la obra antes citada. En concordancia, suscribimos este documento en el momento que hacemos entrega del trabajo final en formato digital a la Biblioteca de la Universidad Politécnica Salesiana.

.....
Diego Paúl Arias Sampedro
CI: 1720179082
Quito, noviembre de 2021

.....
Guíbed Paúl Álvarez Chamorro
CI: 1724733249
Quito, noviembre de 2021

DECLARATORIA DE COAUTORÍA DEL TUTOR

Yo declaro que bajo mi dirección y asesoría fue desarrollado el Artículo Académico, con el tema: SISTEMA DE RECONOCIMIENTO SOBRE LA DISPONIBILIDAD DE ZONAS PARA PARQUEO MEDIANTE REDES NEURONALES CONVOLUCIONALES CON IMÁGENES EN TIEMPO REAL EN EL CAMPUS SUR DE LA UNIVERSIDAD POLITÉCNICA SALESIANA., realizado por Diego Paúl Arias Sampedro y Guíbed Paúl Álvarez Chamorro, obteniendo un producto que cumple con todos los requisitos por la Universidad Politécnica Salesiana, para ser considerado como trabajo final de titulación.

Quito, noviembre de 2021

.....

Lina Patricia Zapata Molina
CI: 0501877278

Sistema de reconocimiento sobre la disponibilidad de zonas para parqueo mediante redes neuronales convolucionales con imágenes en tiempo real en el campus Sur de la Universidad Politécnica Salesiana

1st Guíbed Paúl Álvarez Chamorro
galvarezc3@est.ups.edu.ec

2rd Diego Paúl Arias Sampedro
dariass1@est.ups.edu.ec

3rd Lina Patricia Zapata Molina
lzapata@ups.edu.ec

***Resumen*—El presente trabajo tiene como objetivo desarrollar una aplicación informática de reconocimiento de imágenes, sobre la disponibilidad de espacios de estacionamiento, de diferentes zonas o áreas, en base a redes neuronales convolucionales. En la red neuronal se implementó la arquitectura mAlexnet debido a su precisión y tiempo de respuesta sobre el set de datos objetivo CNRPark+Ext. Las imágenes sobre las zonas de parqueo se extraen en tiempo real mediante el software que provee HIKVISION, fabricante de cámaras de monitoreo. Posteriormente las imágenes son segmentadas, pre procesadas, clasificadas y almacenadas en una base de datos a través de algoritmos codificados en PYTHON. Finalmente, se aplicó programación en paralelo, para la clasificación de imágenes de los espacios de estacionamiento mediante la librería MULTIPROCESSING de Python. Los resultados obtenidos de la clasificación de imágenes fueron del 97.37% de exactitud sobre el set de datos CNRPark+Ext y 95.91% sobre el set de datos local objetivo. Las pruebas de rendimiento en el procesamiento en paralelo permitieron concluir que este enfoque sólo toma ventaja sobre el enfoque secuencial cuando se posee una red neuronal con un largo tiempo de respuesta o gran cantidad de espacios de estacionamiento por procesar a la vez, esto debido al tiempo de inicialización necesario para cada ejecución en paralelo.**

***Palabras Clave*—detección de disponibilidad de parqueo, redes neuronales convolucionales, mAlexNet, CNRPark+EXT, computación en paralelo**

***Abstract*—The objective of this work was to develop an image recognition application about parking spot availability detection, based on convolutional neural networks. The neural network was implemented through mAlexnet architecture due its accuracy and response time the objective data set CNRPark-Ext. Parking zone images are extracted in real time through HIKVISION software, manufacturer of the surveillance cameras. Subsequently images are segmented, preprocessed, classified and stored in a database through algorithms codified in PYTHON. Finally, parallel computing was applied for parking spot classification through the PYTHON MULTIPROCESSING library. Output results from image classification showed a 97.37% accuracy with the dataset CNRPark+Ext and 95.91% with the target dataset. Performance tests with parallel computing allowed us to conclude that this approach only takes advantage from the sequential approach when the processing scenario possess a neural network with a long response time or a high amount of parking spots to process concurrently, this owing to the initialization time needed to begin a parallel execution**

***Keywords*—parking availavility detection, convolutional neural networks, mAlexNet, CNRPark+EXT, parallel computing**

I. Introducción

Debido al aumento exponencial de vehículos en el mundo la congestión es un problema recurrente para todas las ciudades [1], especialmente para aquellas que no fueron planificadas para manejar grandes volúmenes de flujo automotriz [2]. Gran parte de dicha congestión es generada por vehículos que buscan zonas de parqueo, zonas que por lo general no son administradas de forma adecuada, contribuyendo negativamente al aumento del tráfico vehicular, consumo de combustible y aumento en las emisiones de carbono [1]. Tomando en cuenta que una persona puede llegar a gastar en promedio 7.8 minutos en búsqueda de un espacio disponible [3] la necesidad de implementar sistemas de reconocimiento sobre la disponibilidad para zonas de parqueo se ha convertido en un mecanismo esencial para mejorar la movilidad urbana y la calidad de vida de las personas.

Con el desarrollo tecnológico han surgido diversas técnicas de reconocimiento de espacios disponibles para estacionamiento vehicular, espacios que son agrupados en zonas. Éstas técnicas trabajan con gran efectividad sobre áreas de alta concurrencia o de capacidad limitada, como lo son los sensores de monitoreo [4] o técnicas de estimación de disponibilidad [5], pero cuando determinadas áreas presentan una geografía de difícil acceso para la instalación de equipo es necesario recurrir a métodos poco invasivos que aprovechen la infraestructura disponible como el procesamiento de imágenes provistas por cámaras de vigilancia gestionadas mediante inteligencia artificial.

