Estudio sobre el entrenamiento de una Red Neuronal Convolucional para el Reconocimiento de Ubicaciones libres y ocupadas

Gabriel Ignacio Bosco – Fernando Luis Broqua – Rodrigo Javier D'Amico
Guillermo Hernán García – Luis Damián Lancha – Germán Lorenz Vieta
Departamento de Ingeniería e Investigaciones Tecnológicas, Universidad Nacional de La Matanza
San Justo, Buenos Aires, Argentina
gibosco@gmail.com - fernandobroqua@gmail.com - rodrigo.damico92@gmail.com
guillermoh.garcia@gmail.com - damianlancha@gmail.com - germangelv@gmail.com

Resumen

En el presente documento se explicará el proceso de elección del mejor aprendizaje de una red neuronal convolucional (RNC) para la detección de personas en ubicaciones fijas, diferenciando en cada modelo escogido el formato de un mismo conjunto de datos de entrada (imágenes RGB, HSV y escala de grises) y la aplicación de configuraciones de entrenamiento diferentes. Posteriormente, en la etapa de comparación se verán los resultados de las métricas proporcionadas por la plataforma Google Cloud donde se realizaron los entrenamientos, como la precisión y la pérdida de la red. Finalmente, con las redes neuronales ya entrenadas se realizaron pruebas de velocidad de respuesta y detección respecto a regiones de interés preestablecidas. A partir de esto se pudo constatar en cada caso la cantidad de detecciones y el porcentaje de certeza promedio. De esta manera se pudo determinar la RNC más conveniente para el caso de estudio.

1. Introducción

Resulta necesaria en toda organización plural la existencia de un número mínimo de miembros para deliberar o tomar decisiones. Este requisito se denomina *quórum* el cual es imprescindible para que una legislatura "entre en sesión" [1-3]. También cabe destacar el estricto criterio que establece la legislación actual para la determinación de este estado y su importancia para la oficialización de los temas tratados [4].

En muchos casos el control del *quórum* se realiza por personas denominadas "ujieres". Esta tarea representa un gran desafío debido a que deben monitorear múltiples sectores en simultáneo y gestionar el estado de cada ubicación en toda la cámara con el objetivo primordial de brindar transparencia a la hora de ejercer el acto legislativo de votar.

El caso de estudio remite a la problemática de la Honorable Cámara de Diputados de la Provincia de Buenos Aires, en adelante H.C.D. Dado que los miembros dentro del recinto se encuentran en movimiento, el control de esta condición requiere del constante monitoreo del estado de cada una de las ubicaciones por parte del ujier a fin de detectar cada cambio [5]. Así mismo, esta labor no debe afectar al normal desarrollo de las sesiones parlamentarias ni incomodar de modo alguno a los legisladores [6].

En el marco de las asambleas parlamentarias, se propone el desarrollo de una solución completa para el control de *quórum* necesario para la aprobación de leyes.

Las soluciones actuales utilizan sensores de huellas para el reconocimiento de cada diputado y sensores de peso en cada banca para registrar su presencia. Cuando estos sistemas fallan se dispone de personal en el sitio para colaborar con las tareas del ujier.

Debido a que las soluciones actuales no son precisas y entran en conflicto con los puntos mencionados anteriormente, se desarrolla una solución innovadora y superadora aplicando aprendizaje profundo (*deep learning*) para la interpretación de las imágenes de cámaras de video, junto a un análisis comparativo de diferentes estrategias para el entrenamiento de una red neuronal.

2. Objetivo

El objetivo de la presente investigación es identificar aquel conjunto de datos de entre los propuestos que permita una mayor precisión en la detección de ubicaciones ocupadas o vacías en espacios preestablecidos por una red neuronal convolucional luego de un proceso de entrenamiento y la posterior evaluación de sus resultados.

3. Metodología

Se establece como experimento controlado la comparación de variables dependientes e independientes del entrenamiento de una red neuronal convolucional [7]. Como dependiente se utiliza la precisión media promedio (*mAP - mean Average Precision*) y como independiente los datos para entrenar la red neuronal.

Durante el entrenamiento se empleó el mismo conjunto de fotogramas (*frames*) extraídos de un video en tres modelos de color:

- RGB (Red, Green, Blue)
- HSV (Hue, Saturation, Value)
- Escala de grises (Grayscale)

La precisión promedio proporciona una medida de calidad en todos los niveles de entrenamiento para la clasificación de una clase única, y puede verse como el área bajo la curva de precisión al graficar [8]. Entonces, la *mAP* es la media de las precisiones promedio en la clasificación de varias clases; se calcula comparando el grado de solapamiento de la región detectada con la región etiquetada (ver Dataset), así como la cantidad de detecciones falsas y objetos que no se detectaron [9].

