Comparación de marcos de trabajo de Aprendizaje Profundo para la detección de objetos

Jesús Benito-Picazo, Karl Thurnhofer-Hemsi, Miguel A. Molina-Cabello, Enrique Domínguez
Departmento de Lenguajes y Ciencias de la Computación
Universidad de Málaga
Bulevar Louis Pasteur, 35. 29010 Málaga. España.
{jpicazo, karlkhader, miguelangel, enriqued}@lcc.uma.es

***Resumen*—Muchas aplicaciones en visión por computador necesitan de sistemas de detección precisos y eficientes. Esta demanda coincide con el auge de la aplicación de técnicas de aprendizaje profundo en casi todos las áreas del aprendizaje máquina y la visión artificial. Este trabajo presenta un estudio que engloba diferentes sistemas de detección basados en aprendizaje profundo proporcionando una comparativa unificada entre distintos marcos de trabajo con el objetivo de realizar una comparación técnica de las medidas de rendimiento de los métodos estudiados.**

***Index Terms*—detección de objetos, aprendizaje profundo, redes neuronales convolucionales**

I. Introducción

La detección de objetos es una de las tareas de visión por computador más investigadas, donde en la actualidad las Redes Neuronales Convolucionales (CNNs) están mostrando un altísimo rendimiento. Las CNNs están construidas a partir de muchas capas de neuronas intrínsecamente conectadas en un modelo inspirado por la organización jerárquica de la corteza cerebral del ser humano. Las neuronas actúan como una unidad básica en el aprendizaje y extracción de características de la entrada. El rendimiento del aprendizaje y la extracción de características de la entrada se ve mejorada con el aumento de la complejidad de las redes la cual es causada principalmente por la profundidad de las capas de neuronas. Las técnicas de aprendizaje profundo o Deep Learning (DL) en general, y particularmente las CNNs, son capaces de aprender automáticamente, a partir de imágenes genéricas de entrada, datos con múltiples niveles de abstracción debido a la arquitectura profunda que facilita al modelo el proceso de captura y generalización del mecanismo de filtrado realizando operaciones de convolución en el dominio de la imagen. En la literatura se pueden encontrar muchas CNNs de alto rendimiento tales como AlexNet [1], VGG [2], GoogleNet [3], ResNet [4], etc. Algunas como [5], han demostrado superar la precisión del ojo humano en ciertas tareas de reconocimiento de objetos.

A pesar de la popularidad de otros métodos, los métodos basados en DL están superando a otras técnicas tradicionales de visión por computador por un amplio margen en términos de precisión y algunas veces incluso en eficiencia [6]. No obstante, el estado cambiante del DL producido por una falta de trabajos unificadores y revisiones del estado del arte, hacen la iniciación en este campo tediosa y difícil de mantener actualizada.

El objetivo del presente artículo es realizar un estudio de los marcos de trabajo basados en Deep Learning más prometedores en detección de objetos proporcionando unas medidas de comparación unificadas.

El resto del artículo se organiza como sigue: En la Sección II, se explican las diferentes características de los sistemas de detección y clasificación estudiados, incluidos los conjuntos de datos utilizados para este estudio. La Sección III presenta todas las medidas de rendimiento que han sido utilizadas para realizar los experimentos con los distintos sistemas y los resultados obtenidos de dichos experimentos. La Sección IV se dedicará a la extracción de conclusiones a partir de los resultados obtenidos y a perfilar los siguientes pasos que se darán para mejorar la presente investigación.

II. Metodología

Como ya se comentó en la sección I, el objetivo de este trabajo es realizar una comparación entre diferentes tipos de marcos de trabajo orientados a realizar la detección y localización de objetos basada en redes neuronales profundas. Por tanto, nuestro estudio consistirá en el entrenamiento y prueba de diferentes sistemas fundamentados en modelos de redes neuronales profundas. Como consecuencia, en este trabajo se ha realizado la comparación de cuatro de los citados sistemas: Tensorflow, con su modelo de Faster-RCNN, Pytorch también con su modelo de Faster-RCNN y otros dos sistemas basados en el marco de trabajo de Deep Learning proporcionado por Darknet: uno basado en el modelo YOLO 2.0 (YOLOv2) y otro desarrollado a partir de una combinación del modelo YOLOv2 y el modelo de red neuronal convolucional VGG-16. Dichos modelos serán desglosados con más detalle en las subsecciones II-A, II-B, y II-C, respectivamente. Todos ellos están disponibles para los sistemas operativos Windows, Mac OS X y Linux. La comparación entre los diferentes modelos será llevada a cabo utilizando el conjunto de datos VOC que también será explicado de forma más detallada en la subsección II-D.