En el caso de la Campus Sur de la Universidad Politécnica Salesiana existen cinco zonas de parqueo, con un promedio de 50 espacios de estacionamiento cada zona, distribuidas en todo el campus, además estas zonas son de alta concurrencia vehicular, lo que limita al personal de seguridad en sus funciones de gestión sobre el uso de espacios de estacionamiento por parte de los estudiantes y del personal administrativo y docente

Ante esta situación es necesaria la implementación de una aplicación informática de reconocimiento de imágenes, sobre la disponibilidad de espacios de estacionamiento, para las

diferentes zonas o áreas, en base a redes neuronales convolucionales para el Campus Sur de la Universidad Politécnica Salesiana

El presente trabajo está organizado de la siguiente manera, en la sección A. se presentan diferentes trabajos relacionados con el área de la detección de disponibilidad de zonas para parqueo, introduciendo diferentes conceptos cuando se considera oportuno. La sección 2 presenta las herramientas y métodos del trabajo, detallando las técnicas que se implementaron. La sección 3 muestra los resultados relevantes que se obtuvieron y, finalmente, la sección 4 presenta las conclusiones obtenidas del presente trabajo

A. Trabajos Relacionados

Las redes neuronales se definen como una abstracción de la conciencia humana, computacionalmente se componen de un conjunto de nodos llamados "capas", al interconectarse, a cada relación se le asigna un valor numérico llamado "peso" [6]. El valor asignado a cada peso es ajustado mediante el entrenamiento de la red mediante muestras o ejemplos simulando el aprendizaje humano, este ajuste determinará la capacidad de la red para efectuar con éxito su objetivo.

La función de una red neuronal es identificar y responder hacia patrones similares a los que ha visto anteriormente pero que no son exactamente iguales [7], la disposición de nodos, capas, como el tipo de interconexiones tiene un gran impacto sobre el comportamiento final, a este conjunto de características se le llama "arquitectura".

Para el reconocimiento sobre la disponibilidad de zonas para parqueo existen múltiples arquitecturas propuestas que generan óptimos resultados. En [8] se presenta una alternativa para la detección de disponibilidad mediante la arquitectura VGGNet, la cual es una red que se encuentra pre entrenada en el reconocimiento de objetos en imágenes y cuenta con una arquitectura robusta de 19 capas que permitió alcanzar una exactitud elevada del 99% sobre el dataset objetivo, aunque la capacidad de clasificación de la red mostraba resultados excepcionales el tiempo de respuesta para 30 lugares de estacionamiento fue de 2 segundos, esto indica que el tiempo de procesamiento de múltiples zonas para parqueo requeriría hardware dedicado adicional con esta arquitectura.

En [9] se desarrolló una solución para la detección de disponibilidad de una estación de combustible mediante una versión reducida de la red VGGNet llamada VGG16, esta versión mantiene el pre entrenamiento en detección de objetos, pero reduce el número de capas de 19 a 16, en este sistema se alcanzó una exactitud del 94

En [10] se propone realizar una comparativa en el campo de estudio de la detección de disponibilidad con los resultados de la red Alexnet, esta red también pre entrenada se compone de una arquitectura de 9 capas para alcanzar un desempeño de alto nivel, pero se menciona que su estructura sigue siendo demasiado profunda para la clasificación de espacios de estacionamiento en dos grupos de elementos (disponible y no disponible)

En [11] se buscaba proponer una arquitectura efectiva para la detección de disponibilidad de zonas para parqueo, los resultados obtenidos bajo una metodología experimental sobre las arquitecturas propuestas y métodos convencionales de extracción de características entregaron resultados sobresalientes, destacando una versión que reduce el número de capas de la arquitectura Alexnet de 9 a 5 capas conocida como mAlexnet, la cual obtuvo con un 99.49% de exactitud sobre el set de datos objetivo.

En [12] se realizó experimentación con 12 arquitecturas distintas (tres de ellas variaciones de Alexnet) sobre la detección de disponibilidad de zonas para parqueo, aunque todas ellas alcanzaron grandes resultados de exactitud, solo mAlexnet logró clasificar un total de 30 imágenes con 18 espacios de estacionamiento en 25 segundos (mejor tiempo). Debido a la cantidad de estudios sobre esta arquitectura empleando el set de datos objetivo CNRPark+Ext y en base a los resultados de exactitud y tiempo de respuesta frente a otras arquitecturas, se ha seleccionado a mAlexnet como la arquitectura base para el desarrollo de nuestra aplicación informática.

En [13] se presenta un sistema para la detección de disponibilidad que trabaja con flujos o videos para la generación de imágenes estáticas que serán analizadas más adelante. Posteriormente se explica cómo se pre-procesan las imágenes al momento de entrar al clasificador, basándose en un estándar de escala de grises y segmentación por espacio de estacionamiento para mejorar el umbral de clasificación.

En [14] se presentó una propuesta para la detección automática de parqueaderos libres, misma que se apoya con sensores para la detección de los parqueaderos. Dentro de los resultados se pudo evidenciar que para todo el subsistema se tenía información y atributos que contemplaban posibles cambios en el futuro como la capacidad de administrar globalmente desde un solo lugar los parámetros de funcionalidad del sistema, recalcando la importancia de la escalabilidad en un sistema de detección de disponibilidad.

Python MULTIPROCESSING es una librería del lenguaje de programación PYTHON que permite implementar paralelismo de procesos de alto nivel mediante el paradigma de memoria compartida [15]. A pesar de que existen diversos enfoques para mejorar la eficiencia de procesos dentro de este lenguaje nuestra selección de esta librería se debe a su alto nivel de abstracción el cual nos permite dividir la carga de trabajo entre los distintos procesadores sin ser necesarios grandes cambios en la codificación.

La distribución de las tareas de MULTIPROCESSING se administra automáticamente según la disponibilidad de núcleos, esto quiere decir que si el número de ejecuciones asíncronas excede al número de núcleos existentes o desocupados en la máquina, las ejecuciones excedentes tendrán que esperar a que un núcleo se desocupe para ser procesadas, esto beneficia la implementación de la computación paralela sobre la aplicación ya que el número de núcleos que posea la maquina únicamente tendrá impacto en el rendimiento y no en su desarrollo [16].

