Memoria visual de un robot usando redes neuronales

Alexandre Rodríguez Rendo Estudiante
http://jderobot.org/Arodriguez-tfm

Máster Oficial en Visión Artificial
Universidad Rey Juan Carlos
Móstoles, España

Abstract

En este documento se pretende mostrar el contexto y el Estado del Arte del Trabajo de Fin de Máster en curso. En la introducción se hablará del contexto en cuanto a visión artificial, robótica y redes neuronales. En cuanto al Estado del Arte se hablará de las diferentes técnicas utilizadas en seguimiento (*tracking*), clasificación, detección y segmentación. Este trabajo tiene como objetivo la construcción de una memoria visual en un robot.

1 Introducción

El objetivo general de este trabajo se centra en diseñar e implementar una memoria visual en un robot utilizando redes neuronales y una técnica novedosa para *segmentación semántica* de imágenes, *Mask R-CNN*. Se hará uso de una de las plataformas existentes para el desarrollo de dichas redes, Keras. Esta memoria visual tiene como objetivo que el robot pueda "entender" el entorno. En él deberá ser capaz de diferenciar clases como humanos, puertas o mesas.
En esta sección se situará el trabajo en el contexto actual, comenzando por explicar de forma genérica en qué consisten la Robótica y la Visión Artificial. Tras ello, se realizará una introducción a las redes neuronales y algunos de los entornos de Deep Learning, así como algunos de los datasets más empleados. Por último, se expondrán los objetivos concretos de este proyecto.

1.1 Robótica y Visión Artificial

Los campos de la Robótica y la Visión Artificial se encuentran íntimamente relacionados en la actualidad y afrontan los retos del desarrollo de las capacidades básicas requeridas por sistemas semi-autónomos y autónomos para ejecutar tareas complejas en el mundo real. Así, la tendencia indica que deberán seguir un camino paralelo para resolver los problemas futuros que van a surgir. Algunas de estas capacidades requeridas son el procesamiento e interpretación de los datos del sensor o sensores, la construcción de modelos espacio-temporales del mundo exterior y el razonamiento sobre ellos, la auto-localización o las interfaces entre humano y máquina [4]. Cuando se utilizan robots cuyos sensores principales son cámaras, como el proyecto que se trata, la interpretación que el robot hace de los datos obtenidos no

es tan inmediata como puede ser la obtenida con otros sensores como láser. Aquí es donde entra en juego la importancia de la visión artificial en robótica.

Una de las áreas donde más se están utilizando los robots es la industria. Esta se ha caracterizado por su constante cambio y en la actualidad nos encontramos en la cuarta revolución industrial, más conocida como Industria 4.0. Se basa en sistemas de producción cibernéticos y automatizados, utilizando grandes volúmenes de datos y tecnologías para la manufactura. Con ella los robots han ido evolucionando desde la automatización de determinados procesos industriales hasta la actualidad, donde se buscan sistemas robóticos inteligentes y perceptivos [28].
Los sistemas que utilizan Visión Artificial en la industria se conocen como sistemas de visión o visión por máquina (*Machine Vision*) y son ampliamente empleados. Sus propósitos principales son los procesos de inspección y control de calidad (Figura 1), donde a través de una serie de medidas en las características, componentes u otros parámetros de un producto se verifica si cumple con los requisitos especificados. Pero existen muchas otras aplicaciones como la manufactura de componentes electrónicos, la biometría o los sistemas de seguridad en entornos industriales [21]. También cabe mencionar la importancia de los sistemas de visión en el movimiento del robot y el posicionamiento preciso. Este es un aspecto crucial en muchas de las operaciones que realizan los robots en áreas como la automoción o la industria aeroespacial. Así, en los procesos se requiere de una precisión y un margen de error en el movimiento que varía en cada una de las áreas. Por ello, la combinación con métodos de calibrado óptico que pueden ofrecer los sistemas de visión es de gran importancia.

Figura 1: Algunas aplicaciones de la robótica: inspección / logística de almacenes

Otros sectores como el de la logística están siendo transformados a través de robots que se encargan de realizar tareas como el embalaje, transporte o recogida del producto (Figura 1). Con esto se pretende, entre otros motivos, satisfacer la exigencia del cliente en términos de entregas rápidas y eficientes. Además de esto existen numerosas aplicaciones que surgen de la unión de Robótica y Visión Artificial, principalmente, que se están comenzando a utilizar en el día a día. Ejemplo de ello es la irrupción completa del coche autónomo en el panorama automovilístico donde se afrontan problemas que van desde la reconstrucción estéreo, la calibración de las cámaras del vehículo hasta la detección de objetos [17]. Por último, es necesario mencionar una de las aplicaciones más conocidas de la robótica, se trata del robot aspiradora al que ya no es novedad encontrar en muchos hogares (Figura 2).