4. Entorno de desarrollo

Google Cloud

Se utiliza Google Cloud debido a la capacidad de integración con TensorFlow, TensorBoard y sus repositorios de investigación.

TensorFlow

Se utiliza la librería de código abierto TensorFlow debido a que facilita la construcción, entrenamiento y pruebas de redes neuronales [10]. A su vez se elige el detector SSD con extractor de características MobileNet V2 debido a su alto *mAP* y baja latencia [8]. MobileNet se basa en una arquitectura optimizada que utiliza convoluciones separables en profundidad para construir redes neuronales profundas y livianas [11].

TensorBoard

Debido a la complejidad y confusión que pueden generar los datos de entrenamiento se utiliza TensorBoard ya que permite una visualización del progreso de los entrenamientos contribuyendo con la comprensión, depuración y optimización.

OpenCV y NumPy

Se emplea la librería OpenCV para la obtención de *frames* a partir de un flujo de video y su posterior conversión [12]. Por otro lado, se utiliza la librería NumPy para realizar operaciones matriciales en procesos pre y post detección.

Python

Debido a que los repositorios de investigación de TensorFlow se encuentran desarrollados en Python por su estilo de programación productiva, flexibilidad y legibilidad, se desarrolló un software que permite probar los modelos entrenados y que emplea múltiples hilos (*threads*) para paralelizar la captura de las imágenes de video de múltiples fuentes. Esto último otorga una mejora considerable en la detección en tiempo real de las clases entrenadas.

5. Procedimiento general de entrenamiento

A continuación, se detallan los algoritmos y procesos empleados desde la preparación de los datos para el entrenamiento de la red neuronal hasta el reconocimiento de las ubicaciones.

Dataset

Los datos de entrada para el entrenamiento se definen como el conjunto de *frames* obtenidos de un flujo de video a razón de un *frame* por segundo. Se emplea la herramienta LabelImg [13] para realizar el etiquetado o anotación manual de cada *frame* en formato PASCAL VOC 2012. Los paquetes anotados conforman 209 imágenes para entrenamiento y 52 imágenes para evaluación que incluyen 27 etiquetas de reconocimiento (*bounding boxes)* cada una.

Ubicaciones y tomas fijas

Una característica propia del caso de estudio consiste en que cada banca dentro del recinto se encuentre en una posición determinada y la toma de video se dirija a un sector preestablecido. Esto simplifica el proceso tanto de entrenamiento de la red neuronal como el de reconocimiento del estado de las bancas ya que se conoce a priori las regiones de interés (*ROI - region of interest*) para el análisis.

Transformaciones

Con imágenes RGB (Red, Green, Blue)

Se emplean imágenes de 32 bits de color sin transformación alguna. Cada píxel de la imagen viene definido por tres capas con un valor cada una que fusionados brindan el color final: rojo (red), verde (green), azul (blue).

Figura 1. Detección RGB a la izquierda, esperado a la derecha

Con imágenes HSV (Hue, Saturation, Value)

Se emplean imágenes de 32 bits de color convertidas de RGB a HSV mediante la librería OpenCV aplicando el siguiente método [12]:

$$V \leftarrow max(R,G,B)$$

$$S \leftarrow \begin{cases} \dfrac{V - min(R,G,B)}{V} & si\ V \neq 0 \\ 0 & en\ otro\ caso \end{cases}$$

$$H \leftarrow \begin{cases} \dfrac{60(G - B)}{V - min(RGB)} & si\ V = R \\ 120 + \dfrac{60(B - R)}{V - min(RGB)} & si\ V = G \\ 240 + \dfrac{60(R - G)}{V - min(RGB)} & si\ V = B \end{cases}$$