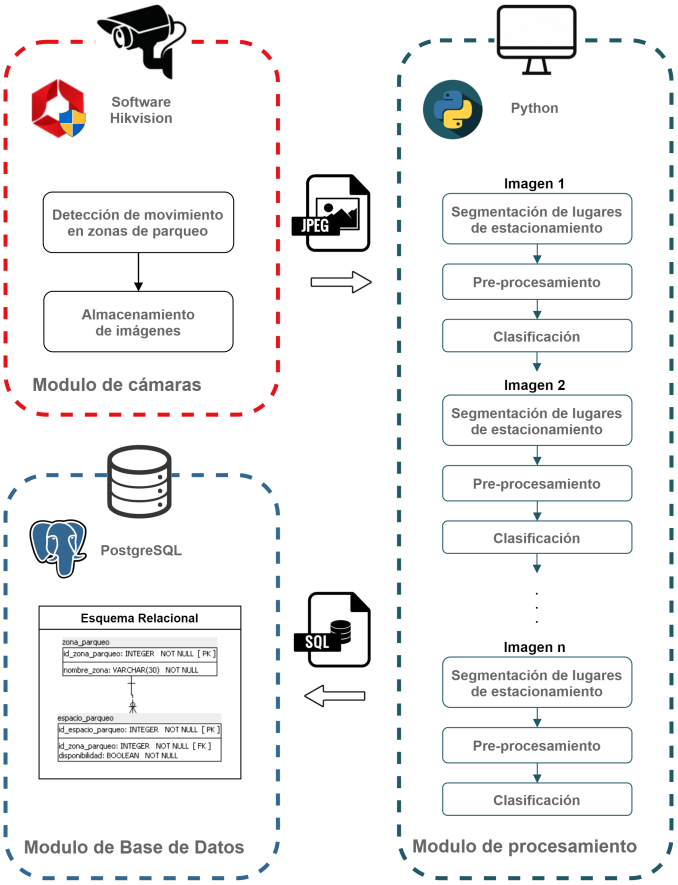

Fig. 1: Arquitectura Secuencial

II. Herramientas y Métodos

La aplicación informática está compuesta por tres secciones o módulos. El módulo de monitoreo de espacios de parqueo que a través de cámaras cumple con la función de capturar y almacenar imágenes. El módulo de procesamiento de imágenes encargado de determinar la disponibilidad de los espacios de parqueo agrupados en zonas y el módulo de base de datos donde se registra información referente al nombre de la zona, numero de espacio o parqueadero y estado del mismo (disponible, no disponible). Adicionalmente existe un archivo de configuración global xml que facilita el traslado del sistema a otros equipos y aumenta la escalabilidad del mismo, cuyo contenido a modo de ejemplo es el siguiente:

Los parametros contenidos en el archivo XML se describen a continuación:

1) carpeta_zonas: Almacena la ruta de la carpeta donde se almacenarán las imágenes de las zonas de parqueo que ingresarán al sistema.
2) carpeta_zonas: Almacena la ruta de la carpeta donde se almacenarán las imágenes segmentadas (recortes) de los espacios de estacionamiento.
3) tiempo_espera: determina el tiempo de espera entre cada busqueda de imagenes sobre la carpeta de imagenes.

```
<?xml version="1.0"?>
<data>
<carpeta_imagenes>
C:Users/Documents/carpeta_zonas/
</carpeta_imagenes>

<carpeta_imagenes_recortes>
C:Users/Documents/imagenes_lugares/
</carpeta_imagenes_recortes>

<archivo_log>
C:/Users/Paul/Desktop/
</log_errores>

<ruta_archivo_pesos>
D:/Downloads/pesos.pth
</ruta_archivo_pesos>

<tiempo_espera>10</tiempo_espera>
```

Fig. 2: Contenido del archivo de configuración global XML

4) archivo_log: Almacena la ruta donde se generará un archivo .txt que almacenará en forma de lista los errores encontrados al ejecutar la aplicación.
5) zona_parqueo: Almacena el nombre de una zona de parqueo
6) espacio: Almacena las coordenadas (en pixeles) pertenecientes a un espacio de estacionamiento dentro de la imagen de su zona correspondiente.
7) procesamiento_paralelo: determina si las imágenes se procesaran mediante la arquitectura secuencial o paralela.

Los componentes de hardware y software utilizados para el desarrollo y uso de la aplicación informática propuesta se describen a continuación:

1) Cámara IP HIKVISION DS-2CD2423G0-IW
2) Software gestor del sistema de cámaras HIKVISION IVMS4200
3) Lenguaje de programación Python v3.8
4) Gestor de base de datos PostgreSQL

Con la definición de los módulos y de los componentes de hardware y software a utilizarse para el desarrollo de la aplicación informática hemos definido dos arquitecturas de software para el diseño de la estructura de la aplicación y se describen a continuación:

A. Arquitectura Secuencial

La arquitectura secuencial hace referencia a la ejecución de las instrucciones de los programas una tras otra, es decir que una instrucción no se ejecuta hasta que finaliza la anterior. Dentro de la aplicación informática la arquitectura secuencial procesará una imagen correspondiente a una zona de parqueo a la vez, determinará si sus espacios individuales se encuentran disponibles o no disponibles y continuará con la siguiente imagen hasta que no existan más imágenes de zonas de parqueo por procesar.

En la Figura 1 tenemos los componentes de la arquitectura secuencial, donde se puede observar el módulo de de

cámaras, procesamiento y base de datos que se describen a continuación:

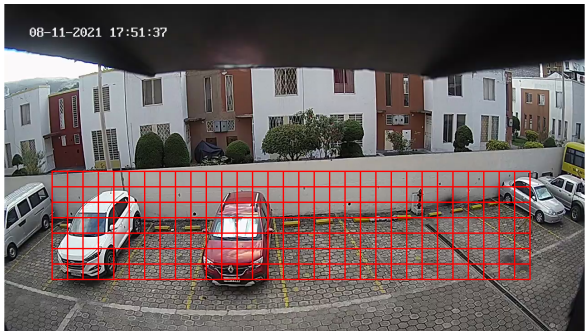

Fig. 3: Area de monitoreo predefinida

1) Modulo de Cámaras: Es responsable de almacenar imágenes de las zonas para parqueo, las cuales servirán como datos de entrada para el sistema, estas imágenes son almacenadas en una carpeta predefinida junto con el nombre de la zona capturada para poder distinguir entre diferentes zonas en caso de que el sistema cuente con múltiples cámaras. Para asegurar que el sistema no malgaste recursos, el almacenamiento de imágenes se realiza únicamente cuando se detecta movimiento en el área de monitoreo, la misma que es definida mediante la opcion "Motion capture" embebida dentro de la cámara IP y configurable mediante el software HIKVISION IVMS 4200 como se puede ver en la imagen de la zona de parqueo local en la Figura 3, donde tenemos una zona de parqueo local marcada con la malla roja.