Figura 2: Aspiradoras robóticas

1.2 Redes Neuronales en Visión Artificial

Desde el nacimiento de la Inteligencia Artificial (IA) en 1956 la robótica y la visión artificial han seguido un gran ritmo de evolución. Llevando a las máquinas a igualar a los humanos en la resolución de algunas tareas, y en ciertos casos, a superarlos. La inteligencia artificial se define en [25] como "el subcampo de las Ciencias de la Computación dedicado a desarrollar programas que permitan a los ordenadores presentar comportamientos que se puedan caracterizar como inteligentes". El aprendizaje máquina o *Machine Learning* (*ML*) se define en [33] como "un campo de las Ciencias de la Computación que dá a los ordenadores la capacidad de aprender sin ser explícitamente programados". Por tanto, dada esta definición, el ML se puede considerar un subcampo de la IA.
Uno de los subcampos del ML más conocidos y en auge actualmente es el denominado *Deep Learning* [9]. Este tipo de algoritmos se encuentra íntimamente ligado con las Redes Neuronales Artificiales (*ANN*) y en la práctica se suelen usar de forma equivalente aunque no son lo mismo. Uno de los aspectos a destacar en los algoritmos de Deep Learning es que ya no es necesario extraer vectores de características para la entrada a la red. Esto es así porque dichos algoritmos "aprenden" a representar los datos de forma jerárquica. Dentro de estas redes tienen especial interés en la visión artificial y en el problema que afronta este proyecto las redes neuronales convolucionales. Este tipo de redes se caracterizan por utilizar una operación convolución en al menos una de las capas de la red y están diseñadas para el procesado de datos bidimensionales como son las imágenes [23]. En los últimos años, su evolución ha sido constante llegando a superar los resultados conseguidos por algoritmos anteriormente en tareas como clasificación o detección de objetos.
Para el uso de Deep Learning han surgido numerosos entornos, estos son algunos de los más empleados en la actualidad:

- **Tensorflow**: ofrece un API de bajo nivel que permite un control completo sobre los diseños de los modelos y también un API de alto nivel más simplificado pero con una funcionalidad limitada. Además permite la visualización del entrenamiento mediante la herramienta Tensorboard.

- **Keras**: proporciona un API de alto nivel para uso de redes de neuronas. Puede correr sobre distintos *backends* como Theano o Tensorflow y dispone modelos de redes preentrenadas que permiten crear una red de forma sencilla. Escrita en Python, ofrece un entorno amigable y modular.

- **Caffe**: emplea una arquitectura C++/CUDA optimizada para uso en GPU y proporciona interfaces para Python o Matlab, por ejemplo. La definición del modelo se hace mediante Protobuf, formato creado por Google, creando una estructura de datos serializada. También dispone de modelos pre-entrenados e interfaz gráfico.

- **Darknet**: este entorno escrito en C y CUDA es rápido y de fácil instalación y proporciona soporte para CPU y GPU. Conocido por su uso en YOLO [30], una de las técnicas del estado del arte en detección de objetos que se comentará posteriormente.

En [19] se realiza un estudio comparativo que incluye los tres primeros entornos citados donde se pueden destacar las buenas prestaciones de los mismos respecto a otros.

1.3 Datasets en Visión Artificial

Los conjuntos de datos empleados a la hora de implementar un determinado sistema o probarlo son clave pues influyen en el rendimiento que pueda llegar a obtener el mismo. Además permiten hacer una comparativa de la solución encontrada respecto a otras que conforman el Estado del Arte en la tarea que se realiza, pues suelen estar asociados a algún tipo de competición. Por ello es necesario elegir correctamente el dataset o los datasets empleados en un problema de visión artificial. A continuación se comentan algunos de los datasets más relacionados con las tareas que se pretenden realizar en este trabajo:

- **COCO (Common Objects in Context)**[1]: es un dataset de larga escala para detección y segmentación de objetos principalmente. Contiene 80 categorías de objetos y 330000 imágenes de las cuales más de 200000 están etiquetadas. Es un dataset ampliamente utilizado entre la comunidad y en congresos como el ICCV (International Conference on Computer Vision).

