ISBN 978 -987-633-574-4

Aprendizaje automático aplicado al procesamiento de imágenes para la clasificación de objetos reciclables

Salina Mauro[1], Osio Jorge[1,2], Cappelletti Marcelo[1,2] , Morales Martín[1]

1 Programa de Tecnologías de la Información y la Comunicación (TIC) en aplicaciones de interés social, IIyA, UNAJ

2 Línea CeTAD, Grupo de Control Aplicado (GCA), Instituto LEICI, UNLP-CONICET

{maurosalina85} @gmail.com {josio, mcappelletti, martin.morales } @unaj.edu.ar

Abstract. El presente proyecto se basa en la utilización de técnicas de Deep Learning, específicamente se realizó el modelado de redes neuronales convolucionales (CNN) capaces de clasificar distintas imágenes de objetos reciclables, estos modelos fueron probados con una clasificación binaria (reciclable-no_reciclable) y una clasificación multiclase (plástico-vidrio-metal-papel-carton, orgánico, no_reciclable). Además, se realizaron pruebas con modelos pre entrenados, utilizando aprendizaje por transferencia (Transfer Learning) para comparar resultados. Estos modelos fueron implementados utilizando como lenguaje de programación Python, apoyándose en el Framework de backend TensorFlow y la librería de alto nivel Keras. El modelo final se probó en una aplicación (beta) implementada también en Python sobre un mini computador Raspberry Pi y un módulo de cámara (picam) en donde se toman fotos y se aplica el modelo para realizar una clasificación en tiempo real.

Keywords: Machine Learning, Deep Learning, IoT, sistema de reciclaje, Visión por computadora, procesamiento de imágenes.

1 Introducción

Una de las problemáticas actuales se centra en la importancia del reciclaje y cuidado del medio ambiente. Reciclar es de suma importancia para la sociedad, debido a que, supone la reutilización de elementos u objetos ya utilizados, los que de otro modo serían desechados contribuyendo al aumento de la formación de basura y al daño ambiental permanente [1].

En nuestro país cada dos segundos se produce una tonelada de basura y una fracción grande de ella termina en rellenos sanitarios que están al borde del colapso [2].

Se estima que los RSU son la mayoría de los desechos. Entre ellos, la basura doméstica encarna la problemática más significativa: aproximadamente la tercera parte está formada por papel y derivados, mientras que el resto se compone por plásticos, vidrio, metales y pilas [2].

Es fundamental la separación en origen, puesto que discriminar una vez que los residuos están mezclados es poco práctico y costoso. En nuestro país, aunque se han tomado

medidas para fomentar el reciclado, solo un 24% de la población se esfuerza por separar los residuos para minimizar su generación y la contaminación. Gran parte del problema radica en el esfuerzo que requiere clasificar y separar los residuos inorgánicos, es por eso que la propuesta busca desarrollar un sistema de visión por computadora que permita detectar y clasificar objetos reciclables para minimizar la cantidad de residuos que se generan diariamente.

La propuesta de trabajo se enfocó en el desarrollo de una aplicación de software encargada de realizar una clasificación de imágenes en tiempo real, mediante la cual se busca detectar la presencia de objetos reciclables contenidos en la imagen. El desarrollo se basó específicamente en el uso de redes neuronales convolucionales, las cuales han demostrado ser las más eficientes en el área del procesamiento de imágenes [3].

Para llevar a cabo la implementación del presente trabajo se utilizó una herramienta que existe desde mediados del siglo pasado, pero que ha tenido un avance significativo en la última década, la Inteligencia Artificial (IA) [4].