2) Módulo de Procesamiento: El módulo de Procesamiento de imágenes consta de tres actividades, como se puede ver en la Figura 1, las mismas que se describe a continuación:

a) Segmentación de los espacios de estacionamiento: Cuando la imagen de una zona de parqueo ingresa al sistema se realiza una segmentación o recorte de la misma para obtener una imagen individual de cada uno de los espacios de estacionamiento que conforman dicha "zona".

La segmentación de una imagen de entrada se realiza según el nombre de la "zona" y el número de "espacios" parametrizados en el archivo XML como se observa en la Figura 4.

```
<zona_parqueo name="Local">
     <espacio>[436, 170, 140, 170, 1]</espacio>
     <espacio>[433, 196, 264, 196, 2]</espacio>
     <espacio>[416, 222, 410, 222, 3]</espacio>
     <espacio>[435, 198, 595, 198, 4]</espacio>
     <espacio>[393, 212, 763, 212, 5]</espacio>
     <espacio>[395, 174, 941, 174, 6]</espacio>
</zona_parqueo>
```

Fig. 4: Zona y espacios de estacionamiento predefinidos en el archivo XML

Donde tenemos una zona configurada con 6 espacios de estacionamiento, correspondiente a las regiones de la Figura 6, todos etiquetados con "Zona_parqueo" en conjunto con el parámetro "name", las etiquetas "espacio" indican la posición en pixeles del espacio de estacionamiento en la imagen original de la zona, como se describe en la Figura 5.

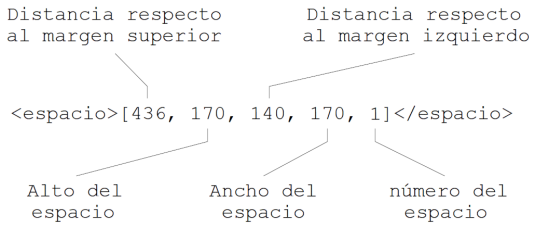

Fig. 5: Parametros de la región del espacio de estacionamiento número 1 del archivo XML

Donde podemos observar los parámetros de posicionamiento en pixeles que generan la region del espacio de estacionamiento "número 1" ubicado a la izquierda de la Figura 6 con la etiqueta "1".

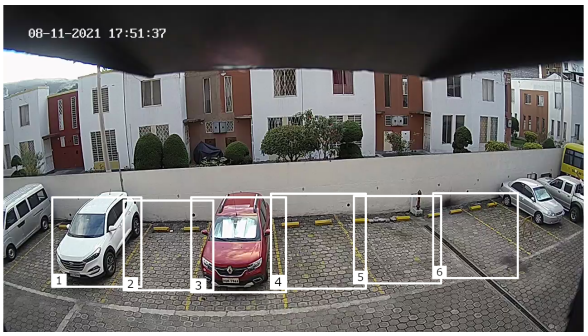

Fig. 6: Imagen de cámara IP con regiones en cada espacio para parqueo

El módulo de procesamiento almacena en la ruta "carpeta_espacios" los recortes de cada una de las regiones en las imagenes de zonas de parqueo utilizando PYTHON para ser preprocesadas.

b) Pre procesamiento de imagenes de espacios de estacionamiento: Antes de ingresar a la red neuronal las imágenes de los espacios de estacionamiento deben ser acondicionados para aumentar la exactitud en la clasificación, el flujo de pre procesamiento se compone de redimensión, recorte centrado, transformación a tensor y normalización, dichas funciones se describen a continuación:

1) Transform.Rezise.- Esta función permite redimensionar la imagen original al tamaño adecuado para la red neuronal de 256 pixeles.
2) Transform.CenterCrop.- En caso de que el recorte obtenido siga siendo demasiado grande, esta función permite reemplazarlo por un recorte de menor tamaño que se ajuste a los 256 pixeles necesarios como entrada para la red neuronal.
3) Transform.ToTensor.- Un tensor es un tipo de dato que almacena valores numéricos en una arreglo o matriz de n

dimensiones, esta función permite almacenar los valores correspondientes a los pixeles RGB de nuestro recorte en tres vectores independientes correspondientes a cada color para poder ingresar a la red neuronal.

4) Transform.Normalize.- Permite ajustar los valores del tensor mediante el cálculo de la media y la desviación estándar, dentro del sistema todos los tensores son normalizados en base al valor 0.5 en todas sus dimensiones.

Al finalizar la fase de pre procesamiento los recortes de los espacios de estacionamiento de 256 pixeles de alto y 256 pixeles de ancho se habrán transformado al tipo de dato tensor para su ingreso a la red neuronal

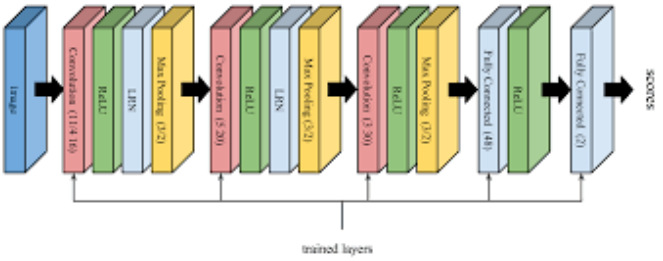

Fig. 7: Arquitectura de la red neuronal convolucional mAlexNet [11]

c) Clasificación de espacios de estacionamiento: Posterior al pre procesamiento, los espacios de estacionamiento en forma de tipo de dato tensor ingresan a la red neuronal. Como se mencionó anteriormente se seleccionó la arquitectura mAlexnet como base para la red neuronal convolucional para asegurar el mejor tiempo de respuesta en el proceso de clasificación de espacios de estacionamiento, en la Figura 7 podemos observar la distribución de las 5 capas a que la componen, las primeras tres comparten operaciones de convolución, ReLU y Max Pooling, la capa cuatro y cinco son capas totalmente conectadas, pero se diferencian mediante la rectificación lineal ReLU presente en la capa cuatro.