- **ImageNet**[2]: este dataset de imágenes consta en la actualidad con 14 millones de imágenes aproximadamente y una media de 500 imágenes por cada categoría o nodo. Organiza el conocido *challenge* ILSVRC de localización y detección de objetos en imágenes y vídeos (hasta el año 2012 en conjunto con PASCAL VOC) y se está convirtiendo en uno de los datasets de referencia en este área.

- **Cityscapes**[3]: este dataset se centra en la segmentación semántica en escenas urbanas. Contiene 30 clases de objetos, 5000 imágenes etiquetadas con etiqueta "fina" (*fine*) y 20000 etiquetadas con etiqueta "gruesa" (*coarse*) en 50 ciudades diferentes.

Existen numerosos datasets más como INRIA, los datasets del MIT o Caltech, por ejemplo, que contribuyen a la mejora continua en este área.

1.4 Objetivos

Como se ha comentado al comienzo de esta sección de introducción el objetivo principal de este trabajo es la implementación de una memoria visual en un robot utilizando para ello redes neuronales. El robot deberá ser capaz de "comprender" el entorno que le rodea

[1] COCO Dataset
[2] ImageNet Dataset
[3] Cityscapes Dataset

detectando clases como humanos, puertas, mesas o ventanas y con esto poder tomar buenas decisiones. Para ello se va a probar el uso de una técnica novedosa dentro del campo de la segmentación semántica conocida como *Mask R-CNN* que ha demostrado resultados prometedores [15].

2 Trabajos previos en tracking

Una vez vistas las bases sobre las que se sustenta el proyecto es necesario ver cuáles han sido y son los trabajos que forman el Estado del Arte en las técnicas que se van a utilizar para resolver el problema que se plantea. Por ello, en esta sección se hablará de métodos de tracking, clasificación, detección y segmentación de objetos.
En el campo de la visión artificial existen una gran variedad de áreas de estudio, una de las más importantes es el denominado seguimiento de objetos (*object tracking*). Su objetivo principal es estimar el estado del objeto (*target*) durante el tiempo en una serie de secuencias de imágenes (*frames*). Este estado puede ser definido por diferentes características como la forma, la apariencia, la posición o la velocidad.
Se trata de un campo difícil ya que se pueden dar una o más circunstancias que deben ser resueltas por el algoritmo. Entre ellas están el manejo de variaciones en la iluminación y en el punto de vista del objeto que puede llevar a cambios en la apariencia del mismo. Así mismo, las oclusiones que ocurren cuando los objetos se mezclan con otros elementos de la escena o la calidad de la imagen en sí pueden ser un problema a tener en cuenta en este área. Para afrontar estos problemas se han seguido los siguientes paradigmas [37]:

- **Tracking using matching**: este grupo de algoritmos hace un *matching* entre la representación del modelo del objeto creado del frame anterior y los posibles candidatos en el siguiente frame. Los métodos más destacados son *Normalized Cross-Correlation* [3], *Lucas-Kanade Tracker* [2], *Kalman Appearance Tracker* [27] y *Mean Shift Tracking* [6].

- **Tracking-by-detection**: en estos se construye un modelo para distinguir el objeto del fondo [26]. Una vez se tiene la detección se asocia con las demás detecciones. En la actualidad, la comunidad está girando hacia las redes neuronales para computar las detecciones.

- **Tracking, learning and detection**: se trata de una extensión del grupo anterior que incluye un mecanismo para actualizar el modelo que se aprende durante la ejecución. Por ejemplo, se pueden usar los resultados de un *optical flow tracker* para esta actualización [18]. Así se consigue que el algoritmo sea invariante ante cambios en el objeto.

Previamente a las técnicas modernas que se van a comentar existen formas más clásicas de hacer seguimiento de objetos y que pueden ser útiles en problemas que exijan tiempo real a la hora de realizar el tracking, por ejemplo. Una de las más conocidas es el *feature tracking*. Esta técnica emplea puntos característicos que se puedan encontrar en las imágenes y permitan estimar el movimiento. Estos puntos deben de cumplir unos requerimientos para poder ser característicos de la imagen como son la repetibilidad (la característica se podrá encontrar en las imágenes aunque estas hayan sufrido alguna transformación), la compatibilidad (cada característica debe de ser descriptiva y fácil de encontrar) o la eficiencia (la representación

de la información característica de la imagen se debe realizar con el menor número de características posible). Unos de los puntos característicos más utilizados son las esquinas que se caracterizan por gradientes con valor más alto en ellas en dos o más direcciones. Entre estas técnicas se destacan los puntos de Harris [13] y Shi-Tomasi [35].
Existen sistemas de tracking que aprovechan la rapidez del feature tracking y el acierto de las redes neuronales para crear un tracking híbrido. En él se realizan las detecciones cada N frames utilizando algún tipo de red neuronal y el seguimiento intermedio se realiza mediante feature tracking.
Con la llegada de las redes neuronales esta forma de agrupar los diferentes métodos de tracking cambia para adaptarse a ellas [16]:

