Deep Learning como modelo predictivo para clasificar dígitos manuscritos

Deep Learning as a predictive model to classify handwritten digits

Omar Alexander Ruiz-Vivanco

***Resumen*—En este trabajo se presentan los resultados de aplicar modelos de predicción Deep Learning para identificar el dígito de una imagen que contiene un número escrito a mano de la base de datos MNIST, este conjunto de dataset se obtiene de la competencia de Kaggle: Digit Recognizer. Como primer paso se utilizan técnicas de preprocesamiento de imágenes las cuales se enfocan en obtener una imagen lo más nítida posible y la reducción de tamaño de la misma, objetivos que se logran con técnicas de umbralización por el Método de Otsu, transformada de Wavelet de Haar y el Análisis de sus Componentes Principales (ACP), para obtener como resultado un conjunto de nuevos dataset para ser evaluados. En el siguiente paso se toman estos dataset y se aplican los modelos Deep Learning MxNET y H2o ejecutados en el lenguaje estadístico R, obteniendo varias predicciones. Por último se envió a la competencia Digit Recognizer la mejor de las predicciones obtenidas en el experimento, y el resultado de la evaluación de la misma fue de 99,129 % de predicción.**

***Palabras Claves*—Inteligencia Artificial, Deep Learning, Dígitos manuscritos, Método Otsu, Wavelet Haar.**

***Abstract*—In this research work, the results of applying Deep Learning prediction models to identify the digit of an image, that contains a handwritten number of the MNIST database, are presented. This set of dataset was acquired from the competition of Kaggle: Digit Recognizer. The following process was applied: First, image preprocessing techniques were used, which focus on obtaining a pretty clear image and to reduce the size of the same, these objectives that are achieved with Otsu Method, transformed from Haar Wavelet and the Principal Component Analysis (PCA), thus obtaining as a result, one set of new dataset to be evaluated. Second, the Deep Learning MxNET and H2o models, which were executed in the statistical language R, were applied to these datasets obtained, this way, several predictions were acquired. Finally, the best obtained predictions in the experiment were sent to the Digit Recognizer competition, and the results of this evaluation scored 99,129% of prediction.**

***Index Terms*—Artificial Intelligence, Deep Learning, Handwritten digits, Otsu Method, Wavelet Haar.**

I. Introducción

La investigación se fundamenta en la competición de Kaggle denominada Digit Recognizer [1], cuya finalidad es el reconocimiento de dígitos escritos a mano. Kaggle es una plataforma en donde las empresas y los investigadores publican datos y estadísticas con el objetivo de brindar a participantes de cualquier parte del mundo la posibilidad de producir los mejores modelos predictivos a través de competiciones en donde se presentan un sinnúmero de estrategias para el mismo problema [2]. Los dataset para esta competición en particular son tomados de MNIST [3] que es una gran base de datos de dígitos escritos a mano de acceso público y usada ampliamente para entrenamiento y pruebas en el campo de aprendizaje automático [4].

De acuerdo con [3] sacar conclusiones razonables a partir de experimentos de aprendizaje requiere que el resultado sea independiente de la elección de los conjuntos de entrenamiento y de prueba. La base de datos MNIST tiene 70.000 registros particionada en dos dataset, el primero con 60.000 ejemplos utilizados como dataset de entrenamiento (training) y el segundo con 10.000 ejemplos utilizados como dataset de prueba (test). Esta partición es estándar en la definición de MNIST [3] y utilizada en ese formato por la mayoría de trabajos que han experimentado con el dataset [5], [6], [7], [8], [13]. La base de datos MNIST se elaboró con la colaboración de estudiantes de secundaria y empleados de la oficina del censo mostrando diferencias en los trazos, siendo los dígitos formado por el segundo grupo mis limpio y más fácil de reconocer, MNIST es la mezcla aleatoria de ambos [3], [4].