Una de las áreas en donde se avanzó notablemente fue en la de detección de objetos y clasificación de imágenes. Esto se debe en su mayor parte al desarrollo de nuevas técnicas de Machine Learning (Aprendizaje Automático [5-7]) como el Deep Learning (Aprendizaje Profundo [8-10]), además de las innovaciones en el manejo de Big Data (datos a gran escala) y el aumento en la capacidad de cómputo mediante el uso de diferentes tecnologías como cloud computing (computación en la nube) o el uso de GPU (unidad de procesamiento gráfico) para el análisis de información. Este avance puede verse en distintas áreas como: medicina, seguridad, turismo, finanzas, robótica, entre otras.

Machine Learning es un subcampo de la IA en el que se utilizan diferentes algoritmos para recolectar datos, y con estos realizar un aprendizaje para luego hacer una predicción o sugerencia sobre algo [7]. De esta manera se permitirá resolver problemas de forma intuitiva y automatizada, sin que el mecanismo de elección se encuentre previamente programado. En la práctica esto se traduce en una función matemática en la que se parte de una entrada y se obtiene una salida, por lo que el desafío reside en construir un modelado automático de esta función matemática.

Deep Learning es un subcampo de Machine Learning, pero existen técnicas de Machine Learning que no utilizan Deep Learning. Este último es utilizado para realizar procesos de Machine Learning empleando redes neuronales artificiales compuestas por varios niveles jerárquicos [10]. En el nivel inicial la red aprende patrones simples, y esta información se envía al siguiente nivel de la jerarquía. Este segundo nivel toma la información obtenida en el primero y la combina con nuevos patrones aprendidos en este, generando información un poco más compleja, la cual es pasada a un tercer nivel, y así sucesivamente.

2 Implementación

En este apartado se detallan las herramientas y los procedimientos que se desarrollaron para implementar el sistema.

2.1 Herramientas:

ISBN 978 -987-633-574-4

- Python: Es un lenguaje de programación multiparadigma soportando la orientación a objetos, programación imperativa y programación funcional. Es interpretado, usa tipado dinámico, open source y es multiplataforma. El framework Tensorflow y la librería de alto nivel Keras utilizados para el entrenamiento de redes neuronales son APIs que pueden ser integradas en Python, siendo esta la razón de la elección para la integración en el proyecto ([11]y[12]).
- Anaconda es una distribución libre y abierta de los lenguajes Python y R, utilizada en ciencia de datos y aprendizaje automático (machine learning). Esto incluye procesamiento de grandes volúmenes de información, análisis predictivo y cómputos científicos. Está orientado a simplificar el despliegue y administración de los paquetes de software.
- Google Colab: es un servicio cloud, basado en los Notebooks de Jupyter, que permite el uso gratuito de las GPUs y TPUs de Google, con librerías como: Scikit-learn, PyTorch, TensorFlow, Keras y OpenCV. Todo ello bajo Python 2.7 y 3.6, aún no está disponible para R y Scala.
- TensorFlow: es una librería de código abierto para Machine Learning. Esta librería de computación matemática ejecuta de forma rápida y eficiente gráficos de flujo [13]. Estos gráficos están formados por operaciones matemáticas representadas sobre nudos y cuyas entradas y salidas son vectores multidimensionales de datos, conocidos como tensores (ver Fig. 1)[14] y[15].
- Keras: Es una Interfaz de Programación de Aplicaciones (API) de alto nivel y de código abierto escrita en Python. Está especialmente diseñada para facilitar una rápida experimentación en Deep Learning [16].

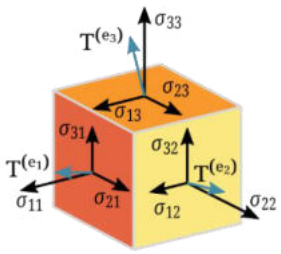

Fig. 1. Representación de un tensor

Otras librerías:

- Jupyter Notebook: El Jupyter Notebook es una aplicación web de código abierto que permite crear y compartir documentos que contienen código vivo, ecuaciones, visualizaciones y texto narrativo.

- Numpy: Librería de Python orientada al cálculo vectorial y matricial. Permite trabajar con matrices N-dimensionales, además de implementar algebra lineal, transformada de Fourier y muchas funciones relacionadas con estadísticas.