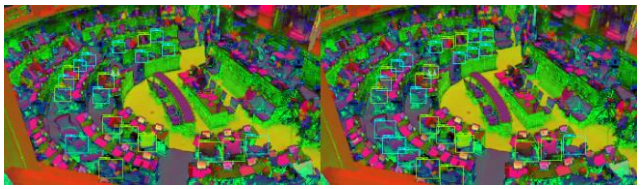

Figura 2. Detección HSV a la izquierda, esperado a la derecha

Con imágenes en escala de grises

Se emplearon imágenes de 32 bits de color convertidas de RGB a escala de grises (*grayscale*) a través de la librería OpenCV que aplica la siguiente transformación lineal [12]:

$$Escala\ de\ Gris \leftarrow 0{,}299R + 0{,}587G + 0{,}114B$$

Figura 3. Detección Grayscale a la izquierda, esperado a la derecha

6. Red neuronal

Existe un gran número de enfoques para adquirir, procesar, analizar y comprender las imágenes [14] para luego ser tratadas, lo que se denomina *computer vision*. Uno de los más populares es el que utiliza una red neuronal convolucional [15] y es el que se desarrolla a lo largo de este documento.

Una red neuronal artificial posee varias capas que realizan diferentes tareas de computabilidad las cuales trabajan en conjunto para darle un significado a la información que reciben. Cada capa está compuesta de nodos con diferentes pesos en las conexiones que afectan en el cálculo que realiza mediante la activación o no de la neurona correspondiente a la conexión [16-17].

A la hora de entrenar una red neuronal esos pesos se van ajustando de manera que la red realice la estimación más precisa posible. Cada capa de convolución aplica un filtro sobre la imagen de entrada o la salida de la capa anterior, calculando un producto matricial sobre una selección de la fuente el cual se guarda en un mapa de activación. Al hacer que un filtro pequeño atraviese la entrada, la capa convolucional conserva la estructura espacial de la entrada [18].

Se eligió para las pruebas la red convolucional SSD con MobileNetV2 debido a que tiene un alto mAP y velocidad de respuesta utilizando bajos recursos computacionales [9] y se encuentra implementada en los repositorios de investigación públicos de TensorFlow de Google la cual fue pre-entrenada con un conjunto de datos de detección, segmentación y subtitulación de objetos a gran escala denominado COCO [19] (ssd_mobilenet_v2_coco). A este modelo se lo ajustó mediante un entrenamiento y validación con un conjunto de datos generados por anotaciones extraídas de un video registrado en la H.C.D.

Las imágenes de entrada fueron redimensionadas a un tamaño de 300 x 300 píxeles y se le aplicaron filtros que fueron reduciendo su tamaño y ampliando su profundidad, para luego ser procesado por capas ocultas mediante el proceso de retropropagación que emplean la función de activación denominada RELU hasta entregar valores de certeza para las clases entrenadas [11].

7. Entrenamiento

La estrategia de entrenamiento empleada busca aprovechar un modelo pre-entrenado brindado para competiciones [20] debido a que permite una inferencia inmediata al utilizar categorías que ya están en sus conjuntos de datos. De esta forma se dio viabilidad al proyecto del cual desprende esta investigación y posteriormente se modificaron las clases de la red pre-entrenada.

Configuración inicial de SSD-MobileNet-v2

La configuración inicial (*pipeline*) utilizada fue la provista por el repositorio de investigación (ssd_mobilenet_v2_coco.config) [21]. Posteriormente se realizaron ajustes para mejorar su *mAP*. En este caso en particular se opta por definir dos clases de objetos a reconocer: *person* (persona) y *empty* (vacío).

```
item {id: 1
  name: 'person'}
item {id: 2
  name: 'empty'}
```

El aumento de datos (*data augmentation*) es el método más sencillo para reducir el sobreajuste (*overfitting*) de los

pesos que tienen los nodos de una red neuronal cuando se trabaja con imágenes como datos. En una primera instancia se entrenó la red aumentando datos con un espejado horizontal (*random_horizontal_flip*) y corte de imagen para solucionar problemas de superposición (*ssd_random_crop*). Posteriormente con otro archivo de configuración se entrenó aumentando datos con dichos parámetros activados y además se aplicaron ajustes de contraste (*random_adjust_contrast*) y brillo (*random_adjust_brightness*). Estas manipulaciones adicionales brindan nuevos casos sobre los anotados que aumentaron significativamente el *mAP* [22].

Los parámetros relevantes de configuración se encuentran detallados en la Tabla 1.