II-A. Tensorflow

Publicado en Febrero de 2011 como una evolución del sistema de aprendizaje máquina *DistBelief* desarrollado por Google

Brain, Tensorflow es un sistema de aprendizaje máquina propietario basado en redes neuronales profundas que es capaz de llevar a cabo operaciones en arrays multidimensionales con soporte para ejecutarse en múltiples GPUs y CPUs utilizando sus extensiones CUDA y SYCL. Oficialmente, Tensorflow presenta APIs para los lenguajes C y Python y, de forma no oficial, presenta soporte para los lenguajes C++, Go y Java. Entre las aplicaciones de este sistema podemos encontrar la localización y clasificación en imágenes y el procesamiento del lenguaje natural o NLP. El sistema que vamos a considerar para la realización de nuestro estudio es la implementación de la red neuronal convolucional Faster-RCNN [7] presentada por Tensorflow. Dada la circunstancia de que las Faster-RCNNs trabajan aplicando un clasificador basado en redes neuronales convolucionales a múltiples regiones de una imagen, en nuestro estudio utilizaremos una Faster-RCNN basada en el bien conocido modelo de red neuronal convolucional VGG-16 en las tareas de detección y localización de objetos.

II-B. Pytorch

Pytorch es una biblioteca de código abierto orientada al aprendizaje máquina basada en redes neuronales convolucionales que está programada en Python y construida para su integración profunda con este lenguaje de programación. Proporciona tres características principales de alto nivel: cálculo de tensores, aceleración mediante GPUs y redes neuronales convolucionales construidas sobre un sistema de integración automática.

Pytorch proporciona una gran variedad de rutinas para operaciones con tensores para satisfacer las necesidades de computación aprovechando el uso rápido y sencillo del desarrollo basado en el lenguaje Python. También incorpora Redes Neuronales Dinámicas que utilizan una eficiente implementación de una técnica denominada derivación automática en modo inverso, la cual permite a los usuarios cambiar arbitrariamente la forma en que la red se comporta.

Tal y como sucede con el caso del modelo de Tensorflow, hemos seleccionado como nuestro sistema de detección y clasificación la implementación de la Faster-RCNN desarrollada por Pytorch que está basada en el bien conocido modelo de red neuronal convolucional VGG-16.

II-C. Darknet-YOLO

Darknet es un marco de trabajo desarrollado en lenguaje C++ orientado al diseño, entrenamiento y ejecución de redes neuronales profundas destinadas a la detección y clasificación de objetos en imágenes 2D [8]. Las principales ventajas de este sistema son su simplicidad en términos de uso, tamaño reducido, facilidad de compilación y una documentación en línea clara y concisa. Todas ellas hacen de Darknet-YOLO un sistema fácil de utilizar nada más instalarlo. También es de destacar su capacidad para utilizar el marco de trabajo CUDA de NVIDIA, el cual permite al sistema utilizar la capacidad de cómputo de las GPUs para llevar a cabo tanto procesos de entrenamiento como de validación. Incluido en el marco de trabajo de Darknet, YOLOv2 [9] es un sistema de detección de objetos que aplica una sola red neuronal a la imagen completa utilizando una sola evaluación de la red para dividirla en regiones, las cuales se utilizarán para predecir las localizaciones de los bounding boxes, en contraposición con otros sistemas tales como las Faster-RCNN, cuyo modo de operación consiste en la aplicación de un modelo de CNN conocido a miles de regiones en la misma imagen. Esta diferencia hace a YOLO un sistema mucho más rápido y computacionalmente ligero que los que llevan otros marcos de trabajo tales como Tensorflow o Pytorch, al mismo tiempo que conserva tasas de precisión aceptables en sus predicciones.

