Investigación en redes convolucionales y una aplicación a la detección de Peatones

Gianfranco Ferro, Estudiante de Ciencia de la Computación[1]
[1]Universidad Nacional de Ingenieria, Peru, g.ferro.palomino@gmail.com

***Abstract*– El reconocimiento automatizado de la actividad humana ha sido uno de los principales desafíos desde la concepción de la inteligencia artificial, dentro de este desafío uno de los problemas más estudiados es la detección de peatones. Es con la llegada en los últimos años del aprendizaje automático (Machine Learning), y más precisamente del Deep Learning, que se le está haciendo frente al problema con buenos resultados.**

El presente artículo tiene como fin describir la teoría de aprendizaje profundo o Deep Learning y desarrollar un modelo de detección de peatones haciendo uso de redes convolucionales.

I. INTRODUCCIÓN

Desde la concepción del computador, la gente se preguntó si dichas máquinas podrían volverse inteligentes; ya en el siglo pasado, se sentaron las bases y conceptos de lo que implicaba una máquina inteligente, ideando una prueba para evaluar si una máquina lo era tanto como un ser humano (prueba de Turing). Alcanzar tal nivel de abstracción era imposible para los paradigmas en computación de la época: establecer reglas formales con complejidad suficiente para describir con precisión el mundo que nos rodea. Estas dificultades presentes en los sistemas que se apoyan en el conocimiento codificado (pre-programado) sugerían que el rumbo a tomar era otro, los sistemas requerían de la habilidad de adquirir su propio conocimiento extrayendo patrones de datos crudos. Esto último es lo que se conoce como Machine Learning [1].

Los algoritmos de Machine Learning aprende de la data de entrada e identifica los patrones para luego tomar decisiones que en principio pueden parecer subjetivas. Podemos dividir la forma de aprendizaje en tres:

- Supervisado: Predice un resultado. Puede ser de clasificación (predice una clase) o regresión (Predice una cantidad). La data debe contar con etiquetas o valores esperados.
- No-Supervisado: Identifica el parentesco entre los datos y los segmenta.
- Reforzamiento: Aprende constantemente del ambiente, cambiando su respuesta en función del estado actual.

Los algoritmos de Machine Learning supervisados buscan aprender de los datos de entrada y no memorizarlos. Cuando se da lo segundo, se dice que el modelo está sobre-ajustado o en overfitting ya que responde adecuadamente a los datos de entrenamiento pero de manera pobre a nuevos datos. El aprendizaje, por el contrario, consiste en la generalización del modelo sobre una determinada data de entrada, respondiendo satisfactoriamente a cualquier dato de entrada siempre que sea del mismo tipo.

A pesar de las ventajas de los algoritmos de Machine Learning, para algunas tareas estaban limitadas. Por décadas, construir un reconocedor de patrones o sistema de aprendizaje automático requería de una ingeniería cuidadosa y gran conocimiento del dominio (como el Histogram Of Gradients) para diseñar un extractor que transformara la data (como los píxeles de una imagen) en una representación adecuada o vector de características de las cuales el sistema pudiera detectar o clasificar patrones. No fue hasta la última década que un algoritmo supervisado cobro relevancia: Las redes neuronales.

Se les llama redes neuronales por estar ligeramente inspiradas en el sistema nervioso. Asimismo, el proceso de conexión entre capas es una interpretación libre de la sinapsis, donde la función de activación (perceptrón de Rosenblatt) representaba el umbral a superar para la iniciación del potencial de acción. Hoy en día las redes neuronales tienen mayor base en disciplinas de matemática, siendo más una máquina de aproximación diseñada para alcanzar la generalización, que un modelo sencillo que busca representar al cerebro [1].