Tabla 1
Comparación de Pipeline empleado en Primeras Pruebas y Pruebas Optimizadas

Parámetro	Primeras pruebas	Pruebas optimizadas
feature_extractor		
depth_multiplier	1.0	
min_depth	16	
feature_extractor > conv_hyperparams > regularizer > l2_regularizer		
weight	3.9999999e-05	3.0e-06
box_predictor > convolutional_box_predictor		
kernel_size	1	3
box_predictor > post_processing > batch_non_max_suppression		
max_detections_per_class	100	20
max_total_detections	100	30
activation	RELU_6	
batch_non_max_suppression		
max_detections_per_class	100	20
max_total_detections	100	30
score_converter	SIGMOID	
image_resizer > fixed_shape_resizer		
height	300	100
width	300	
classification_loss	weighted_sigmoid	
hard_example_miner		
num_hard_examples	3000	
max_negatives_per_positive	3	
min_negatives_per_image	3	
eval_config		
num_visualizations	52	
num_examples	50	
metrics_set	"coco_detection_metrics"	
shuffle	shuffle	
num_readers	1	
train_config		
batch_size	24	16
batch_size > data_augmentation_options	random_horizontal_flip ssd_random_crop	random_horizontal_flip ssd_random_crop random_adjust_contrast random_adjust_brightness
num_steps	50000 y 100000	
train_config > optimizer > rms_prop_optimizer > learning_rate		
initial_learning_rate	0.004	0.005
decay_steps	800720	5000
decay_factor	0.94999999	0.8

8. Resultados

Mediante el análisis de la evaluación de entrenamiento

Se obtuvieron datos de referencia cuando la red se encontraba en 50000 (50K) y 100000 (100K) pasos o iteraciones de entrenamiento.

Los resultados que arrojaron los entrenamientos en la plataforma de TensorBoard en las primeras pruebas con el *pipeline* por defecto se muestran en la Tabla 2.

Tabla 2
Comparación de Porcentaje de Precisión en Resultados de Entrenamiento con Primeras Pruebas

Pasos	50K			100K		
Filtros	RGB	HSV	Gray	RGB	HSV	Gray
mAP	72,13	69,49	**75,57**	**76,69**	75,69	71,95
mAP@.75	84,17	83,09	**88,93**	**89,42**	89,40	89,11
mAP@.50	95,27	95,28	**95,73**	95,04	**95,42**	95,35

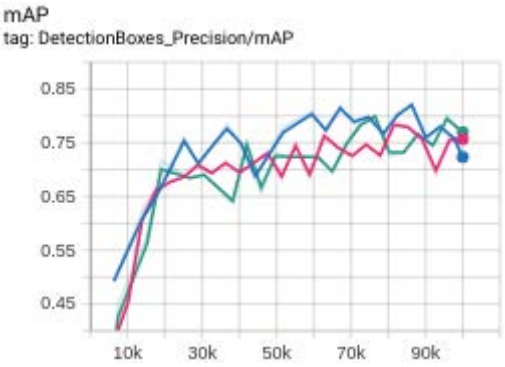

Figura 4. Gráfico de Porcentaje de Precisión en Detección por Pasos con Primeras Pruebas

Las pérdidas que arrojaron los entrenamientos con las primeras pruebas se muestran en la Tabla 3.

Tabla 3
Comparación de Porcentaje de Pérdida en Resultados de Entrenamiento con Primeras Pruebas

Pasos	50K			100K		
Filtros	RGB	HSV	Gray	RGB	HSV	Gray
Classification	0,0948	0,3183	**0,0692**	0,0996	0,1883	**0,0769**
Localization	**0,0688**	0,1143	0,0723	**0,0641**	0,0734	0,0751
Regularization	0,2354	0,2354	**0,2348**	0,2172	**0,2169**	0,2173

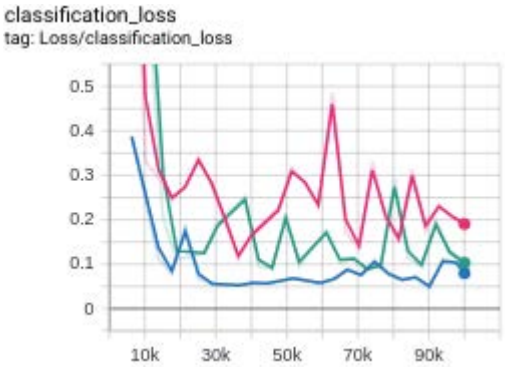

Figura 5. Gráfico de Porcentaje de Pérdida al Clasificar Objetos por Pasos con Primeras Pruebas