ISBN 978 -987-633-574-4

- Pandas: Es una librería de Python concebida como una extensión Numpy, que facilita el procesamiento de tablas, series temporales y otras estructuras de datos complejas.
- ScikitLearn: es una biblioteca de código abierto, que provee de herramientas científicas para el análisis de datos y el data mining [13].
- Pillow: es una librería para el manejo de todo tipo de imágenes.
- Matplotlib: es una librería específica para la generación de gráficos en Python, incluye la representación 2D y 3D de funciones e imágenes.
- OS: Permite acceder a funcionalidades dependientes del sistema operativo. Sobre todo, aquellas que refieren información sobre el entorno de este y facilitando la manipulación de la estructura de directorios.

2.2. Hardware

El uso de técnicas de Machine Learning [17], en especial de Deep Learning como es el caso de las redes neuronales convolucionales implica un costo computacional muy elevado, debido a que están compuestas por un gran número de capas y neuronas y, además, trabajan con gran cantidad de imágenes que en muchos casos tienen una alta resolución [18].

El CPU de una computadora realiza operaciones matemáticas con una velocidad muy elevada, pero las ejecuta de a una por vez, o al menos una operación por núcleo. En cambio, las tarjetas gráficas o GPU como se puede apreciar en la Fig. 2 disponen de muchos más núcleos por lo que pueden realizar muchas más operaciones en el mismo tiempo. La GPU (Unidad de Procesamiento Gráfico) es un coprocesador dedicado al procesamiento de imágenes, tiene miles de núcleos optimizados para trabajar en paralelo con operaciones sencillas, por lo que está especialmente preparada para el cálculo matricial. Básicamente, al ser otro procesador añadido, su función es la de liberar la carga del CPU, aumentando el rendimiento de nuestro ordenador. Toda esta potencia de cálculo aritmético puede emplearse para la computación de los algoritmos de Deep Learning.

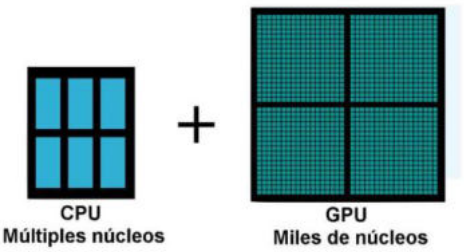

Fig. 2. Comparación entre núcleos de CPU y núcleos de GPU

2.3 Desarrollo

Durante el desarrollo del proyecto se implementaron aproximadamente 80 modelos de redes neuronales convolucionales ([19] y [20]), 60 de estos modelos fueron implementados con aprendizaje desde cero y el resto de los modelos fueron desarrollados utilizando técnicas de transferencia de aprendizaje. A su vez, las pruebas

ISBN 978 -987-633-574-4

se implementaron con 2 sets de datos, el primero con una división de dos clases y, el segundo conjunto de datos, fue dividido en 6 clases. A continuación, se detallará mejor el contenido de cada set de datos.

2.3.1 Modelo de clasificación binaria

En este caso los datos (set de datos 1) fueron divididos en 2 clases (reciclable y no reciclable), la cantidad total de imágenes de este set es de 8340 imágenes muy bien balanceadas en ambas clases, a su vez, se dividieron los datos en tres partes separando 100 imágenes para el testeo del modelo, 1708 imágenes para la validación y 6532 imágenes utilizadas para el entrenamiento. Antes de ser enviadas al modelo como entradas, se realizó un redimensionamiento por software a cada una de las imágenes para que todas tengan un tamaño de 200 x 200 píxeles. El mismo proceso de redimensionamiento se debe realizar a las nuevas imágenes que el modelo deberá clasificar luego del entrenamiento.