A continuación, se describe la funcionalidad de las capas que componen la arquitectura:

1) Capa de Convolusión
   Esta capa funciona como un filtro para las imágenes al encontrar "características" o "patrones" como líneas o curvas, mientras más capas convolutivas se combinen más complejas serán las formas que podrá "reconocer" la red neuronal, esto quiere decir que obtendrá información relevante de las imágenes de espacios de estacionamiento como la forma que tienen cuando se encuentran disponibles o no disponibles [17].
2) Capa ReLu
   Esta capa también es conocida como la capa de rectificación linear unitaria, la cual tiene como objetivo permitir el paso de las características e ignorar el resto de valores que no sean considerados relevantes, cuando se aplica una capa convolutiva sobre una imagen (en forma de tensor) las características importantes se transmiten como valores numéricos positivos hacia la siguiente capa, pero debido a la naturaleza de la convolución sobre una imagen también obtendremos valores negativos que no aportan al proceso de reconocimiento de características por lo que la capa ReLu se encarga de convertir esos valores negativos a cero permitiendo que la red "aprenda" de forma adecuada [18] ademas de aportar un soporte matemático a la estabilidad de las redes neuronales [17].
3) Capa MaxPooling
   La capa de pulido tiene la función de reducir el tamaño de la imagen de entrada mediante la extracción sistemática del valor máximo sobre conjuntos de pixeles (valores numéricos del tensor). La particularidad de esta capa es su capacidad para mantener las características o patrones durante la reducción [18].
4) Capa Fully-Connected
   Esta capa tiene la función de tomar el resultado de las capas anteriores y traducirlo en categorías permitiendo generar una predicción o salida numérica [19], nuestra red neuronal posee dos categorías de salida que corresponden al posible estado del espacio de estacionamiento y son "disponible" o "no disponible".

Cuando un recorte de un espacio de estacionamiento (como tensor) ingresa a la red neuronal que acabamos describir, la predicción o salida que genera será un valor numérico en el rango de 0 a 1, entonces el algoritmo de PYTHON marcará al recorte como no "disponible" si dicho valor numérico es mayor a 0.9 o "no disponible" en caso de que sea menor.

3) Modulo Base de Datos: El módulo de base de datos tiene la función de actualizar el estado de los espacios de estacionamiento en base a la predicción resultante de la red neuronal, dicha actualización se realiza mediante una consulta UPDATE utilizando PYTHON hacia una base de datos relacional generada mediante POSTGRES, el esquema de la base se puede observar en la Figura 9 en donde encontramos dos tablas llamadas "tb_zona" y "tb_parqueadero".

La tabla "tb_zona" almacena información de las zonas para parqueo, en donde una zona puede tener múltiples espacios de estacionamiento a la vez mediante una relación uno a varios, el campo "zona" almacena el nombre del área monitoreada por la cámara y es el campo al que apuntan los algoritmos de PYTHON para la actualización de los espacios de estacionamiento en cada área correspondiente, la utilidad del resto de campos se centra en agregar más detalles al sistema y facilitar el análisis y futura administración del área.

La tabla "tb_parqueadero" almacena información del estado de un espacio de estacionaimento mediante el campo "disponible" en donde se almacena 0 en caso de que el espacio no se encuentre disponible o 1 en caso de que se encuentre disponible, el campo "id9_zona" es el identificador que conecta al espacio de estacionamiento con la zona o área a la que pertenece. Al igual que con la tabla "tb_zona" la utilidad del campo "estado" de la tabla "tb_parqueadero" se centra en agregar funcionalidad administrativa al espacio de estacionamiento.