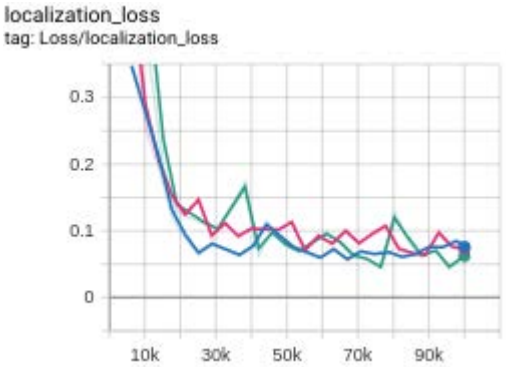

Figura 6. Gráfico de Porcentaje de Pérdida al Localizar Objetos por Pasos con Primeras Pruebas

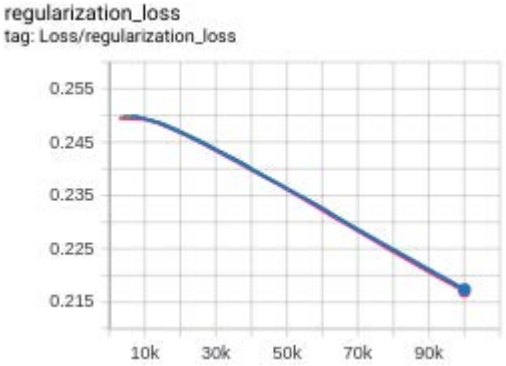

Figura 7. Gráfico de Porcentaje de Pérdida al Aprender por Pasos con Primeras Pruebas

Posteriormente optimizado el *pipeline*, los resultados que arrojaron los entrenamientos en las pruebas optimizadas se presentan en la Tabla 4.

Tabla 4
Comparación de Porcentaje de Precisión en Resultados de Entrenamiento con Pruebas Optimizadas

Pasos	50K			100K		
Filtros	RGB	HSV	Gray	RGB	HSV	Gray
mAP	78,88	77,31	**80,94**	80,02	77,95	**81,48**
mAP@.75	87,87	86,96	**89,53**	87,40	87,49	**89,50**
mAP@.50	89,59	89,60	**91,09**	89,59	89,60	**91,09**

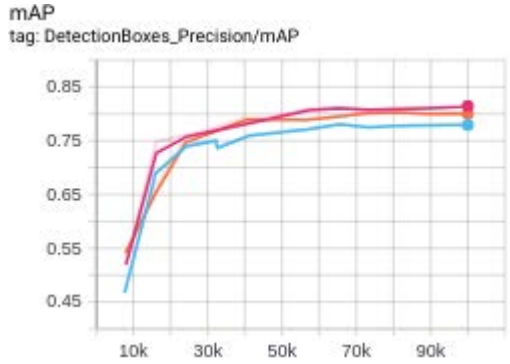

Figura 8. Gráfico de Porcentaje de Precisión en Detección por Pasos con Pruebas Optimizadas

Las pérdidas que arrojaron los entrenamientos en las pruebas optimizadas se muestran en la Tabla 5.

Tabla 5
Comparación de Porcentaje de Pérdida en Resultados de Entrenamiento con Pruebas Optimizadas

Pasos	50K			100K		
Filtros	RGB	HSV	Gray	RGB	HSV	Gray
Classification	0,0414	0,0438	**0,0250**	0,0398	0,0396	**0,0255**
Localization	**0,0172**	0,0329	0,0186	**0,0143**	0,0271	0,0191
Regularization	0,2119	0,2114	**0,2110**	0,2106	0,2101	**0,2097**

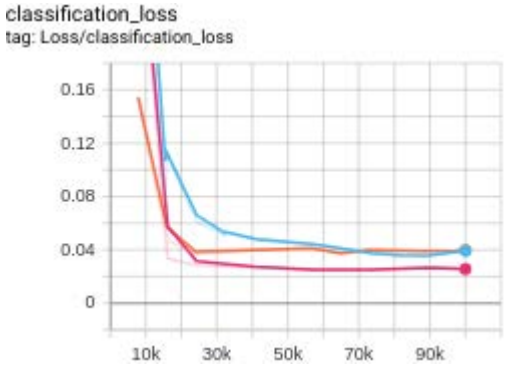

Figura 9. Gráfico de Porcentaje de Pérdida al Clasificar Objetos por Pasos con Pruebas Optimizadas

Figura 10. Gráfico de Porcentaje de Pérdida al Localizar Objetos por Pasos con Pruebas Optimizadas