2.3.2 Modelo de clasificación multiclase

Para el set de datos 2 se incrementó significativamente la cantidad de imágenes y se realizó una división en 6 clases (orgánico, papel/cartón, metal, vidrio, plástico, y no reciclable). La cantidad total de imágenes de este set es de 15105 que se buscó balancear entre las 6 clases disponibles, a su vez, se dividieron los datos en tres partes separando 90 imágenes para el testeo del modelo, 3039 imágenes para la validación y 11976 imágenes utilizadas para el entrenamiento. Antes de ser enviadas al modelo como entradas, se realizó un redimensionamiento como en el modelo anterior.

3 Resultados

Durante todo el proyecto se modelaron aproximadamente 80 redes neuronales convolucionales, de las cuales los primeros 60 modelos fueron entrenados desde el inicio y los restantes 20 modelos fueron entrenados utilizando transfer learning basados en varias redes pre-entrenadas como VGG16, VGG19, ResNet50, MobileNet y XceptionV3.
Para el entrenamiento de los distintos modelos se necesitaron alrededor de dos meses de horas máquina, ya que los modelos más simples llevaron un tiempo promedio de entrenamiento de 16 horas y los modelos más complejos tuvieron un tiempo promedio de entrenamiento de 60 horas, llegando a un promedio general de 36 horas de entrenamiento por modelo propuesto.
Uno de los principales problemas que se detectó durante el desarrollo de los primeros modelos fue la presencia de overfitting, logrando buenos resultados para el set de entrenamiento y malos para el set de validación. Para los primeros modelos los resultados rondaban el 60% de acierto para la predicción de nuevas imágenes, por lo tanto, para lograr mejorar estos resultados se fueron probando pequeñas variaciones en los hiperparámetros de la red y la profundidad de los modelos. Además, se implementaron algunas de las técnicas como data augmentation, early stopping y

ISBN 978 -987-633-574-4

dropout, para lograr reducir el sobreajuste del modelo, siendo de mucha utilidad las funciones de “callbacks” que permite utilizar la librería de alto nivel Keras [16].
Las funciones “callbacks” son funciones que se ejecutan luego de cada iteración de entrenamiento, permitiendo por ejemplo realizar modificaciones al LR (learning rate) o guardar los pesos del modelo en ese momento si se cumple alguna condición predefinida. Esto se consigue debido a que estas funciones tienen acceso a todas las variables involucradas durante el entrenamiento como, por ejemplo, el porcentaje de acierto ya sea para el set de entrenamiento como para el set de validación. Las funciones elegidas fueron tres, en primer lugar, se utilizó la función “EarlyStopping” a la que se le indica la cantidad de épocas (iteraciones) que se debe esperar para terminar el entrenamiento siempre y cuando se cumpla una condición aplicada a una variable que se va monitoreando; en este caso se monitorea el valor de la función de pérdida de los datos de validación y la función corta el entrenamiento si este dato no disminuye luego de las iteraciones indicadas. En segundo lugar, se utilizó la función “ModelCheckpoint”, que permite guardar los valores de los pesos del modelo obtenido en cada iteración y una vez terminado el entrenamiento restaurar los mejores pesos, en donde el modelo se desempeña de mejor manera. Por último, la función utilizada fue “ReduceLROnPlateau”, que permite reducir el hiperparámetro LR en un factor que se define cuando se llama a la función y debe ser mayor a 0 y menor a 1, permitiendo una mayor reducción cuando el número es más cercano al nivel inferior permitido. Para el caso de los modelos planteados la función monitoreaba el valor del porcentaje de acierto del set de validación y los mejores resultados se obtuvieron cuando el factor por el que se multiplicaba el LR era “0.9” y el porcentaje de épocas a esperar para realizar el cambio era de alrededor de un 15% a un 20% de la cantidad total de épocas definidas para el entrenamiento.
Por otra parte, también se aplicó la técnica de dropout a la capa completamente conectada del modelo, variando el porcentaje de neuronas al que se le aplicaba dependiendo de la cantidad de neuronas que contenía cada capa, (este porcentaje se fue variando entre un 25% y un 75% en los distintos modelos). Esta técnica permite la desconexión de un % de neuronas para evitar que el modelo memorice un resultado.