- **Tracking-by-detection**: están diseñados para seguir una determinada clase de objetos (*model-based*) y obtener un clasificador específico. En fase de test las detecciones obtenidas con redes neuronales son vinculadas usando información temporal. Se encuentran limitados a una sola clase de objetos.

- **Tracking, learning and detection**: se caracterizan por ser entrenados completamente *online*. Un ejemplo típico de tracker de este grupo muestrea zonas cercanas al objeto y las considera "*foreground*", lo mismo ocurre con las zonas lejanas que serían asignadas al "*background*". Con esto permite construir un clasificador que los diferencie y estimar la nueva localización del objeto en el siguiente frame [1]. Se ha probado a introducir redes neuronales en entornos con entrenamiento online pero debido a la lentitud de las redes a la hora de entrenar los resultados son lentos en fase de test.

- **Siamese-based tracking**: usando técnicas de *patch-matching* [40] en este tipo de redes se reciben múltiples patch candidatos del nuevo frame y se escoge como mejor candidato aquel con mayor *matching score* respecto al frame anterior, es decir, el más similar según la función de matching.

- **Tracking as regression**: en este grupo, en cambio, la red recibe solamente dos imágenes y regresa directamente la localización del objeto [16].

- **Tracking con RNN**: a partir de la detección obtenida este tipo de algoritmos emplean *Recurrent Neural Networks* para modelar la secuencia del movimiento de los objetos mejorando así la respuesta a oclusiones prolongadas en el tiempo, por ejemplo [32]. Son el estado del arte en tracking en la actualidad.

3 Trabajos previos en clasificación y detección usando redes neuronales

Mucho del progreso realizado durante los últimos años en la clasificación en visión artificial se puede asociar directamente a un conjunto de arquitecturas de redes neuronales. Gran parte de las mismas vienen implementadas y entrenadas en Keras[4]: VGG16, VGG19, ResNet50, Inception v3, Xception o MobileNet, son algunas de ellas.
Pero el primer gran paso adelante vino en 2012 cuando AlexNet [20] bate todas las propuestas del estado del arte en ese momento en el concurso (*challenge*) de ImageNet, ILSVRC, un challenge de clasificación en imágenes de referencia en visión artificial en la actualidad.

[4] Keras Applications

AlexNet obtuvo una tasa de error en test del 15,3% en comparación con el ganador el año anterior que fue del 26,2%. Esta red, junto con las redes VGG [36], sigue el arquetipo de diseño básico de las redes convolucionales: una serie de capas de convoluciones, seguido de max-pooling y capas de activación antes de las capas finales de clasificación *fully-connected*. MobileNet es una versión simplificada de Xception [5] para aplicaciones móviles que en la actualidad está debajo de las aplicaciones de visión usadas en los dispositivos móviles de Google.
Las arquitecturas ResNet e Inception, principalmente, se han convertido en bloques que sirven de base para numerosos trabajos posteriores en visión artificial y se comentan a continuación:

- **Res-Net** [14]: esta red trata de resolver el problema que parece aparecer al añadir capas a una red y es que esta se comporta peor generalmente. Por ello, los autores en lugar de tratar de aprender el *mapping* oculto de x a H(x), aprenden la diferencia entre los dos, esto es, el residuo (*residual net*). Y para calcular H(x) simplemente añaden el residuo a la entrada y vuelve a entrar en la siguiente capa.
  Esto supone un gran cambio en su momento ya que soluciona el problema de los *vanishing gradients* que sufrían las redes neuronales hasta la fecha. Además permite crear redes mucho más profundas, es decir, con más capas, que tengan buenos resultados. Con ello ResNet gana el ILSVRC de 2015 con una tasa de error del 3,57%.

