COLOQUIO NACIONAL DE ESTADÍSTICA
ESCUELA DE ESTADÍSTICA
Tendencias del pasado, presente y futuro de la estadística

UNIVERSIDAD EAFIT

Sistemas Automáticos de Conteo de Vehículos Basados en Video y Otras Soluciones de Smart Parking

Simón Fallon, Isabel Urrego & Santiago Ortiz
Departamento de Ciencias Matemáticas, Universidad EAFIT, Medellín, Colombia
sfallong@eafit.edu.co - iurregog@eafit.edu.co - sortiza2@eafit.edu.co

1. Resumen

En este estudio se emplean una serie de técnicas matemáticas y computacionales para elaborar sistemas automáticos de conteo de vehículos basados en video, en el contexto de un sistema de smart parking para un campus universitario. Nuestra propuesta se basa en tres elementos: **I)** un sistema de detección de vacancia para el monitoreo de celdas de parqueo, desarrollado a partir de segmentación de imágenes y clasificación binaria con técnicas de machine y deep learning, **II)** un detector de vehículos para filas de entrada, usando el algoritmo de detección y clasificación de objetos YOLOv5 , **III)** la implementación de técnicas de data augmentation para crear datasets de imágenes más grandes y robustos y mejorar el desempeño. Finalmente, **IV)** se presenta un algoritmo detector de placas usando EasyOCR para automatizar el proceso de entrada y pago.

2. Metodología

2.1 Detección Automática de la Ocupación

El primer objetivo de esta investigación es evaluar el desempeño de diferentes modelos en la clasificación de imágenes de celdas de parqueo en las clases “vacío” y ”ocupado”. Para esto, se extrajeron las características que se describen a continuación:

- **Pixeles**: Computacionalmente, una imagen es una matriz de pixeles caracterizados por tres canales (RGB). Sea P_i a la i-ésima imagen con dimensiones $M \times N \times 3$, donde M es el número de pixeles a lo alto, N a lo ancho y 3 describe la cantidad de canales por pixel.

- **HoG** (Histograma de gradientes orientados) [1]: Para obtener los gradientes se aplica un filtro a las imágenes con un kernel horizontal y uno vertical (Figura 1) y finalmente se calcula la magnitud $\| \nabla f \|$ y dirección de los gradientes θ como se muestra en la ecuación (1).

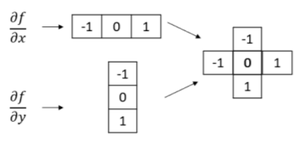

Figura 1: Kernels para un HoG.

$$||\nabla f|| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2} \qquad (1)$$

$$\theta = \tan^{-1}\left(\frac{\partial f}{\partial x} \Big/ \frac{\partial f}{\partial y}\right)$$

- **VGG-16** [2]: Las redes neuronales convolucionales se componen de dos partes. Primero, un extractor de características, seguido de un clasificador. En este trabajo se ha utilizado únicamente el extractor de características de la red neuronal VGG-16.

Para medir la efectividad de las características extraídas, se contrastó el desempeño de los siguientes modelos: Regresión Logística, SVM con kernels lineal y polinomial, Random Forest y Gradient Boosting

2.2 Monitoreo de Filas

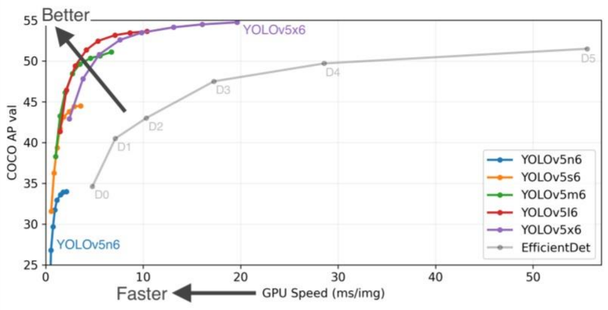

Figura 2: Performance: YOLOv5 vs. EfficientDet (MS COCO).
Tomado de: https://github.com/ultralytics/yolov5

Se entrena un algoritmo de detección de objetos conocido como YOLOv5, desarrollado por Ultralytics [3]. Este algoritmo end-to-end de detección y clasificación de objetos, fue previamente entrenado con el dataset Microsoft COCO. En la Figura 1 se muestra una comparación de las diferentes versiones del YOLOv5 con la red neuronal de Google EfficientDet. Se puede observar el gran desempeño en términos de la métrica de precisión media (AP val) en relación a la velocidad de cómputo, motivo por el cual este algoritmo es considerado el estado del arte para este tipo de tareas.