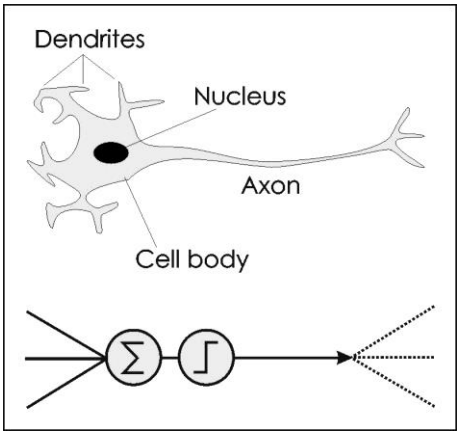

Fig. 1 Comparación entre una neurona biológica y una artificial [2].

En un principio, las redes neuronales multicapa no fueron útiles debido a diversos problemas que acarreaba su cálculo, siendo uno de ellos el de la función de activación. El algoritmo de backpropagation permitía minimizar el error de

aproximación de una manera analítica pero para su ejecución se requería de una función de activación derivable; así fue como se tomó en cuenta la función sigmoid (regresión logística). Sin embargo, el mayor problema fue su alto costo computacional para la época y su necesidad de grandes cantidades de data, lo que ocasionó que todas las expectativas depositadas en este modelo se perdieran produciendo el invierno de la inteligencia artificial. Es en dicho período, aproximadamente en 1990, que Yann LeCun discretamente desarrolla una red conectada localmente, con pesos compartidos, la cuál mejora al rendimiento de las redes neuronales en tareas de reconocimiento simple.

Gracias a las redes y el desarrollo tecnológico a fines de los 90, se volvió factible el uso de las redes neuronales como modelo predictivo masivo.
Pero es recién en el 2012 cuando se ve el verdadero poder de las redes multicapa, y más precisamente, de la arquitectura planteada por LeCunn.

Alex Krizhevsky, Ilya Sutskever y Geoffrey Hinton crearon una red neuronal convolucional profunda que les permitió ganar el concurso ImageNet Large-Scale Visual Recognition Challenge del 2012; este concurso es el equivalente a las olimpiadas de visión computacional, donde se busca el mejor modelo para tareas de clasificación, localización, detección, etc. A este modelo lo llamaron AlexNet, el cuál obtuvo una tasa de error top5 de 15.4% a comparación del 2do lugar que obtuvo 26.2% de top5 (donde el modelo no predice la etiqueta o clase correcta de la imagen dentro de sus 5 mejores predicciones) [3].

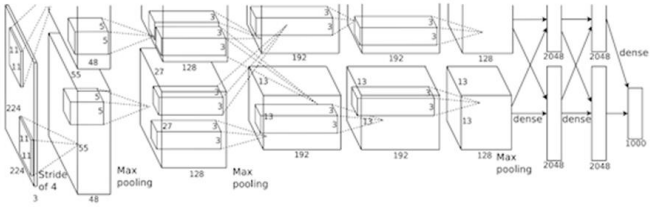

Fig. 2 Arquitectura AlexNet para entrenamiento en 2 GPUs.

Aquí nos deberíamos preguntar ¿Qué es Deep Learning?

El Deep learning o aprendizaje profundo se puede entender como una extensión de los algoritmos de Machine Learning, tomando como base a las redes neuronales. Son modelos computacionales compuestos de múltiples capas de unidades de procesamiento no lineal (neuronas) que aprenden a representar la data con múltiples niveles de abstracción. Con deep learning podemos descubrir estructuras complejas en un gran conjunto de datos usando el algoritmo de backpropagation para actualizar los parámetros usados en el modelo.

A pesar que se concibe desde las redes neuronales, éstas no son las únicas presentes en un modelo de Deep Learning. Podemos contar también con capas con propiedades diferentes como la de convolución, arquitecturas diferentes como las redes recurrentes, aunque ésto último está fuera del alcance del presente texto.

A. Deep Feedforward Networks

Las redes multicapa, también llamadas MLP por sus siglas en inglés y en ocasiones Deep Feedforward Networks, son los modelos de Deep Learning por excelencia. Tienen por objetivo el aproximar alguna función f^* para diversas tareas como pueden ser la clasificación, transcripción, traducción, estructuramiento de datos, etc [1].
Un ejemplo de lo mencionado, para el clasificador $y = f^*(x)$ mapea una entrada x a la categoría y. Una MLP mapea $y = f(x,\theta)$ y aprende el valor de los parámetros θ que resultan en la mejor aproximación de la función.