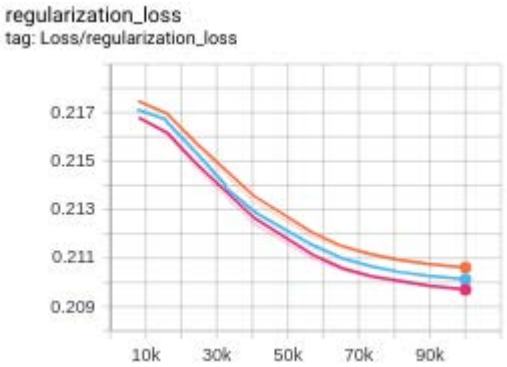

Figura 11. Gráfico de Porcentaje de Pérdida al Aprender por Pasos con Pruebas Optimizadas

Mediante el desarrollo del prototipo de proyecto

Una vez entrenada la red neuronal se prosiguió a medir su tiempo de respuesta. Para esto se creó un programa que toma un *frame* cada 3 segundos del flujo de video registrado en la H.C.D y lo procesa por la red neuronal obteniendo una *bounding box* por cada objeto reconocido, la clase asociada y el nivel de certeza correspondiente. Cada *bounding box* de la clase *person* se compara con cada ubicación fija (previamente configuradas) y se analiza el porcentaje de solapamiento entre cada par para determinar a cuál banca corresponde. En el caso de presentar un solapamiento mayor al 50% se considera a la banca como ocupada. Las ubicaciones fijas que no presentan solapamiento con ninguna *bounding box* de la clase *person* conforman el conjunto de bancas vacías. De esta manera se obtiene periódicamente el estado de cada una de las bancas.

Se tomaron los tiempos de respuesta de la red neuronal entrenada bajo cada uno de los conjuntos de datos contemplados, empleando como entrada un flujo de video de prueba. Los resultados se muestran en la Tabla 6.

Tabla 6
Comparación de Tiempos de Respuesta por Ciclo de Ejecución entre Primeras Pruebas y Pruebas Optimizadas

	Primeras pruebas (milisegundos)	Pruebas optimizadas (milisegundos)
RGB	113,44	80,13
HSV	127,30	93,77
Grayscale	**110,79**	**76,72**

Se evaluó también la capacidad de las redes entrenadas de generar *bounding boxes* bajo los conjuntos de datos de prueba. Para esto se definió la métrica que mide la cantidad promedio de *bounding boxes* obtenidas por *frame* procesado por la red neuronal. A su vez, se midieron los porcentajes de certeza promedio en las detecciones, mostrando los datos obtenidos en la Tabla 7.

Tabla 7
Comparación de Porcentajes de Certeza Promedio en las Detecciones entre Primeras Pruebas y Pruebas Optimizadas

	Primeras pruebas		Pruebas optimizadas	
	Porcentaje promedio	Cantidad promedio	Porcentaje promedio	Cantidad promedio
RGB	83,94%	3.85	78,41%	4.34
HSV	89,47%	3.16	60,49%	0.60
Grayscale	**97,85%**	**12.57**	**99,55%**	**12.84**

El primer resultado evidenciado por la comparación en la Tabla 7 es la capacidad superior de la red entrenada con *Grayscale* por sobre las otras para obtener *bounding boxes* desde las imágenes. En el caso de RGB, por ejemplo, aunque el porcentaje promedio de certeza en la obtención de las *bounding boxes* fue alto, solo logró detectar 4 objetos de interés en promedio, en contraposición con los cerca de 13 hallados por *Grayscale*. Esta red neuronal también destaca por sobre las otras por los niveles de certeza ampliamente superiores y por haber sido capaz de obtener, a lo largo de cada ciclo de detección, *bounding boxes* en la totalidad de los lugares previamente etiquetados como *ROI*.

Por último, se puede ver reflejado en los valores de la Tabla 6 que el *pipeline* (Tabla 1) de las pruebas optimizadas logró mejorar el comportamiento de la red neuronal y el nivel de confianza con respecto al *pipeline* de las primeras pruebas.

Mediante el análisis de la evaluación de un nuevo entrenamiento para RGB y HSV

Con los resultados de las pruebas anteriores se consideró realizar un tercer entrenamiento buscando mejorar los resultados para RGB y HSV obtenidos de las pruebas optimizadas.