- **Inception** [38]: esta familia de redes busca redes más "anchas", esto es, con más operaciones intermedias entre capas. Los autores tratan de aumentar las redes neuronales sin un incremento del coste computacional. Introduciendo diferentes operaciones de convolución en paralelo incrementan la densidad de información extraída pero también los costes computacionales. Para resolver el problema usan convoluciones de 1x1 para reducir la dimensionalidad a la vez que realizan diferentes transformaciones en paralelo. Obteniendo como resultado redes que son simultáneamente profundas y anchas.
  La primera versión de Inception, conocida como GoogLeNet, fue la ganadora del ILSVRC de 2014. Posteriormente se ha mejorado en Inception v2 y v3 hasta la última versión de Inception, v4. Inception v4 crea un híbrido con ResNet, Inception-ResNet [39].

Con la llegada de los vehículos autónomos, la videovigilancia inteligente, la detección facial y numerosas aplicaciones que están surgiendo, se demanda cada vez unos sistemas de detección más rápidos y precisos. Esto no sólo incluye reconocer y clasificar cada objeto en la imagen sino también localizarlo con su correspondiente *bounding box*. Esto hace de la detección de objetos una tarea significativamente más complicada que la tradicional clasificación de imágenes. A pesar de ello, los algoritmos de detección de objetos más exitosos en la actualidad son extensiones de modelos de clasificación de imágenes.
A continuación, se van a introducir los principales modelos de detección de objetos [10].

- **Faster R-CNN** [31]: es uno de los modelos actuales de referencia y uno de los últimos detectores conocidos como *region-based* de Girshick *et al*. Estos modelos funcionan básicamente de la siguiente forma: usan algún mecanismo para extraer regiones de una imagen que probablemente sean un objeto y luego clasifican esas regiones propuestas con una CNN.
  El padre de este modelo es el R-CNN y fue el verdadero impulsor de este tipo de técnicas [12]. En las regiones propuestas obtenidas mediante un algoritmo denominado

Selective Search se extraen las características mediante una CNN por región y luego se clasifican esas regiones basándose en las características. Pero su funcionamiento era lento.

Se mejora llega con Fast R-CNN [11] con dos motivos principales. El primero es que se aplica la CNN sobre toda la imagen en lugar de por cada región y luego se obtienen las regiones a partir del último mapa de características de la red. El segundo se debe a la introducción de una capa de Softmax que simplifica la clasificación. Su funcionamiento era más rápido y fácil de entrenar que R-CNN pero aún existía un cuello de botella en la generación de regiones.

Para resolverlo se introduce la RPN (*Region Proposal Network*) que sumado al Fast R-CNN crea Faster R-CNN. La RPN devuelve regiones propuestas basándose en un *score* que hace referencia a la probabilidad de que la bounding box sea un objeto (Figura 3). Y estas regiones son pasadas directamente a la Fast R-CNN.

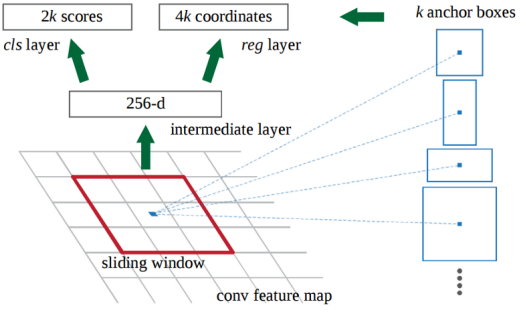

Figura 3: Region Proposal Network (RPN)

- **Overfeat** [34]: ganador del ILSVRC del 2013 en localización y detección de objetos, este trabajo mostró que entrenar una red convolucional para simultáneamente clasificar, localizar y detectar objetos en imágenes puede potenciar el acierto tanto en clasificación como en detección y localización. Posteriormente, ha sido reemplazado por SSD y YOLO para tareas que requieran una mejor detección en tiempo real.

- **SSD** [24]: existen modelos más rápidos que Faster R-CNN como R-FCN, que trata de mejorar la velocidad del sistema maximizando el cómputo compartido [8], o SSD. Esta última proporciona grandes ganancias en velocidad respecto a Faster R-CNN realizando las fases de generación de regiones de interés y posterior clasificación de forma conjunta (*Single Shot MultiBox Detector*). Como resultado obtiene una gran cantidad de bounding box de las cuales la mayoría no son útiles. Aplicando en ellas las técnicas *non-maximum suppression* y *hard-negative mining* se consiguen las detecciones finales.

- **YOLO** [30]: este sistema usa un enfoque diferente a los anteriores ya que aplica una sola red neuronal a la imagen completa. Esta red divide la imagen en regiones y predice los bounding box y probabilidades de cada región. Posteriormente estas son ponderadas con las probabilidades para obtener las detecciones definitivas. Esto lo

hace, según indican los autores, cien veces más rápido que Fast R-CNN, por ejemplo, manteniendo un acierto similar. En la última versión, YOLOv2, se introducen algunas mejoras como FCN.