Todas estas modificaciones permitieron una mejora considerable en el porcentaje de acierto de los modelos propuestos llegando a valores superiores al 72% para el set de datos que estaba dividido en seis clases y valores superiores al 84% para el set de datos dividido en dos clases. Una vez obtenidos estos resultados se comenzaron a aplicar los modelos pre entrenados con los cuales se logró obtener resultados mayores al 90% de acierto para el set de dos clases y valores cercanos al 80% para el set de seis clases. En el caso de los modelos que utilizaron transfer learning también se aplicaron las funciones “callbacks”.

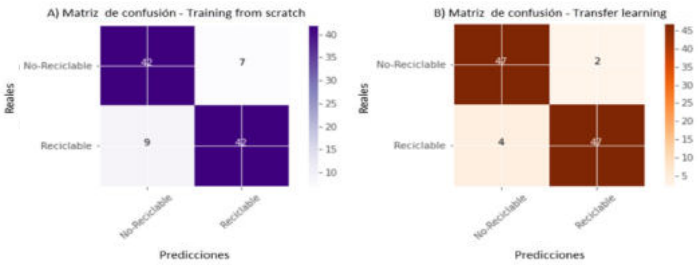

Fig. 3 – Matrices de confusion modelos binarios

ISBN 978 -987-633-574-4

Se presenta a continuación las matrices de confusión obtenidas de los 4 mejores modelos implementados, en la figura 3-A la matriz del modelo binario entrenado desde cero y la figura 3-B el modelo binario con transfer learning.

En la figura 4-A se representa la matriz de confusión correspondiente al modelo multiclase con entrenamiento desde cero, y en la figura 3-B la matriz del modelo con transfer learning.

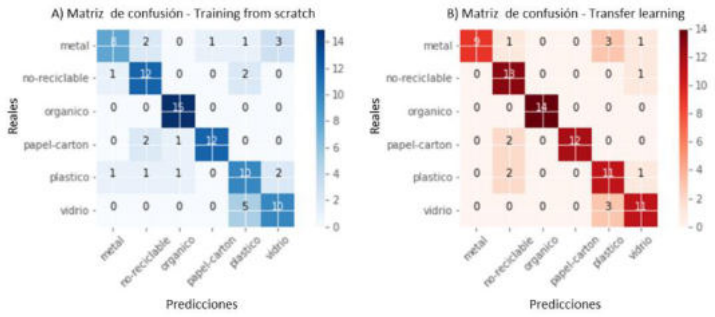

Fig. 4 – Matrices de confusión modelos multiclase

A continuación, se muestran los fragmentos de código utilizados para realizar nuevas predicciones con uno de los modelos planteados:

```
#se carga el modelo guardado ya entrenado
altura, ancho= 200,200
models='C:/temporal pps/modelo-8.4.1.vgg19/modelo.h5'
weights='C:/temporal pps/modelo-8.4.1.vgg19/pesos.h5'
cnn=tf.keras.models.load_model(models)
cnn.load_weights(weights)
#función que realiza la predicción
def predict(file):
    image=load_img(file, target_size=(altura, ancho))
    image=img_to_array(image)
    image=np.expand_dims(image, axis=0)
    prediction=cnn.predict(image)
    result=prediction[0]
    reply=np.argmax(result)
    if reply==0:
        clase='0 - METAL'
    elif reply==1:
        clase='1 - NO-RECICLABLE'
    elif reply==2:
        clase='2 - ORGANICO'
    elif reply==3:
        clase='3 - PAPEL-CARTON'
    elif reply==4:
        clase='4 - PLASTICO'
    elif reply==5:
        clase='5 - VIDRIO'
```