Se procedió a agregar en el *pipeline* del entrenamiento HSV dos parámetros de *data augmentation* que permitieran mejorar su mAP, relacionados con el matiz (*random_adjust_hue*) y la saturación (*random_adjust_saturation*) de las imágenes a entrenar. El resultado se compara con el entrenamiento de la prueba optimizada en la Tabla 8.

Tabla 8
Comparación de Porcentaje de Precisión en Resultados de Entrenamiento con Pruebas Optimizadas en HSV y HSV con Aumento de Datos en Matiz y Saturación

Pasos	100K	
Filtros	HSV	HSV (H,S)
mAP	77,95	**80,00**

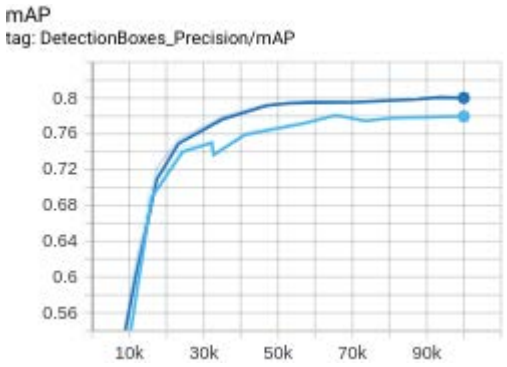

Figura 12. Gráfico de Porcentaje de Precisión en Detección por Pasos con Pruebas Optimizadas en HSV y HSV con Aumento de Datos en Matiz y Saturación

Se procedió a agregar al *pipeline* del entrenamiento RGB un parámetro que permite entrenar la red con imágenes en RGB y Grayscale en simultáneo mediante *data augmentation* con parámetro *random_rgb_to_gray*. El resultado arrojado se compara en la Tabla 9 con los entrenamientos RGB y Grayscale optimizados.

Tabla 9
Comparación de Porcentaje de Precisión en Resultados de Entrenamiento con Pruebas Optimizadas en RGB, Grayscale y RGB + Grayscale

Pasos	100K		
Filtros	RGB	Gray	RGB + Gray
mAP	80,02	**81,48**	80,82

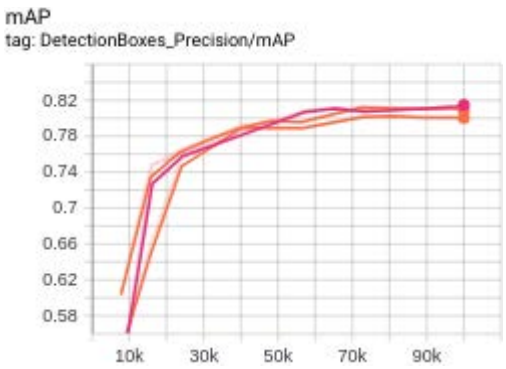

Figura 13. Gráfico de Porcentaje de Precisión en Detección por Pasos con Pruebas Optimizadas en RGB, Grayscale y RGB + Grayscale

Se puede observar en las Tablas 8 y 9 que se logra una mejora en la precisión para HSV y RGB, y que a pesar de ello Grayscale mantiene su superior mAP en el proceso de evaluación.

9. Discusión

A lo largo de las pruebas se aprecia que toda *bounding box* obtenida se correspondía con alguna de las *ROI* definidas, no pudiendo hallar otras *bounding boxes* en regiones distintas del *frame*. Esto es característico de una red entrenada con sobreajuste *(overfitting)*. Es decir que se adapta al conjunto de datos de entrenamiento a tal nivel que solo logra detectar formas idénticas con baja capacidad de generalización.

Para mejorar esta situación es necesario entrenar la red neuronal con un conjunto de datos de mayor variedad (distintas calidades de imagen, diferentes ángulos de cámara, personas y vestimentas), y también probar diversos ajustes en los factores que se utilizan para parametrizar el propio proceso de instanciación del modelo (*hyperparameters*) y configuraciones de entrenamiento (*pipeline*).

10. Conclusión

Se diseñó esta investigación con el fin de obtener la solución de red neuronal, entre tres planteadas, que mejor se adapte a la detección de ocupación de bancas fijas en un recinto, priorizando la precisión por sobre la velocidad. Las redes propuestas se diferenciaron en los conjuntos de datos de entrenamiento, utilizando imágenes en tres modelos de color distintos: RGB, HSV y escala de grises.