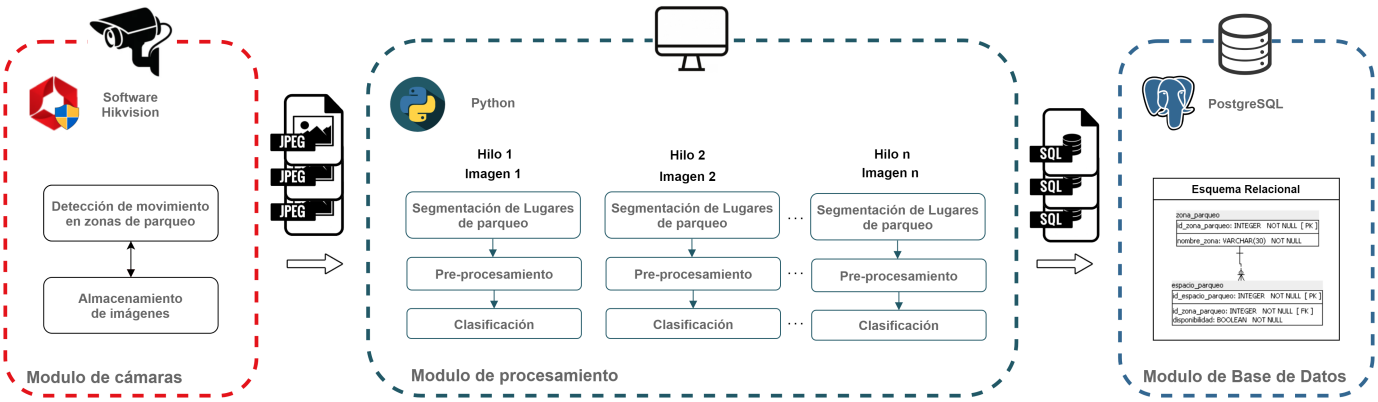

Fig. 8: Arquitectura en Paralelo

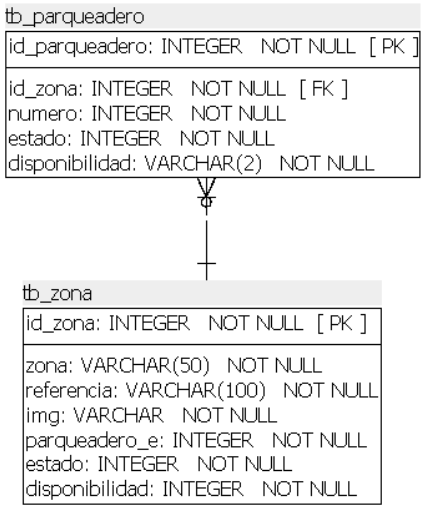

Fig. 9: Esquema de la base de datos del proyecto

La Figura 10 muestra el algoritmo general de la clase principal del módulo de procesamiento, dicha clase tiene la función de monitorear la carpeta "carpeta_zonas" en busca de imágenes almacenadas por el módulo de cámaras, en caso de encontrarlas, se realizará un llamado al método "process_zone", la cual será el encargado de determinar la disponibilidad o no disponibilidad de todos los espacios de estacionamiento que contenga cada una de las imágenes en la carpeta.

B. Arquitectura en Paralelo

La arquitectura en paralelo posee los mismos módulos considerados en la arquitectura secuencial, la diferencia radica en la forma de procesamiento paralelo o simultáneo de las imágenes referentes a los espacios de estacionamiento, como se observa en la Figura 8. El objetivo es que todas las imágenes a ser procesadas sean distribuidas entre todos los procesadores del equipo de computo, en lugar de que un solo procesador se haga responsable de dichas imágenes como es el caso de la arquitectura secuencial. Esto permite que todas las imágenes

Clase principal que gestiona el procesamiento de zonas de forma secuencial o paralela en base a la configuración del archivo global XML

```
1.  procesar = Verdadero
2.  Mientras procesar = Verdadero hacer
3.  lista_rutas_imagenes = buscar_imagenes(ruta_carpeta_imagenes)
4.  Si longitud(lista_rutas_imagenes) ≥ 1 hacer
5.      Si procesamiento_paralelo = Falso hacer
6.          Para recorrer rutas de imagenes
7.              Process_zone(ruta_imagen)
8.          FinPara
9.      SiNo
10.         with concurrent.futures.ProcessPoolExecutor()
            as executor: future = executor.map(process_zone, lista_rutas_imagenes)
11.     FinSi
12. FinSi
13. time.sleep(tiempo_espera)
14. Repetir
15. Fin
```

Fig. 10: Pseudocódigo clase principal

disponibles sean segmentadas, pre procesadas, clasificadas y actualizadas a la vez mediante el multiprocesamiento programado en PYTHON.

1) Multiprocesamiento de zonas para parqueo: El multiprocesamiento en Python se implementó a través de la función "concurrent.futures.ProcessPoolExecutor", la cual permite realizar llamadas asíncronas empleando todos los núcleos desocupados de la máquina, y la función "execute.map" que maneja dos parámetros, el primero es la función que será ejecutada de forma asíncrona (process_zone) y el segundo una lista iterable donde cada elemento de la lista será enviado como parámetro a una ejecución asíncrona (lista_rutas_imagenes), esto quiere decir que al enviar una lista con las rutas de las imágenes a cada ruta le corresponde una ejecución asíncrona

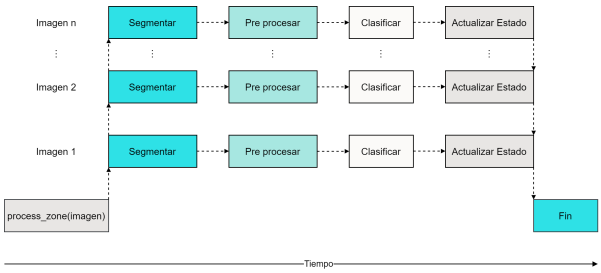

Fig. 11: Esquema de multiprocesamiento

con la función "process_zone" como se puede ver en la Figura 3

III. Pruebas y análisis de resultados

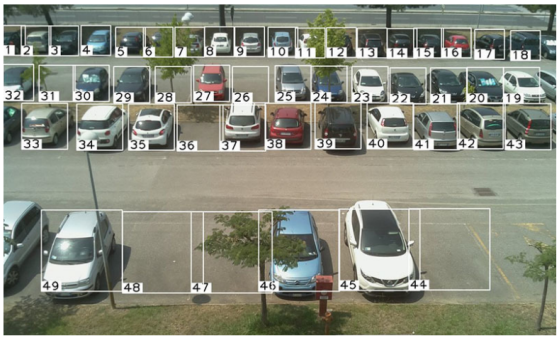

Fig. 12: Imagen tomada del dataset CNRPark+EXT

A. Prueba 1 Set de pruebas CNRPark-Ext

La prueba consiste en clasificar imágenes de espacios de estacionamiento del set de datos de prueba CNRPark+Ext con la arquitectura de red neuronal propuesta mAlexNet y comparar sus predicciones o salidas con la clasificación realizada manualmente mediante mediante el porcentaje de exactitud calculado con la fórmula:

$$Exactitud = \frac{(\text{verdaderos positivos} + \text{verdaderos negativos})}{(\text{positivos} + \text{negativos})}$$

El objetivo de esta prueba es es concluir si la red neuronal es óptima sobre sobre el set de datos seleccionado, los componentes utilizados se explican a continuación:

1) Hardware: como componente principal para ejecutar las pruebas se usó una maquina Dell Inspiron 5559 con procesador Intel i7 de octava generación con 4 núcleos.
2) Set de datos de prueba: CNRPark+EXT es un set que cuenta con más de 144925 imágenes de espacios de estacionamiento con varias perspectivas y tres escenarios climáticos como se puede observar en la Figura 12. El motivo por el cual se seleccionó este set de datos para el presente trabajo es la similitud de sus imágenes con el ambiente objetivo.