En cuestión de detección y localización de objetos basados en CNNs, Darknet es un sistema interesante porque utiliza un algoritmo diferente del modelo clásico de Faster-RCNN pero consigue resultados similares con una menor carga computacional, lo cual le confiere una mayor velocidad. Por esta razón, en la realización del presente trabajo se han considerado dos modelos distintos de redes neuronales para ser utilizadas con este sistema:

- Uno es YOLOV2 que era el modelo más avanzado de YOLO desarrollado por sus autores, J. Redmon y A. Farhadi, en el momento en que se llevó a cabo el presente estudio.
- El otro es una adaptación de YOLOv2 basada en el modelo de CNN VGG-16 que se ha realizado en este mismo trabajo con el objetivo de poder realizar una comparación con los otros basados en las CNN VGG-16 y que ya se citaron en las subsecciones II-A y II-B del presente documento.

Con el objetivo de que que este estudio sea suficientemente fiable y ofrezca unos resultados significativos, es muy importante la selección de un conjunto de datos adecuado que comprenda un conjunto de clases el cual sea lo suficientemente genérico como para representar un buen número de objetos cotidianos, y suficientemente específico como para que ofrezca la posibilidad de extraer conclusiones relacionadas con el rendimiento de cada modelo en las tareas de detección de objetos pertenecientes a clases específicas, y su posibilidad de ser utilizado en determinados entornos. Estas razones han llevado a los autores de este trabajo a seleccionar el popular conjunto de datos VOC2007.

II-D. Los conjuntos de datos VOC

Estos conjuntos de datos fueron originados en las competiciones *PASCAL Visual Object Classes* [10]. El principal objetivo de dichas competiciones era el de reconocer objetos en escenarios tomados del mundo real. Para ese propósito, fueron seleccionadas veinte categorías diferentes de objetos para las cuales se generaron sendos conjuntos de entrenamiento y validación. El número de clases se incrementó de 10 a 20 respecto al proyecto anterior. Así, se consideraron las siguientes clases: persona, pájaro, gato, vaca, perro, caballo, oveja, aeroplano, bicicleta, barco, autobús, coche, motocicleta, tren, botella, silla, mesa de comedor, maceta, sofá y tv/monitor.

De entre todos los conjuntos VOC, los más utilizados son aquellos que se generaron para las competiciones de los

años 2007 [11] y 2012 [12]. El primero de ellos consta de 9963 imágenes que contienen 24640 objetos etiquetados y el segundo contiene 11530 imágenes con un total de 27450 objetos etiquetados. Los conjuntos se encuentran divididos, estando el 50 % dedicado al entrenamiento de los sistemas de clasificación y el 50 % restante, a la validación o prueba de dichos sistemas. Por cada uno de los ficheros de imagen, estos conjuntos de datos contienen un fichero que contiene la localización y dimensiones de las regiones rectangulares mínimas así como la clase asociadas a cada uno de los objetos que se encuentran en dicha imagen. Si el objeto no está claro o está visible menos de un 20 % del mismo, entonces dicho objeto será descartado.

III. Resultados experimentales

En la red pueden encontrarse algunas implementaciones similares del modelo de red Faster-RCNN tanto para el marco de trabajo Tensorflow como para Pytorch. Dadas las similitudes de dichas implementaciones, se ha elegido *tf-faster-rcnn* [13] y *faster-rcnn.pytorch* [14] respectivamente. Para el marco de trabajo de Darknet-YOLO solamente está disponible el código original proporcionado por sus creadores.

Como ya se indicó en la Sección II, con el objetivo de realizar una comparación lo más justa posible entre los diferentes modelos y de obtener resultados lo más significativos posibles para nuestro estudio, se han utilizado los conjuntos de datos VOC2007+VOC2012 para el proceso de entrenamiento y el conjunto de datos VOC2007 para realizar las pruebas de las diferentes propuestas.

Todos los experimentos se han llevado a cabo en un ordenador con una CPU modelo Intel(R) Core(TM) i7-5930K de 64 bits con doce núcleos a 3.50GHz, 64GB de RAM y una GPU NVidia GTX Titan X. El sistema operativo base es Linux-Ubuntu 16.04.3 LTS.

Desde un punto de vista cuantitativo, se han seleccionado diferentes medidas bien conocidas para comparar el rendimiento de la detección y la clasificación de las diferentes propuestas estudiadas. Para ser exactos, se ha considerado la exactitud espacial (S) y la F-medida. Estas medidas proporcionan valores en el intervalo $[0, 1]$, donde mayor es mejor, y representan el porcentaje de aciertos del sistema.