Se concluye que la estrategia que emplea imágenes en escala de grises (*Grayscale*) tanto para el entrenamiento como en la etapa de pruebas fue la que obtuvo los resultados más satisfactorios para nuestro caso de estudio por brindar mayor *mAP* (Tabla 2 y 4) y menores pérdidas (Tabla 3 y 5) en los entrenamientos, y tiempos de respuesta menores (Tabla 6), mayor cantidad de *bounding boxes* obtenidas y con mayor grado de certeza (Tabla 7) en las pruebas para un mismo *frame* respecto de las alternativas.

Dentro de las futuras líneas de trabajo se continuará la investigación empleando esta estrategia y se hará énfasis en procesos que permitan aumentar la precisión, eficiencia y velocidad actual.

Referencias

[1] Honorable Cámara de Diputados de la Nación, Glosario Quórum. Recuperado de https://www.hcdn.gob.ar/secparl/dgral_info_parlamentaria/dip/glosario/M/mayorias.html, 2019

[2] Constitución de la Nación Argentina. Ley 24430, Art. 14. Recuperado de https://www.argentina.gob.ar/normativa/nacional/ley-24430-804/texto, 2019

[3] Reglamento de la H. Reglamento Interno, Art. 19. Recuperado de https://intranet.hcdiputados-ba.gov.ar/refleg/REGLAMENTO%2009-12-2015.pdf, 2019

[4] Honorable Cámara de Diputados de la Nación. Reglamento interno comentado. Recuperado de https://www.hcdn.gob.ar/institucional/reglamento-comentado.html, 2019, pp. 111-112

[5] Reglamento de la H. Reglamento Interno, Art. 174. Recuperado de https://intranet.hcdiputados-ba.gov.ar/refleg/REGLAMENTO%2009-12-2015.pdf, 2019

[6] Reglamento de la H. Reglamento Interno, Art. 216 y 217. Recuperado de https://intranet.hcdiputados-ba.gov.ar/refleg/REGLAMENTO%2009-12-2015.pdf, 2019

[7] Numerentur.org, , CNN-RNA Convolucional. Recuperado de http://numerentur.org/convolucionales, 2019

[8] Agarwal, Kirti. Object Detection in Refrigerators using TensorFlow. preprint Recuperado de http://dspace.library.uvic.ca/bitstream/handle/1828/10464/Agarwal_Kirti_MSc_2018.pdf, 2019

[9] Filip Bång, Computer vision as a tool for forestry. Recuperado de http://www.diva-portal.org/smash/get/diva2:132373 , 2019

[10] TensorFlow Object Detection API. Recuperado de https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/installation.mdt, 2019

[11] T. Sheng, C. Feng, S. Zhuo, X. Zhang, L. Shen, and M. Aleksic. A quantization-friendly separable convolution for mobilenets. arXiv preprint arXiv:1801.04381, 2019

[12] OpenCV: Color conversions Open Source Computer Vision. Recuperado de https://docs.opencv.org/4.1.1/de/d25/imgproc_color_conversions.html, 2019

[13] Awesome Open Source LabelImg, Labelimg. Recuperado de https://awesomeopensource.com/project/tzutalin/labelImg, 2019

[14] Sumit Saha, “A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way” . Recuperado de https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53, 2019

[15] A. G. Howard, “Rich feature hierarchies for accurate object detection and semantic segmentation” arXiv Prepr. arXiv1311.2524, pp. 1–2, 2014.

[16] Redes Neuronales Artificiales: Fundamentos, modelos y aplicaciones , J Hilera, V Martinez. Alfaomega- rama, 2000.

[17] Redes de Neuronas Artificiales : Un enfoque práctico, Pedro Isasi Viñuela, Ines Galvan de Leon. Person Prentice Hall, 2004. pp. 13-14

[18] Serena Yeung (2017, Aug. 11). Lecture 5, Convolutional Neural Networks. Recuperado de

https://www.youtube.com/watch?v=bNb2fEVKeEo, 2019
[19] Google, “TensorFlow Object Detection Repository”. Recuperado de https://github.com/tensorflow/models/tree/master/research/object_detection, 2019
[20] Google TensorFlow, Model Zoo Repository. Recuperado de https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/detection_model_zoo.md, 2019
[21] Google TensorFlow, Sample configs. Recuperado de https://github.com/tensorflow/models/tree/master/research/object_detection/samples/configs, 2019
[22] A. G. Howard, “Some Improvements on Deep Convolutional Neural Network Based Image Classification,” arXiv Prepr. arXiv1312.5402, pp. 1–6, 2013.