La competencia Digit Recognizer [1] propone la utilización de la base de datos MNIST con una partición diferente, los registros se han mezclado y dividido con 42.000 elementos para formar el dataset de entrenamiento y 28.000 elementos para el dataset de prueba. Esta partición fué la que se tomó para la experimentación en el presente trabajo, siendo mejor en el sentido de que al mezclar la partición original de MNIST obliga a que en el aprendizaje de los dígitos se tomen nuevos elementos al igual que en el test. La imagen del dígito manuscrito está formada por 784 columnas con valores entre 0 para el color blanco y 255 para el color negro que representan los píxeles que conforman la imagen de 28x28 en escala de grises. El dataset de entrenamiento es el único que contiene una columna adicional en cada fila de elementos que etiqueta el dígito de la imagen. En la Figura 1 se muestran 100 dígitos tomados al azar del dataset de entrenamiento de Kaggle, el dígito manuscrito de color negro y la etiqueta de color rojo.

Otras bases de datos utilizadas para el reconocimiento de patrones con varias publicaciones han sido **CENPARMI** (Centre for Pattern Recognition and Machine Intelligence) [10] fundado en 1972 por los profesores del Departamento de Ciencias de la Computación de la Universidad de Concordia en Canadá, cuyos miembros se dedican a la investigación multidisciplinaria del reconocimiento de escritura a mano y

Figura 1. Dígitos del dataset de entrenamiento de Kaggle

verificación de firma, análisis de imágenes y procesamiento de documentos, combinando diferentes técnicas de clasificación. **CEDAR** [11] imágenes de caracteres japoneses creada en la Universidad Estatal de Nueva York en Buffalo desde 1978, estas imágenes se extrajeron de una variedad de fuentes de documentos, incluidos libros, faxes, revistas, impresoras láser, revistas y periódicos. Sin embargo MNIST es la base de datos de mayor utilización en experimentos de aprendizaje máquina especializada en patrones de dígitos escritos a mano.

La principal contribución de este trabajo es la combinación de las técnicas de preproceso de imágenes con los modelos de predicción, trabajos previos no realizan un análisis de preprocesamiento de imágenes como en [12] donde se utiliza una arquitectura de redes neuronales, en [7] se compara el rendimiento de varios algoritmos clasificadores en una base de datos estándar de dígitos manuscritos, en [8] se crea un clasificador neuronal Limited Receptive Area (LIRA), en [13] se utiliza retro-propagación en línea para perceptrones lisos de múltiples capas.

Tomando en consideración lo mencionado se estima que la combinación de técnicas previas para el tratamiento de imágenes y los modelos predictivos, según los resultados del presente trabajo, se obtienen predicciones comparables con los principales artículos científicos escritos sobre el tema, lo que justifica por si mismo la razón de dar conocer esta investigación que es la síntesis del trabajo de tesis ***Análisis predictivo para clasificar dígitos escritos a mano utilizando la base de datos MNIST*** en [14] del autor.

II. TÉCNICAS Y MODELOS

Las utilidades que se obtienen hoy en día del reconocimiento automático de dígitos a partir de una imagen pueden ser muy variadas, lo importante es desarrollar una técnica eficiente a partir de datos de entrenamiento. El aprendizaje de máquina (machine learning) consiste en "enseñar", a un computador a reconocer dígitos escritos a mano por distintas personas en donde factores como intensidad del trazo, curvatura, posición de la mano, etc., generan múltiples variaciones del dibujo lo que constituye el problema a resolver.

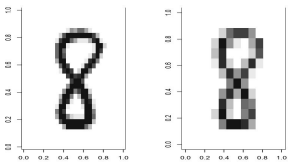

Figura 2. Reducción del dígito 8 dataset Kaggle

En esta sección se mencionan las técnicas de preprocesamiento de imágenes utilizadas para mejorar la calidad de las mismas. En la minería de datos se extrae información de un conjunto de datos para transformarla en una estructura más comprensible y manejable. La técnica a utilizar es el aprendizaje supervisado en donde se deduce una función a partir de datos de entrenamiento para obtener una etiqueta de clase [15], es decir la clasificación de dígitos del 0 al 9. Esta tarea se lleva a cabo realizando un análisis de los datos iniciales para encontrar las técnicas de preproceso más adecuadas para luego aplicar el modelo predictivo que maximice el grado de acierto en la predicción de los dígitos. Estas técnicas utilizadas son Wavelet Haar [20] para reducir el tamaño de la imagen. Umbralización por Otsu [16] para obtener una imagen más nítida y por último PCA [21] para el análisis de los componentes principales.