También se consideran en este trabajo los verdaderos positivos (True Positives o TP), falsos negativos (False Negatives o FN), falsos positivos (False Positives o FP), la tasa de falsos negativos (False Negative Rate o FNR), Precisión (Precision o PR) y exhaustividad (Recall o RC). Entre todas estas medidas, la exactitud espacial y la F-medida proporcionan una buena evaluación general del rendimiento de un método dado, mientras que FN debe ser considerado contra FP (menor es mejor), y PR contra RC (mayor es mejor).

La definición de cada medida puede ser descrita como sigue:

$$S = \frac{TP}{TP + FN + FP} \qquad \text{F-medida} = 2 * \frac{PR * RC}{PR + RC} \tag{1}$$

$$RC = \frac{TP}{TP + FN} \qquad PR = \frac{TP}{TP + FP} \qquad FNR = \frac{FN}{TP + FN} \tag{2}$$

La medida típica para obtener la bondad de un método del tipo Faster-RCNN es la métrica conocida como precisión para la detección de objetos [15]–[17], que utiliza la precisión y la exhaustividad para dibujar la curva precisión-exhaustividad y la precisión media es calculada a partir de las puntuaciones de precisión media para cada clase de objeto.

En este trabajo, la detección y la clasificación son distinguidas y otras evaluaciones han sido consideradas. En este caso, hemos considerado la detección como la habilidad del sistema para localizar un objeto, independientemente de la predicción de la clase del objeto. Por otro lado, la clasificación de los objetos es analizada de acuerdo a su tamaño respecto al tamaño de la imagen.

Por tanto, dada una imagen de test de tamaño $height \times width$ píxeles que considera M objetos reales en la máscara de verdad, con centroide c_gt_m y área a_gt_m para un objeto real $m \in \{1, \ldots, M\}$, y dada una propuesta que estima N objetos predichos, con centroide c_bw_n y área a_bw_n para el objeto predicho $n \in \{1, \ldots, N\}$, la detección del objeto real m es correcta si existe un objeto predicho n con:

$$dist(c\_gt_m, c\_bw_n) <= p \cdot \sqrt{height^2 + width^2} \tag{3}$$

donde *dist* es la función de distancia euclídea y p es una constante de proporcionalidad. Es decir, si la distancia entre ambos centroides es menor que p % del tamaño de la diagonal de la imagen de test entonces el objeto m está correctamente detectado. En este estudio se ha considerado $p = 0{,}03$.

Por otro lado, en el proceso de clasificación también se han considerado objetos lejanos y objetos cercanos. Un objeto real m es lejano si:

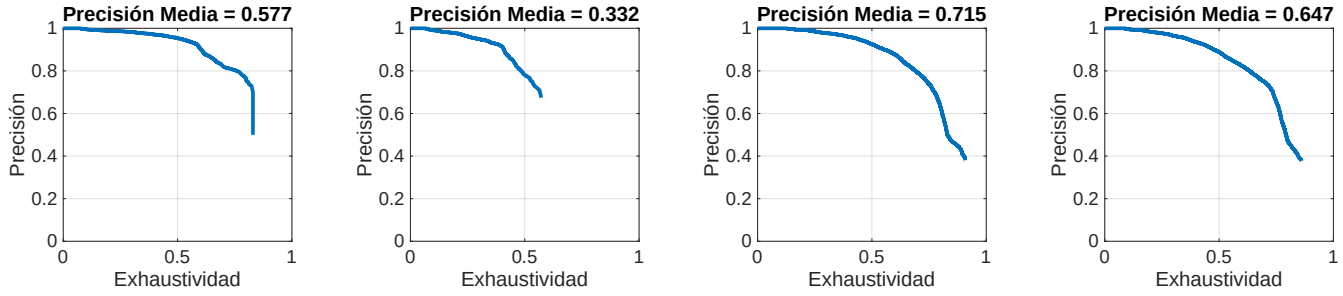
$$a\_gt_m < u \cdot (height \cdot width) \tag{4}$$

donde u es una constante. Un objeto real m es cercano si no es lejano. En este estudio se ha considerado $u = 0{,}05$.