Las redes neuronales convolucionales (CNN), creadas por LeCun, están inspiradas en la corteza visual de los mamíferos, variante de las MLP. Se tiene conocimiento que la corteza visual contiene un arreglo complejo de células, las cuales son sensitivas a sub-regiones pequeñas del campo visual también llamado campo receptivo. Estas sub-regiones se superponen para cubrir el campo visual [4].

Las células actúan como filtros locales sobre el espacio de entrada y son adecuadas para representar la fuerte correlación espacial local en imágenes.
A esto se le suma la identificación de dos tipos de celulas: Las simples que responden a patrones marcados dentro del campo visual, y las complejas con un campo mas amplio y localmente invariantes a la posición de los patrones.

En resumen, la red convolucional se basa en 3 conceptos

- Conexiones locales.
- Apilamiento de capas.
- Invariante espacial o como una abstracción no cambia con el tamaño, contraste, orientación y rotación.

Una red convolucional consiste de muchas capas, las cuales pueden ser de 3 tipos [5]:

- Convolucional: consiste en una malla rectangular de neuronas. Requiere que la capa anterior tambien sea una malla rectangular de neuronas. Cada neurona toma una entrada desde una sección rectangular de la

capa anterior, los pesos se mantienen para cada neurona en la misma capa convolucional. Por ello, la capa convolucional es una convolución de imagen de la capa previa, donde los pesos especifican el filtro de convolución. Adicionalmente, pueden haber diversas mallas en cada capa convolucional; cada malla toma entradas desde todas las mallas en la capa anterior, usando diferentes filtros potenciales.

- Max-Pooling: Después de cada capa convolucional, podría haber una capa pooling. Esta capa toma bloques rectangulares (ventanas) pequeños de la capa convolucional para producir una única salida, reduciendo la dimensión. Hay diferentes maneras de realizar el pooling como tomar el valor promedio o una combinación lineal de las neuronas, pero normalmente se recurre al máximo del bloque.

- Totalmente conectada: Luego de las capas mencionadas anteriormente, se recurre a una red neuronal multicapa. Esta capa toma todas las neuronas previas (ya sea totalmente conectada, pooling o convolucional) y las conecta a cada neurona que posee. Tomar en cuenta que estas capas no se encuentran espacialmente ubicadas (solo cuentan con 1 dimensión), por lo que no puede haber más capas convolucionales después.

La capa convolucional además cuenta con tres hiperparámetros que controlan el resultado de dicha capa [6]:

- Profundidad: Consideramos a la capa convolucional como un tensor o bloque donde la profundidad será la cantidad de filtros a aplicar sobre la imagen o capa anterior.
- Stride: Controla el salto entre píxeles del filtro sobre la imagen o capa anterior. Este parámetro controla la superposición de un mismo filtro.
- Zero-padding: Agrega ceros al borde de la imagen, aumentando el tamaño de la imagen que será afectado por la convolución.

Como parte complementaria, una arquitectura convolucional contiene capas de tipo:

- Normalización: Se encarga de evitar problemas de escala. Usualmente se usa la operación ReLu, que consiste en la función $f(x) = \max\{0, x\}$. Ello convierte cualquier valor negativo en cero.
- Regularización: Ésta capa se encarga de evitar el overfitting o sobre-ajuste. Usualmente se usa una capa Dropout, que consiste en una capa con neuronas que tienen una probabilidad (usualmente 0.5) de estar activas o desactivadas por cada iteración.

B. Aprendizaje

El algoritmo de backpropagation puede ser separado en 4 secciones diferentes: El recorrido, la función de pérdida (o costo), el retroceso y la actualización de los pesos [4].

1. Recorrido o forward pass, consiste en ingresar el dato (una imagen) como arreglo de números y pasarlo por la red. Tomar en cuenta que los parámetros como los pesos y valores de los filtros iniciales tienen valores aleatorios.
   La red con dichos pesos no puede encontrar los patrones de bajo nivel u obtener alguna conclusión sobre la clasificación.