Para la realización de las pruebas se seleccionó aleatoriamente el 30% de las imágenes del set de datos (10397 espacios de estacionamiento) y se procesó para cada una de ellas las fases de pre procesamiento y clasificación con la aplicación desarrollada, los resultados obtenidos de la red neuronal se encuentran en la tabla III, donde podemos ver que la clasificación es optima al alcanzár una tasa de exactitud de 97%.

Set de datos CNRPark+EXT			
Tipo	Espacios	No. de Ocupados	No. de vacios
Entrenamiento	94493	63622	30871
Prueba	31825	21428	10397
Subtotal	126318	85050	41268

TABLE I: Segmentación del dataset CNRPark+EXT

Set de datos Local				
Clima	Muestras	No. de Ocupados	No. de vacios	Total
Soleado	103	380 (61.49%)	238 (38.51%)	618
Nublado	35	101 (48.09%)	109 (51.91%)	210
Subtotal	138	481 (58.09%)	347 (41.91%)	828

TABLE II: Segmentación del dataset Local

B. Prueba 2 Set de pruebas Local

LA prueba a realizarse es muy similar a la prueba 1, la diferencia radica en clasificar imágenes de espacios de estacionamiento del set de datos de prueba local

los componentes necesarios para la prueba se listan a continuación:

1) Hardware: como componente principal para ejecutar las pruebas se usó una maquina Dell Inspiron 5559 con procesador Intel i7 de octava generación con 4 núcleos.
2) Set de datos de prueba: el set de datos local se compone de 828 imágenes de espacios de estacionamiento obtenidos durante 7 días de monitoreo mediante una cámara de vigilancia ubicada en una zona de parqueo particular, el número de espacios disponibles y no disponibles de este dataset se puede observar en la figura II, mientras que la zona de parqueo local puede ser vista en la Figura 6.

Para la realización de las pruebas se procesó a las 828 imágenes con las fases de pre procesamiento y clasificación con la aplicación desarrollada, los resultados obtenidos de la red neuronal se encuentran en la tabla III, donde podemos ver que la clasificación alcanzó una tasa de exactitud de 95%.

Adicionalmente, se realizó una gráfica de la curva ROC generada a partir de los experimentos sobre el set de datos local, dicha curva muestra la tasa de falsos positivos en el eje y en conjunto con la tasa de falsos positivos en el eje x, esta gráfica nos permite obtener una visualización de la calidad de la clasificación realizada donde una línea cercana a los extremos indica una clasificación perfecta mientras que una línea diagonal indica una clasificación totalmente aleatoria [20]. A partir de los datos desplegados en el grafico el puntaje del área bajo la curva obtenido fue de 0.99% indicando que el modelo es óptimo sobre los datos evaluados.

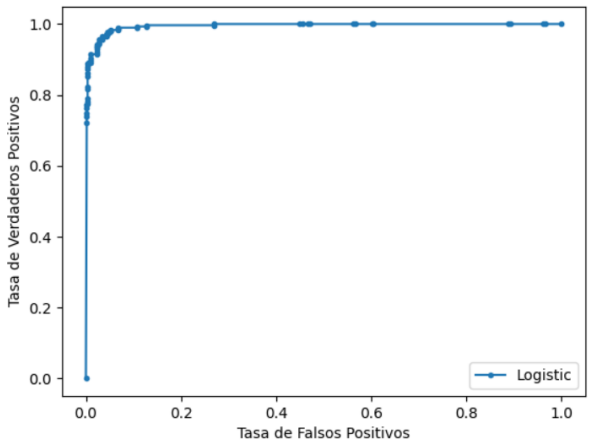

Fig. 13: Curva ROC para el dataset local

Evluación de las muestras		
Medida	Valor set Local	Valor set CNRPark
Sensitividad	0.9898	0.9901
Especificidad	0.9425	0.9377
Presición	0.9034	0.9721
Exactitud	0.9591	0.9737
Valor predictivo negativo	0.9941	0.9773
Tasa falsos positivos	0.0575	0.0623
Tasa falsos descubrimientos	0.0966	0.0279
Tasa falsos negativos	0.0102	0.0099
F1 Score	0.9446	0.9810
AUC Score	0.9943	0.9986

TABLE III: Resultados del set de Prueba y Local

C. Prueba 3: Pruebas de rendimiento

La prueba consiste en procesar varios grupos de imágenes de zonas de parqueo y comparar el tiempo promedio de ejecución tanto de la arquitectura secuencial como de la arquitectura en paralelo. El objetivo es determinar cuál de las arquitecturas muestra mejor rendimiento sobre cada grupo de imágenes, es decir determinar qué arquitectura logra completar las fases de segmentación, pre procesamiento, clasificación y actualización a la base de datos en el menor tiempo.

Los componentes de la prueba se describen a continuación:

1) Hardware: como componente principal para ejecutar las pruebas se usó una maquina Dell Inspiron 5559 con procesador Intel i7 de octava generación con 4 núcleos.
2) Set de datos de prueba: se utilizarán imagenes segmentadas de lugares de estacionamiento tanto del set de datos local como del set de datos CNRPark+Ext seleccionadas aleatoriamente.

Para la realización de las pruebas se procesaron grupos de 10, 25, 50, 200 y 400 imágenes, para determinar el resultado final de cada grupo se calculó el tiempo promedio de 10 ejecuciones por cada grupo de imágenes y tipo de arquitectura.

La tabla IV registra los tiempos de ejecución promedio en base al grupo de imágenes procesado, como se puede observar que con el grupo de 10 imágenes la arquitectura secuencial

Tiempo de Ejecución		
No. de Muestras	Procesamiento Secuencial	Procesamiento Parelelo
10	1.3684 seg.	6.6247 seg.
25	3.1162 seg.	7.3505 seg.
50	14.7078 seg.	11.1911 seg.
200	43.2905 seg.	27.6448 seg.
400	80.4736 seg.	46.2092 seg.