II-A. Material y métodos

II-A1. Análisis del dataset: Los dígitos del dataset contienen varias formas de escritura con diferentes trazos, curvatura, forma, etc., es decir diferencias muy particulares de cada escritor. Con el objetivo de encontrar tendencias en la escritura de los dígitos se analiza el nivel de intensidad de cada uno, añadiendo esta nueva característica al dataset mediante la obtención de la media. Esta característica de intensidad de trazo puede ser útil al momento de realizar el análisis predictivo. Con esta técnica se obtiene el dataset llamado: "train_kaggle_int".

II-A2. Reducción de la imagen: La reducción de la dimensión de los datos se logra aplicando la transformada de Wavelet Haar, esta función transforma cualquier secuencia 1

$$(a_0, a_1, ..., a_{2n}, a_{2n+1}), \tag{1}$$

de cualquier longitud en una secuencia de dos componentes vectoriales 2

$$((a_0, a_1), ..., (a_{2n}, a_{2n+1})), \tag{2}$$

Si se multiplica por la derecha cada vector con la matriz $H2$, se obtiene el resultado 3

$$((s_0, d_0), ..., (s_n, d_n)), \tag{3}$$

de una etapa de la transformada rápida de wavelet de Haar. La función devuelve las secuencias s y d y se utiliza la secuencia s para continuar con los cálculos [20]. En el dataset train de Kaggle se utiliza dos etapas de cálculo equivalentes a tomar cuatro píxeles continuos y promediar su valor para obtener un nuevo píxel que remplace a los anteriores. El objetivo es reducir la imagen de cada dígito manuscrito de 28x28 píxeles

a una imagen de 14x14 píxeles. Para lo cuál se utiliza el principio de Wavelet de Haar. En R la función ht, de la librería binhf [22], proporciona la transformada. El Algoritmo 1 es utilizado para lograr la reducción del dataset original de 784 columnas a 196.

Algoritmo 1 Transformada Wavelet de Haar

Entrada: Seleccionar matriz original (28x28)
Salida: Matriz Reducida (14x14)

```
para i = 1 hasta 14 veces hacer
    para i = 1 hasta 14 veces hacer
        vector = matriz original[i, j]
        haar = funcion wavelet haar(vector)
        haar = funcion wavelet haar(haar$s)
        matriz reducida[i, j] = funcion redondear(haar$s)
    fin para
fin para
devolver matriz reducida
```

El resultado de esta fase devuelve un dataset con una reducción a 196 columnas más la columna de etiqueta, Figura 2. Se obtiene la reducción del dataset llamado: "train_kaggle_r".

II-A3. Binarización de la imagen: La binarización es el método mediante el cuál se convierte los píxeles de una imagen de varios valores a únicamente dos (negro: 255 y blanco: 0). Para lo cual se establece un umbral sobre el cual todos los valores se convierten en 255 y por debajo de este en 0. El método de Otsu [16], [17] encuentra el umbral óptimo (threshold) maximizando la varianza entre clases (between-class variance) mediante una búsqueda exhaustiva. La umbralización se utiliza cuando la diferencia entre la imagen y el fondo existe una clara diferencia [18].

Se fundamenta en la similitud entre los píxeles que forman el objeto principal y sus diferencias respecto al resto de objetos. La escena debería tener un fondo uniforme y objetos parecidos. A continuación se describe la fórmula de la técnica de Otsu basada en estadísticas sobre el histograma unidimensional de una imagen [19].

$$T = max(\sigma^2) \tag{4}$$

siendo

$$\sigma^2 = w_B(\mu_B - \mu)^2 + w_F(\mu_F - \mu)^2 \tag{5}$$

$$w_k = \sum_{i=0}^{k} p_i \tag{6}$$

$$\mu_k = \sum_{i=0}^{k} i.p_i \tag{7}$$

$$\mu_c = \frac{\mu_k}{w_k} \tag{8}$$

Donde pi es la probabilidad de aparición de un determinado nivel i. Se aplica la binarización, Figura 3, al dataset train original de 784 columnas, y al dataset train reducido de 196 columnas, además se aumenta la columna de intensidad a ambos para obtener un total de 4 dataset como resultado de esta fase: "train_kaggle_bo", "train_kaggle_br", "train_kaggle_boi", y "train_kaggle_bri".