ISBN 978 -987-633-574-4


```
    image=mpimg.imread(file)
    plt.imshow(image, interpolation=None)
    plt.title('Categoría predicha de imagen: ' + clase)
    plt.axis('off')
    return
```

La función definida realiza la predicción sobre la imagen que recibe como parámetro e imprime el resultado de la predicción junto con la imagen analizada. A continuación, en la Fig. 5 se presentan tres ejemplos de predicciones tomadas con la cámara de la Raspberry y pasadas a la función, además se presenta un ejemplo de un falso positivo en donde se fotografió un envase de plástico y fue tomado por la red como uno de vidrio, aunque al mirar la imagen no es tan sencillo para el ojo humano hacer una buena predicción sobre la misma.

La aplicación (versión Beta), fue implementada en un miniordenador Raspberry Pi 3 Model B+, en donde se hace uso del módulo de la cámara (pi camera) de la Raspberry para tomar fotos en tiempo real y realizar la clasificación de dicha imagen determinando que tipo de objeto reciclable se encuentra en ella. Dentro de los resultados obtenidos se está evaluando no solo el porcentaje de acierto, sino también, los tiempos de predicción. Si bien se detectó que la carga inicial del modelo de red neuronal tiene una latencia de entre 30 y 40 segundos, luego, la captura de la imagen y posterior clasificación arrojó tiempos aproximados a los 10 segundos.

(a) Categoría plástico (b) Categoría metal

(c) Categoría No reciclable (d) falso positivo

Fig. 5. Resultados más significativos de las pruebas realizadas

ISBN 978 -987-633-574-4

4 Conclusiones

Durante el desarrollo e implementación de este trabajo se estudiaron algunas técnicas de Machine Learning aplicadas a la clasificación de imágenes y particularmente se buscó colaborar con el cuidado del medio ambiente utilizando la inteligencia artificial aplicada a la clasificación de objetos reciclables.

De por sí la implementación del sistema se encuentra atada a la complejidad de los altos requerimientos de una red neuronal convolucional, pero de todas formas, se logró implementar una red funcional que supera los objetivos planteados durante las primeras etapas del trabajo.

Se ha elegido Deep Learning (DL) para llevar a cabo este trabajo debido a la alta capacidad que presenta para el análisis de imágenes. Sin embargo, al programar algoritmos de DL, específicamente redes neuronales convolucionales, se presentaron algunas dificultades.

Es muy importante y complejo determinar los parámetros e hiperparámetros que mejor se adecuen al modelo propuesto impactando directamente en el resultado final. Este ajuste “fine tuning” muchas veces se considera un arte y no una ciencia para los que se inician en el campo del DL, debido a que se requiere cierta experiencia e intuición para encontrar los valores óptimos de estos hiperparámetros. En este caso, la investigación ha jugado un papel fundamental, además los parámetros e hiperparámetros se deben especificar antes de iniciar el proceso de entrenamiento. Otro aspecto que se debe tener en cuenta es la profundidad y la cantidad de capas de los modelos propuestos.

Por otro lado, la correcta elección y confección del set de datos que se utilizará para entrenar los modelos es vital para llegar a los resultados deseados. La red debe conocer de igual manera todos los elementos o clases que se quieren predecir o clasificar, por lo que, las distintas clases deben estar correctamente balanceadas, contener ejemplos representativos y, además, contar con la mayor cantidad posible de datos para lograr buenas soluciones, aunque esto se traduce a un mayor tiempo de entrenamiento. Durante el trabajo se demostró que las redes neuronales convolucionales presentan excelentes resultados en el campo de la “computer vision”, aunque uno de los inconvenientes encontrados ha sido el tiempo, ya que, a medida que los modelos se hacen más complejos crecen los tiempos de entrenamiento, llegando en el caso de este trabajo a un promedio de 36 horas para el entrenamiento de cada uno de los modelos propuestos.

Se debe tener en cuenta también que trabajar con CNN y grandes conjuntos de datos implica un alto costo computacional, por lo tanto, es muy importante contar con equipos que presenten altas prestaciones e incluso contar con GPUs (unidades de procesamiento gráfico).