2. La función de pérdida o costo, nos permite determinar el error entre el resultado obtenido del paso anterior y el valor esperado. De forma más técnica, se compara el resultado final $a^l(x)$ con el esperado $y(x)$. Para conseguir que la red aproxime adecuadamente definimos una función de costo J.
   Como queremos el error mínimo, minimizamos la función de costo tomando en cuenta que depende de los pesos y de las capas anteriores.

3. El retrocesos o backward pass consiste en determinar que pesos contribuyen más a la función de pérdida, para ajustarlos y reducir al mínimo la función.

4. Obteniendo el gradiente de la función, podemos actualizar los pesos según la siguiente ecuación:

$$w = w_i - \eta \frac{\partial J}{\partial w}$$

   Donde w_i es el peso previo; η el ratio de aprendizaje y w el peso actualizado.

II. APLICACIÓN

Como se comentó anteriormente, el Deep Learning tiene buenos resultados para la clasificación de imágenes y objetos, siendo hoy el estado del arte en dicho tópico. Ahora nos enfocaremos en un problema específico a resolver: La detección de objetos.

La detección de objetos, y más precisamente, la detección de peatones es un tema de investigación popular debido a su

importancia en un buen número de aplicaciones, especialmente en el campo vehicular, vigilancia y robótica.

Sin embargo, ello no implica que se encuentre en un estado de perfección. Esto se puede verificar en las métricas de dichos modelos respecto a diferentes repositorios de muestra, más aún, podemos notarlo en el concurso anual de ILSVRC, donde cada año compiten por ver que modelo identifica mejor objetos de diferentes clases (entre ellas se encuentran los peatones) en imagen y video.

Generalmente, un detector de peatones involucra un sistema con la siguiente secuencia:

1. Identificar las regiones que potencialmente contienen peatones y sus límites.
2. Extraer patrones de dichas regiones.
3. Evaluar si una persona se encuentra en dicha región o no.

Para la identificación de regiones se suelen usar dos tipos de algoritmos [7]:

- Sliding Window: Consiste en una ventana o cuadro que se desliza sobre la imagen para elegir una sección y clasificarla con el modelo de reconocimiento de objetos. No solo evalúa el modelo en cada sección posible sino también a diferentes escalas, lo que termina siendo costoso en términos de rendimiento.

- Region Proposal: Recibe una imagen y responde con un cuadro delimitador correspondiente a todas las secciones de la imagen que probablemente contengan objetos. Para esto usa segmentación, donde se agrupan regiones adyacentes similares entre si basados en criterios como color, textura, etc. De esta forma se agrupan en una menor cantidad de segmentos, con diferentes escalas y proporciones, obteniéndose una menor cantidad secciones propuestas para evaluar el modelo.

Luego se podrá verificar que este sistema no es tan rígido y que aplicando conceptos de otras fases se puede mejorar el rendimiento y efectividad de la detección.

De esta forma, si bien se han tenido grandes avances y una estructura preestablecida, se tiene aún una brecha que superar.

A. Investigaciones Previas

En 2016 Tomé, Monti y compañía [8] generaron un sistema de detección de peatones basado en Deep Learning, adaptando una red convolucional de propósito general a mano. El sistema propuesto supera a los alternativos basados en patrones aprendidos y desarrollados, con un costo computacional inferior. La versión ligera del sistema se empleó con éxito en una tarjeta Jetson TX1, el cuál puede ser usado como sistema embebido para proyectos de autos autónomos.

Para esto, mejoraron las propuestas de regiones a analizar en las imágenes basándose en tres estrategias: Sliding Window, Selective Search y Locally Decorrelated Channel Features o LDCFS en conjunto con una red neuronal. Ésta última es usada principalmente en la 2da fase de procesamiento, la cual consiste en identificación de patrones en la región seleccionada de la imagen.