Primero, en la primera fila de la Tabla I se muestra la velocidad de procesado para la tarea de detección de objetos. Estos valores fueron medidos teniendo en cuenta solo la función de detección, que esencialmente hace uso de la GPU a través de CUDA v9.0. Darknet es, como se puede comprobar en la tabla, el mejor marco de trabajo en términos de velocidad de procesado en el proceso de detección. Es seguido por Pytorch, mientras que Tensorflow estaría en la última posición. Es muy interesante remarcar que la diferencia entre los tiempos de computación obtenidos por YOLO y los otros dos marcos de trabajo son significativamente mayores que las diferencias entre Tensorflow y Pytorch. La razón es que Pytorch y Tensorflow están utilizando una implementación basada en una Fast-RCNN, mientras que darknet_yolo utiliza

Cuadro I: Tiempos de computación

	Darknet(yolo)	Darknet(yolo-vgg)	Tensorflow	Pytorch
Media segundos por fotograma	0.009	0.011	0.089	0.048
Media fotogramas por segundo	111.738	90.9	11.297	21.011



(a) darknet_yolo (b) darknet_yolo-vgg (c) tensorflow (d) pytorch

Figura 1: Precisión vs exhaustividad. Las imágenes muestran la figura con la representación del rendimiento en precisión frente a exhaustividad de los diferentes marcos de trabajo (mayor es mejor). La precisión media se calcula como la media de todas las clases de objetos.

un método de evaluación de una sola imagen. Todas las medidas de rendimiento se ven afectadas por este motivo.

Sin embargo, si nos centramos en la exactitud de las detecciones, en la Figura 1 y Tabla II, se puede observar que el mejor rendimiento es obtenido por Tensorflow, seguido por Pytorch. Ambos marcos de trabajo tienen menos falsos negativos en las detecciones de los objetos ya que la exhaustividad es mayor, aunque también es cierto que la precisión es menor que la obtenida por darknet_yolo, lo que indica un mejor comportamiento en términos de falsos positivos para el sistema.

En cuanto al rendimiento de la clasificación, de los resultados obtenidos en los experimentos y mostrados en la Tabla III y Figura 2, se puede observar que, hablando sobre la correcta identificación de los objetos, Tensorflow tiene el mayor número de objetos localizados correctamente clasificados, seguido de Pytorch. También tiene la mayor exactitud media para los objetos localizados correctamente clasificados. Tensorflow también presenta el menor error de distancia medio y la mayor confianza media en la clasificación.

Para algún usuario que quiera utilizar uno de los sistemas presentados en este estudio sería muy útil saber cuántos objetos del total que son detectados, son clasificados erróneamente. A este respecto, comprobando la segunda fila de la Tabla III, se puede ver que el sistema que presenta el menor número de objetos localizados clasificados erróneamente es YOLO-VGG. Parece que el sistema YOLO-VGG es el que comete menos errores en la tarea de identificación una vez que ha detectado un objeto determinado en la imagen.

Las últimas tres filas de la Tabla III muestran una visión general sobre cuántos objetos no han sido identificados por el sistema y la posible correlación de este número con la posición dentro de la imagen y el tamaño de esos objetos respecto al tamaño de la imagen de test. En la tercera fila se muestra el número total de objetos no identificados por el sistema y de los valores que aparecen allí, se puede observar que Tensorflow nuevamente supera a sus competidores mostrando el menor número de objetos no identificados, una vez más seguido de Pytorch. En esta fila también aparece el nombre de la clase de aquellos objetos que permanecen como no identificados más frecuentemente. En este caso, los cuatro sistemas concluyen que, en general, los objetos de la clase "persona" son los que más frecuentemente aparecen como no identificados.

También es interesante conocer dónde están esos objetos no identificados en la imagen de test y su tamaño. En este sentido, la cuarta fila de la Tabla III muestra el número de objetos lejanos no identificados y la clase de los objetos lejanos que más frecuentemente no se identifican. Nuestros experimentos señalan que el sistema con el menor número de objetos no identificados es una vez más Tensorflow con su asociado modelo de red basado en VGG-16. Del mismo modo, los objetos lejanos de clase "botella" son los más frecuentemente no identificados. En la última fila de la Tabla III se muestran los resultados para los objetos cercanos no identificados. Ahí se puede observar nuevamente que Tensorflow supera a sus competidores teniendo el menor número de objetos lejanos no identificados, seguido de Pytorch. Igualmente, se puede observar que los objetos de la clase "persona" son los más frecuentemente no identificados por cada sistema.