En referencia a los resultados obtenidos para el conjunto de datos de prueba se puede decir que fueron satisfactorios, estos superaron las expectativas propuestas al inicio del trabajo. Además, todas las pruebas realizadas sirven como aprendizaje para futuros proyectos.

4.1 Líneas futuras

En primer lugar, se puede mejorar y aumentar el set de datos permitiendo el reentrenamiento de los modelos planteados con estas mejoras en los datos, lo que presentará una mayor robustez y mejor balanceo en las clases a clasificar.

En segundo lugar, y luego de haber mejorado el conjunto de datos, se deberían implementar nuevos modelos con distintas variaciones en los parámetros e hiperparámetros, buscando una mejora en los resultados para las distintas predicciones.

Por último, se podría implementar esta aplicación en algún sistema robótico como, por ejemplo, un cesto inteligente para separar los residuos reciclables.

Referencias

1. Maquituls España (2017), La importancia del reciclaje. Cuidemos el Medio Ambiente, https://www.maquituls.es/noticias/la-importancia-del-reciclaje-cuidemos-el-medio-ambiente/#:~:text=El%20reciclar%20o%20el%20reciclaje,de%20manera%20continua%20al%20planeta., recuperado 10 de mayo 2020.
2. Diario El Cronista (2018), Producción de basura: cuál es la realidad en Argentina y que se podría hacer, https://www.cronista.com/responsabilidad/Produccion-de-basura-cual-es-la-realidad-en-Argentina-y-que-se-podria-hacer-20180302-0075.html, recuperado 10 de mayo 2020.
3. Transfer Learning Wikipedia (2020), Transfer Learning, https://en.wikipedia.org/wiki/Transfer_learning, recuperado 01 julio de 2020.
4. Wolfgang Ertel, Introduction to Artificial Intelligence, second edition, Springer International Publishing AG 2017.
5. Kevin Murphy, Machine Learning - A probabilistic perspective, University of Cambridge, 2012
6. Zoubin Gharamani, Automatic Machine Learning, University of Cambridge, 2018
7. Miroslav Kubat, An Introduction to Machine Learning, second edition, Springer International Publishing AG 2017.
8. Ian Goodfellow, Yoshua Bengio, Aaron Courville, Deep Learning, MIT 2017.
9. Nikhil Buduma, Fundamentals of Deep Learning, Editorial O'reilly. 2017
10. Sandro Skansi, Introduction to Deep Learning – From Logical Calculus to Artificial
11. Andreas Muller, Sarah Guido, Introduction to Machine Learning with Python, Editorial O'reilly, 2016.
12. Francois Chollet, Deep Learning with Python, MEAP edition, Manning Publications 2017.
13. Aurelien Gerón, Hands-On Machine Learning withs cikitLearn & TensorFlow, Editorial O'reilly, 2017
14. Tom Hope, Yehezkel Resheff, Itay Lieder, Learning TensorFlow, Editorial O'reilly. 2017
15. TensoFlow (2020), TensorFlow API documentation, https://www.tensorflow.org/api_docs/, recuperado 01 octubre de 2020.
16. Keras io (2020), Keras API references, https://keras.io/api/, recuperado 01 octubre de 2020.
17. Aprende Machine Learning (2020), https://www.aprendemachinelearning.com/, recuperado 25 de abril 2020.
18. Charu C. Aggarwal, Neural Network and Deep Learning, Springer International Publishing AG part of Springer Nature 2018.
19. C. Jay Kuo, Understanding Convolutional Neural Networks with A Mathematical Model, Department of Electrical Engineering University of Southern California 2016
20. Dominik Scherer, Andreas Muller, Sven Behnke, Evaluation of Pooling Operations in Convolutional Architectures for Object Recognition, University of Bonn, Institute of Computer Science VI ,2010.