En 2017, Ribeiro y Bernardino [9] hacen uso de una red profunda convolucional para el problema de detección de peatones. Se enfocaron en mejorar el rendimiento de los métodos que no dependen del Deep Learning, comparándolos con un modelo pre entrenado con Imagenet y aplicando diferentes seccionamientos de imagen (filtros de frontera). Finalmente ejecutaron su propuesta en la base de datos de imágenes y video de peatones INRIA y Caltech.

En principio, se preguntaron si un modelo de CNN profunda puede mejorar el rendimiento de detectores de peatones no profundos. Para resolver esta duda, propusieron detectores no profundos en cascada, como ACG, LDCF y Spatial pooling con un modelo de redes convolucionales con varias capas produciendo una mejor generalización.

Li, C Wang [10] implementó una red convolucional para detectar peatones, usando un detector AdaBoost por capa y encontró la capa óptima comparando el rendimiento. Obtuvo 16.5\% de log-average miss rate al combinar dos detectores en la base de datos de peatones de Caltech. De igual forma, realizó la misma prueba en la base de datos de INRIA obteniendo un 9.91% al hacer la combinación de detectores.

Sin embargo, en el presente año (2017), Zhang [11] hace un recuento sobre las técnicas actuales para la detección de peatones y menciona como es que aún queda un largo camino por recorrer, debido principalmente a las limitaciones de las redes convolucionales para producir buenos resultados en la localización de objetos pequeños. Asi mismo, recalca que aplicando criterios como la regresión Bounding Box y NMS podrán dar una mejora a este inconveniente.

B. Desarrollo de un modelo

Para poder desarrollar un modelo, debemos tener en cuenta que el proceso de entrenamiento es largo y costoso. Durante la última década se ha vuelto más accesible gracias al creciente desarrollo de las GPUs y sus directivas a bajo nivel

(OpenCL o CUDA) para realizar las operaciones de manera más rápida.

En la actualidad, existen frameworks o infraestructuras de trabajo para Deep Learning que soportan (y en algunos casos exigen) el uso de GPU para el entrenamiento y ejecución (feedforward) del modelo. Ejemplos de esto son TensorFlow, Caffe, PyTorch y Theano.

Estos frameworks nos ofrecen flexibilidad ya que los conceptos anteriormente mencionados respecto a las redes profundas ya se encuentran abstraídos. Esto permite enfocarse netamente en la definición del modelo (hiperparámetros, tipo de capas, etc.).

Entre los frameworks de Deep learning, si nos centramos en la detección, podemos destacar Darknet y el modelo YOLO (You Only Look Once) creado por Joseph Redmond [12]. Éste modelo, considerado el estado del arte en detección de objetos, usa una única CNN para clasificación y localización de objetos usando cuadros delimitadores (bounding boxes). YOLO considera la detección como un problema de de regresión volviendo la arquitectura menos compleja conllevando a mayor rendimiento (45 fps en un Nvidia Titan X).

A diferencia del sliding window y regiones propuestas, éste modelo observa la imagen completa y no secciones evitando cometer errores por omisión del contexto, problemas que si se generan con las R-CNN rápidas.

Para cada cuadro, la celda predice una clase como un clasificador, resolviendo en una distribución de probabilidades sobre las clases posibles.

El valor de confianza para cada cuadro y la clase predicha se combinan en un solo valor que indica la probabilidad de que el cuadro contengo un determinado objeto.

Como solo tiene 169 celdas con 5 cuadros cada una, tenemos 845 cuadros en total. Sólo nos quedamos con los cuadros que tengan valor final superior a 30% (se puede regular con el parámetro threshold). Éstas 845 predicciones se hacen en una sola ejecución del modelo, resultando en un menor tiempo de ejecución.

Por ello, se desarrolló el modelo de predicción de peatones/personas usando el modelo YOLOv2 con Darknet, tomando el dataset de INRIA para entrenamiento y validación. Para esto, solo se consideró 30 filtros para la última capa de convolución y una sola clase; siguiendo la documentación de Darknet.

Asimismo, se siguió la guía de Alexey [13] para la conversión del dataset al formato aceptado por Darknet. Las modificaciones en los scripts se pueden encontrar en el repositorio del proyecto de éste texto [14].