TABLE IV: Pruebas de tiempo de ejecución

posee una ventaja de 5 segundos con respecto a la arquitectura en paralelo, mientras que con el grupo de 400 imágenes la arquitectura en paralelo tiene una ventaja de 34 segundos concluyendo que la arquitectura en paralelo únicamente es más eficiente que la arquitectura secuencial a medida que aumenta el tamaño del grupo de imágenes.

IV. Conclusiones

Se logró extraer imágenes mediante una cámara de seguridad y procesarlas mediante computación en paralelo, aunque la arquitectura en paralelo sobre grupos pequeños de imágenes sigue siendo aceptable para una aplicación en tiempo real, debido al tiempo de inicialización del paralelismo este solo demuestra tener ventaja sobre la arquitectura secuencial únicamente cuando se procesa un grupo elevado de imágenes a la vez, esto nos permite concluir que la arquitectura secuencial es una mejor opción para ambientes con pocas zonas de parqueo y siendo la arquitectura paralela la mejor opción para el procesamiento de imágenes en grandes cantidades.

Se logró implementar un clasificador de espacios de estacionamiento que determina si dichos espacios de parqueo están disponibles o no disponibles en base a redes neuronales convolucionales y a la arquitectura mAlexNet.

Se logró desarrollar con éxito una aplicación informática para la Universidad Politécnica Salesiana capaz de actualizar en tiempo real el estado de la disponibilidad de los espacios de estacionamiento de múltiples zonas de parqueo en una base de datos relacional para su análisis futuro.

References

[1] T. G. Crainic, N. Ricciardi, and G. Storchi, "Advanced freight transportation systems for congested urban areas," *Transportation Research Part C: Emerging Technologies*, vol. 12, no. 2, pp. 119–137, 2004. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0968090X04000142

[2] B. Giles-Corti, A. Vernez-Moudon, R. Reis, G. Turrell, A. L. Dannenberg, H. Badland, S. Foster, M. Lowe, J. F. Sallis, M. Stevenson, and N. Owen, "City planning and population health: a global challenge," *The Lancet*, vol. 388, no. 10062, pp. 2912–2924, 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0140673616300666

[3] R. Arnott and E. Inci, "An integrated model of downtown parking and traffic congestion," *Journal of Urban Economics*, vol. 60, no. 3, pp. 418–442, 2006. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0094119006000386

[4] A. Zajam and S. Dholay, "Detecting efficient parking space using smart parking," in *2018 9th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, 2018, pp. 1–7.

[5] B. Xu, O. Wolfson, J. Yang, L. Stenneth, P. S. Yu, and P. C. Nelson, "Real-time street parking availability estimation," in *2013 IEEE 14th International Conference on Mobile Data Management*, vol. 1, 2013, pp. 16–25.

[6] S.-C. Wang, *Interdisciplinary computing in Java programming*. Springer Science & Business Media, 2012, vol. 743.

[7] P. Benardos and G.-C. Vosniakos, "Optimizing feedforward artificial neural network architecture," *Engineering Applications of Artificial Intelligence*, vol. 20, no. 3, pp. 365–382, 2007. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0952197606001072

[8] D. Acharya, W. Yan, and K. Khoshelham, "Real-time image-based parking occupancy detection using deep learning." in *Research@ Locate*, 2018, pp. 33–40.

[9] X. Xiang, N. Lv, M. Zhai, and A. El Saddik, "Real-time parking occupancy detection for gas stations based on haar-adaboosting and cnn," *IEEE Sensors Journal*, vol. 17, no. 19, pp. 6360–6367, 2017.

[10] S. Nurullayev and S.-W. Lee, "Generalized parking occupancy analysis based on dilated convolutional neural network," *Sensors*, vol. 19, no. 2, 2019. [Online]. Available: https://www.mdpi.com/1424-8220/19/2/277

[11] G. Amato, F. Carrara, F. Falchi, C. Gennaro, C. Meghini, and C. Vairo, "Deep learning for decentralized parking lot occupancy detection," *Expert Systems with Applications*, vol. 72, pp. 327–334, 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S095741741630598X

[12] A. Farley, H. Ham, and Hendra, "Real time ip camera parking occupancy detection using deep learning," *Procedia Computer Science*, vol. 179, pp. 606–614, 2021, 5th International Conference on Computer Science and Computational Intelligence 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050921000533

[13] N. Bibi, M. N. Majid, H. Dawood, and P. Guo, "Automatic parking space detection system," in *2017 2nd International Conference on Multimedia and Image Processing (ICMIP)*. IEEE, 2017, pp. 11–15.

[14] D. Ordóñez-Camacho, E. Gómez, and H. Ávalos, "An architectural proposal for an automatic vacant parking detection system," in *2018 International Conference on Information Systems and Computer Science (INCISCOS)*. IEEE, 2018, pp. 351–355.

[15] P. Hadjidoukas, A. Bartezzaghi, F. Scheidegger, R. Istrate, C. Bekas, and A. Malossi, "torcpy: Supporting task parallelism in python," *SoftwareX*, vol. 12, p. 100517, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2352711020300091

[16] J. Palach, *Parallel programming with Python*. Packt Publishing Ltd, 2014.

[17] N. Sharma, V. Jain, and A. Mishra, "An analysis of convolutional neural networks for image classification," *Procedia Computer Science*, vol. 132, pp. 377–384, 2018, international Conference on Computational Intelligence and Data Science. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050918309335

[18] T. Guo, J. Dong, H. Li, and Y. Gao, "Simple convolutional neural network on image classification," in *2017 IEEE 2nd International Conference on Big Data Analysis (ICBDA)*, 2017, pp. 721–724.

[19] F. Sultana, A. Sufian, and P. Dutta, "Advancements in image classification using convolutional neural network," in *2018 Fourth International Conference on Research in Computational Intelligence and Communication Networks (ICRCICN)*, 2018, pp. 122–129.

[20] G. Amato, F. Carrara, F. Falchi, C. Gennaro, and C. Vairo, "Car parking occupancy detection using smart camera networks and deep learning," in *2016 IEEE Symposium on Computers and Communication (ISCC)*, 2016, pp. 1212–1217.