IV. Conclusiones

En este trabajo se ha propuesto un estudio de diferentes sistemas de detección y clasificación de objetos en imágenes digitales. Para ello se han realizado pruebas con cuatro sistemas de detección y clasificación basados en redes neuronales convolucionales: Tensorflow, Pytorch, YOLOv2 y YOLOv2-VGG. Se han llevado a cabo pruebas especializadas con el conjunto de datos estándar VOC calculando medidas de rendimiento adaptadas específicamente para la medición del rendimiento en sistemas de detección donde la posición y el

Cuadro II: Rendimiento de detección

	darknet(yolo)	darknet(yolo-vgg)	tensorflow	pytorch
Número de verdaderos positivos	6216	3404	7992	7198
Número de falsos negativos	5816	8628	4040	4834
Número de falsos positivos	1879	1581	6674	6504
Exhaustividad	0.517	0.283	0.664	0.598
Tasa de falsos negativos	0.483	0.717	0.336	0.402
Precisión	0.768	0.683	0.545	0.525
F-medida	0.618	0.400	0.599	0.559
Exactitud	0.447	0.250	0.427	0.388

Cuadro III: Rendimiento de clasificación

	darknet(yolo)	darknet(yolo-vgg)	tensorflow	pytorch
Objetos correctamente identificados				
Número de objetos localizados correctamente clasificados	5974	3253	7531	6738
Exactitud de objetos localizados correctamente clasificados	0.961	0.956	0.942	0.936
Error de distancia media	7.827	8.815	7.096	7.567
Confianza media	0.827	0.805	0.960	0.969
Objetos localizados incorrectamente clasificados				
Número de objetos localizados incorrectamente clasificados	242	151	461	460
Exactitud de objetos localizados incorrectamente clasificados	0.039	0.044	0.058	0.064
Error de distancia media	9.460	9.353	8.451	8.709
Confianza media	0.773	0.748	0.855	0.880
Objetos no identificados				
Número de objetos no identificados	5816	8628	4040	4834
Área media de los objetos no identificados (píxeles)	37199.04	37199.94	48008.87	47350.94
Clase menos identificada (porcentaje de todos los objetos no identificados)	persona (0.156)	persona (0.145)	persona (0.167)	persona (0.164)
Objetos no identificados (lejanos)				
Número de objetos lejanos no identificados	2060	2889	946	1175
Área media de los objetos lejanos no identificados (píxeles)	3772.41	3931,125	3.962	3.977
Clase de objetos lejanos menos identificada (porcentaje de todos los objetos lejanos no identificados)	botella (0.234)	botella (0.231)	botella (0.290)	botella (0.305)
Objetos no identificados (cercanos)				
Número de objetos cercanos no identificados	3756	5739	3094	3659
Área media de los objetos cercanos no identificados (píxeles)	55532.07	53947.39	61476.31	61279.5
Clase de objetos cercanos menos identificada (porcentaje de todos los objetos cercanos no identificados)	persona (0.182)	persona (0.165)	persona (0.191)	persona (0.188)

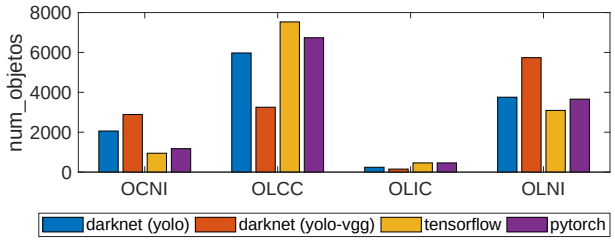


Figura 2: Comparativa de rendimiento de la clasificación de objetos. El número de objetos clasificados por marco de trabajo se muestra de acuerdo a las condiciones consideradas: Objetos Localizados Correctamente Clasificados (OLCC), Objetos Localizados Incorrectamente Clasificados (OLIC), Objetos Cercanos No Identificados (OCNI) y Objetos Lejanos No Identificados (OLNI).

tamaño de un objeto en la imagen de entrada son desconocidos *a priori*.