III. MEDICIONES

A. Métricas

En la detección de objetos, el objetivo es el de determinar si el objeto de una cierta categoría es representado en una imagen y si provee una estimación aproximada de su ubicación, esto último se encuentra con un cuadro delimitador. La detección se considera correcta si el cuadro del objeto y la anotación se superponen en un rango mayor al 50%. Este criterio se conoce como el criterio Pascal ya que se usa en el desafío PVOC.

A partir de ello, se pueden generar otros indicadores para la detección:

$$\text{Precisión} = \frac{Objeto \cap Cuadro}{Cuadro}$$

$$\text{Recall} = \frac{Objeto \cap Cuadro}{Objeto}$$

$$IoU = \frac{Objeto \cap Cuadro}{Objeto \cup Cuadro}$$

B. *Resultados*

Para el entrenamiento de la red se utilizó, como ya se mencionó, el framework Darknet y el modelo YOLOv2 con una GPU Nvidia GTX1050Ti.

- Tiempo de ejecución: 19.47 horas.
- Se realizaron 14303 iteraciones hasta alcanzar un error promedio de 0.994734.
- La intersección de uniones (IoU) dió como resultado 0.824 y el Recall 0.948.

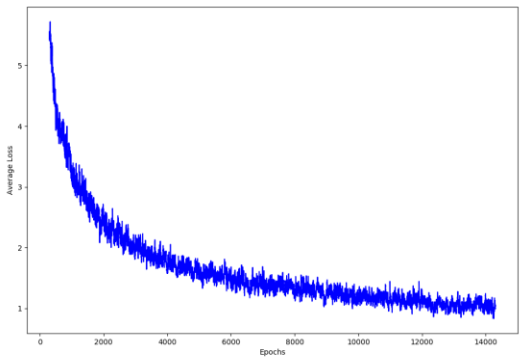

Fig. 3 Entrenamiento de la red para detección de peatones.

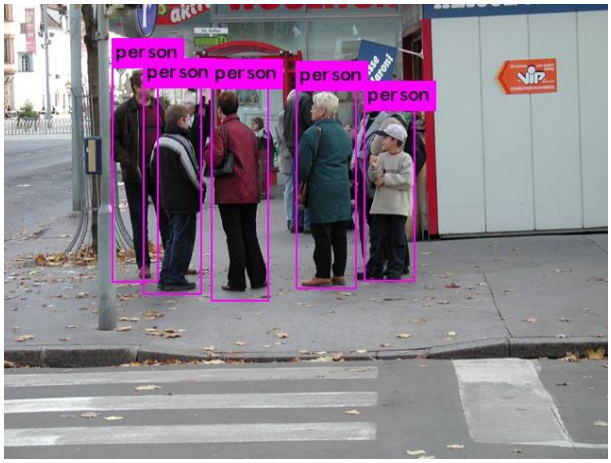

Fig. 4 Imagen de prueba del dataset INRIA con el modelo de detección.

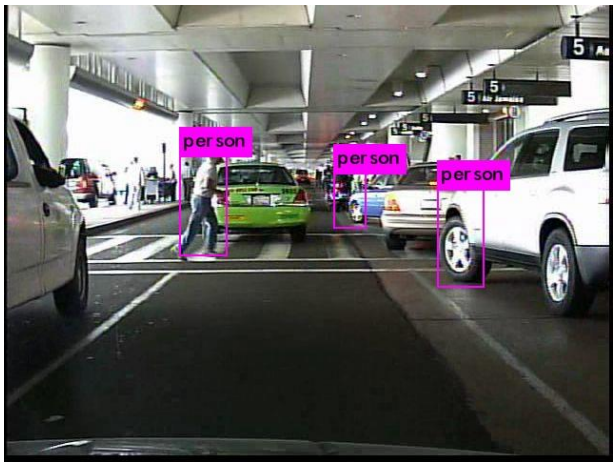

Fig. 5 Imagen del Set00 de Caltech con el modelo de detección.