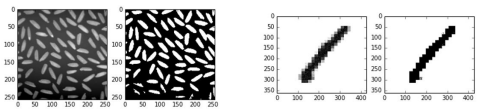

Figura 3. Binarización: Izq: grano de arroz, Der: dígito 1 dataset Kaggle

II-A4. Extracción de nuevas variables: El análisis de componentes principales es una técnica de transformación con el objetivo de identificar los principales factores que explican las diferencias en los datos y una descripción más comprimida de sus correlaciones. Constituye una herramienta para extraer tendencias generales de un conjunto de datos. PCA ordena los componentes principales por su significado ofreciendo una excelente base para la reducción de la dimensión de datos multidimensionales [21]. Este proceso identifica un grupo de variables ficticias que se forman de la combinación de las anteriores observadas. De esta forma se sintetizan los datos y se relacionan entre sí, sin recurrir a ninguna hipótesis previa sobre el significado de cada factor inicial. Los componentes principales obtenidos mediante el proceso de cálculo de raíces y vectores característicos de una matriz simétrica contienen la mayoría de la varianza observada, evitando la información redundante. Para esto las variables deben ser incorreladas entre sí y se expresan como combinación lineal de las demás variables que han sido observadas. El porcentaje de mayor varianza mostrada en cada una de estas componentes se traduce en que las mismas contiene mayor cantidad de información. Se aplica la reducción por PCA para las 64 primeras componentes de los dataset: train kaggle bo y train kaggle br describiendo el 99.21 % y el 99.28 % de las características respectivamente. Dataset resultantes de este proceso: "train_kaggle_pca_br", "train_kaggle_pca_bo".

II-A5. Modelo predictivo: El Deep Learning [9] forma parte de un conjunto amplio de métodos de aprendizaje automático los cuales se basan en aprender representaciones de datos, mediante una serie de operaciones matemáticas sobre una lista de números, dando como resultado otra lista de números. Por ejemplo el reconocimiento de imágenes, se codifica como una lista de números, por tanto, la red neuronal recibiría tantos números en su entrada como píxeles tienen las imágenes o el caso de imágenes en color tres por cada píxel. Su salida bastaría con un solo número cercano a 1.0 para determinar si se trata de un rostro (figura, letra, dígito, etc.) o cercano a 0.0 si no reconoce un rostro, los valores intermedios se interpretan como inseguridad o probabilidad.

La arquitectura de las redes de neuronas [23] se basa en la conexión que tienen las neuronas en cada capa con la siguiente, estas conexiones tienen asociado un número llamado peso. La principal operación dentro de las redes consiste en multiplicar los valores de una neurona por los pesos de sus conexiones salientes, luego cada neurona de la siguiente capa suma todos los input que recibe de las conexiones entrantes. En cada capa se utiliza una función de activación que se encarga de mantener los números producidos por cada neurona dentro de un rango razonable, números reales entre 0 y 1. La función más habitual es la sigmoide que es no lineal, es decir que si se grafican los valores de entrada y de salida el resultado no sería una línea.

Tabla I
RESULTADOS DEL PREPROCESAMIENTO

Preprocesamiento del dataset train de Kaggle			
Tratamiento	Dataset Resultante	Dimensión	Tiempo (seg.)
ninguno	train_kaggle	42000 x 785	0,00
int	train_kaggle_int	42000 x 786	4,57
red	train_kaggle_r	42000 x 197	110,17
bin	train_kaggle_bo	42000 x 785	288,19
red + bin	train_kaggle_br	42000 x 197	162,91
bin + int	train_train_kaggle_boi	42000 x 786	292,76
red + bin + int	train_kaggle_bri	42000 x 198	167,48
bin + pca	train_kaggle_pca_bo	42000 x 65	315,68
red + bin + pca	train_kaggle_pca_br	42000 x 65	164,96

bin = binarización (umbralización por método de Otsu)
red = reducción (transformada Wavelet de Haar)
int = intensidad (aumento de nueva característica)
pca = análisis de componentes principales