En la siguiente tabla obtenida de [30] se puede observar como YOLO se encuentra casi a la par en acierto con métodos como SSD o Faster R-CNN, mejores en acierto. En cambio, tiene un mejor equilibrio rapidez-acierto ya que consigue trabajar en algunos casos a 91 FPS (*frames per second*) cuando Faster R-CNN apenas llega a 10 FPS (ver Tabla 3 en [30]).

Method	data	mAP	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv
Fast R-CNN [5]	07++12	68.4	82.3	78.4	70.8	52.3	38.7	77.8	71.6	89.3	44.2	73.0	55.0	87.5	80.5	80.8	72.0	35.1	68.3	65.7	80.4	64.2
Faster R-CNN [15]	07++12	70.4	84.9	79.8	74.3	53.9	49.8	77.5	75.9	88.5	45.6	77.1	55.3	86.9	81.7	80.9	79.6	40.1	72.6	60.9	81.2	61.5
YOLO [14]	07++12	57.9	77.0	67.2	57.7	38.3	22.7	68.3	55.9	81.4	36.2	60.8	48.5	77.2	72.3	71.3	63.5	28.9	52.2	54.8	73.9	50.8
SSD300 [11]	07++12	72.4	85.6	80.1	70.5	57.6	46.2	79.4	76.1	89.2	53.0	77.0	60.8	87.0	83.1	82.3	79.4	45.9	75.9	69.5	81.9	67.5
SSD512 [11]	07++12	74.9	87.4	82.3	75.8	59.0	52.6	81.7	81.5	90.0	55.4	79.0	59.8	88.4	84.3	84.7	83.3	50.2	78.0	66.3	86.3	72.0
ResNet [6]	07++12	73.8	86.5	81.6	77.2	58.0	51.0	78.6	76.6	93.2	48.6	80.4	59.0	92.1	85.3	84.8	80.7	48.1	77.3	66.5	84.7	65.6
YOLOv2 544	07++12	73.4	86.3	82.0	74.8	59.2	51.8	79.8	76.5	90.6	52.1	78.2	58.5	89.3	82.5	83.4	81.3	49.1	77.2	62.4	83.8	68.7

Tabla 1: Comparativa de acierto en detección en test de PASCAL VOC 2012

4 Trabajos previos en segmentación usando redes neuronales

La comunidad en visión artificial ha mejorado los resultados obtenidos en detección de objetos y segmentación semántica en un corto período de tiempo gracias, en gran parte, a poderosos sistemas de base como Faster R-CNN. En este proyecto se tratará de realizar segmentación de instancias, lo cual requiere de la correcta detección de todos los objetos en la imagen junto con la segmentación precisa de cada instancia. Así, cada píxel pertenece a alguna de las diferentes categorías sin diferenciar que se encuentre o no en un determinado objeto.

Llevados por la efectividad de la familia de R-CNN muchos de los métodos propuestos para segmentación de instancias están basados en propuestas de segmento, donde la segmentación precede al reconocimiento del tipo de objeto [29]. Esto ha demostrado ser más lento e impreciso que si se realizase de forma paralela y separada la predicción de las máscaras del objeto y las etiquetas de las clases. Li *et al*. proponen un sistema conocido como FCIS (*Fully Convolutional Instance Segmentation*) [22] que trata de predecir la salida de un conjunto de canales sensibles a la posición de forma completamente convolucional. Estos canales realizan las tareas de obtención de las clases, bounding box y máscaras de forma simultánea lo que los hace más rápidos, pero muestran errores en instancias que se solapan creando bordes espurios de forma sistemática (Figura 4).

Recientemente surge Mask R-CNN [15] para resolver muchos de estos problemas y situarse como técnica estado del arte en segmentación de instancias como se puede ver en la Figura 4. Por ello esta sección se va a centrar en esta técnica y sus resultados.