2.3 Detección Automática de placas:

Para agilizar los procesos de ingreso y pago, se desarrolló una herramienta para detectar placas de vehículos automáticamente a partir de imágenes. El algoritmo consiste en aislar la placa del resto de la imagen y finalmente con la librería EasyOCR de Python se procede a extraer el texto asociado.

3. Resultados

A partir de las fotos obtenidas con un Drone, se desarrolla un algoritmo que permita segmentar las celdas individuales como se observa en la Figura 3. Manualmente, las imágenes son separadas en las dos categorías objetivo: ocupadas y vacías.

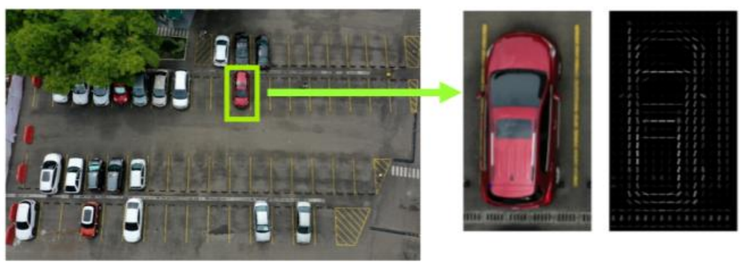

Figura 3: Ejemplo de imagen segmentada y su respectivo HoG.

Se realizó el proceso de extracción de características para cada imagen segmentada y se construyeron 3 conjuntos de datos diferentes: la información de los pixeles de todas las imágenes segmentadas, el HoG y las características extraidas de la red neuronal VGG-16. Para el caso de los pixeles, la matriz para cada P_i se transformó en un vector $V_i \in \mathbb{R}^n$. Así, se construye un conjunto de datos a partir de los pixeles donde cada fila corresponde a una imagen. Un proceso análogo se realiza con la información obtenida a partir del HoG y la red neuronal VGG-16.

Cuadro 1: Tamaño de los datos por imagen según el conjunto de características

Conjunto de Características	Tamaño del Conjunto de datos
Pixeles	(2254, 24576)
HoG	(2254, 8192)
VGG16	(2254, 25089)

Para el entrenamiento se utilizaron 1803 imágenes (80% de los datos) con mitad ocupadas y mitad vacías y luego se evaluó el desempeño de los modelos con el 20% restante (451 fotos). Los resultados del Cuadro 2 muestran que un conjunto de características más especializado para la tarea de clasificación, como los obtenidos a partir de HoG o la red VGG16, mejora el desempeño de los algoritmos en el conjunto de prueba. Adicionalmente, el HoG añade un beneficio al reducir considerablemente la dimensionalidad del problema en comparación con los otros conjuntos utilizados.

Cuadro 2: Error Train/Test con los tres conjuntos de datos

Modelo	Error Pixeles	Error VGG16	Error HoG
Regresión Logística	0.00 % / 8.42 %	0.00 % / 1.11 %	0.00 % / 3.32 %
SVM kernel lineal	0.00 % / 7.98 %	0.00 % / 0.66 %	0.00 % / 4.21 %
SVM kernel polinomial	0.16 % / 7.32 %	0.17 % / 1.77 %	0.83 % / 1.99 %
SVM kernel base radial	0.88 % / 2.43 %	0.22 % / 2.44 %	0.28 % / 1.11 %
Random Forest	0.05 % / 3.32 %	0.27 % / 2.88 %	0.44 % / 4.65 %
Gradient Boosting	0.00 % / 2.21 %	0.00 % / 2.43 %	0.00 % / 1.77 %

Para adaptar la arquitectura YOLOv5 a nuestro problema, se realizó un proceso fine-tuning para calibrar los 7.255.094 parámetros de las 283 capas de la red, con el objetivo de detectar únicamente automóviles. Fue necesario crear y etiquetar nuestro dataset a partir de fotos tomadas con un drone. Se emplearon técnicas de data augmentation (rotaciones, reflexiones y transformaciones gamma) para aumentar el conjunto de entrenamiento de 64 a 2240 imágenes, dado que estas redes neuronales tan profundas requieren una alta cantidad de datos para mejorar su desempeño, como probó Solawetz [3] con la versión v4.