Tabla II
RESULTADOS DE LA EXPERIMENTACIÓN DEL ANÁLISIS PREDICTIVO

Experimentación del modelo predictivo con los dataset obtenidos			
Dataset	Accuracy (%)	T. (seg.)	T. Total (seg.)
Modelo h2o			
train_kaggle	98,41	22371,26	22371,26
train_kaggle_bo	98,22	18331,55	18619,74
train_kaggle_br	97,59	26224,96	26387,87
train_kaggle_boi	98,09	15515,57	15808,33
train_kaggle_bri	97,53	27115,42	27282,90
train_kaggle_pca_bo	98,49	15027,91	15343,59
train_kaggle_pca_br	97,68	18342,16	18507,12
Modelo MXNet			
train_kaggle	98,04	4291,57	4291,57
train_kaggle_bo	98,82	4875,65	5163,84
train_kaggle_br	97,15	879,83	1042,74
train_kaggle_pca_bo	81,89	416,20	731,88
train_kaggle_pca_br	85,60	413,43	578,39
Modelo RN 2l			
train_kaggle	93,55	806,36	806,36
train_kaggle_bo	94,33	1025,34	1313,53
train_kaggle_br	91,12	369,55	532,46

Cuando se nombra a una red neuronal se lo hace asi: n-layer. La capa de salida se toma para representar los resultados de clase, 10 en el caso de los dígitos manuscritos de Kaggle. Se dimensiona la red neuronal por el número de neuronas o por el número de parámetros.

III. EXPERIMENTACIÓN Y RESULTADOS

Se procura que la experimentación con los modelos predictivos sea en las mismas condiciones, para esto se fija la misma semilla[1] en todas las pruebas y se realiza el mismo procedimiento. Teniendo en cuenta que el dataset test proporcionado por Kaggle no contiene la columna de etiqueta se hace difícil la comprobación de la predicción realizada en la experimentación. Se crea por lo tanto una muestra a partir del dataset train para obtener nuevos conjuntos de training y testing que contengan ambos la etiqueta del dígito a predecir. Para estas pruebas en particular se obtienen un subconjunto de 10.000 dígitos para testing y otro de 32.000 dígitos para training.

III-A. Experimentos con los modelos predictivos

III-A1. Preprocesamiento de datos: En el transcurso del presente trabajo se obtuvieron algunos dataset con diferente tipo de tratamiento previo. La Tabla I muestra el resumen del preprocesamiento realizado con el dataset train oiginal de Kaggle. Se puede identificar el tratamiento realizado, el nombre del dataset resultante con la descripción de sus dimensiones y el tiempo empleado en el proceso expresado en segundos.

III-A2. Análisis predictivo con los modelos: Tres son los modelos predictivos utilizados para la experimentación, el primero h2o con su implementación de Deep Learning ofrece escalabilidad y rapidez con configuraciones que permiten el multiprocesamiento, es decir el uso de dos o más procesadores o CPUs, y entre otras, la posibilidad de especificar el tamaño de memoria, aspectos importantes que hacen de esta herramienta una de las más utilizadas en problemas de clasificación y con suficiente documentación disponible en [24]. La versión y configuración utilizada para estas pruebas permiten el uso de máximo 4 unidades de procesamiento con un tamaño de memoria de 12 GB, pero las restricciones de hardware de la máquina utilizada permite un máximo de 2 CPUs y un máximo de 10,67 GB, lo que hace que los tiempos para este modelo no sean de los mejores presentados.

El segundo modelo MXNet [25] dispone de una flexible y eficiente librería para Deep Learning, se ejecuta en las CPU o GPU4, en los clústeres, servidores, ordenadores de sobremesa, o teléfonos móviles [26]. Este modelo evidencia los mejores resultados en tiempo de ejecución con el mismo hardware. Y por último el tercer modelo es una implementación sencilla codificada en R de una red neuronal 2-layer en la que se pretende realizar un ensayo didáctico enfocado en el conocimiento más que en obtener los mejores resultados. La Tabla II muestra el resumen de la experimentación en donde se identifica el modelo predictivo, el dataset insumo para los cálculos, la exactitud de predicción, el tiempo empleado y el tiempo total que consolida los tiempos globales tomando en cuenta el tiempo de preproceso.