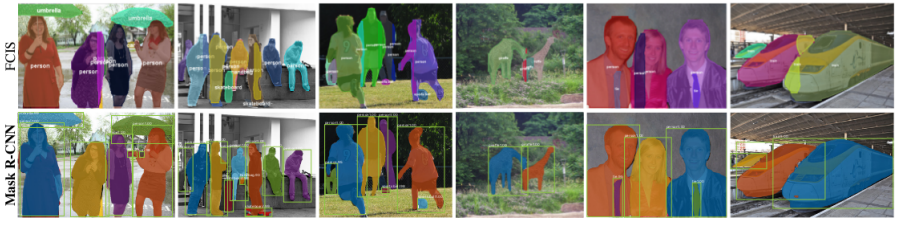

Figura 4: Resultados obtenidos por FCIS y Mask R-CNN en imágenes de test de COCO Dataset

Conceptualmente, Mask R-CNN añade una tercera etapa a Faster R-CNN en la que obtiene la máscara del objeto. La primera etapa de RPN coincide con la de Faster R-CNN mientras que en la segunda calcula, en paralelo con la predicción de la clase y la bounding box, una máscara binaria para cada región de interés (RoI). La generación de máscaras para cada clase se hace sin que las clases compitan entre ellas lo que permite separar la máscara y la predicción de la clase del objeto. Según los autores, esto demuestra ser clave para unos buenos resultados en la segmentación final.
Otro de los factores clave en el buen funcionamiento de este método es el alineamiento correcto entre las RoI y las características extraídas. Esto se suele hacer mediante *RoIPool* en Fast R-CNN pero introduce desalineamientos si lo que se desea es segmentar en lugar de clasificar, por lo que en Mask R-CNN se crea *RoIAlign* para corregir este problema. Para demostrar la generalidad del método propuesto los autores introducen la rama de predicción de la máscara sobre varias arquitecturas de redes neuronales existentes como Faster R-CNN con ResNet como extractor de características, por ejemplo, y logran superar a los ganadores del COCO Challenge de segmentación de 2015 y 2016, MNC [7] y FCIS [22]. En estos experimentos se utilizan las métricas *standard* de COCO. Como se aprecia, todas las implementaciones del modelo Mask R-CNN superan los modelos base previos del estado del arte en segmentación de instancias.

	backbone	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
MNC [10]	ResNet-101-C4	24.6	44.3	24.8	4.7	25.9	43.6
FCIS [26] +OHEM	ResNet-101-C5-dilated	29.2	49.5	-	7.1	31.3	50.0
FCIS+++ [26] +OHEM	ResNet-101-C5-dilated	33.6	54.5	-	-	-	-
Mask R-CNN	ResNet-101-C4	33.1	54.9	34.8	12.1	35.6	51.1
Mask R-CNN	ResNet-101-FPN	35.7	58.0	37.8	15.5	38.1	52.4
Mask R-CNN	ResNeXt-101-FPN	**37.1**	**60.0**	**39.4**	**16.9**	**39.9**	**53.5**

Tabla 2: Resultados obtenidos por Mask R-CNN vs el estado del arte en segmentación hasta el momento en imágenes de test de COCO Dataset

Los autores también han realizado experimentos del rendimiento de esta técnica en el dataset Cityscapes (comentado en la sección 1.3). Para las categorías *persona* y *coche* este dataset presenta un gran número de instancias de diferentes clases solapadas, esto hace aún más complicada la correcta segmentación de instancias. Sin embargo, Mask R-CNN mejora los mejores métodos actuales en esta tarea y se convierte en método estado del arte.

Referencias

[1] Boris Babenko, Ming-Hsuan Yang, and Serge Belongie. Visual tracking with online multiple instance learning. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 983–990. IEEE, 2009.

[2] Simon Baker and Iain Matthews. Lucas-kanade 20 years on: A unifying framework. *International journal of computer vision*, 56(3):221–255, 2004.

[3] Kai Briechle and Uwe D Hanebeck. Template matching using fast normalized cross correlation. In *Proc. SPIE*, volume 4387, pages 95–102, 2001.

[4] Mario FM Campos and Alberto Elfes. Introduction to the special issue on robotics and computer vision. *Journal of the Brazilian Computer Society*, 4(3), 1998.

[5] François Chollet. Xception: Deep learning with depthwise separable convolutions. *arXiv preprint arXiv:1610.02357*, 2016.

[6] Dorin Comaniciu, Visvanathan Ramesh, and Peter Meer. Real-time tracking of non-rigid objects using mean shift. In *Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on*, volume 2, pages 142–149. IEEE, 2000.

[7] Jifeng Dai, Kaiming He, and Jian Sun. Instance-aware semantic segmentation via multi-task network cascades. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3150–3158, 2016.

[8] Jifeng Dai, Yi Li, Kaiming He, and Jian Sun. R-fcn: Object detection via region-based fully convolutional networks. In *Advances in neural information processing systems*, pages 379–387, 2016.