Se puede encontrar pruebas en video del modelo sobre el set de validación en el repositorio del proyecto [14].

IV. Conclusiones

- El Deep learning con CNN representa el estado del arte para la detección de objetos y personas, pero presenta desafíos que van desde el algoritmo de segmentación de la imagen hasta los hiperparámetros de la red convolucional. El entrenamiento de estas arquitecturas requiere de un hardware con GPU potente para que alcance un error promedio aceptable en el menor tiempo posible.

- Dentro de los factores más importantes en el entrenamiento del modelo, la calidad de la data juega un papel importante. El dataset INRIA cuenta con diferentes imágenes de personas, pero éstas en su mayoría no se encuentran en las calles y pistas; resultando en un detector que identifica personas en actividades comunes, pero se le complica el identificarlas cuando éstas son peatones.

- Para finalizar, cabe decir que Deep learning es un campo amplio y creciente. Lo planteado en el presente texto resume conceptos básicos y obvia detalles más profundos y complejos que pueden ser revisados en la bibliografía correspondiente.

V. Trabajo a Futuro

- Entrenar el modelo con el dataset de Caltech, ya que éste es más adecuado para el problema de peatones. De esta forma, podremos validar la efectividad de la arquitectura del modelo YOLOv2.

- Evaluar otras arquitecturas de Deep Learning con los datasets INRIA y Caltech.

- Recientemente Hinton [15] explicó como las CNN pierden patrones en el reconocimiento de imágenes al aplicar la capa de pooling. Para evitar esto, propone el uso de cápsulas, con las que obtuvo resultados del estado del arte para el dataset MNIST. De ésta forma, las cápsulas se posicionan como un nuevo tema de investigación para la detección.

REFERENCIAS

[1] I. Goodfellow, et al, *Deep learning*, Vol. 1, Cambridge: MIT press, 2016, pp. 1-4, 164-166.

[2] A Basic Introduction to Feedforward Backpropagation Neural Networks, http://www.webpages.ttu.edu/dleverin/neural_network/neural_networks.html

[3] The 9 Deep Learning Papers You Need To Know About (Understanding CNNs Part 3), https://adeshpande3.github.io/The-9-Deep-Learning-Papers-You-Need-To-Know-About.html

[4] A Beginner's Guide To Understanding Convolutional Neural Networks, https://adeshpande3.github.io/adeshpande3.github.io/A-Beginner's-Guide-To-Understanding-Convolutional-Neural-Networks

[5] Convolutional Neural Networks, http://andrew.gibiansky.com/blog/machine-learning/convolutional-neural-networks/

[6] A Beginner's Guide To Understanding Convolutional Neural Networks Part 2, https://adeshpande3.github.io/adeshpande3.github.io/A-Beginner's-Guide-To-Understanding-Convolutional-Neural-Networks-Part-2/

[7] Selective Search for Object Detection (C++ / Python), https://www.learnopencv.com/selective-search-for-object-detection-cpp-python/

[8] D. Tomè, et al, "Deep convolutional neural networks for pedestrian detection", *Signal Processing: Image Communication*, vol. 47, pp. 482-489, 2016.

[9] D. Ribeiro, et al, "Improving the performance of pedestrian detectors using convolutional learning.", *Pattern Recognition*, vol. 61, pp. 641-649, 2017.

[10] C. Li and X. Wang and W. Liu, "Neural features for pedestrian detection.", *Neurocomputing*, vol. 238, p. 420-432, 2017.

[11] S. Zhang, et al. "How far are we from solving pedestrian detection?.", *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1259-1267, 2016.

[12] J. Redmon, et al, "You only look once: Unified, real-time object detection.", *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 779-788, 2016.

[13] Yolo-v2 Windows and Linux version, https://github.com/AlexeyAB/darknet#how-to-train-to-detect-your-custom-objects

[14] PedestrianDetection, https://github.com/lArkl/PedestrainDetection

[15] S. Sabour and N. Frosst, and G. E. Hinton, "Dynamic routing between capsules.", *Advances in Neural Information Processing Systems*, pp. 3859-3869, 2017.