III-B. La competencia de Kaggle Digit Recognizer

El insumo para la presentación en la competencia de Kaggle se obtiene utilizando el modelo y preproceso con el mejor resultado, esto es el modelo MXNet con el dataset binarizado que obtuvo el registro de 98,82 % con un tiempo de 5163,84 segundos. La Tabla III muestra el resultado de la participación en la competencia, alcanzando un 99,129 % de efectividad en la predicción de dígitos manuscritos de la competencia. Participación realizada el 30 de agosto de 2016.

[1]En R se utiliza la función set.seed( ) para obtener números aleatorios reproducibles, el argumento es un número entero cualquiera.

Tabla III
RESULTADOS DE LA COMPETENCIA DE KAGGLE: DIGIT RECOGNIZER

Tratamiento	Tiempo (seg.)	Modelo	Accuracy (%)	Tiempo (seg.)
bin	288,19	MXNet	99,129	6940,212

bin = binarización (umbralización por método de Otsu)

IV. CONCLUSIONES

- El modelo h2o y el modelo MXNet alcanzan similares resultados sin embargo el segundo tiene un coste computacional mucho más bajo, constituyéndose en el modelo con los mejores resultados de la presente investigación.
- La técnica de añadir una nueva característica al dataset mediante la obtención de la media de la intensidad no mejora la efectividad predictiva para ninguno de los dos dataset probados.
- Binarizar la imagen constituye el mejor preproceso de la presente investigación y se evidencia en las imágenes producidas, siendo estas más nítidas que las originales.
- La reducción no mejora los resultados obtenidos con el dataset original esto puede ser comprensible desde el hecho que visualmente se nota la diferencia en las imágenes producidas por cada dataset.
- El porcentaje de efectividad predictiva (accuracy) es alto para toda la experimentación. El mejor resultado global de las pruebas es de 98,82 %, y la calificación obtenida en la participación de la competencia Digit Recognizer es de 99,129 % de efectividad de predicción para datos reales. Esta calificación da como resultado un margen de error del 0,871 %.
- La utilización del método de Otsu combinado con el modelo MXNet presenta el mejor porcentaje de predicción de este trabajo.

AGRADECIMIENTOS

El autor agradece al Dr. Iñaki Inza y al Dr. Yosu Yurramendi director de tesis y responsable, respectivamente, del Máster Universitario en Ingeniería Computacional y Sistemas Inteligentes de la Universidad del País Vasco / Euskal Herriko Unibertsitatea (UPV/EHU), por su apoyo en la consecución de los objetivos académicos planteados en la realización del máster.