Los resultados experimentales revelan que Darknet-YOLO es, por una amplia diferencia, el método más rápido de los cuatro siendo un 18 % más rápido que su competidor más directo y el mejor en términos de precisión en la fase de detección. Sin embargo, cuando se trata de precisión en la clasificación, Tensorflow consigue las mejores puntuaciones tanto en la media de la medida de equilibrio precisión-exhaustividad y en número de objetos localizados correctamente clasificados. Los resultados también indican que Pytorch presenta un buen equilibrio entre detección, clasificación y velocidad, lo que lo presenta como un sistema conveniente para su utilización en tareas de detección y clasificación de objetos cuando el usuario está de acuerdo en renunciar a una pequeña cantidad de precisión a cambio de obtener un sistema de detección y clasificación más rápido que funcione casi en tiempo real.

Como una posible extensión del presente trabajo, proponemos extender este estudio a un mayor número de marcos de trabajo para deep learning que actúen como base de sistemas de detección y clasificación de objetos.

AGRADECIMIENTOS

Este trabajo ha sido financiado parcialmente por el Ministerio de Economía, Industria y Competitividad de España, bajo

los proyectos TIN2014-53465-R y TIN2016-75097-P, y por la Junta de Andalucía, bajo los proyectos TIC-6213 y TIC-657. Todos los proyectos incluyen fondos FEDER. Los autores quieren agradecer los recursos computacionales y asistencia técnica proporcionados por el Servicio de Supercomputación y Bioinformática (SCBI) de la Universidad de Málaga. Así mismo, también quieren agradecer a NVIDIA Corporation por la donación de 2 GPUs Titan X para su investigación. Karl Thurnhofer-Hemsi (FPU15/06512) está disfrutando de una beca de doctorado del Ministerio de Educación, Cultura y Deportes bajo el programa FPU.

REFERENCIAS

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proceedings of the Advances in Neural Information Processing Systems*, 2012, pp. 1097—-1150.

[2] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2014. [Online]. Available: http://arxiv.org/abs/1409.1556

[3] Y. J. P. S. S. R. D. A. D. E. V. V. C. Szegedy, W. Liu and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2015, pp. 1–9.

[4] S. R. K. He, X. Zhang and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2016, pp. 770—-778.

[5] C. J. M. A. G. L. S. G. V. D. D. J. S. I. A. V. P. M. L. e. a. D. Silver, A. Huang, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, pp. 484—-489, 2016.

[6] A. Garcia-Garcia, S. Orts-Escolano, S. Oprea, V. Villena-Martinez, P. Martinez-Gonzalez, and J. Garcia-Rodriguez, "A Survey on Deep Learning Techniques for Image and Video Semantic Segmentation," *Applied Soft Computing*, vol. 70, pp. 41–65, 2018.

[7] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Advances in Neural Information Processing Systems (NIPS)*, 2015.

[8] J. Redmon, "Darknet: Open source neural networks in c," http://pjreddie.com/darknet/, 2013–2016.

[9] J. Redmon and A. Farhadi, "Yolo9000: Better, faster, stronger," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017, pp. 6517–6525.

[10] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *International Journal of Computer Vision*, vol. 88, no. 2, pp. 303–338, Jun. 2010.

[11] ——, "The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results," http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html.

[12] ——, "The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results," http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html.

[13] X. Chen and A. Gupta, "An implementation of faster rcnn with study for region sampling," *arXiv preprint arXiv:1702.02138*, 2017.

[14] J. Yang, J. Lu, D. Batra, and D. Parikh, "A faster pytorch implementation of faster r-cnn," https://github.com/jwyang/faster-rcnn.pytorch, 2017.

[15] H. Schütze, C. D. Manning, and P. Raghavan, *Introduction to information retrieval*. Cambridge University Press, 2008, vol. 39.

[16] D. Hoiem, Y. Chodpathumwan, and Q. Dai, "Diagnosing error in object detectors," in *European conference on computer vision*. Springer, 2012, pp. 340–353.

[17] P. Dollar, C. Wojek, B. Schiele, and P. Perona, "Pedestrian detection: An evaluation of the state of the art," *IEEE transactions on pattern analysis and machine intelligence*, vol. 34, no. 4, pp. 743–761, 2012.