El desempeño del modelo se evaluó en términos del mean average precision (mAP), definido como el cociente entre el área de intersección y el área de la unión de las bounding boxes reales y estimadas (Intersection over Union). Los resultados se muestran en la Figura 4. A continuación, en la Figura 5, se presentan algunas instancias de imágenes etiquetadas del conjunto de prueba. Cada bounding box está acompañada de la etiqueta del objeto (Cars) y la probabilidad estimada de que este pertenezca a tal clase.

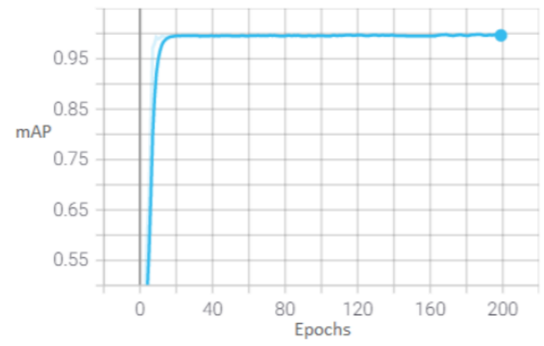

Figura 4: Desempeño del modelo a través del mAP.

Figura 5: Ejemplo YOLOv5 en el conjunto de imágenes de prueba.

En la Figura 6 se muestra como el algoritmo en funcionamiento logra extraer únicamente la placa de la imagen poniéndole una máscara a los demás detalles y luego a partir de la librería EasyOCR se extrae el texto de la placa. El código fue desarrollado replicando los tutoriales de Renotte [5] y OMES [6].

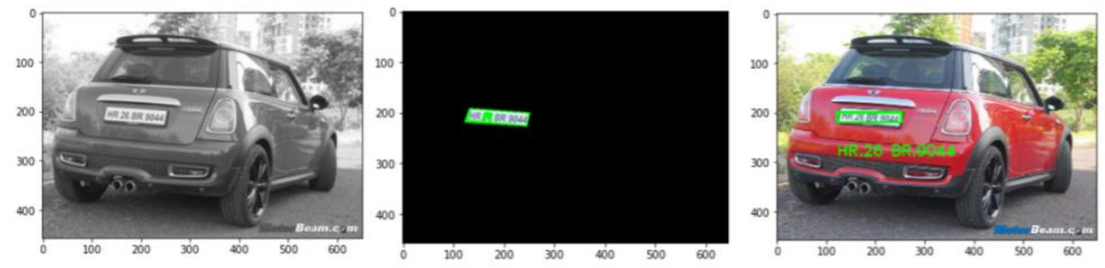

Figura 6: Ejemplo de detección de placas para un vehículo.

4. Referencias

[1] Dalal, N. ; Triggs, B. (2005). Histograms of oriented gradients for human detection.IEEE Computer SocietyConference on Computer Vision and Pattern Recognition (CVPR05).1, 886–893.

[2] Simonyan, K. ; Zisserman, A. (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition.arXiv 1409.1556.

[3] Ultralytics. (2020). YOLOv5 Pytorch implementation. [En línea]. [Consultada en Marzo de 2021]. Disponible en: https://github.com/ultralytics/yolov5

[4] Solawetz, J. (2020), Data Augmentation in YOLOv4. [En línea]. [Consultada en Septiembre de 2021]. Disponible en: https://towardsdatascience.com/data-augmentation-in-yolov4-c16bd22b2617

[5] Nicholas Renotte (2020), Python ANPR with OpenCV and EasyOCR in 25 Minutes|Automatic Number Plate Recognition Tutorial. [En línea]. [Consultada en septiembre de 2021]. Disponible en: https://www.youtube.com/watch?v=NApYP5wlKYt=1262s

[6] OMES (2020), Reconocimiento de placas vehiculares|OpenCV OCR en Python. [En línea]. [Consultada en Septiembre de 2021]. Disponible en: https://www.youtube.com/watch?v=8rgqi8mjjk