REFERENCIAS

[1] Kaggle, (2016). Digit Recognizer. *Learn computer vision fundamentals with the famous MNIST data.* Recuperado de: https://www.kaggle.com/c/digit-recognizer.
[2] Kaggle, (2016). The Home of Data Science & Machine Learning. *Kaggle helps you learn, work, and play.* Recuperado de: https://www.kaggle.com/.
[3] Y. LeCun and C. Cortes and C. Burges, (2015). *THE MNIST DATABASE of handwritten digits*, Recuperado de: http://yann.lecun.com/exdb/mnist/.
[4] L. Deng. (2012). The MNIST Database of Handwritten Digit Images for Machine Learning Research [Best of the Web]. *IEEE Signal Processing Magazine*, 29, pp. 141142.
[5] Y. LeCun, L. Bottou, Y. Bengio, y P. Haffner. (1998). Gradient-based learning applied to document recognition. *IEEE*, 86(11), pp. 2278-2324.
[6] P. Y. Simard, D. Steinkraus, y J. C. Platt. (2003). Best Practices for Convolutional Neural Networks Applied to Visual Document Analysis. Document Analysis and Recognition. *IEEE Computer Society Washington, DC, USA*, pp. 958.
[7] L. Bottou, C. Cortes y J.S. Denke. (2002). Comparison of classifier methods: a case study in handwritten digit recognition. *IEEE*. DOI: 10.1109/ICPR.1994.576879.
[8] E. Kussul, T. Baidyk. (2004). Improved method of handwritten digit recognition tested on MNIST database. *Image and Vision Computing*, 22(12), pp. 971-981.
[9] J. Schmidhuber, (2015). Deep learning in neural networks: An overview. *Neural Networks*, 61, pp. 85 - 117.
[10] CENPARMI, (2010). About CENPARMI. *Centre for Pattern Recognition and Machine Intelligence.* Recuperado de: http://www.concordia.ca/research/cenparmi.html.
[11] CEDAR, (2008). About CEDAR. *Handwriting Recognition.* Recuperado de: https://www.cedar.buffalo.edu/Databases/JOCR/.
[12] D. Ciregan, U. Meier, y J. Schmidhuber. (2012). Multi-column deep neural networks for image classification. *IEEE*, DOI: 10.1109/CVPR.2012.6248110.
[13] D. Ciregan, U. Meier, L. M. Gambardella y J. Schmidhuber. (2012). Deep Big Simple Neural Nets Excel on Handwritten Digit Recognition. *IEEE*, DOI: 10.1162/NECO_a_00052.
[14] Ruiz-Vivanco, Omar. (2016). *Análisis predictivo para clasificar dígitos escritos a mano utilizando la base de datos MNIST* (tesis de maestría). Universidad del País Vasco / Euskal Herriko Unibertsitatea (UPV/EHU), España.
[15] Google Brain Team. (2016). *TensorFlow is an Open Source Software Library for Machine Intelligence*. Recuperado de: https://www.tensorflow.org/.
[16] L. Jianzhuang, L. Wenqing y T. Yupeng. (2002). The Automatic Thresholding of Gray-Level Pictures via Two-Dimensional Otsu Method. *IEEE*, DOI: 10.1109/CICCAS.1991.184351 .
[17] L. Hui, C. Shi, A. Min-si y W. Yi-qi. (2008). Application of an Improved Genetic Algorithm in Image Segmentation. *IEEE*, DOI: 10.1109/CSSE.2008.794.
[18] J. Rodríguez. (2010). *Método otsu*, Recuperado de: https://es.scribd.com/doc/45875911/Metodo.Otsu/.
[19] Universidad Nacional de Quilmes. (2005). *Segmentacioón por umbralizacioón - método de otsu*. Recuperado de: www.iaci.unq.edu.ar.
[20] G. Jin-Sheng, y J. Wei-Sun. (2007). The Haar wavelets operational matrix of integration. *International Journal of Systems Science*, 27, pp. 623-628.
[21] J. Yang, D. Zhang y A.F. Frangi. (2004). Two-dimensional PCA: a new approach to appearance-based face representation and recognition. *IEEE*, DOI: 10.1109/TPAMI.2004.1261097.
[22] M. Nunes. (2014). *Package binhf*, Recuperado de: https://cran.rproject.org/web/packages/binhf/binhf.pdf.
[23] J. Johnson y A. Karpathy. (2016). *Cs231n: Convolutional neural networks for visual recognition*. Recuperado de: http://cs231n.github.io/.
[24] H2O Community. (2016). H2O Open Source Software Documentation. *H2O and Sparkling Water Documentation*. Recuperado de: http://docs.h2o.ai/h2o/latest-stable/index.html.
[25] KDnuggets, (2016), *Top 10 Deep Learning Projects on Github*. Recuperado de: http://www.kdnuggets.com/2016/01/top-10-deeplearninggithub.html.
[26] MXNet Community. (2016). MXNet Architecture. *Flexible and Eficient Library for Deep Learning*. Recuperado de: https://mxnet.readthedocs.io/en/latest/.

Omar Alexander Ruiz-Vivanco Ingeniero en Sistemas por la Universidad Nacional de Loja (2008), Máster Universitario en Ingeniería Computacional y Sistemas Inteligentes por la Universidad del País Vasco / Euskal Herriko Unibertsitatea (UPV/EHU) (2016). Actualmente es Docente - Investigador en la Universidad Técnica Particular de Loja en la Sección de Inteligencia Artificial del Departamento de Ciencias de la Computación y Electrónica.