[9] Li Deng, Dong Yu, et al. Deep learning: methods and applications. *Foundations and Trends® in Signal Processing*, 7(3–4):197–387, 2014.

[10] Cheng-Yang Fu, Wei Liu, Ananth Ranga, Ambrish Tyagi, and Alexander C Berg. Dssd: Deconvolutional single shot detector. *arXiv preprint arXiv:1701.06659*, 2017.

[11] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[12] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[13] Chris Harris and Mike Stephens. A combined corner and edge detector. In *Alvey vision conference*, volume 15, pages 10–5244. Manchester, UK, 1988.

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[15] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. *arXiv preprint arXiv:1703.06870*, 2017.

[16] David Held, Sebastian Thrun, and Silvio Savarese. Learning to track at 100 fps with deep regression networks. In *European Conference on Computer Vision*, pages 749–765. Springer, 2016.

[17] Joel Janai, Fatma Güney, Aseem Behl, and Andreas Geiger. Computer vision for autonomous vehicles: Problems, datasets and state-of-the-art. *arXiv preprint arXiv:1704.05519*, 2017.

[18] Zdenek Kalal, Jiri Matas, and Krystian Mikolajczyk. Pn learning: Bootstrapping binary classifiers by structural constraints. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 49–56. IEEE, 2010.

[19] Vassili Kovalev, Alexander Kalinovsky, and Sergey Kovalev. Deep learning with theano, torch, caffe, tensorflow, and deeplearning4j: Which one is the best in speed and accuracy? 2016.

[20] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[21] Remigiusz Labudzki, Stanislaw Legutko, and Pero Raos. The essence and applications of machine vision. *Tehnicki Vjesnik*, 21(4):903–909, 2014.

[22] Yi Li, Haozhi Qi, Jifeng Dai, Xiangyang Ji, and Yichen Wei. Fully convolutional instance-aware semantic segmentation. *arXiv preprint arXiv:1611.07709*, 2016.

[23] Tianyi Liu, Shuangsang Fang, Yuehui Zhao, Peng Wang, and Jun Zhang. Implementation of training convolutional neural networks. *arXiv preprint arXiv:1506.01195*, 2015.

[24] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.

[25] John McCarthy, Marvin L Minsky, Nathaniel Rochester, and Claude E Shannon. A proposal for the dartmouth summer research project on artificial intelligence, august 31, 1955. *AI magazine*, 27(4):12, 2006.

[26] Hieu T Nguyen and Arnold WM Smeulders. Robust tracking using foreground-background texture discrimination. *International Journal of Computer Vision*, 69(3): 277–293, 2006.

[27] Hieu Tat Nguyen and Arnold WM Smeulders. Fast occluded object tracking by a robust appearance filter. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26 (8):1099–1104, 2004.

[28] Luis Pérez, Íñigo Rodríguez, Nuria Rodríguez, Rubén Usamentiaga, and Daniel F García. Robot guidance using machine vision techniques in industrial environments: A comparative review. *Sensors*, 16(3):335, 2016.

[29] Pedro O Pinheiro, Ronan Collobert, and Piotr Dollár. Learning to segment object candidates. In *Advances in Neural Information Processing Systems*, pages 1990–1998, 2015.

[30] Joseph Redmon and Ali Farhadi. Yolo9000: better, faster, stronger. *arXiv preprint arXiv:1612.08242*, 2016.

[31] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[32] Amir Sadeghian, Alexandre Alahi, and Silvio Savarese. Tracking the untrackable: Learning to track multiple cues with long-term dependencies. *arXiv preprint arXiv:1701.01909*, 2017.

[33] Arthur L Samuel. Some studies in machine learning using the game of checkers. *IBM Journal of research and development*, 44(1.2):206–226, 2000.

[34] Pierre Sermanet, David Eigen, Xiang Zhang, Michaël Mathieu, Rob Fergus, and Yann LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. *arXiv preprint arXiv:1312.6229*, 2013.

[35] Jianbo Shi et al. Good features to track. In *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR'94., 1994 IEEE Computer Society Conference on*, pages 593–600. IEEE, 1994.

[36] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[37] Arnold WM Smeulders, Dung M Chu, Rita Cucchiara, Simone Calderara, Afshin Dehghan, and Mubarak Shah. Visual tracking: An experimental survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(7):1442–1468, 2014.

[38] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[39] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *AAAI*, pages 4278–4284, 2017.

[40] Ran Tao, Efstratios Gavves, and Arnold WM Smeulders. Siamese instance search for tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1420–1429, 2016.